17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11)

17. GI/ITG Fachtagung *Kommunikation in Verteilten Systemen*, March 8–11, 2011, Kiel, Germany

Edited by

Norbert Luttenberger

Hagen Peters

Editors

Norbert Luttenberger, Hagen Peters
Institut für Informatik
Christian-Albrechts-Universität zu Kiel
{nl|hap}informatik.uni-kiel.de

ACM Classification 1998
C.2.4 Distributed Systems

ISBN 978-3-939897-27-9

Published online and open access by
Schloss Dagstuhl – Leibniz–Zentrum für Informatik gGmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany.

Publication date
March, 2011.

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Digital Object Identifier: 10.4230/OASIcs.KiVS.2011.i

ISBN 978-3-939897-27-9 **ISSN 2190-6807** **http://www.dagstuhl.de/oasics**

OASIcs – OpenAccess Series in Informatics

OASIcs aims at a suitable publication venue to publish peer-reviewed collections of papers emerging from a scientific event. OASIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 2190-6807

www.dagstuhl.de/oasics

Preface

Our world is in its public and private spheres, and equally in its economic, political and cultural domains more and more affected and influenced by the permanent availability of a huge variety of different kinds of information. It seems natural that in this world communication itself, communication in all its facets (i.e. from bearer networks over endsystems to information syntax and semantics) is undergoing a fundamental process of change—a process of change that, though being a constant, is only seldom of such fundamental depth as can be observed today in the networking and communications arena.

Network providers around the world have started to consolidate their infrastructure into All-IP networks. The well-beloved Plain Old Telephone System is going to vanish in the next 10 to 15 years, and its successor is not the long believed-in global ATM network with its controllable QoS, but it is ... plain old IP! (Is POIP being the new abbreviation that we have to teach to our students?) This process is as ambitious as chancy, but seems to be inevitable when regarding the ever increasing data streams that cross the Internet.

Present-day mobile endsystems incorporate capabilities that until yesterday have been reserved to devices that find themselves on desks, but not in pockets. Smart environments hosting ubiquitous computing devices let people experience the world as both a real and a virtual "thing" at the same time. And no doubt: It is the virtual side that today fascinates most people! The emphatic acceptance of the new species of devices lets us easily speculate on even faster, more powerful devices tomorrow.

Upper layers functions (Yes–the authors of these lines are strong followers of ISO's OSI reference model!), though never fully spelt out in the Internet world, are re-animated and augmented in new shape, being called the Semantic Web Layer Cake, where the somewhat disrespectful "cake" seems to reflect the wish for a less bureaucratic kind of architecture definition, something more homebrew. Related efforts strive to bring "better" information to people and even let machines do the information provisioning work. The goal is still very far, but semantic technologies are available today that provide a sound base to build upon.

Only the ongoing security and privacy problems seem to be a permanent companion on the path to a new communications world. He/she changes face, but we seem unable to get rid of this nasty free rider ...

KiVS'11 reflects several aspects of these world trends, and KiVS'11 does it on a high-quality level. From 55 submissions, the program committee accepted 14 papers as full papers, 5 as short papers, and 4 as industry papers. This careful selection ensures that KiVS is going to stay the most important conference on all kinds of communication-related matters in the German-speaking countries.

Though KiVS' focus is clearly on contributions from these countries, it takes another step to becoming more international: The conference proceedings are published in Open Access format for the first time, which makes them easily accessible via the Internet, and English is the only accepted language for the proceedings.

KiVS'11 for the first time hosts an Industry Track. This is to show the close intertwining between academic and industrial research. The publishers think that this track should also enhance further KiVS conferences.

We thank all authors for their efforts in providing their input and hope that both scientific and personal discussions during the conference are rewarding for all participants.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Contents

Full Papers

Short Papers

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Industry Papers

Summaries of Prize-Winning Theses

List of Authors

Nils Aschenbruck
University of Bonn - Institute of Computer Science 4, Germany

Robert Basmadjian
University of Passau, Department of Computer Network and Communication, Germany

Michael Beigl
TecO, Chair for Pervasive Computing Systems, Karlsruhe Institute of Technology (KIT), Germany

Joachim Bleidiessel
Funkwerk Information Technologies GmbH, Germany

Uli Bornhauser
University of Bonn – Institute of Computer Science 4, Germany

Lothar Braun
Network Architectures and Services, Technische Universität München, Germany

Georg Carle
Network Architectures and Services, Technische Universität München, Germany

Vasilios Darlagiannis
Informatics and Telematics Institute, Centre for Research and Technology Hellas, Greece

Michael Dürr
Mobile and Distributed Systems Group, Institute for Informatics, Ludwig-Maximilians-Universität, Germany

Stefan Fischer
Institute of Telematics, University of Lübeck, Germany

Daniel Fischer
SAP AG, Germany

Hannes Frey
University of Paderborn, Germany

Jochen Furthmüller
Institute of Telematics, Karlsruhe Institute of Technology, Germany

Kurt Geihs
Distributed Systems, Universität Kassel, Germany

Elmar Gerhards-Padilla
University of Bonn - Institute of Computer Science 4, Germany

Tobias Gerlach
Operations Research and Stochastics Research Group, Ilmenau University of Technology, Germany

Nils Glombitza
Institute of Telematics, University of Lübeck, Germany

Wolfgang Goerigk
b+m Informatik AG, Germany

Nils Gruschka
NEC Laboratories Europe, Germany

Stephan Heckmüller
Telecommunications and Computer Networks, Universität Hamburg, Germany

Martin Horneffer
Deutsche Telekom AG – Technical Engineering Center, Germany

Oliver Hunte
Fachhochschule Hannover, Fakultät IV (Wirtschaft und Informatik), Germany

Luigi Lo Iacono
European University of Applied Sciences (EUFH), Germany

Carsten Kleiner
Fachhochschule Hannover, Fakultät IV (Wirtschaft und Informatik), Germany

Uwe Koch
Fachhochschule Hannover, Fakultät IV (Wirtschaft und Informatik), Germany

Andrey W. Kolesnikov
University of Hamburg, Department of Informatics, Germany

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Björn Koschel
edatasystems GmbH, Germany

Arne Koschel
Fachhochschule Hannover, Fakultät IV (Wirtschaft und Informatik), Germany

Reinhold Kroeger
RheinMain University of Applied Sciences, Distributed Systems Lab, Germany

Winfried Lamersdorf
Distributed Systems and Information Systems, Computer Science Department, University of Hamburg, Germany

Nicolas C. Liebau
Multimedia Communications Lab, Technische Universität Darmstadt, Germany

Peter Lieven
Heinrich Heine University Düsseldorf, Mobile and Decentralized Networks Group, Germany

Martin Lipphardt
Institute of Telematics, University of Lübeck, Germany

Holger Machens
Hamburg University of Technology, Institute of Telematics, Germany

Alexander Marold
University of Duisburg-Essen, Institute for Experimental Mathematics, Germany

Peter Martini
University of Bonn – Institute of Computer Science 4, Germany

Andreas U. Mauthe
Computing Department, Lancaster University, United Kingdom

Martin Mauve
Institute of Computer Science, Heinrich Heine University Düsseldorf, Germany

Hermann de Meer
University of Passau, Department of Computer Network and Communication, Germany

Gerhard Münz
Network Architectures and Services, Technische Universität München, Germany

Marc C. Necker
Institute of Communication Networks and Computer Engineering, University of Stuttgart, Germany

Jana Neumann
Institute of Information Systems, University of Lübeck, Germany

Stefan Nitz
Fachhochschule Hannover, Fakultät IV (Wirtschaft und Informatik), Germany

Andriy Panchenko
Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, Luxembourg

Benjamin Pesch
Institute of Computer Science, Heinrich Heine University Düsseldorf, Germany

Ranjith Pillay
Amrita Vishwa Vidyapeetham, India

Tobias Pögel
Institute of Operating Systems and Computer Networks, Technische Universität Braunschweig, Germany

Rico Radeke
Technische Universität Dresden, Chair for Telecommunications, Germany

Wolfgang Renz
Multimedia Systems Laboratory, Hamburg University of Applied Sciences, Germany

Jedrzej Rybicki
Institute of Computer Science, Heinrich Heine University Düsseldorf, Germany

Daniel Saur
Distributed Systems, Universität Kassel, Germany

Reiner Saykowski
Funkwerk Information Technologies GmbH, Germany

Guenter Schaefer
Telematics and Computer Networks Research Group, Ilmenau University of Technology, Germany

Björn Scheuermann
Heinrich Heine University Düsseldorf, Mobile and Decentralized Networks Group, Germany

Sebastian Schmerl
Brandenburg University of Technology, Germany

Elferik Schultz
Funkwerk Information Technologies GmbH, Germany

Stephan Sigg
TecO, Chair for Pervasive Computing Systems, Karlsruhe Institute of Technology (KIT), Germany

Hendrik Skubch
Distributed Systems, Universität Kassel, Germany

Ralf Steinmetz
Multimedia Communications Lab, Technische Universität Darmstadt, Germany

Jeanne Stynes
Cork Institute of Technology, Department of Computing, Ireland

Jan Sudeikat
Multimedia Systems Laboratory, Hamburg University of Applied Sciences, Germany

Holger Teske
Institute of Telematics, Karlsruhe Institute of Technology, Germany

Andreas Textor
RheinMain University of Applied Sciences, Distributed Systems Lab, Germany

Volker Turau
Hamburg University of Technology, Institute of Telematics, Germany

Stefan Türk
Technische Universität Dresden, Chair for Telecommunications, Germany

Ante Vilenica
Distributed Systems and Information Systems, Computer Science Department, University of Hamburg, Germany

Michael Vogel
Brandenburg University of Technology, Germany

Oliver P. Waldhorst
Institute of Telematics, Karlsruhe Institute of Technology, Germany

Christian Werner
Institute of Telematics, University of Lübeck, Germany

Kevin Wiesner
Mobile and Distributed Systems Group, Institute for Informatics, Ludwig-Maximilians-Universität, Germany

Bernd E. Wolfinger
University of Hamburg, Department of Informatics, Germany

Sander Wozniak
Telematics and Computer Networks Research Group, Ilmenau University of Technology, Germany

Full Papers

The Scope of the IBGP Routing Anomaly Problem

Uli Bornhauser[1], Peter Martini[1], and Martin Horneffer[2]

1 **University of Bonn – Institute of Computer Science 4**
Roemerstr. 164 – D-53117 Bonn – Germany
{ub, martini}@cs.uni-bonn.de

2 **Deutsche Telekom AG – Technical Engineering Center**
Hammer Str. 216 – D-48153 Muenster – Germany
martin.horneffer@telekom.de

Abstract

Correctness problems in the iBGP routing, the de-facto standard to spread global routing information in Autonomous Systems, are a well-known issue. Configurations may route cost-suboptimal, inconsistent, or even behave non-convergent and -deterministic. However, even if a lot of studies have shown many exemplary problematic configurations, the exact scope of the problem is largely unknown: Up to now, it is not clear which problems may appear under which iBGP architectures.

The exact scope of the iBGP correctness problem is of high theoretical and practical interest. Knowledge on the resistance of specific architecture schemes against certain anomaly classes and the reasons may help to improve other iBGP schemes. Knowledge on the specific problems of the different schemes helps to identify the right scheme for an AS and develop workarounds.

1998 ACM Subject Classification Computer-Communication Networks: Network Protocols – Protocol verification, Routing protocols

Keywords and phrases Inter-domain Routing, Border Gateway Protocol, Routing Anomalies

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.2

1 Introduction

The Internet we know today is a system of independent, interconnected Autonomous Systems (ASs). The Border Gateway Protocol (BGP) [12] is the de-facto standard used to exchange global routing information between ASs. The process of spreading global routing information within ASs is usually implemented via internal BGP (iBGP), a particular operational mode of BGP. However, even if iBGP is widely used, it may come along with correctness problems.

1.1 Motivation and Objectives

While classical iBGP is based on a logical full-mesh, expected scalability problems led quickly to the development of alternative information exchange schemes. Two of these schemes, BGP Route Reflection [2] and AS Confederations [14], were finally standardized, implemented, and frequently deployed. Today, it is well known that these schemes are prone to serious correctness problems [6]. Various studies, cf. [6, 9], for example, have shown different classes of conflicts with basic correctness properties; *iBGP anomalies*. However, even if the causes are known [5], the scope of the problem is still unknown: Are iBGP anomalies a direct consequence of using scalability techniques or may similar problems also appear if full-mesh iBGP is used? What causes may appear under which iBGP information exchange schemes? Is – and how is – it possible to address the causes with operational means, i.e. without any protocol modifications? Adequate answers are of high theoretical and practical interest: They help protocol designers to combine the advantages of different schemes and support operators addressing anomalies.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 2–13

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.2 Directly Related Work

In 2000, studies of Cisco Systems [6] and McPherson et al. [12] showed that the iBGP routing is prone to consistency problems in general. Two years later, Griffin et al. [9] could show that consistency problems are only the tip of the iceberg: They presented a series of simple and largely realistic configurations revealing further correctness problems. In the same year, Basu et al. [1] proved that at least verifying the consistency of an iBGP configuration is NP-hard. Based on this insight, different concepts which try to alleviate or avoid the observed problems were developed in the following years. However, since none of them can be used unconditionally in production systems, iBGP anomalies are still an issue today. An interesting aspect of all previous studies is that anomalies are usually understood as a side-effect of an information reduction compared to fully meshed iBGP [10, 11]. Some studies even go that far to implicitly assume full-mesh iBGP as inherently anomaly-free [7] – without any proof.

Even if the root causes for iBGP anomalies are well-known [5], the exact scope of the problem was never studied. Today, it is known that scalability techniques facilitates effects inducing anomalies. However, a complete, structured overview of the problem does not exist. Problems in full-mesh iBGP are unknown so far. This is the starting point for our analyses.

1.3 Paper Outline

The rest of the paper is organized as follows. At first, in section 2, we summarize the most important aspects of iBGP relevant in the context of this paper. In section 3, we introduce the problems that may come along with iBGP and sketch their root causes. Knowing the different anomaly classes, in section 4 we study the standardized iBGP schemes in terms of their vulnerability to the anomaly classes. In section 5, we give an outlook on possible workarounds and solutions. Finally, we close with a short conclusion and aspects for future work in section 6.

2 The Border Gateway Protocol

The Border Gateway Protocol is a classical path vector protocol. In this section, we outline the main aspects of BGP that are relevant in the context of the following analyses. Nevertheless, readers of this paper should be familiar with internal BGP, BGP Route Reflection, and AS Confederations. Details on iBGP and its exchange schemes may be found in [2, 12] and [14].

2.1 Basics on the Border Gateway Protocol

Today, BGP is the de-facto standard Exterior Gateway Protocol. *BGP speakers* forward traffic via a unique *best path*. This path is determined by applying the BGP *path selection function* on all known paths. Paths are known due to *local configuration* and *advertisements* of system-*external* and -*internal BGP peers*. BGP peers only advertise their best path and only if this path is not already known. To be able to focus on iBGP issues, we assume that the external and local paths each speaker knows are time-invariant. Traffic forwarding via a path is realized via the *BGP next-hop* it specifies. Under common edge conditions (as assumed in this paper), it specifies the router where the path is externally learned or locally configured. Traffic that reaches this router is correctly forwarded out of the AS. The forwarding to the BGP next-hop is realized on a hop-by-hop basis via the shortest Interior Gateway Protocol (IGP) route.

A *BGP path* is defined by *path attributes*. The Local Preference, AS-path Length, Origin Code, Multi Exit Discriminator (MED), and peer-AS are *global* path attributes. In contrast to *local attributes*, they are equal on every router within an AS. They do not depend on the topological position within the AS. Based on the path attributes, paths can be compared.

2.2 The BGP Path Selection Process

Let π denote the paths known by a BGP speaker v for a certain destination. After removing unresolvable paths from π, v's best path is determined by the following selection process. If π is empty, no best path for the destination exists on router v.

Step 0) Remove all paths from π that are not tied for having the lowest **Local Preference**.
Step 1) If any path is **locally configured** on v, remove all non-local paths from π.
Step a) Remove all paths from π that are not tied for having the shortest **AS-path Length**.
Step b) Remove all paths from π that are not tied for specifying the lowest **Origin Code**.
Step c) Remove all paths from π that do not have the best **MED** for their peer-AS group.
Step d) If paths are **externally learned**, remove all internally learned paths from π.
Step e) Remove all paths from π that are not tied for having the lowest **IGP distance** to the specified BGP next-hop.
Step f) Remove all paths from π which are not tied for specifying the lowest **BGP ID**.
Step g) Choose and return the path from π that specifies the lowest **Peer-address**.

An important aspect of the selection process is the fact that MEDs can only be compared if the paths specify the same peer-AS. The peer-AS specifies the source of a routing information. It can be derived from the AS-path attribute, cf. [12]. As MEDs can only be compared within a peer-AS group, *path rankings* may behave non-transitive: For arbitrary paths p, q, and r, it may hold that $p<q$, $q<r$, and $r<p$, where $<$ is the order defined by the selection process, cf. figure 5.a). Details on the other BGP path attributes may be found in [12].

2.3 Terms and Identifiers

To establish a common language, a few terms and identifiers are introduced at first. A path, i.e. the external or local BGP routing information a router knows, is denoted as *BGP path*. A BGP path is propagated via *signaling paths* according to the applied iBGP scheme through an AS. The tuple of BGP path and BGP next-hop specifies all forwarding information a signaling path contains. It is denoted as *delivery path*. A signaling path resolvable at router v that is propagated according to the applied iBGP scheme is called *permitted* at $v \in V$. The set V covers all routers within the AS. The set of paths permitted at v is written as $\mathcal{P}^v$. The selection process of BGP defined above is formalized by the *selection function* λ. λ is always applied for the local view of a router v, written as λ^v, to a subset of its permitted paths. If λ^v is applied up to and including step e) to all paths $\mathcal{P}^v$ permitted at v, the router's *cost-optimal paths* $\Omega_c^v = \lambda^v(\mathcal{P}^v)$ are calculated. They optimize the forwarding costs. The definition stresses the fact that the BGP ID and Peer-address are only tie-breaker criteria. We label signaling paths usually with the letters p or q. A signaling path p permitted at v is usually identified as p_v. The underlying delivery path is labeled as $\dot{p}_v$, the BGP path as $\bar{p}$.

Once traffic reaches the BGP next-hop, it leaves the system by definition. The forwarding to the BGP next-hop is realized via the AS's IGP. The IGP *forwarding route* from v to the BGP next-hop a delivery path $\dot{p}_v$ specifies is labeled as $f_{v \to \dot{p}_v}$. The route without the BGP next-hop (the last physical hop on the IGP route) is written as $f^1_{v \to \dot{p}_v}$.

In what follows, we study the BGP routing in an *iBGP configuration*. Such a configuration is defined by the routers V, the physical network topology in the AS, the BGP paths available at each $v \in V$, and the AS's iBGP peering structure. The routing decision v has made is defined by the delivery path the selected best path represents. The tuple $s = (\dot{p}_1, ..., \dot{p}_n) \,|\, 1 \leq i \leq n, i \in V$ of delivery paths specifies the *state* a configuration has entered. Over time, the configuration moves from state to state until a *stable* state is *entered*: The delivery paths the routers select do not change any more. The set of all states the configuration may enter is written as S.

3 IBGP Routing Anomalies and their Root Causes

Even if the specific demands on routing protocols and information exchange schemes generally differ, correctness specifies a universally valid requirement. Details on this requirement and the causes for the problems iBGP comes along with are discussed in what follows.

3.1 Correctness Properties and Routing Anomalies

The correctness of a routing protocol is determined by three basic properties: *Expressiveness*, *Global Consistency*, and *Robustness* [5, 7, 8]. A protocol behaves correct, if these properties are fulfilled in arbitrary configurations that match the protocol specification. A definition of the three correctness properties is given in what follows. Details may be found in [5].

3.1.1 Expressiveness

Expressiveness is the first correctness property. It states that the routing implements traffic forwarding according to the "natural forwarding costs". This means that if a destination is reachable via a policy-conform path, traffic forwarding is implemented via a *cost-optimal path*. The cost-optimal paths are determined by the distance metric the protocol uses. As specified for iBGP, cf. section 2.3, they remain after applying the cost-relevant steps of the protocol's path selection process to all available paths. Tie-breaker rules are cost-irrelevant. If a router does not comply with expressiveness, it routes *cost-suboptimal*, cf. figure 1.a). Cost-suboptimal routing decisions lead to unnecessary costs and may even foil traffic transition arrangements like – in case of BGP – defined by the MED.

3.1.2 Global Consistency

Global consistency is an essential property for hop-by-hop based routing protocols. It states that the local routing decisions routers make are free of conflicts: If traffic is forwarded by a hop to another to reach a destination via a certain route, the next-hop has to continue the intended route. If a next-hop on an intended route redirects traffic onto another route or to another destination, *inconsistency* is present, cf. figure 1.b). Local policies may be overridden and – in special circumstances [9] – forwarding loops may be induced.

3.1.3 Robustness

Robustness means that the local processes the protocol defines always lead to a stable state. The routing must converge. The state the configuration finally enters must be predictable in advance if the initial state and configuration are known. The routing processes must behave deterministic. If the routing may not converge, it is *vulnerable to divergence*. If the final state is not predictable, the routing is *vulnerable to non-determinism*, cf. figure 1.c). Divergence and non-determinism may cause paket re-orderings, losses, and complicate routing analyses.

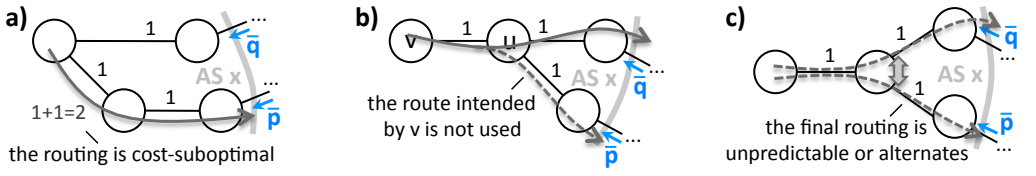


Figure 1 Conceptual illustration of cost-suboptimal routing, inconsistency, and non-determinism.

3.2 IBGP Anomalies and their Root Causes

Using the identifiers defined in section 2.3, anomalies in the iBGP routing can be traced back to six *root causes*, cf. [5]. They are introduced in what follows. Note that in contrast to the generic definitions given in section 3.1, the root causes are specific for the protocol.

3.2.1 Cost-suboptimal Routing

Obviously, it only makes sense to assess a routing decision as cost-suboptimal if the decision is permanently. This is the case, if the configuration has entered a stable state $s \in S$. In this state, every router $v \in V$ selects its best path by applying the selection process to the known paths. Consequently, cost-suboptimal routing is present in state s, if and only if

$$\exists v \in V : \lambda^v(\pi_s^v) \notin \Omega_c^v,$$

where π_s^v denotes the paths known in state s. Due to the design of BGP, this may have two reasons: Obviously, a router cannot select a cost-optimal path as best one if no such path is known. If such paths are known, they may be masked since no transitive ordering relation is defined, cf. figure 5.a). Both root causes can formally be differed as

$$v \in V : (\pi_s^v \cap \Omega_c^v) = \emptyset \quad (1) \qquad \text{and} \qquad v \in V : (\pi_s^v \cap \Omega_c^v) \neq \emptyset \wedge \lambda^v(\pi_s^v) \notin \Omega_c^v \quad (2).$$

3.2.2 Inconsistency

Similar to cost-suboptimal routing, referring to inconsistency only makes sense if a configuration has entered a stable state $s \in S$. By assumption, traffic is correctly delivered once it has reached the BGP next-hop, cf. section 2.1. Inconsistent local views may thus only appear on the IGP route to the BGP next-hop: A router may take away the traffic from the intended route and redirects it to another BGP next-hop. Another delivery path is used. It holds that

$$\exists v \in V, u \in f^1_{v \to \dot{p}_v} \setminus \{v\} : \dot{p}_v \neq \dot{p}_u,$$

where $p_x = \lambda^x(\pi_s^x)$ denotes the best path of and π_s^x the paths known by router $x \in \{v, u\}$ in state s. If inconsistency is not a side-effect of a cost-suboptimal routing, it is caused by *dissenting local views*. Dissenting views of v and u may have two reasons: Firstly, they may appear due to a simple lack of routing information required for consistent routing decisions. Provided with adequate paths, u and v choose consistent delivery paths. The basic problem is illustrated in a simplified manner in figure 2.a). Secondly, inconsistent views may be a result of *dissenting path rankings*. The BGP paths are not equivalently preferred by v and u. Induced by cost-relevant local path attributes that behave non-isotonicly [13] on a forwarding route, additional routing information is not helpful, cf. figure 2.b). Both root causes can be formalized as

$$\begin{aligned} &\forall p_v, q_v \in \mathcal{P}^v, p_u, q_u \in \mathcal{P}^u \mid u \in f^1_{v \to \dot{p}_v}: \\ &\lambda^v(\{p_v, q_v\}) = p_v \Rightarrow \lambda^u(\{p_u, q_u\}) = p_u \ (3) \\ &\qquad \mid \dot{p}_v = \dot{p}_u, \bar{q}_v = \bar{q}_u \end{aligned} \quad \text{and} \quad \begin{aligned} &\exists p_v, q_v \in \mathcal{P}^v, p_u, q_u \in \mathcal{P}^u \mid u \in f^1_{v \to \dot{p}_v}: \\ &\lambda^v(\{p_v, q_v\}) = p_v \not\Rightarrow \lambda^u(\{p_u, q_u\}) = p_u \ (4). \\ &\qquad \mid \dot{p}_v = \dot{p}_u, \bar{q}_v = \bar{q}_u \end{aligned}$$

3.2.3 Divergence and Non-determinism

In contrast to cost-suboptimal routing and inconsistency, robustness problems do not affect single states. They refer to an unwanted behavior of transitions between states $s_x \in S$. If a configuration is vulnerable to divergence, a cycle may be traversed. Formally, it holds that

$$\exists s_0, s_x, s_y \in S : s_0 \dddot{\to} s_x \dddot{\to} s_y \dddot{\to} s_x \mid s_x \neq s_y,$$

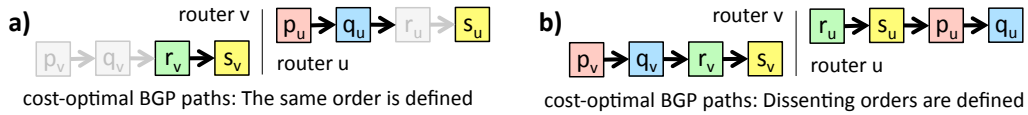


Figure 2 Inconsistency is either a result of an information lack (b) or dissenting path rankings (b).

where s_0 denotes an initial state of the configuration. If a configuration is vulnerable to non-determinism, state transitions from an initial state to at least two different stable states exist. If s_x and s_y denote stable states, it formally holds that

$$\exists s_0, s_x, s_y \in S : s_x \dddot{\leftarrow} s_0 \dddot{\rightarrow} s_y \mid s_x \neq s_y.$$

Both, vulnerability to divergence and non-determinism can be traced back to a conceptual defect of the state transition function. It holds that:

An iBGP extension or architecture scheme causes vulnerability to divergence, if and only if its update process definition does not define a partial order on the possible states arbitrary iBGP configuration may enter. [5] (5)

An iBGP extension or architecture scheme causes vulnerability to non-determinism, if and only if the partial order does not define a least element on the states an arbitrary configuration may enter. [5] (6)

The root causes for robustness problems are shown in figure 3.a) and .b). Figure 3.c) illustrates the basic idea behind robust state transitions.

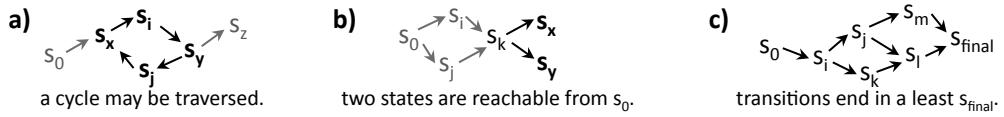


Figure 3 Divergent (a), non-deterministic (b), and robust (partial order, least state) (c) transitions.

4 Studies for Common IBGP Information Exchange Schemes

In literature, correctness problems in the iBGP routing are usually associated with information reduction schemes, cf. section 1.2. Anomalies in fully meshed iBGP configurations were never demonstrated. However, neither the correctness of full-mesh iBGP was ever entirely proven nor systematically studied to which root causes the information reduction schemes are prone to. Evaluating these aspects is tackled in what follows.

4.1 Full-meshed iBGP

Full-meshed iBGP, the original AS-internal BGP scheme specified in the protocol standard [12], realizes a logical full-mesh for the information exchange. As shown in [3, 5], the state transition process based on such a full-mesh can be mapped to a partial order relation with a least element. This property excludes the root causes (5) and (6) for arbitrary configurations; full-mesh iBGP always behave robust. Whether the remaining root causes can appear is unclear.

4.1.1 Cost-suboptimal Routing

As explained in section 3.2.1, cost-suboptimal routing decisions may have two different causes: Either no cost-optimal path is known (1) or sub-optimal paths mask the cost-optimal ones

(2). In a fully meshed configuration, the first root cause may never appear while the second one may appear. Both properties are proven in what follows.

Firstly, we show that if a configuration is fully meshed, every router knows a cost-optimal path. Let $v \in V$ be a router of a configuration that has entered a stable state. By assumption, it knows none of its cost-optimal paths. Furthermore, let $p \in \Omega_c^v$ be an arbitrary cost-optimal path of v. Let $u \in V$ denote the BGP next-hop where the underlying BGP path $\bar{p}$ is local or external. It knows a signaling path $\bar{p}(u)$ corresponding to the same BGP path $\bar{p}$. Both paths p and $\bar{p}(u)$ specify equal global path attributes. As shown in figure 4, three cases are possible: If $v = u$, cf. figure 4.a), it holds that $p = \bar{p}(u)$. A cost-optimal path is known. If $v \neq u$, u may have chosen path $q \in \mathcal{P}^u \,|\, q \neq \bar{p}(u)$ as its best path, cf. figure 4.b). Compared to $\bar{p}(u)$ and p, this path specifies global attributes of equal preference. Otherwise, as it can be verified easily, u would either not choose q as its best path or p would not be cost-optimal on v. Since $\bar{p}(u)$ is local or external on u, q must be local or external on u, too. If this would not be the case, u would not select q as its best path. Consequently, the signaling path $\bar{q}(u, v)$ permitted and known at v representing $\bar{q}$ specifies u as BGP next-hop. It comes along with the same IGP distance like path p. All in all, $\bar{q}(u, v)$ and p specify cost-relevant path attributes of equal preference. It holds that $\bar{q}(u, v) \in \Omega_c^v$. As this path is known at v, v knows a cost-optimal path. Finally, it may be that $u \neq v$ has selected $\bar{p}(u)$ as its best path. Due to the fact that a full-mesh is applied, this path is expanded to v and thus known. In all three cases, at least one cost-optimal path is known by v. This is a contradiction to the assumptions.

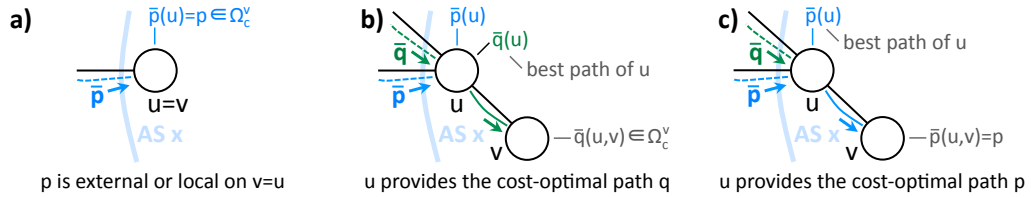


Figure 4 Realizing iBGP via a full-mesh, each router $v \in V$ knows at least one cost-optimal path.

Path maskings, root cause (2), may appear since BGP does not define a transitive ordering relation on arbitrary paths. The basic problem is shown in figure 5.a): Knowing three with respect to the first four attributes equally preferred paths p, q, and r for two peer-AS groups x and y, MEDs are evaluated in step c). For those paths q and r specifying the same peer-AS group y, the MEDs are compared. If different MEDs are specified, the path with the worse MED, here r, is discarded. If path q is now beaten by p in the remaining steps of the selection process, knowledge on q may be essential for a cost-optimal routing decision: If q is unknown, p may lose the decision process against r due to attributes evaluated after step c), the MED. The cost-suboptimal path r can mask path p if q is unknown, even if p is cost-optimal.

As shown in figure 5.b), a situation where a cost-optimal path is masked may also appear if an iBGP configuration is fully meshed. Router u knows all three available exit points. In the decision process, $\bar{q}(u)$ beats $\bar{r}(v, u)$ in step c) (lower MED), and $\bar{p}(u)$ beats $\bar{q}(u)$ in step f) (lower BGP ID). Consequently, v is not aware of $\bar{q}(u, v)$. The known cost-optimal path $\bar{p}(u, v)$ is masked by $\bar{r}(v)$, cf. figure 5.a). For all that is known, this possibility was unknown so far.

4.1.2 Inconsistency

As explained in section 3.2.1, inconsistent routing decisions can have two different root causes: Missing information on relevant paths (3) and dissenting path rankings (4). Using an iBGP full-mesh, a lack of information that leads to inconsistent local views, root cause (3), cannot

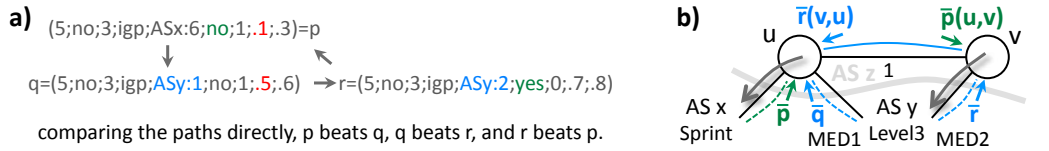


Figure 5 The BGP path selection process does not define a transitive ordering relation (a). This may cause path maskings even in fully meshed iBGP configurations (b).

appear. As depicted in figure 6, the full-mesh structure ensures that v and u learn the same delivery paths from all other routers $w \in V \setminus \{v, u\}$. Since v forwards traffic system-internally, its best path is internally learned. It does not advertise any path. Router u has either chosen an internally learned path as best path and advertises no path or else has chosen a local or external path which is advertised to v. All in all, the information basis v and u have only differ in externally learned and locally configured non-best paths.

By assumption, we know that if we take only the common information basis into account, v and u select a representation of the same BGP path. This is guaranteed by the edge condition defined by equation (3). The routing is consistent. As local paths are preferred in step 1) of the selection process, a router's non-best local path certainly has a lower Local Preference than its best path. Consequently, if such a path is known on v or u, it has no effect on the routers' routing decisions. External paths are preferred in step d). Thus, if a non-best external path is known, the best path specifies better global path attributes: A higher Local Preference (LP), equal LP and shorter AS-path Length (ASpL), equal LP, ASpL, and lower Origin Code (OC), or equal LP, ASpL, OC, and same peer-AS group with lower MED. Again, in all possible cases externally learned paths have no effect on the routing decision. Consequently, even if such paths are known, the same routing decision is made as if only the common information base would be known. Root cause (3) can never occur if full-meshed iBGP is used.

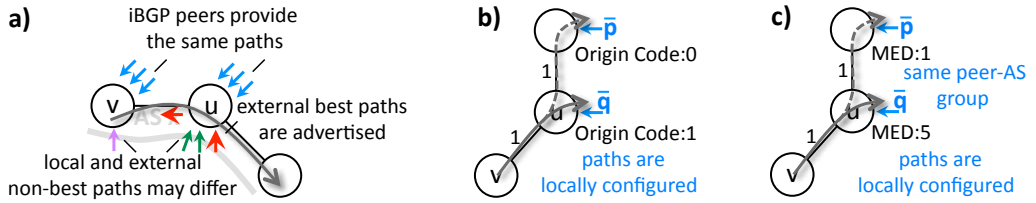


Figure 6 Using a full-mesh, the routing information of v and u differ only by local and external non-best paths (a). Despite of the full-mesh, inconsistency due to root cause (4) may appear (b, c).

Next, we check whether full-mesh iBGP is prone to root cause (4). For that purpose, we analyze the decision process and try to identify paths satisfying the equation. For paths that match the equation, we sketch an exemplary configuration. If equation 4 is satisfied, it holds that $\dot{p}_v = \dot{p} = \dot{p}_u$, $\bar{q}_v = \bar{q} = \bar{q}_u$, and that router v prefers p_v to q_v. Based on these assumptions and the fact that v forwards traffic to a BGP next-hop "behind" u, all possible preferences for the representations permitted at v and u are shown in the RC4-tree, cf. figure 7. Equation (4) is satisfied, if v prefers p_v, while u prefers q_u. Note that $\dot{p}_v = \dot{p} = \dot{p}_u \Rightarrow \bar{p}_v = \bar{p} = \bar{p}_u$.

Local Preference If path p_v is preferred to q_v by v due to a higher Local Preference, path p_u is also preferred over q_u by u. This is the case, since the Local Preference is a global path attribute. It is the same for all representations of a BGP path. If both underlying BGP paths specify the same Local Preference, the subsequent attributes define the path preferences of v and u. Following steps of the decision process are of interest.

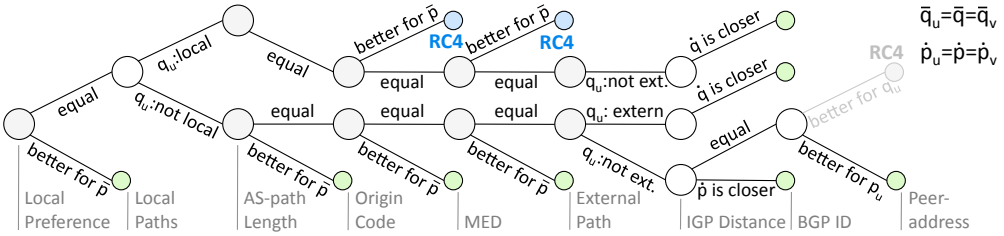


Figure 7 Possible path preferences under the assumptions given by root cause (4). Blue leaves specifying preferences satisfying root cause (4) (RC4), green ones specify a conflict.

Local Paths If step 1) is relevant, neither v nor u prefers any of its paths yet. By assumption, $\bar{p}$ is neither local on v nor on u. Since v does not prefer q_v, $\bar{q}$ is not local on v, too. The cases that $\bar{q}$ is local on u (q_u: local) and that it is not local on u (q_u: not local) remain.

AS-path Length If $\bar{q}$ is local on u, it specifies an AS-path Length of zero. As v prefers path p_v, the AS-path Length of $\bar{p}$ must be zero, too. Following path attributes define the final preferences. This is also the case if $\bar{q}$ is not local on u, while both BGP paths specify the same AS-path Length. If $\bar{p}$ specifies a shorter AS-path, v prefers p_v and u prefers p_u.

Origin Code As p_v is preferred by v, path $\bar{p}$ specifies the same or a better Origin Code than $\bar{q}$. In the former case, independent of whether $\bar{q}$ is local on u or not, v has no preference yet. Following attributes are crucial. In the latter case, v prefers p_u. If $\bar{q}$ is not local on u, u also prefers p_u, equation (4) is not satisfied. If $\bar{q}$ is locally configured on u, u prefers path q_u. This satisfies equation (4). An example that realizes this case is illustrated in figure 6.b).

MED Since p_v is preferred by v, $\bar{p}$ specifies either the same (or an incomparable) or a better MED than $\bar{q}$. In the former case, the following attributes are crucial, independent of whether $\bar{q}$ is local on u or not. In the latter case, p_u is preferred on u if q_u is not local on u. The root cause is not matched. If q_u is local on u, path q_u is already preferred by u. Root cause (4) is satisfied. However, this scenario seems unusual: Since both paths specify an empty AS-path, both paths are locally configured in the AS. For local paths, the MED is usually not specified. A default value is evaluated instead. Nevertheless, a conceivable example is shown figure 6.c).

External Paths According to local paths, no path is externally learned on v and $\bar{q}$ may be external on u. If $\bar{q}$ is local on u, it cannot be external on u, too. If $\bar{q}$ is not local on u, it may be external on u or not. Three cases that may satisfy equation (4) remain, cf. figure 7.

IGP Distance As we assume that IGP distances increase strictly monotonically, the distance from v to u is smaller than the distance to any BGP next-hop behind u. Consequently, if $\bar{q}$ is local or external on u while the preceding attributes did not define a preference on v yet, v prefers path q_v. This is a contradiction to the assumptions. It remains the case that $\bar{q}$ is neither local nor external on u. If the BGP next-hop given by $\dot{p}$ is closer to v than the one given by $\dot{q}$, this also holds for u. Equation (4) may only hold if both IGP distances are equal.

BGP ID In the remaining open case, p_v and q_v as well as p_u and q_u are equally preferred by v and u, respectively. They are internally learned on v and u. Because every router advertises only its best path and $\bar{p} \neq \bar{q}$, both p_v and q_v as well as p_u and q_u specify different BGP IDs. As v prefers p_v, p_v specifies a lower BGP ID than q_v. If path p_u also specifies a lower BGP ID than q_u, consistent rankings are defined. If p_u specifies a worse ID than q_u, root cause (4) is matched. However, in a full-mesh, this case cannot appear in practice: Since p_v and p_u as well as q_v and q_u are advertised by the same router, p_u specifies the same BGP ID like p_u and q_u like q_u. Consequently, p_u has a lower BGP ID than q_u.

All in all, full-meshed iBGP turned out to be prone to expressiveness and global consistency problems. The correctness problems that may appear are summarized in table 1.

Expressiveness Problems		Global Consistency Problems		Robustness Problems	
No known optimal path (1)	Path maskings (2)	Lack of relevant paths (3)	Inconsistent path rankings (4)	Cyclic state transitions (5)	No least state exists (6)
impossible	possible	impossible	possible	impossible	

Table 1 Possible root causes for iBGP anomalies in fully-meshed iBGP configurations.

4.2 BGP Route Reflection AS Confederations

Since Route Reflectors and routers within a member-AS are fully meshed, Route Reflection and AS Confederations are prone to the same problems like full-mesh iBGP. However, as both concepts realize an information reduction compared to a full-mesh, further causes may appear.

Anomalies caused by a lack of information (root cause (1) and (3)) as well as robustness problems are well-known and well-studied phenomenas if Route Reflection or a Confederation is applied. For that reason and due to the space limitation, we confine ourselves to sketch only the conceptual problems for these problem classes.

A cost-suboptimal routing decision due to a lack of cost-optimal paths, root cause (1), is very easy to induce. It appears if the Route Reflectors or the member-AS Border Routers do not reflect a router's cost-optimal paths. This behavior occurs if none of the relevant paths is the best one from the reflectors' points of view. A simple example for such a configuration is shown in figure 8.a). An example for AS Confederations can be designed analogously. Root cause (3), inconsistent routing decisions due to a lack of relevant information, can be induced in a similar way. Inconsistent routing decisions are made, if router u on the IGP forwarding route to the BGP next-hop knows a better path $\bar{q}$ that is not reflected. Again, this is the case if $\dot{q}$ is not the best delivery path from the reflectors' points of view. An example configuration is depicted in figure 8.b). A comparable configuration can be designed for Confederations, too. The fact that information reduction may cause cyclic state transitions and avoid that a least state is defined, root cause (5) and (6), is thoroughly studied. Readers who are interested in the details may be referred to [9], for example.

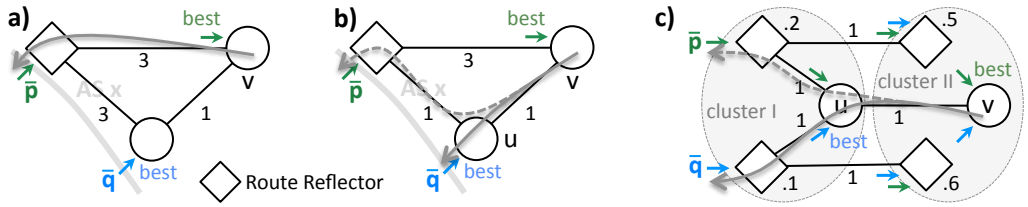


Figure 8 Classical examples for cost-suboptimal routing due to root cause (1) (a), inconsistency due to root cause (3) (b), and dissenting rankings due to the BGP ID (c). Example (b) is taken from [9].

Finally, we take a quick look on inconsistency due to dissenting path rankings, root cause (4), in ASs using Route Reflection and AS Confederations. An important difference between full-mesh iBGP and schemes which reflect paths is that representations of the same delivery path on different routers may specify different BGP IDs. As a result of this property, the third possible path preference that matches root cause (4), cf. the RC4-tree in figure 7, may occur. An example for such a configuration is shown in figure 8.c): Routers v and u both learn two cost-optimal paths $\dot{p}$ and $\dot{q}$ from two different routers. Due to the BGP ID, u prefers $\dot{q}$, while v prefers $\dot{p}$. Note that this case is of high relevance in practice. Network Operators in large ASs usually use several redundant reflectors per cluster to avoid single points of failure. A similar configuration can also be designed for AS Confederations. The root causes for iBGP anomalies that may appear if scalability techniques are used are summarized in table 2.

Expressiveness Problems		Global Consistency Problems		Robustness Problems	
No known optimal path (1)	Path maskings (2)	Lack of relevant paths (3)	Inconsistent path rankings (4)	Cyclic state transitions (5)	No least state exists (6)
trivial	possible	trivial	possible	possible	

Table 2 Possible root causes for anomalies in ASs using Route Reflection or AS Confederations.

5 Workarounds and Solutions

Understanding the protocol and configuration properties that induce iBGP anomalies, possible countermeasures can be discussed. We differ between *solutions* and *workarounds*: Solutions correct conceptual defects but require protocol changes. Workarounds are quickly deployable patches that avoid unwanted behavior in production systems. Due to the space limitation, we only sketch the basic ideas.

Unknown cost-optimal paths, root cause (1), appear if a router's local view is not matched by any of its BGP peers' best path selection. Information on the cost-optimal paths is not forwarded. To avoid this problem, reflectors could advertise paths matching to the receivers' point of view – independent of their own best path decision. A scheme realizing this concept is the *iBGP Route Server architecture*, cf. [4]. Using common information reduction schemes, problems are avoided if every cluster shares a common local view. To reach this, the distances of the links must ensure that cluster-internal BGP next-hops are always closer than -external ones. However, this workaround is a serious limitation for the IGP routing. Path maskings, root cause (2), are avoided if the selection process defines a transitive ordering relation on arbitrary paths. This can be reached if the MED is not evaluated at all or across all peer-AS, cf. [11]. Alternatively, it can be ensured that every router knows an AS-cost-optimal path for each peer-AS group [4]. However, this is not trivial in arbitrary configurations in general.

Consistency problems due to a lack of relevant routing information, root cause (3), are avoided if routers on a future forwarding route have a comparable information basis for their routing decisions. Improvements can be achieved here if additional cluster-internal BGP sessions are kept. However, this workaround counteracts with the intended information reduction and – which is the greater problem – can be the source of inconsistent routing, cf. [9]. Dissenting rankings, root cause (4), are avoided if all routers on a future forwarding route prefer the same delivery paths. However, to reach this, modifications on the selection process are necessary. Using a full-mesh, operators can achieve consistent rankings by means of a simple workaround, given by two design rules: Firstly, paths for a destination originated within the own AS must specify the same Origin Code. Secondly, MEDs must not be used for local paths. Controlled by the operator, both properties can be realized easily in ASs.

Robustness problems can be addressed in various different ways. Redesigning the protocol and the information exchange scheme, it is possible to ensure that an optimal state always exists and that this state is certainly entered, cf. [4]. Making use of common iBGP schemes, cyclic dependencies between the cost-optimal paths of routers in different clusters can be studiously avoided, cf. [11]. This can be reached by design rules comparable to those specified above. However, also similar problems are induced. Other concepts are based on modified path preferences [15], the exchange of additional information [16], and cycle detection [10].

6 Conclusion and Future Work

In this paper, we systematically analyzed the possibility of anomalies when using the standardized iBGP routing information exchange schemes. The most interesting results were obtained

for full-mesh iBGP: While we could prove that full-mesh iBGP is resistant to problems caused by an information lack, we could also reveal that path maskings and inconsistent rankings may appear. This observation clearly falsifies that iBGP anomalies are only a consequence of an information reduction, a position many studies convey. For BGP Route Reflection and AS Confederations, we could give a systematic overview on the for the great part known effects.

Understanding that and why full-mesh iBGP is clearly more resistant to anomalies seems to be an important increase of knowledge. Drafts for new iBGP extensions come along with similar conceptual defects like Route Reflection. Knowledge why certain iBGP schemes are prone or resistant to different root causes could help to avoid that new correctness problems are caused by the final standards. Besides this, the knowledge could also help to improve the existing schemes in the long term. In addition to protocol designers, the results gained here are also of interest for Network Operators. Knowing why certain root causes appear helps to develop and deploy adequate workarounds like network design rules for the existing schemes. Moreover, it becomes easier to evaluate iBGP scheme alternatives and identify the right one for the own AS. Even if we sketched first ideas in section 5, addressing anomalies so that the solution is usable in production systems is an important aspect for future research.

References

1 A. Basu, C. Luke Ong, A. Rasala, F. B. Shepherd, and G. Wilfong. Route Oscillations in I-BGP with Route Reflection. In *SIGCOMM '02*, pages 235–247. ACM Press, 2002.

2 T. Bates, E. Chen, and R. Chandra. BGP Route Reflection - An Alternative to Full Mesh IBGP. April 2006. RFC 4456.

3 U. Bornhauser and P. Martini. A Divergence Analysis in Autonomous Systems using Full-mesh iBGP. In *CNSR 2008*, pages 600–608. IEEE Computer Society, May 2008.

4 U. Bornhauser, P. Martini, and M. Horneffer. An Inherently Anomaly-free iBGP Architecture. In *IEEE LCN 2009*, pages 145–152. IEEE Computer Society, October 2009.

5 U. Bornhauser, P. Martini, and M. Horneffer. Root Causes for iBGP Routing Anomalies. In *IEEE LCN 2010*. IEEE Computer Society, October 2010.

6 Cisco Systems. Field Notice: Endless BGP Convergence Problem in Cisco IOS Software Releases. Technical report, October 2000.

7 N. Feamster and H. Balakrishnan. Correctness Properties for Internet Routing. *Allerton Conference 2005*, September 2005.

8 T. Griffin, A. D. Jaggard, and V. Ramachandran. Design Principles of Policy Languages for Path Vector Protocols. In *SIGCOMM '03*, pages 61–72, 2003.

9 T. Griffin and G. Wilfong. On the Correctness of IBGP Configuration. *SIGCOMM Comput. Commun. Rev.*, 32(4):17–29, August 2002.

10 T. Klockar and L. Carr-Motyckova. Preventing Oscillations in Route Reflector-based I-BGP. In *Proceedings of ICCCN 2004*, pages 53–58, Chicago, IL, October 2004.

11 D. McPherson, V. Gill, D. Walton, and A. Retana. Border Gateway Protocol (BGP) Persistent Route Oscillation Condition. August 2002. RFC 3345.

12 Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4. January 2006. RFC 4271.

13 J. L. Sobrinho. An Algebraic Theory of Dynamic Network Routing. *IEEE/ACM Trans. Netw.*, 13(5):1160–1173, 2005.

14 P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. August 2007. RFC 5065.

15 M. Vutukuru, P. Valiant, S. Kopparty, and H. Balakrishnan. How to Construct a Correct and Scalable iBGP Configuration. In *IEEE INFOCOM*, Barcelona, Spain, April 2006.

16 D. Walton, A. Retana, E. Chen, and J. Scudder. BGP Persistent Route Oscillation Solutions. May 2010. Internet Draft.

A Reputation-Based Approach to Self-Adaptive Service Selection

Jan Sudeikat[1], Wolfgang Renz[1], Ante Vilenica[2], and Winfried Lamersdorf[2]

1 **Multimedia Systems Laboratory**
Hamburg University of Applied Sciences
Berliner Tor 7, 20099 Hamburg, Germany
[jan.sudeikat;wolfgang.renz]@haw-hamburg.de

2 **Distributed Systems and Information Systems**
Computer Science Department, University of Hamburg
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
[vilenica;lamersd]@informatik.uni-hamburg.de

Abstract

Service-orientation provides concepts and tools for flexible composition and management of large-scale distributed software applications. The automated run-time management of such loosely coupled software systems, however, poses still major challenges and is therefore an active research area, including the use of novel computing paradigms. In this context, the dynamic and adaptive selection of best possible service providers is an important task, which can be addressed by an appropriate middleware layer that allows considering different service quality aspects when managing the adaptive execution of distributed service workflows dynamically. In such an approach, service consumers are enabled to delegate the adaptive selection of service providers at run-time to the execution infrastructure. The selection criteria used are based on the *cost* of a service provision and the continuous, dynamic evaluation of *reputations* of providers, i.e. maintained track records of meeting the respective service commitments. This paper discusses the design and operating principle of such an automatic service selection middleware extension. Its ability to balance different quality criteria for service selection, such as service *cost* vs. the *reliability* of provision, is empirically evaluated based on a multi-agent platform approach.

1998 ACM Subject Classification I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence — Multiagent systems; D.2.11 [Software]: Software Architectures — Service-oriented architecture (SOA)

Keywords and phrases Service, Workflow, Multagent System, Self-Adaptivity

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.14

1 Introduction

Service-orientation is a successful paradigm for the construction of flexible, loosely coupled distributed software systems. This loose coupling can be exploited to enable flexible compositions, orchestrations, and choreographies of such services and, thus, forms a technical foundation for the construction of *self-adaptive* systems (e.g. see [10], p. 20). For such systems, their respective adaptivity allows them to operate in dynamic execution contexts without explicit human or explicitly pre-programmed intervention. Especially when the availability of service providers and the demand for service invocations are (dynamically) changing, the management of service executions, in particular their respective selection and composition, has to be *adapted* according to changes in the environment (see Section 4).

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 14–25

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The adaptive operation in dynamic execution contexts is also of economic interest when service providers and consumers interact in open markets. In such cases, enterprises may schedule the achievement of individual tasks, i.e. the execution of workflows and the selection of services, based on economical as well as on technical criteria. An example for such an application may be provided by a car manufacturer that requires well-specified parts for the completion of a specific manufacturing process. Here, the automated selection of suppliers — based on given criteria — could provide means for controlling lean production processes.

Enabling flexible executions of workflows in these markets of service providers requires the adaptive selection of providers. Self-managing and self-organizing design techniques are attractive for controlling service selections while taking into account different quality aspects (see Section 4). Self-adaptive self-managing approaches introduce software components that control adjustments of subordinate system elements (see e.g. [10]). Self-organizing approaches realize system-level coordination in decentralized system architectures (see e.g. [11]), where the coaction of system elements gives rise to adaptive system level features.

In this paper, we propose a flexible service selection mechanism and related middleware extension for the management of service-based applications that can be used to equip service-oriented software infrastructures with adaptive service selection strategies. This framework provides a software layer that can be supplemented to service-oriented execution infrastructures. It provides an architectonic bluprint for integrating both self-adaptive and self-organizing dynamics in service-oriented execution infrastructures. The management of individual processes, i.e. workflows of service invocations, is handled by *software agents*. These agents do not manage the complete application but only those partitions that are influential for the completion of a specific workflow. Therefore, this architecture allows for the *self-organization* of concurrent, parallel workflows that are executed in an open (and dynamically changing) market of service providers. In general, self-adaptivity and self-organization are typically understood as opposing concepts for the creation of adaptive software systems (e.g. see e.g. [10], p. 5). The relevance of both concepts for the development of complex distributed systems and the respective need for comprehensive development concepts has been identified before (e.g. see [2]). In this context, the architecture proposed here provides an environment which allows to examine the interplay of managing entities.

We present both (1) the architecture of this middleware extension and (2) a design approach for adaptive selection strategies. First, the corresponding software architecture allows for integrating different quality criteria to be considered during the run-time selection of services. A flexible, agent-based design allows for the integration of different selection strategies. It provides a generic blueprint for integrating both self-adaptive and self-organizing dynamics in service execution middlewares. Secondly, we discuss and exemplify the principled design of distributed adaptation strategies. The conception of adaptive control algorithms, based on integrating feedback loops in software systems, is an active research topic [2] and we discuss the use of a visual modeling techniques. More specifically, a *reputation-based* approach is proposed here in which past experiences can be used, in addition to the service cost, for estimating the reliability of potential service providers. Balancing influences of several of such aspects for dynamic service selections allows even more specific tuning of the economic benefits. First experiences of such an approach in a case study are reported as well.

The following section introduces the agent-based selection-framework and illustrates the operating principle of the reputation-based selection strategy. Then the paper presents and discusses first simulation results (see Section 3) to evaluate the adaptive management. Finally, related work on the self-management of service-based applications is outlined before the paper concludes and gives prospects for future work.

2 An Adaptive Service Selection Middleware

The delegation of service selections to an additional middlware layer allows to separate the adaptiveness of service invocations from the realizations of service providers and consumers. In the middleware layer proposed here, *agent technology* is used as a connector between the service-oriented application elements, which constitute the application infrastructure, and the use of coordination means, e.g. utility-based service selections. In such an approach, software agents interact with service-based workflow executions and provide services themselves. Since the adaptations are delegated to supportive agents, adaptive system level aspects can be supplemented to the service execution middleware. Using the example of reputation-based service selections (see Section 2.2), the run-time invocations of services are observed by the agent system in order to reason from the past experiences to the reliability of service providers. The coordination of the involved agents is built on top of a coordination infrastructure for the creation of *self-organizing multi-agent systems* (MAS) [14]. This infrastructure provides a generic reference model for the integration of decentralized coordination processes in MAS.

Here, a generic model of the selection problem from [8] is adopted in which *Service Agents* (SAs) are providers of services and *Service Market Agents* (SMAs) are responsible to manage the execution of workflow instances. These workflows are distinguished between *Instantiated Workflow* (IW) and *Partial Instantiated Workflow*. In an instantiated workflow, service providers are associated to service invocations and when one or more services are not associated the workflow is partially instantiated.

2.1 Architectural Concept: A Supportive Middleware Layer

The selection middleware automates the allocation of service providers. This supportive system allows to automate and encapsulate the management of service-based software systems. The system is built in different layers (see Figure 1) which separate the business logic of the system elements from the accomplishment of adaptive features. The topmost *Application Layer* contains the functional elements of the managed application. Here these are providable service realizations and the service consumption that is controlled by a *Process Execution Engine*. The underlying *Connector* layer contains software agents that interact with these elements. The *Jadex Processes*[1] framework is used to realize these interactions. A process engine controls the execution of processes in the *Business Process Modelling Notation* (BPMN). The identification of service providers is delegated to SMAs. The provision of services is managed by SAs. As new SMA and SA (types) can be added at runtime the architecture enables the dynamic provision of new service providers/consumers. The selection mechanism (see Section 2.2) is embedded in an underlying *Coordination Layer*. This layer follows the architectonic model from [14] and separates the participation of agents in decentralized coordination processes from the agent models. This separation facilitates the reconfigurability and changeability of processes.

The constituent elements are the *Endpoints* and *Media*. Endpoints contain and control the activities that are conceptually related to the coordination of agents. These are computational elements that are able to observe and influence the execution of software agents [17]. The media are infrastructure elements that connect endpoints. Media provide interaction mechanisms for the coordination of agents that are based on communication infrastructures [17] and/or shared agent spaces [21]. Using a generic publish/subscribe interface, these

[1] http://jadex-processes.informatik.uni-hamburg.de/

infrastructures are integrated as a distributed event-based system.

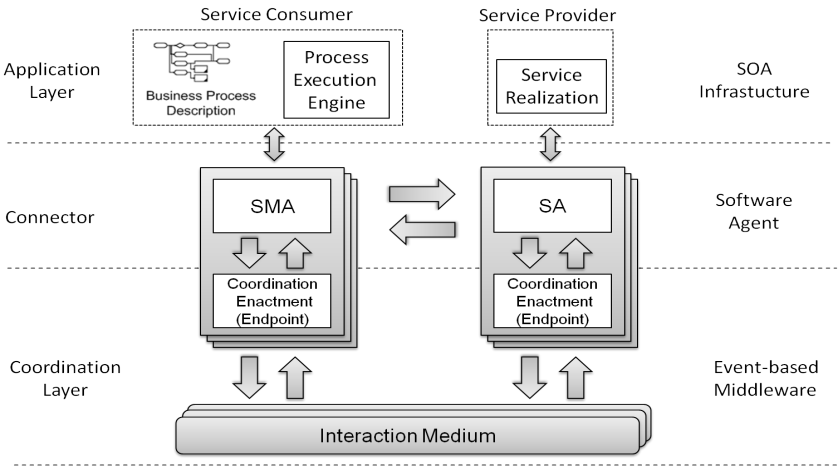


Figure 1 Service composition/selection architecture

Using this architecture, a reputation-based selection mechanism is realized (see the following section). Endpoints control the enactment of the coordination. In this case they control the selection and the update of the measured reputations of service providers. Two media connect the endpoints. One medium handles the *selection* of an appropriate service provider. Providers initially register at this medium and SMAs can inquire a service provider, decided by a given utility function (see Section 2.2.1). A second medium handles the reputations of service providers. This medium maintains the quality indicators of providers that are increased/decreased by the successful/unsuccessful completions of service. The configuration of the adaptive selections is be separated from the business logic, which is given in BPMN-based workflow descriptions. Each service provider and workflow is managed by an agent, e.g. to ensure that the commitments of individual providers do not overlap.

2.2 Adaptation Mechanism: The Dynamic Perspective

Within the Coordination layer (see Figure 1) adaptive, collaborative processes can be integrated into distributed systems. Here, the integration is exemplified by a process that allows to manage service invocations. We assume a setting where multiple providers are able to achieve specific tasks. The quality attributes, e.g. cost, processing time, reliability, etc., of the providers differ and thus the service consumers have to make economic selections. The consideration of different providers is not handled on the application level but is delegated to a supportive middleware.

For the description and configuration of decentralized processes, a dynamical modelling level has been developed that supplements agent-oriented design techniques with descriptions of dynamic aspects within MAS [18]. It makes use of *System Dynamics* [13] modelling concepts. Particularity the *Causal Loop Diagram* (CLD), a formalism for the illustration of system variables and their interdependencies, is extended for the description of the dynamics and coaction within MAS. Using this modelling stance, adaptive system behaviours can be pictured: Nodes in the graphs represent system variables, e.g. the number of agents that play a specific role and connection between these nodes denote influences and interdependencies. Influences describe additive or subtractive relations, e.g. when agents place or remove items

in/from a stock. Interdependencies describe causal relations, e.g. an increase of service requests in a SOA-based application consequently leads to an increase of service invocations.

2.2.1 Conception of the Dynamic System Behaviour

A pragmatic approach to the conception of a decentralized coordination processes is based on the comparison of the system behavior *as is* with the *intended*, i.e. coordinated and adaptive, behavior of the application [16]. The problematic, unintended behaviour is modeled first. Based on this *problematic dynamic*, a corresponding *solution dynamic*, i.e. an additional process fragment, is modelled, which improves the system behaviour.

The problem dynamic of the service allocation scenario is illustrated in Figure 2. A set of service providers (SAs) have committed to provide services in the given time-frame(s). Committed agents exhibit the role *Allocated Service* as they are locally aware of their obligations. One agent instance can commit to the participation in several IW, if the intervals of each service provision do not overlap. Before invocating allocated service instances, service providers may decide to change their commitment. In Figure 2, two possible reasons are denoted. First, agents may decide by themselves (*autonomous reallocation*) that a simultaneous service commitment is more profitable. Secondly, internal failures in the provider component (*execution error*) enforce that providers indicate their inability to satisfy commitments. The impact of these influences is characterized by an interaction rate (*leave IW*). Increases in this rate reduce the overall number of allocations and increases the number of agents that are *falsely committed*, i.e. agents that fail to achieve their commitments. When agents are in this state, the associated IWs are not executable and the number of *executable* IWs is reduced. The system exhibits a balancing feedback loop that limits the number of allocations (α), due to the perturbations, which influence the rate of agents that want to cancel their commitment (leave IW). The objective of the self-adaptive management of workflows is to counteract this feedback, i.e. to re-establish stable system configurations where all IWs are executable.

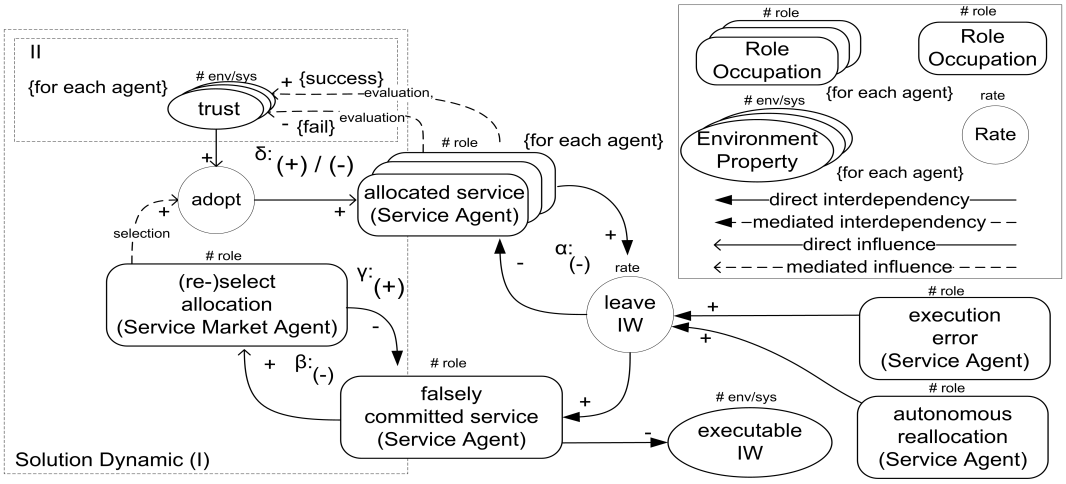


Figure 2 Macroscopic model: Dynamic service selections

The corresponding *solution dynamic* (see Figure 2, I) proposes the supplementation of the application with an additional agent role and related agent-interdependencies to establish additional feedback loops. Service Agents are enabled to communicate their intention to cancel a commitment to the corresponding SMA. This agent then initiates a *selection* of a

service provider to replace the missing commitments. When a replacement has been found the false commitment is revoked. The revoking manifests a second balancing feedback loop (β) that limits the number of agents that are falsely associated, i.e. that are deficient for the provision of the allocated service. As the selections also reinforce the number of allocations, a third, reinforcing feedback is manifested (γ) that reinforces allocations. This reinforcement alleviates the effects of the problematic feedback (α). This reinforcing feedback itself is controlled by the availability of falsely committed agents (β).

The establishment of the counterbalancing feedback γ is a requirement. The exhibited system dynamics is expected to show this feedback and the model neglects how this feedback is established. The approach followed here makes use of a utility function that takes into consideration not only the cost of the service invocation but also the *Reputation* of providers, i.e. the confidence that service providers can meet their commitments.[2] These reputations, called *trust*, are adjusted at run-time (see Figure 2, II). A set of agents (allocated service) participates in the workflow and for each agent such a value is maintained (*evaluation*). The corresponding values are increased when the service execution *succeeds* and are decreased when service providers *fail* to meet their obligations. The current trust value is considered in the service selection and thus influences the overall rate (*adopt*) of provider selection.

An alternative, decentralized design approach would be to equip Service Agents with the ability to negotiate their replacement with other agents. The causal structure of the application dynamics would not be affected by this adjustment, but the role of the negotiation initiator (*(re-)select allocation*) would be played by the Service Agent, not the Service Market Agent. An example for such a completely decentralized approach, embedded within a Coordination Layer, is given in [19], where failing machines in a production line are responsible to find replacements for their designated activities. In the following, the former approach is examined in system simulations.

2.2.2 The Selection Dynamics

The dynamic model in Figure 2 describes the macroscopic dynamics of systems that are equipped with the supportive middleware layer. This modelling level is particularly suited to describe self-organizing dynamics (e.g. see [18, 16]), and is here used to denote the operation of sets of self-adaptive managers (SMAs). For the explanation of concrete simulation experiments (see the following section) a detailed model of the dynamics is needed that does not only refer to macroscopic system variables, e.g. the overall numbers of agents that show specific behaviour, but that describes the concrete influences in system variables. This description level gives a more specific illustration of the relations of system variables and is obtained by refining macroscopic models. A detailed discussion of the different modelling levels for self-organizing systems can be found in [15].

The corresponding model (see Figure 3) describes how the computation of services' utilities influences the overall profit generated by the process executions. The variable *Profit (Process)* describes the economic benefit that results from the execution of workflows. The

[2] Utility functions offer the possibility to specify preferences among (potentially conflicting) objectives and can be mathematically expressed as

$$U(x) = \sum_{i=1}^{N} x_i \cdot w_i$$

where w_i expresses the weight of an objective x_i [5].

major (additive) contributions are the profits, which are generated by the constituent service invocations (*Profit (Service)*). The selection of the corresponding service providers is based on the computation of the utility (*Available Utility*). This value estimates the most appropriate service provider and thus the provider with the maximal utility value is selected. Several factors influence this estimation and in Figure 3 two are exemplary denoted. These are the current *Reputation* and *Cost* values of services. In addition, two disturbance variables, which are externally set to fixed values in the system simulations, influence the generation of profit. First, the failing of a service (*Service failure*) is associated with a penalty cost that subtracts from the overall profit (β). Failures also minimize the trust values and therefore have an indirect influence on the service selection. A second disturbance is the blackout of services (*Black out probability*). When services are temporarily not available, e.g. due to internal errors or unreliable network connections, this affects the current service selection and also minimizes the current reputation value. The successful provisioning of services is recorded by the corresponding medium and increases the reputation values that are associated to specific providers. Therefore, the variables are steadily increased as denoted by positive (+), i.e. reinforcing, feedback loop [13]. On the other hand, the external disturbances, i.e. failures and blackouts, reduce the individual reputations.

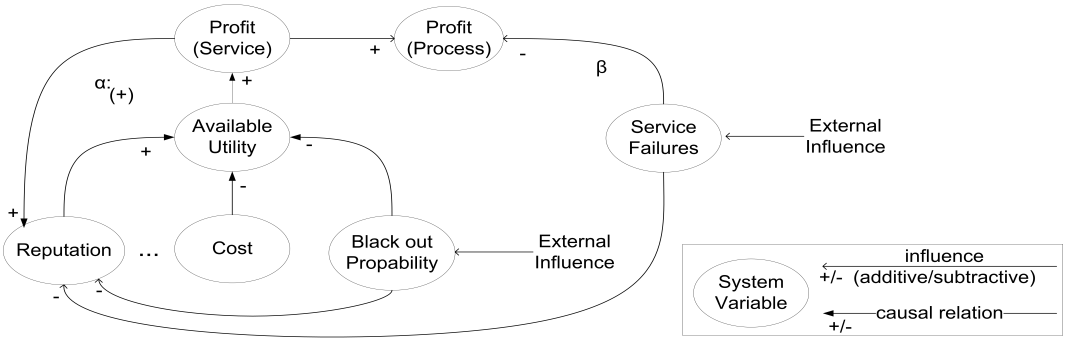


Figure 3 The dynamics of process costs, as shown by the implemented simulation model (see the following section)

3 System Evaluation

3.1 Operating Principle

That the system is self-adaptive, i.e. that services are adaptively selected, is validated by system simulations. A sample simulation run, with a fixed reputation value[3] is shown in Figure 4. The measurements were obtained from a single SMA that repeatedly selects one service. The SMA has three options: an *expensive*, a *moderate* and a *low-cost* service provider. These agents can perform the same service type but differ in their cost and the probability of a service failure, i.e. a blackout, which is detected by the infrastructure applying timeouts. The three service providers were set up with following cost characteristics: expensive (1200), moderate (1000), low-cost (800) and with an initial value of 30 for reputation. For the ease of evaluation the SMAs service selection function, i.e. the utilitfy function, only considered the costs and reputation of respective SAs. Quality of service parameters where not considered

[3] set to: 0.4

but could easily be integrated by extending the arguments of the service selection function with additional parameters.

The history of service invocations shows that the invocations (I) and the recorded reputation values (II) mutually influence each other. These influences validate a set of the relations in the dynamic application model (see Figure 3). In phases where one service type is invoked repeatedly and performs as expected, such as between the time steps 40 to 55,[4] the corresponding reputation value is amplified. This is expressed as the reinforcing feedback (α) in Figure 3. The systems quality also decides the likelihood that service invocations will fail using following function to compute the time between two blackouts: *t = -log(1-x)*ServiceCharacteristic*. Thereby, the three service providers had following *ServiceCharacteristics*: expensive (0.1), moderate (0.5) and low-cost (0.999). Moreover, service failures are drastic events that lead to sudden decrease in the reputations (negative contribution to the Reputation node in Figure 3). Upon these failures the provider with the highest utility is selected.

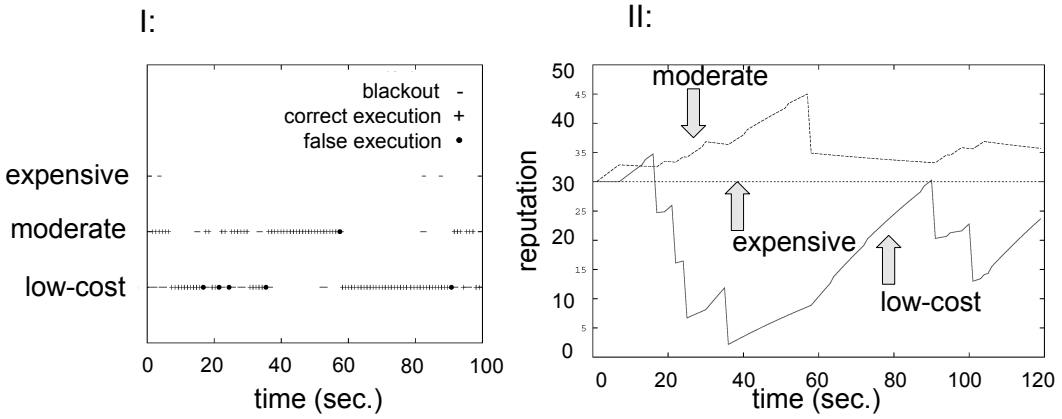


Figure 4 Sample simulation run: Service invocations (I) and the corresponding reputations (II)

3.2 The Impact of Considering Reputations

The adjustment strategy proposed here makes use of past experiences. The weight of the gained reputation is a static parameter that is set initially. In Figure 5 (I), it is shown how this parameter affects the fractions of agents that participate in workflows. Simulations are carried out for two minutes and the SMAs repeatedly select one service provider. For low values (x axis) the historic reputation that services gain have limited impact on the selection. Instead the service cost is the dominant factor and low-cost service providers are preferred. When higher weight values are used, more reliable, i.e. more expensive, service providers are used.

The performance of the workflow executions is indicated by two measurements that are shown in Figure 5 (II). Simulations are carried out for a fixed duration and higher weights for the reputation values lead to a slight increase of the total number of services that can be completed in this time. This effect results from the increased utilization of more reliable, and thus expensive, service providers. Due to their reliability these fail more seldom and re-invocations of services are avoided.

[4] measured in seconds, see Figure 3

The weight of the reputation significantly affects the overall profit that is generated. For small wights (< approx. 0.4), the selection is mainly influenced by the cost of invocations and therefore the fraction of selected services that fail are comparatively high. Failures cost penalties and enforce re-selections of providers. Thus the overall profit that can be generated is limited. For high weights (> approx. 0.8), the selection is biased toward expensive services. Due to their cost, it is not of economic interest to use these providers extensively and the overall profit is reduced. A weight about approx. 0.6 maximizes the profit as the use of expensive/low-cost service providers is balanced and the majority of invocations refer to moderately priced providers. In the simulations, the cost of expensive providers and the penalties of failures are set to values that only allow for a small net earnings area that is reached by using mid-level reputation weights.

Qualitatively, the simulation results indicate that the selection process conforms to the systemic model of the selection dynamics that are given in Figure 3. Service failures, an external influence, affect the generated profit (β). Therefore, unreliable providers have to be avoided. Since failures reduce the reputation, SMAs are repelled from service providers that have shown to be unreliable. This explains that the use of reputation mechanisms can increase the systems performance. On the other hand, the steady increase of reputations (α) leads to a problematic use of expensive providers for high-level reputation weights. Their cost limits the total profit. Thus the balancing of the former influence (external factors) and the latter feedback is required. This balancing depends on the system parametrization and for the simulation setting studied here, moderate weights of the reputation in the utility function are appropriate.

In conclusion it can be stated that the simulation results prove the applicability of the proposed framework to enable self-adaptive service compositions by using reputation as a criterion (among others like costs) for selecting service partners. At the same time, it has to be stated that using this framework for automated service selection requires simulation effort in order to find appropriate settings for an environment that exhibits high dynamics. The mathematical analysis of the dynamics for inferring appropriate parametrizations is left for future work. Additionally, there is the need to provide more implementations of wide-spread negotiation protocols in order to increase the utilization of this framework as well as to study the impact of using reputation with these protocols. Nevertheless, this evaluation shows that using reputation, i.e. upon to a certain factor, to measure the quality of service partners increases the profit of service executions in contrast to settings that take only service costs into consideration.

4 Related Work: Adaptive Service Selections

The fact that agent-orientation can be used as a valuable enrichment for service-oriented architectures has been argued by several authors, e.g. in [12]. Here, an enhancement is realized by a layered architectural model. Software agents represent the consumers and providers of services and form the logical link between the application business logic and the adaptivity-related coordination, i.e. the selection.

Consequently, the automation of the orchestration and composition of services has been studied. Approaches for the adaptive management have to prepare for (possibly) large scale of service-oriented systems and for the handling of dynamic execution contexts. Due to these challenges, the use of novel computing paradigms, which focus on bringing about intended application level dynamics, in this application domain have been proposed. Examples are the use of self-adaptive architectures and self-organizing problem solving strategies. Self-

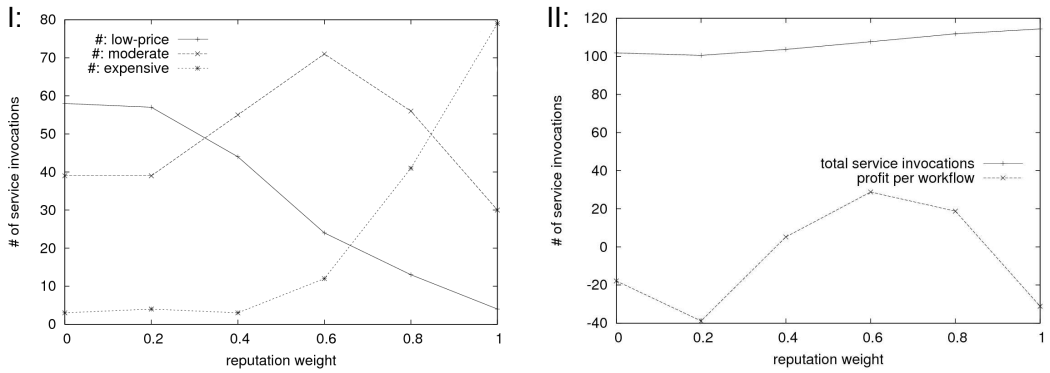


Figure 5 Averaged cumulative invocations of service providers plotted over the fixed reputation value (I), averaged, total sum of service invocations (II) and averaged profit generated plotted over fixed reputation values (II).

organization allows for decentralized management strategies that are particularly suited for large-scale application infrastructures. Examples include the transfer of nature inspired phenomena e.g. the chemical reactions [3] and the differentiation of individuals as found in biological systems [9].

Management of self-organization in service-based applications is discussed in [7] and a corresponding reference architecture is proposed. A *management overlay* is constructed by associating each element in the application with a managing software element. These elements follow the Observer/Controller (O/C) architecture that is developed by the Organic Computing research initiative (e.g. see [20]). The proposed architecture conforms to the framework proposed here as each element is associated with a computational element that control the local adaptations. However, O/C elements are comparatively complex entities that contain among others data analysis, prediction, and simulation components in order to estimate the system states that the system is approaching. The model presented here is comprehensive as it does not only contain the decision making elements but also abstractions of their interactions. Endpoints are comparatively lightweight as they only contain the logic that is required to impose the participation in the collaborative process, which steers the system operation. Simplification of the controlling system elements is enabled by the explicit modelling of the process (see Section 2.2) [16].

Furthermore, there is existing work that targets either the provision of infrastructures for the dynamic selection of services or the development of new negotiation/bidding strategies in open marked-based scenarios. On the one hand, Borrisov et al. [1] propose a framework that automates the generation of bids for the allocation of computing services in grid-based systems. Similar to the approach presented in this work, [1] incorporate a technique, i.e. machine learning, to deal with aspects related to reputation. At the same time the approach focuses on the domain of grid-systems and gives therefore only partial solutions to general problems related to adaptive service selections, e.g. the approach does not deal with blackout of services. On the other hand, Lewis et al. [6] focus on developing a new negotiation strategy that can cope with the challenges of dynamic, decentralized and service-based systems. This strategy is encapsulated within so called "evolutionary markets agents" that act on behalf of service providers and service consumers. Therefore, it would be of interest to extend the framework used within this work with this new bidding strategy of Lewis et al. in order to evaluate its advantages and disadvantages with respect to existing negotiation strategies.

5 Conclusions

This paper proposes a generic management architecture for the adaptive management of service-oriented applications based on multi-agent middleware. The corresponding framework uses agent technology to enable the self-adaptive operation of services. It provides an additional middleware layer that can extend infrastructures in order to provide for decentralized service selection management. The design of this framework is not biased towards specific management strategies, but is, on the contrary, highly configurable. This paper approaches the service selection problem via a *reputation-based strategy*, where past experience with service provision is used as a(n additional) selection criterion for service providers. The corresponding middleware layer is based on a programming approach for *self-organizing multiagent systems* [14]. Accordingly, the work reported here does not only concern the service infrastructures but also exemplifies how the decentralized, agent-based management can be used to supplement conventional software infrastructures. In such an approach, the corresponding MAS serves as a connector between application-level software artefacts and the participation of such artefacts in managing distributed processes with adaptive system properties.

Future work shall address the elaboration of management strategies and the validation of the management framework in more realistic scenarios. This includes studying non-functional aspects of the approach, e.g. scalability, and the mathematical analysis of the system dynamics. The framework presented here and the underlying programming model first demonstrate in principle the ability to supplement decentralized management processes into software systems. This however, requires additional policies, guidelines as well as heuristics that enable development teams to conceive the appropriate management approach for specific application scenarios in order to decide between, e.g. centralized, managing software entities, as addressed by autonomic computing systems (e.g. see [4]), and the use of decentralized coordination processes [19].

Acknowledgements The authors would like to thank *Deutsche Forschungsgemeinschaft* (DFG) for supporting this work through a joint research project on "Self-organization based on decentralized co-ordination in distributed systems" (SodekoVS) and, in addition, Claudia Di Napoli and Maurizio Giordano from C.N.R, Naples, Italy, for related discussions on service selection problems.

References

1 Nikolay Borissov, Dirk Neumann, and Christof Weinhardt. Automated bidding in computational markets: an application in market-based allocation of computing services. *Autonomous Agents and Multi-Agent Systems*, 21:115–142, 2010.

2 Yuriy Brun, Giovanna Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. *Software Engineering for Self-Adaptive Systems*, chapter Engineering Self-Adaptive Systems through Feedback Loops, pages 48–70. Springer-Verlag, Berlin, Heidelberg, 2009.

3 Claudia Di Napoli, Maurizio Giordano, Zsolt Németh, and Nicola Tonellotto. Using chemical reactions to model service composition. In *SOAR '10: Proceeding of the second international workshop on Self-organizing architectures*, pages 43–50. ACM, 2010.

4 Rajarshi Das Jeffrey, O. Kephart, Charles Lefurgy, Gerald Tesauro, David W. Levine, and Hoi Chan. Autonomic multi–agent management of power and performance in data centers.

In *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) – Industry and Applications Track*, pages 107–114, 2008.

5 Henry M. Levin and Patrick C. McEwan. *Cost-Effectiveness Analysis: Methods and Applications*. Sage Publications, 2. edition, 2000.

6 Peter R. Lewis, Paul Marrow, and Xin Yao. Resource allocation in decentralised computational systems: an evolutionary market-based approach. *Autonomous Agents and Multi-Agent Systems*, 21(2):143–171, 2010.

7 Lei Liu, Stefan Thanheiser, and Hartmut Schmeck. A reference architecture for self-organizing service-oriented computing. In *ARCS*, volume 4934 of *LNCS*, pages 205–219. Springer, 2008.

8 Claudia Di Napoli. *Software Agents to Enable Service Composition through Negotiation*, chapter 12, pages 275–296. Studies in Computational Intelligence. Springer, 2009.

9 F. Saffre, J. Halloy, M. Shackleton, and J. L. Deneubourg. Self-organized service orchestration through collective differentiation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(6):1237–1246, Dec. 2006.

10 Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Trans. Auton. Adapt. Syst.*, 4(2):1–42, 2009.

11 G. D. M. Serugendo, M. P. Gleizes, and A. Karageorgos. Self-organisation and emergence in MAS: An overview. In *Informatica*, volume 30, pages 45–54, 2006.

12 Munindar P. Singh and Michael N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons Ltd, 2005.

13 John D. Sterman. *Business Dynamics - Systems Thinking and Modeling for a Complex World*. McGraw–Hill, 2000.

14 Jan Sudeikat, Lars Braubach, Alexander Pokahr, Wolfgang Renz, and Winfried Lamersdorf. Systematically engineering self-organizing systems: The SodekoVS approach. *Electronic Communications of the EASST*, 17, 2009.

15 Jan Sudeikat and Wolfgang Renz. *Applications of Complex Adaptive Systems*, chapter Building Complex Adaptive Systems: On Engineering Self–Organizing Multi–Agent Systems, pages 229–256. IGI Global, 2008.

16 Jan Sudeikat and Wolfgang Renz. On the modeling, refinement and integration of decentralized agent coordination – a case study on dissemination processes in networks. In *Self-Organizing Architectures*, volume 6090/2010 of *LNCS*, pages 251–274. Springer, 2010.

17 Jan Sudeikat and Wolfgang Renz. Separating agent-logic and inter-agent coordination by activated modules: The decomas architecture. *Electronic Proceedings in Theoretical Computer Science*, 27:17–31, 2010. (Proceedings of the Workshop DCDP 2010).

18 Jan Sudeikat and Wolfgang Renz. Qualitative modeling of mas dynamics - using systemic modeling to examine the intended and unintended consequences of agent coaction. In Michael Luck and Jorge J. Gomez-Sanz, editors, *Agent-Oriented Software Engineering X*. Springer, 2011. (to be published).

19 Jan Sudeikat, Jan-Philipp Steghöfer, Hella Seebach, Wolfgang Renz, Thomas Preisler, Peter Salchow, and Wolfgang Reif. Design and simulation of a wave-like self-organization strategy for resource-flow systems. In *Proceedings of The Multi-Agent Logics, Languages, and Organisations Federated Workshops (MALLOW 2010)*, volume 627, 2010.

20 Stefan Thanheiser, Lei Liu, and Hartmut Schmeck. Towards collaborative coping with it complexity by combining service–oriented architectures and organic computing. *System and Information Science Notes*, 2(1):82–87, 2007.

21 Ante Vilenica, Jan Sudeikat, Winfried Lamersdorf, Wolfgang Renz, Lars Braubach, and Alexander Pokahr. Coordination in multi-agent systems: A declarative approach using coordination spaces. In *Proc. of EMCSR 2010 - Int. Work. From Agent Theory to Agent Implementation (AT2AI-7)*, pages 441–446. Austrian Society for Cybernetic Studies, 2010.

A Service-Oriented Operating System and an Application Development Infrastructure for Distributed Embedded Systems

Martin Lipphardt[1], Nils Glombitza[1], Jana Neumann[2], Christian Werner[1], and Stefan Fischer[1]

1 **Institute of Telematics, University of Lübeck, Germany**
2 **Institute of Information Systems, University of Lübeck, Germany**

Abstract

The paradigm of service-orientation promises a significant ease of use in creating and managing distributed software systems. A very important aspect here is that also application domain experts and stakeholders, who are not necessarily skilled in computer programming, get a chance to create, analyze, and adapt distributed applications. However, up to now, service-oriented architectures have been mainly discussed in the context of complex business applications. In this paper we will investigate how to transfer the benefits of a service-oriented architecture into the field of embedded systems, so that this technology gets accessible to a much wider range of users. As an example, we will demonstrate this scheme for sensor network applications. In order to address the problem of limited device resources we will introduce a minimal operating system for such devices. It organizes all pieces of code running on a sensor node in a service-oriented fashion and also features the relocation of code to a different node at runtime. We will demonstrate that it is possible to design a sensor network application from a set of already existing services in a highly modular way by employing already existing technologies and standards.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.26

1 Introduction

Pervasive computing can be found in a large variety of application scenarios. Hence, many different kinds of pervasive systems are realized by different classes of devices. Wireless sensor networks (WSN) represent one particular pervasive scenario, which is characterarized by the usage of resource constraint devices. A lot of effort was put into technical and algorithmical problems in WSN, yet implementing a WSN application is still a tedious task. Many parameters have to be considered like network density, traffic patterns and mobility models. Additionally, possible environmental influences must be preconceived. Therefore, besides profound programming abilities a lot of expert knowledge and understanding is needed to design a functioning and robust sensor network application. This holds especially if we consider the fact that actual users of sensor network technology are not usually skilled in the field of computer science. Application scenarios for WSNs can be found in many different fields stretching from military, science, logistics to health care, where each application has its specific requirements and characteristics.

As state of the art for WSN application development we observe a development cycle that can be subsumed to the graph given in Figure 1. A potential user of WSN technology describes the problems with requirements, e.g., monitoring phenomena or tracking objects. The user drafts the requirements and presents them to a WSN expert. Based on the requirements the WSN expert creates a model of the application scenario. The size and density of the network is determined. Mobility models are designed and the data rates are

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 26–37

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

estimated. With consideration of environmental influences the scenario is evaluated in a simulator. Based on the results, the WSN expert chooses different protocols and algorithms and finally implements the WSN application. After the successful deployment of the sensor network, the user receives data from the network and draws conclusions which may bring up new aspects that might affect the requirements to the WSN. Again the user presents his new requirements to the WSN expert.

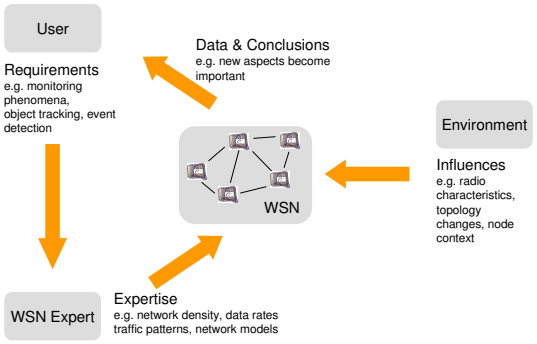


Figure 1 Current WSN application development cycle

Observing this situation we see parallels to the early days of software development in the late 1950s. In these days, a lot of expertise was needed to deploy even programs solving simple tasks. As a consequence the usage of information technology was reserved for the programmer himself. Since then a lot of effort was put in the development of new programming paradigms to enable actual target user groups to get access to new information technologies. Up to now, the main focus of sensor network research is on mastering resource constraints and achieving robustness in WSNs. Having overcome a lot of technical barriers, we have the possibility to make the wireless sensor network techology accessible for users. Our goal is to make the development of a distributed sensor network application easier allowing non-WSN experts to deploy a sensor network. With regard to the distributed nature of WSN applications and the complexity of interactions between network topology and environmental influences, we consider two aspects to be crucial for enabling users without WSN expertise to access WSN technology: abstractions of application development and runtime modification of deployed WSN applications.

An abstraction for application development makes the complexity of the sensor network application more transparent to the user. It must enable the user to develop different applications easily and in a minimum of time. Since a user with no WSN expertise cannot predict the behavior of the WSN or the WSN application, there might be a lot of trial and error during application development and deployment. Enabling the sensor nodes to change their software during runtime can ease the process towards a successfull deployment and is therefore inevitable.

In this work we introduce *Surfer OS*, a service-oriented minimal operating system for embedded devices that allows an integration and replacement of services during runtime. Additionally we present an infrastructure that offers a *Service Repository* and allows graph based service composition and runtime adaptation of WSN applications for *Surfer OS*.

This work is structured as follows: The next section will give an overview over current WSN operating systems, middlewares and frameworks with the focus on the development of distributed WSN applications. In Section 3 we will present our basic idea for WSN application design and introduce the interactions among the components of our infrastructure, namely *Surfer OS*, the *Service Repository* and the *Service Composition*. Section 4 gives a detailed description of *Surfer OS* and its implementation. Section 5 describes the *Service Repository* and the data stored with each service. In Section 6 we give details about the application development for *Surfer OS* with a graphical tool support. Furthermore, we show how the process of *Service Composition* can be integrated into enterprise applications as well as

business processes. In Section 7 we discuss the benefits and downsides of our presented operating system and infrastructure. We conclude the paper with Section 8 summarizing our contribution and giving an outlook on future work.

2 Related Work

The most widespread operating system used for sensor networks is TinyOS [6, 7]. Using a component based programming model, TinyOS is more than an operating system. It is a framework with a set of components that allows for building an application specific operating system for embedded devices. Each component resembles a functionality and exposes one or more interfaces. The fine grained component-based architecture allows the composition of applications before the deployment of the sensor network. This makes the application development for sensor networks more transparent.

An extension to TinyOS which allows modifications of the WSN application during runtime is FlexCup [12]. FlexCup is able to dynamically exchange and link TinyOS components on the sensor nodes for adapting applications to new needs. The five steps of the FlexCup update process are storage of code and meta-data, symbol table merge, relocation table replacement, reference patching, and installation and reboot of the application.

Dunkels et al. developed the Contiki [4] operating system for sensor nodes. With Contiki it is possible to load and unload individual applications or services at runtime [3].

Mantis [1] is a WSN operating system that offers a comprehensive set of system application programming interfaces (API) for I/O and system interaction. Providing this API Mantis hides the complexity of scheduling and concurrent access to ressources claiming a shallow learning curve for programmers with C knowledge. Mantis also allows binary code updates on sensor nodes during the runtime of applications. Based on calls to a system call library provided by the Mantis system kernel applications are updated and written to the EEPROM. By resetting the node the software update process is completed and the new application is executed.

Han et al. introduce SOS [5] a dynamic operating system for sensor nodes. SOS allows the dynamic interaction of independent software modules. SOS uses position independent code to achieve relocation and jump tables for application programs to access the operating system kernel. Application programs can register function pointers at the operating system for performing inter-process communication.

With Maté [8], a virtual machine is available for TinyOS. Maté abstracts from the operating system allowing only a specific set of instructions. A module written with these instructions can be easily integrated into a running application allowing a (re-)configuration and adaptation of the application.

SensorWare [2] is a framework that uses Tcl scripts as abstraction for implementing distributed applications. The scripts can be send into the WSN during runtime. They can migrate within the network and replicate themselves.

The programming framework EnviroSuite [10] by Luo et al. proposes a new programming paradigm called environmentally immersive programming. EinviroSuite offers languages primitives that transparently map onto a library of algorithms for sensor networks allowing the programmer to think diretcly in terms of environmental abstractions.

OASiS [11], a programming framework for service-oriented sensor networks, applies the paradigm of service-orientation on top of TinyOS. In OASiS each activity is implemented as seperate service. The goal is to simplify programming by providing an abstraction, where the applications behavior is described as modular dataflow blocks between services in a specific

service graph.

3 Operating System and Infrastructure

The related work shows that supporting the application development and providing an intuitive network abstraction to the application developer is an important issue. We observe that on different levels from the basal operating system to high level frameworks much work is spent to make the application development easier. Each of these approaches has its strengths but none fulfills our needs to enable users without expertise to create and deploy a WSN application sufficiently.

On operating system level, a lot of understanding of the distributed nature of sensor networks and their constraints is needed. The component based approach of TinyOS allows a composition of the application, but still the right protocols and algorithms for the deployment must be known upfront. Approaches which are based on loadable binary code modules enable the user to change the application during runtime. But often these approaches such as [12] and [4] exploit specific platform characteristics or necessitate a restart of the reprogrammed nodes what can be critical for keeping a consistent network state and is an additional aggravation of the development process. Overall, an operating system alone does not provide a sufficient abstraction for the application developer.

Middlewares and frameworks offer a more intuitive abstraction for the programmer. Using a virtual machine like Maté allows for a platform independent and smooth integration of modules into a running application since no restart of the application is needed. But the programmer is bounded to a fixed subset of instructions offered by the virtual machine. Additionally the programmer has to learn the specific syntax for the virtual machine. Furthermore, such approaches are always tailored to a specific type of application constraining the user to a specifc scenario. Frameworks as well as middlewares already offer a specific set of algorithms and protocols. The developer has no chance to modify or optimize this layer to his needs.

3.1 Overview

In our approach, we want to offer the application developer as much flexibility as possible by providing an easy to use application development support at the same time. The fundament is the service-oriented paradigm, which is commonly used in the field of distributed applications. In contrast to other service-oriented approaches for WSNs we consider all functionalities on a node down to the hardware interfaces as services. Where hardware interfaces are always present on the nodes (non-migratable), all other services need to migrate into the WSN and onto the nodes. We deploy the nodes solely with the non-migrateable services. This implies that neither application logic nor any protocols are present on the node right after deployment. On demand, services representing a routing protocol or a localization algorithm

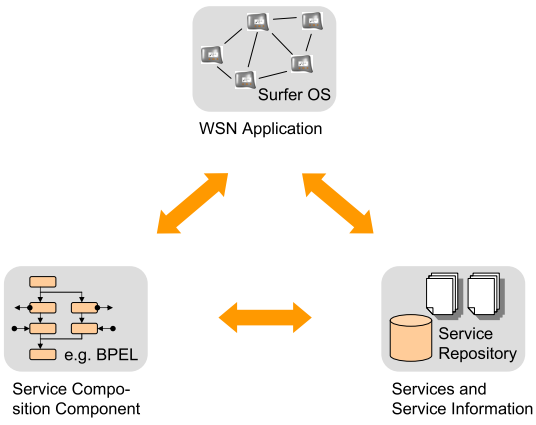


Figure 2 Components of the infrastructure for WSN application development

or part of the application logic migrate successively into the WSN composing themselves to a WSN application.

In order to realize this migration of the services and support an intuitive development of an application, the presented infrastructure consists of three components shown in Figure 2: a minimal operating system called *Surfer OS*, a *Service Repository* and an interface for *Service Compositon*.

The *Service Composition* interface allows for an abstraction for composing services and initiates the migration. The services themselves are stored in the *Service Repository* from where they migrate into the WSN. The operating system on the nodes integrates the services into the running application. In the following we will describe the requirements for each component.

3.2 Operating System

For a successful sensor network deployment, the usage of adequate protocols and algorithms for a specific scenario is crucial. Being not able to predetermine radio characteristic and all network parameters such as data traffic patterns, topology changes or node mobility, the user of WSN technology must be able to exchange all protocols or functionality from the application logic down to the communication layer. In terms of the service-oriented paradigm all functionalities offered by protocols and algorithms are encapsuled into different services. The operating system must provide the possibility to add, remove or exchange services on the nodes during the runtime of the deployed sensor network. As a consequence, right after the deployment the nodes basically offer only two kinds of functionalities: (1) access to the available hardware resources on the node via service calls and (2) management of services.

The access to the hardware resources is realized as very basic service calls. The operating system must publish the access point to the radio interface, the sensors and other hardware resources as services, since such functionality demands for special hardware I/O like, e.g., the reading and writing on special registers. Higher level services such as a radio stack are present as services providing a predetermined interface to other services but the implementation on the node does not offer any higher functionality yet. The service that encapsules the radio stack for sending a packet via the radio is not yet providing any routing layer. This service in its initial version on the node directly accesses the hardware interface sending the “plain” packet.

Managing the services means allowing the integration, removal and replacement of services. The operating system must react on special service packets which are directly processed. When adding the services, the provided functionality must be published on the nodes by the operating system. The functionality must be made accessible in an intuitive way, e.g., via a service and a function name. Every service contains a service type and a version number. The service type is a numerical representation of the functionality provided by the service, e.g., “routing”. The version number distinguishes between different implementations of the service. Based on the type of a service and the version the operating system determines whether to add the provided service, to replace a present service or to ignore the received service. If a service of a specific type is already present, it is replaced if a different version of this service type is received. Additionally, dependencies among services must be preserved by the operating system in case of replacement of services. To realize these demands, we implemented the operating system *Surfer OS* which is described in Section 4.

3.3 Service Repository

In order to compose WSN applications, the user of WSN technology needs a collection of different services providing different functionalitites. Programmers with WSN expertise develop a lot of different algorithms and protocols suited for different scenarios and applications. If these algorithms and protocols are implemented as services for *Surfer OS*, they can be applied by a WSN user in his application. Therefore, we need a place where these services can be uploaded, stored and accessed by a component which manages the service migration. For this task we implemented a so called *Service Repository*. It acts as a container for migrateable services where services can be up- and downloaded or managed by the service programmers via an easy to use user interface.

In order to enable the service programmer as well as the service user to use the *Service Repository* it must be accessable via an intuitive and easy to use user interface. This interface has to provide the above mentioned functionalities namely: displaying all available services including important service information, providing the possibility to insert new services and upload the corresponding service code, and providing the possibility to change the service information as well as the service code itself.

3.4 Service Composition

The third component of our approach addresses the process of *Service Composition*. This component enables abstraction for the user of WSN technology for creating sensor network applications by composing a set of services.

One important characteristic of a service-oriented approach for distributed applications is its flexibility. Thus, the requirements to the design of the *Service Composition* component are very versatile. If we consider how services can be composed, we identify two scenarios: Services can be composed by a *Human Composer* or by a software performing the composition to which we refer to as *Automatic Composer*.

The *Human Composer* is the developer of an application who is able to deploy a WSN application even without expertise in this field. By selecting a set of services out of a service repository, he can compose an application tailored to the demands of his individual application scenario ad-hoc during runtime.

The composition of services can also be performed automatically by a software, the *Automatic Composer*. Today sensor networks become more and more part of enterprise IT systems or even flexible parts of business processes. Therefore, it must also be possible that such systems can control the composition of services for the sensor network automatically.

The realization of the *Service Composition* component has to meet the requirements of both scenarios: the *Human Composer* and the software based composition. Therefore, it must provide an intuitive interface to the *Human Composer* and a standardized interface for the composition of services by an external software.

4 Surfer OS

4.1 Concept and Architecture

As described in Section 3.2, the main task of our operating system *Surfer OS* is to provide access to available hardware resources and to manage services, which compose the complete WSN application including even lower layers such as the communication stack. The management of services includes addition, removal and replacement of services. At the moment of deployment of the sensor network, the nodes do not provide any further functionality.

The central unit within *Surfer OS* that organizes the services is the *ServiceManager*. The *ServiceManager* keeps a dynamic symbol table that maps a tuple consisting of a service type and a function name onto a function pointer. This allows an intuitive usage of services by accessing them via the service type and the function name. Initially, the *ServiceManager* keeps the non-migratable services (dark-colored in Figure 3) representing the hardware interfaces in the symbol table. This includes the access to the sensors, the radio, the RS232, the memory, as well as scheduling as shown in Figure 3. Additionally, basal service implementations representing the radio stack and the RS232 stack are provided using the specific hardware interfaces.

Surfer OS identifies special service packets which are then processed by the *ServiceManager*. With these packets a service can migrate onto a node or the command for erasing a service on a node can be transmitted. If such a migratable service (light-colored in Figure 3) is received, the *ServiceManager* checks the service type and the version number. If a service of this type is not present, the new service is included into the application. The new functionalities are added under the service type in the symbol table of the *ServiceManager* such as the beacon service in Figure 3. The *ServiceManager* offers an interface where services themselves can access other services and the hardware via the services already registered in the symbol table. If a service of the same type but with a different version number is already present, the present service is removed and the new service is added. In this way even non-migratebale services can be replaced as shown in Figure 3. In order to preserve dependencies such as registrations for callback, the registration is provided by a subscription service. This service keeps a table with service type and callback addresses. If a service is replaced, the previously registered callbacks for this service type can be retrieved by the new service. This allows a seamless replacement of services that are used by other services. Thus, we can even exchange, e.g., the services representing the communication stack enabling us to exchange routing protocols during runtime.

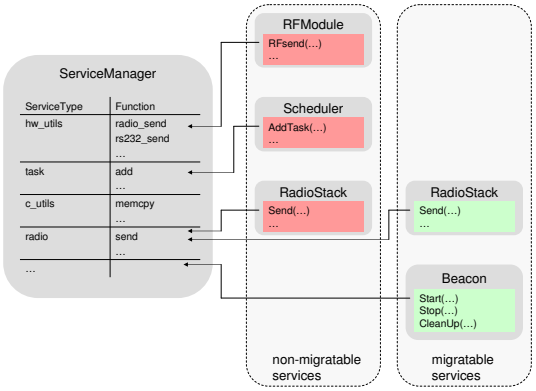


Figure 3 The *ServiceManager* module with the dynamic symbol table

4.2 Implementation

We implemented *Surfer OS* on the *pacemate* sensor node platform [9]. The *pacemate* platform uses the Phillips LPC2136 microcontroller with 256 kbyte Flash memory and 32 kbyte RAM. The microcontroller realizes a 32 Bit ARM architecture. The LPC2136 provides in-application programming allowing for erasing and/or programming the Flash while the application is running. The complete Flash or a single Flash sector can be erased in 400 ms. Writing 256 bytes to the Flash takes 1 ms. Program code is executed from the Flash but can also be executed directly in the RAM.

The *Surfer OS* is implemented in C. The *ServiceManager* keeps the addresses of functions as function pointers. The services are compiled independently. For the migration of the services onto a node we have to solve the problem of code relocation. We use a scheme that provides a solution for the general problem of code relocation. This approach exploits the partition of the ELF format into .TEXT, .DATA and .BSS segments that can be used on

every hardware platform that uses the ELF file format. After a service is received completely and its code is relocated successfully, the *ServiceManager* accesses the new service via a predefined interface.

The services follow a specific service template. Every service implements at least four functions: ServiceInit(...), ServiceStart(), ServiceStop() and ServiceCleanUp(). The ServiceInit(...)-function performs the registration process of the service. The ServiceStart() and ServiceStop() start or stop the actions of the service. The ServiceCleanUp() is executed when the service is removed. The programmer of a service has to make sure that all registrations and callbacks are removed in this function.

The ServiceInit(...)-function is invoked by the *ServiceManager* after the service is received. The linker file ensures that this function is located at the very beginning of the programm code in the .TEXT segment enabling the *ServiceManager* to find the entry point to this function. A predefined signature allows the *ServiceManager* to pass arguments into the ServiceInit(...)-function. In this way the function pointers to the interfaces of the *ServiceManager* are given to the service. The service uses these interfaces to retrieve available functionality from other services and register its own provided functions. If the registration of functions is completed, the ServiceInit(...) returns the status to the *ServiceManager*. If the registration was successful, the process is completed. The *ServiceManager* executes the ServiceStart()-function that is now available in the symbol table. If the registration of functions failed, the *ServiceManager* would remove the service from the symbol table and would free the allocated memory erasing the service.

Since received services are written to the Flash memory, they are persistent even if the node is turned off. If the node is rebooted, the present services can be integrated and restarted. In this way, the software configuration of a node can be preserved, reused and even analyzed in case of failures.

5 Service Repository

In Section 3.3 we introduced the *Service Repository* as a collection of services which are provided by service programmers with WSN expertise and can be applied by the WSN user. To meet the described requirements, we identify three main modules that compose the *Service Repository*: a graphical *User Interface*, a *Repository Service* and a *Machine Code Modification Service*.

The *User Interface* lists available services and visualizes complete information of services such as service type, version, size and author. For the service programmer, it provides input masks to upload implemented services. The user interface additionally offers the possibility to change the service information in the database as well as in the stored service code. Using the migration scheme described in Section 4.2 the services themselves are stored as machine code for the target sensor node platform which is in our case the pacemate platform. We realized the user interface as Web-based application.

The connection to the database is established by the *Repository Service*. This service – realized as Web Service – encapsulates the upload of services into the database and makes the stored data available. More precisely, it offers interfaces to store, delete, update and display the content of the database.

The functionality to allow modifications of the machine code in order to update service type and version is provided by the *Machine Code Modification Service*. If the user changes this information via the input mask provided by the user interface, this modification service writes the new information back into the machine code. This avoids unnecessary recompilations of

the service code and guarantees consistency between the database entry and the according service code.

6 Service Composition

The aim of our contribution is to make WSN technology available for users without any WSN expertise. In Section 3.4 we described the requirements for the *Service Composition* component. We identified two scenarios: either the *Human Composer* or the *Automatic Composer* is controlling the composition process. We demanded that the composition component of our presented infrastructure has to meet the requirements of both scenarios. In order to fulfill these requirements, we have to divide the composition component into two modules which are designed following the service-oriented architecture programming paradigm. We need one module which controls the actual migration process (*Migration Control*). As second module, we need the *Migration Infrastructure* consisting of the *Repository Service* and the *Migration Service* (see Figure 4).

The *Migration Infrastructure* is the basis of the *Service Composition* component. While the *Repository Service* (cf. Section 5) enables the access to a set of *Surfer OS* services, the *Migration Service* realizes the interface to the sensor network. The controlling module of the migration process retrieves services from the *Repository Service*. Calling the *Migration Service* pushes the services into the WSN and completes the migration process outside the sensor network. Analog to the functionality of pushing *Surfer OS* services into the sensor network, the *Migration Service* provides the removal of services from sensor nodes. As the pushing process, the removal process is controlled by controlling modules as well.

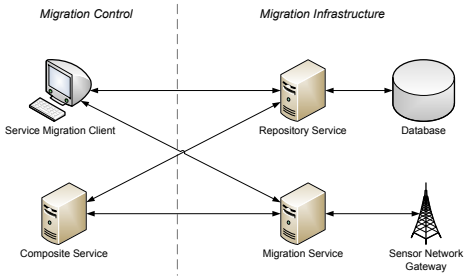


Figure 4 Architecture of the implementation of the *Service Composition* component

The *Migration Control* manages the migration process and offers the abstraction for the user for the WSN application development. As part of our infrastructure we provide a graphical user interface application allowing the *Human Composer* to select services of a repository and to transfer them into the sensor network. In order to embed the service migration process into enterprise IT systems using the *Automatic Composer*, such systems have to provide the capability of controlling the migration. Based on the *Repository Service* and the *Migration Service* and the fact that we use Web Services as communication backbone, we are able to build other high level services on top using classical programming languages like Java or C++ as well as business process modeling languages such as BPEL (Business Process Execution Language, [13]). Especially embedding the migration process into business processes, which are designed with a graphical tool support, enables non IT personnel to easily compose *Surfer OS* services.

7 Discussion

7.1 Benefits

Having implemented all three components, the *Surfer OS*, the *Service Repository* and the interface for *Service Composition*, we now have an infrastructure for quick and easy WSN

application development. Opposed the development cycle for WSN applications previously shown in Figure 1, our approach realizes an application development process shown in Figure 5.

The service-oriented approach and thus the loosely coupled services provide an abstraction that is commonly used for distributed applications.

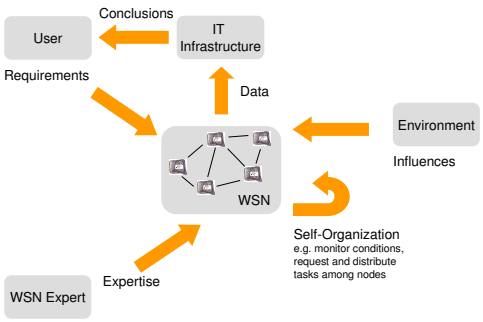


Figure 5 Realized WSN application development process

A user of WSN technology can design and adapt a sensor network application with *Surfer OS* independently. Successively the user migrates services representing application logic as well as different protocols into the WSN and onto the nodes composing his own application for his specific requirements. The ability to add, remove and replace services during runtime gives the user a high degree of flexibility and independence. Especially in contrast to approaches using multi-hop over the air flashing and replacing the whole software image on a sensor node, the service based approach of *Surfer OS* produces less network traffic if changes to an application become necessary. The services themselves, especially those that demand for special WSN knowledge, are provided by WSN experts who have the expertise in the challenges of distributed ad-hoc systems. Services can also be implemented by the user following the service template. In order to compose a WSN application, the user can utilize the provided graphical user interface for manually transfer needed services onto the nodes. Our presented infrastructure also allows for using even professional application design and integration tools such as BPEL, which represent the WSN application as business processes. Using such tools eases the integration of the sensor network into an existing IT infrastructure.

The services provided in the *Service Repository* can also implement selforganizing functionality causing other services or data to migrate or replicate themselves automatically. Even a stateful migration of services is possible by migrating the according .DATA segment from the RAM and the dynamically allocated memory as well. Services providing self organizing functionalities can be realized with *Surfer OS*. They enable the network to react independently on topology changes or depleting energy or memory ressources extending the lifetime of the sensor network application. Furthermore, such approaches again hide network complexity from the user.

Additionally, there is no need for predicting and modeling radio characteristics in order to specify a routing protocol or other topology dependent algorithms. The service-oriented approach in *Surfer OS* that encapsules every functionality as a service allows the replacement even of the radio stack including the replacement of routing protocols. Thus, the right protocols can be chosen during the deployment till satisfactory results are achieved. In the same manner the RS232 stack can be replaced if changed IT infrastructure demands for a new data format.

During the development of different services we realized that the service-oriented approach for applications respectively protocols offers even more benefits than just the composition of applications. When implementing protocols, e.g, for routing or localization, we found out that often some basic functionalities are many protocols in common. E.g., a lot of protocols have to be aware of their current neighborhood size or even of the node ids. Therefore all these protocols send so called hello beacons. In this way the nodes in the direct neighborhood

are aware of the presence of this neighboring node. When using more than one of these protocols within one application, each protocol performs its own beaconing. This results in a waste of radio bandwidth. Additionally, when the node has already transmitted data packets, there is no need for additional beacons, since the neighbors are already aware of the presence of the node. We realized that even protocols can be subdivided into several services where each service can focus on performing as efficiently as possible. In this way we can reduce the needed resources. Memory can be saved since different services may use the functionality of one service instead of providing their own functionality. In the same way, the communication costs can be reduced since a needed action like a beacon is only performed once.

7.2 Downsides

Adding functionality to the network after deployment has its price. Tightly coupled systems such as TinyOS can be optimized for a special application. Unneeded code modules can be ommitted during the linking process of the application before deployment resulting in a minimal code size. In order to support future application modifications by addition or removal of services, *Surfer OS* must provide all basic functionalities of the sensor node. *Surfer OS* must offer software interfaces to all hardware components on the node which might be used by services that migrate onto the node during deployment even if a hardware component stays unused during the deployment. This implies that the code size of the *Surfer OS* itself cannot be optimized for special applications.

The integration of services and thus the relocation of code itself demands for some additional information and program logic. Each service must provide relocation information in order to make the code executable on the target node. Additionally, *Surfer OS* must keep a dynamic symbol table and offer interfaces to the *ServiceManager*. For the relocation information of the integrated services and for the code for the *ServiceManager* and symbol table additional memory is needed on the nodes. Due to this, a loosely coupled *Surfer OS* application offers less potiential for optimization of code sizes than a tightly coupled application.

Tightly coupled monolithic sensor network applications are usually flashed onto the nodes before deployment with special programming tools that might include special programming hardware. Right before deployment the batteries of the nodes can be fully charged. A *Surfer OS* application is adapted after the nodes are already deployed. The *Surfer OS* infrastructure can be connected to any arbitrary node within the network, allowing the addition or removal of services in the whole network. This flexibility implies that services have to be distributed throughout the network. Thus the modification of an application is naturally accompanied by additional network traffic.

8 Conclusion and Future Work

As state of the art, it is a very tedious task to design and deploy a sensor network application. Special skills in distributed application design and WSN expertise are inevitable. In this work we presented a service-oriented minimal sensor node operating system called *Surfer OS* and an application development infrastructure which allows an application design from a set of existing services making wireless sensor networks accessible for a wider range of users.

In contrast to current monilithically designed applications the highly modular architecture of the *Surfer OS* provides a high degree of flexibility allowing the user to constantly adapt an application to his individual needs. The possibility to migrate stateful services allows for new strategies to efficiently use the resources provide by the network. The presented solution

can be transferred into other areas of pervasive computing, where flexible applications on resource constrained devices are needed. With our work we aim to ease the development process of pervasive technology in various fields of application.

References

1 H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. Mantis: System support for multimodal networks of in-situ sensors. In *2nd ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 50–59, 2003.

2 Athanassios Boulis, Chih-Chieh Han, and Mani B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *Proceedings of the 1st international conference on Mobile systems, applications and services (MobiSys '03)*, pages 187–200, New York, NY, USA, 2003. ACM.

3 Adam Dunkels, Niclas Finne, Joakim Eriksson, and Thiemo Voigt. Run-time dynamic linking for reprogramming wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems (SenSys '06)*, pages 15–28, New York, NY, USA, 2006. ACM.

4 Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.

5 Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services (MobiSys '05)*, pages 163–176, New York, NY, USA, 2005. ACM.

6 Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, and Kristofer S. J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

7 P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. *Ambient Intelligence*, pages 115–148, 2005.

8 Philip Levis and David Culler. Maté: a tiny virtual machine for sensor networks. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems (ASPLOS-X)*, pages 85–95, New York, NY, USA, 2002. ACM.

9 Martin Lipphardt, Horst Hellbrueck, Dennis Pfisterer, Stefan Ransom, and Stefan Fischer. Practical experiences on mobile inter-body-area-networking. In *Proceedings of the Second International Conference on Body Area Networks (BodyNets'07)*, 2007.

10 Liqian Luo, Tarek F. Abdelzaher, Tian He, and John A. Stankovic. Envirosuite: An environmentally immersive programming framework for sensor networks. *Trans. on Embedded Computing Sys.*, 5(3):543–576, 2006.

11 Xenofon Koutsoukos Sandeep Neema Manish Kushwaha, Isaac Amundson and Janos Sztipanovits. Oasis: A programming framework for service-oriented sensor networks. In *IEEE/Create-Net COMSWARE 2007*, January 2007.

12 Pedro José Marrón, Matthias Gauger, Andreas Lachenmann, Daniel Minder, Olga Saukh, and Kurt Rothermel. Flexcup: A flexible and efficient code update mechanism for sensor networks. In *Proceedings of the Third European Workshop on Wireless Sensor Networks (EWSN 2006*, pages 212–227. Springer, 2006.

13 OASIS WS-BPEL Technical Committee. Webservices – Business Process Execution Language Version 2.0, 2005.

An adaptive protocol for distributed beamforming

Stephan Sigg[1] and Michael Beigl[2]

1 **TecO, Chair for Pervasive Computing Systems**
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
sigg@teco.edu

2 **TecO, Chair for Pervasive Computing Systems**
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
michael@teco.edu

Abstract

We study distributed adaptive beamforming in networks of wireless nodes. In particular, we observe that for the synchronisation of carrier phases, distinct algorithmic configurations are optimal in various environmental settings and propose a protocol that utilises organic computing principles to find optimum parameters. Furthermore, we study the impact of different modulation schemes on the bit error rate of a signal sequence transmitted collaboratively by distributed devices via adaptive beamforming.

1998 ACM Subject Classification I.1.2 Algorithms, I.2.6 Learning

Keywords and phrases Distributed beamforming, Adaptive synchronisation protocol

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.38

1 Introduction

Beamforming is the approach to combine transmission signals from distinct transmit antennas simultaneously in order to create a superimposed signal with improved channel characteristics at a remote location [18]. A necessary condition for beamforming is that the signal streams that are placed onto the transmit antennas are tightly synchronised. One period of a signal transmitted at 2.4 GHz, for instance, lasts only for $0.0004\mu s$. With inaccurate synchronisation among signal streams, the relative phase allocation is therefore likely random. While this very tight synchronisation is achied for centralised beamforming [4], where all antennas are located on one device and the signal streams are controlled by a single controller on this device, it poses a major challenge for distributed beamforming where antennas of distributed devices are utilised for signal transmission [7, 8, 2].

Several open-loop and closed-loop carrier synchronisation approaches have been proposed that enable sufficient synchronisation among carrier signals. These classic approaches are, however, computationally very complex so that an application in wireless sensor networks is not suggestive.

A computationally cheaper but more time consuming randomised interactive closed-loop carrier synchronisation was proposed in [10]. It was analysed for its computational complexity in [9, 15, 16] and several algorithmic improvements have been proposed in [5, 14, 13, 3].

This approach is feasible to be applied in wireless sensor networks due to its low computational complexity for individual nodes. However, despite the numerous studies on this approach, only carrier synchronisation but no actual data transmission was yet studied with this transmission protocol.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 38–48

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In our work we present a protocol for this iterative distributed adaptive beamforming scheme and demonstrate the accuracy of data transmission for several modulation schemes in mathematical simulations. Furthermore, we show that the performance of the protocol is impacted by environmental impacts such as the noise figure, movement or activity on the wireless channel but that by adapting several parameters of the transmission protocol, an optimum performance can be achieved for each distinct environment.

We introduce some related work on distributed adaptive beamforming in wireless sensor networks in section 2. In section 3 we show that the synchronisation performance is impacted by environmental settings and introduce a protocol for distributed adaptive transmit beamforming that utilises organic computing principles. In section 4 we study the impact of various modulation schemes on the performance of distributed adaptive transmit beamforming in networks of wireless devices. Finally, in section 5 we draw our conclusion.

2 Distributed beamforming in wireless sensor networks

Algorithms for distributed adaptive beamforming can be distinguished by closed-loop phase synchronisation and open-loop phase synchronisation techniques. Closed-loop carrier synchronisation can be achieved by a master-slave approach as detailed in [17]. The central idea is that nodes transmit a synchronisation sequence simultaneously on code-divisioned channels to a destination node. The destination calculates the relative phase offset of the received signals and broadcasts this information to all transmitters which adapt their carrier signals accordingly.

Due to the high computational complexity burden for the source node to derive the relative phase offset of all received signals and for all nodes due to the utilisation of code division techniques, this implementation is not suggestive in some applications.

Alternatively, a Master-slave-open-loop synchronisation can be applied [6]. In this approach, the relative phase offset among nodes is determined by the transmit nodes with a method similar to [17] but only among transmit nodes. The receiver then broadcasts a carrier signal once so that the transmit nodes are able to correct their frequency offsets. In this method, however, the generally high complexity for the nodes is only shifted from the receiver node to one of the transmit nodes. Therefore, this approach is also hardly feasible in wireless sensor networks.

A simpler and less resource demanding beamforming scheme to synchronise carrier singal components for distributed beamforming is the one-bit feedback based closed-loop approach considered in [17, 8]. The central optimisation procedure of this iterative process consists in n devices $i \in [1, \ldots, n]$ randomly altering the phases γ_i of their carrier signal

$$\Re\left(m(t)e^{j(2\pi(f_c+f_i)t+\gamma_i)}\right) \tag{1}$$

In this signal representation, $m(t)$ denotes the transmit message and f_i the frequency offset of device i to a common carrier frequency f_c. Initially, i.i.d. phase offsets γ_i of carrier signals are assumed. When a transmission from the devices is requested, carrier phases are synchronised in the following iterative process.

1. Each transmitter i randomly alters its carrier phase offset γ_i and frequency offset f_i.
2. Devices transmit as a distributed beamformer.
3. A receiver estimates the amount of synchronisation among carrier phases (for instance by the Signal-to-Noise ratio (SNR) of the received sum signal).
4. This information is broadcast to the transmit devices that interpret it and adapt their carrier phase accordingly.

These four steps are iterated repeatedly until sufficient synchronisation is achieved [9, 11, 12]. Observe that in each of these iterations, a reduction of the SNR is not accepted. A new configuration of phase offsets for transmit carrier components is accepted only when the SNR was increased. Since the search space for the algorithm is weak multimodal [15], this means that the method converges to the optimum with probability 1 [9]. This result achieved in [9] considered an idealised environment without noise and interference. In a realistic environment, the impact of the noise figure determines the accuracy that can be achieved by this approach.

The distinct approaches proposed for this transmission scheme differ in the implementation of the first and the fourth step specified above. In [11, 3] devices alter their carrier phase offset γ_i according to a normal distribution with small variance. In [12] a uniform distribution with a small probability to alter the phase offset of one individual device is utilised instead.

In [9] it was determined that the expected optimisation time of this approach for a network of n transmit nodes is $\mathcal{O}(\log n)$ when in each iteration the optimum probability distribution is chosen. For a fixed uniform distribution over the whole optimisation process, we were able to derive a sharp asymptotic bound of $\Theta(n \cdot k \cdot \log n)$ for the expected optimisation time [15]. Here, k denotes the maximum number of distinct phase offsets a physical transmitter can generate.

When we assume a reasonable number of signal periods for one iteration, this means that, although this synchronisation time is greatly higher than the time required for synchronisation with the classical methods, it is still in the order of milliseconds for reasonable network sizes.

The strength of feedback based closed-loop distributed adaptive beamforming in wireless sensor networks is its simplicity and low processing requirements that make it feasible for the application in networks of tiny sized, low power and computationally restricted devices. However, the impact of environmental settings as well as the bit error rate (BER) achieved by several devices transmitting collaboratively has not yet been considered.

In the following we present a transmission protocol that utilises this synchronisation scheme and that is in addition responsive to environmental impacts.

3 Adaptive distributed beamforming protocol

In one-bit feedback based distributed adaptive transmit beamforming, devices combine their carrier signals to collaboratively reach a remote receiver. In recent studies we have already observed some hints that the phase synchronisation performance might be dependent on environmental parameters. These parameters are, for instance, the probability to alter phases of transmit carriers, the noise figure, the distance between transmit devices and a remote receiver or the count of devices participating in the synchronisation [14, 13, 16]. Figure 1 exemplary shows the impact of the network size on the synchronisation process. We observe that the best phase alteration probability to achieve an optimum BER for collaborative transmission differs among various network sizes. Consequently, optimum parameters have to be derived individually for each concrete scenario.

Therefore, we propose a protocol for distributed adaptive beamforming that incorporates self-adaptation and self-optimisation features. The protocol adapts the parameters of the iterative synchronisation to a given environment so that an optimum synchronisation performance is achieved.

All devices utilise the iterative distributed carrier synchronisation detailed in [10, 6]. In order to adapt to different environments, devices maintain and adapt the following parameters.

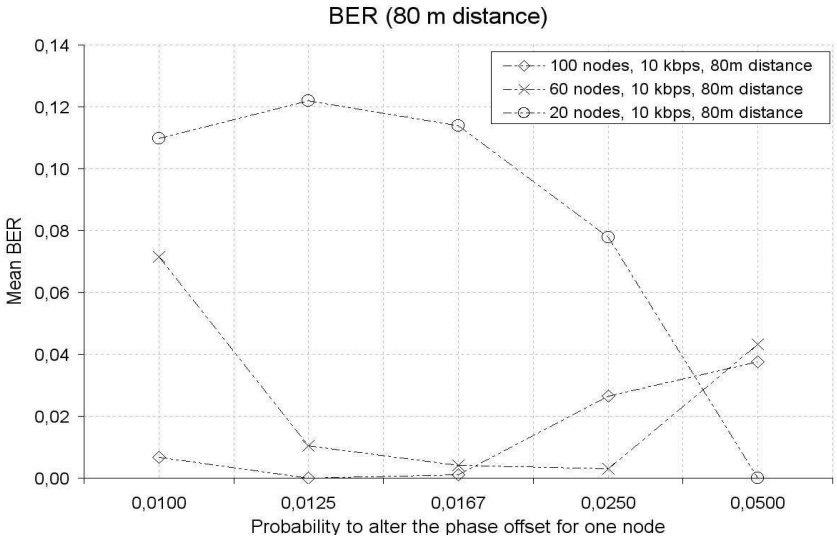


Figure 1 Impact of the network size on the BER at distinct phase alteration probabilities for one bit feedback based distributed phase synchronisation and adaptive transmit beamforming

$P_{mut,i}$ Probability to alter the phase offset of an individual device i ($P_{mut,i} \in [0,1]$)
$P_{dist,i}$ Probability distribution for the random process of device i ($P_{dist,i} \in \{\text{normal, uniform}\}$)
var_i Variance for the random process ($var_i \in [0,\pi]$)

The transmission protocol consists of the following steps that are executed in order.

1. An individual device broadcasts a data sequence s_d to devices in its proximity.
2. Devices decide whether to participate in the transmission. Possible decision parameters are, for instance, the energy level, a required count of participating devices or current computational load.
3. Closed-loop one bit feedback based carrier synchronisation is achieved (cf. section 2). Devices utilise $P_{mut,i}$, $P_{dist,i}$, var_i.
4. Upon sufficient synchronisation the receiver broadcasts an acknowledgement.
5. Optimisation parameters $P_{mut,i}$, $P_{dist,i}$ and var_i are adapted.
6. Devices collaboratively transmit s_d.

For a given environment, a set of devices can with this protocol improve their synchronisation performance after several transmissions.

The protocol is self-adaptive to a given environment and self-healing as it automatically adapts the optimisation parameters to changing numbers of participating devices or communication topologies.

We evaluate this protocol in mathematical simulations in a network of 100 devices. The scenario of distributed adaptive beamforming in an environment of distributed devices was implemented in Matlab. In the simulation, we calculate the phase offset of the dominant signal component from each transmit device at the remote receiver based on the transmission distance between the nodes in a direct line of sight (LOS) scenario. Path loss was calculated by the Friis free space formula $(P_{tx}\left(\frac{\lambda}{2\pi d}\right)^2 G_{tx}G_{rx})$ with antenna gain for the transmitter and the receiver as $G_{rx} = G_{tx} = 0dB$. Signals are transmitted at $2.4GHz$ with transmit power $P_{tx} = 1mW$.

Table 1 Configuration of the simulations. P_{rx} is the the received signal power, d is the distance between transmitter and receiver and λ is the wavelength of the signal

Property	Value
Node distribution area	$30m \times 30m$
Location of the receiver	$(15m, 15m, 30m)$
Mobility	stationary devices
Base band frequency	$f_{base} = 2.4$ GHz
Transmission power of devices	$P_{tx} = 1mW$
Gain of the transmit antenna	$G_{tx} = 0$ dB
Gain of the receive antenna	$G_{rx} = 0$ dB
Iterations per simulations	6000
Identical simulation runs	10
Random noise power [1]	-103 dBm
Pathloss calculation (P_{rx})	$P_{tx} \left(\frac{\lambda}{2\pi d}\right)^2 G_{tx} G_{rx}$

All received signal components calculated in this manner are then summed up in order to achieve the superimposed sum signal

$$\zeta_{\text{sum}}(t) = \sum_i \left(\Re \left(m(t) e^{j(2\pi(f_c + f_i)t + \gamma_i)} \right) \right) \tag{2}$$

Finally, a noise signal $\zeta_{\text{noise}}(t)$ is added on to $\zeta_{\text{sum}}(t)$ to calculate the signal at the receiver. We utilise ambient white Gaussian noise (AWGN) with $-103dBm$ as proposed in [1].

Short of multipath propagation the simulation therefore captures all relevant aspects of the radio channel in our scenario. In particular, the channel is not modelled by a statistical distribution but calculated on a LOS basis.

Devices are distributed uniformly at random on a $30m \times 30m$ square area with a receiver located up to $200m$ above the centre of this area. All devices are stationary and frequency and phase stability are considered perfect (cf. Table 1).

Each simulation lasts for 6000 iterations and one iteration consists of the devices transmitting, feedback computation, feedback transmission and feedback interpretation. It is possible to perform these steps within few singal periods so that the time consumed for a synchronisation of 6000 iterations is in the order of milliseconds for a base band signal frequency of 2.4 GHz.

Signal quality of a signal during the synchronisation phase is measured by the Root of the Mean Square Error (RMSE) of the received signal ζ_{sum} to an expected optimum signal ζ_{opt}:

$$RMSE = \sqrt{\sum_{t=0}^{\tau} \frac{\left(\zeta_{\text{sum}}(t) + \zeta_{\text{noise}}(t) - \zeta_{\text{opt}}(t)\right)^2}{n}} \tag{3}$$

In equation (3), τ is chosen to cover several signal periods.

The optimum signal is calculated as a perfectly aligned and properly phase shifted received sum signal from all transmit sources. For the optimum signal, noise is disregarded.

We utilise an optimisation approach that implements a uniform distribution to alter the phase offset of distributed carrier signals. Consequently, only the mutation probability $P_{mut,i}$ of devices $i \in [1..n]$ is altered. After each 10 successful synchronisations, the mean achieved RMSE is compared to recently achieved RMSE values and the phase alteration probability is adapted accordingly. As all nodes receive identical feedback from the receiver device, this adaptation process is identical among devices.

We implement the search for the optimum mutation probability as a divide and conquer approach. Nodes start with a mutation probability of 0.5 and then subsequently approximate

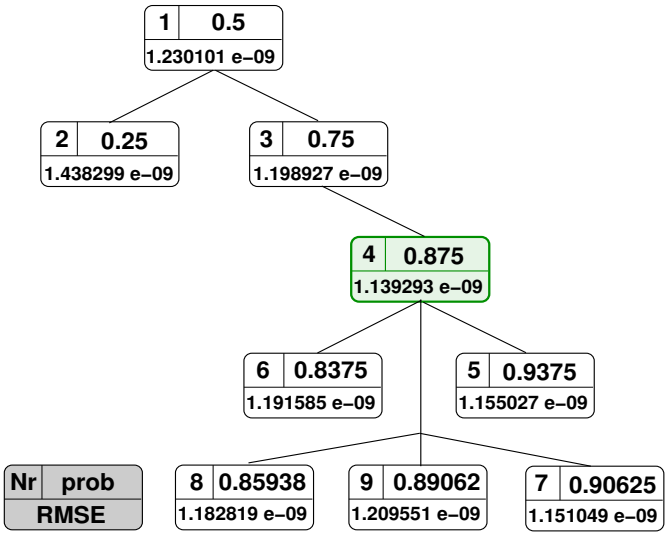


Figure 2 Schematic of the optimisation process of the proposed protocol. RMSE values depicted denote the mean RMSE after 10 synchronisations with identical $P_{mut,i}$

the optimum mutation probability by testing those parts of the search space with lower and higher probability. The process always follows the phase alteration probability with the best achieved RMSE after 10 synchronisations. Figure 2 schematically illustrates this optimisation process in a network of 100 devices that are located approximately 30 meters from a remote receiver.

All devices first complete synchronisations with $P_{mut,i} = 0.5$ and then derive the mean RMSE values for $P_{mut,i} = 0.25$ and $P_{mut,i} = 0.75$. Since the latter probability achieves a better RMSE in this simulation, the lower half of the probability space ($P_{mut,i} \in [0, 0.5]$) is disregarded in the synchronisation process. With 0.875 a probability is reached for which no further improvement is found. In order to derive an optimum value, the algorithm tests additional three probability values in the proximity of the best value reached so far and then exits with the optimum derived mutation probability of $P_{mut,i} = 0.875$ in this case.

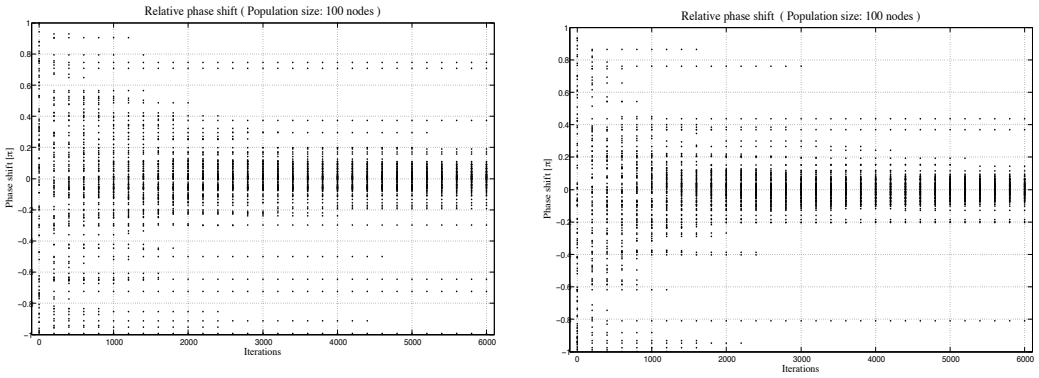
Figure 3 depicts the relative phase offset of all carrier signal components during the course of the synchronisation for $P_{mut,i} = 0.5$ and $P_{mut,i} = 0.875$. We observe that the synchronisation in the latter case is better since the variance in the phase offsets achieved for all nodes is lower.

We can see this also from the sequence of RMSE values observed by the remote receiver as depicted in figure 4. The synchronisation with $P_{mut,i} = 0.875$ is faster and achieves an improved RMSE during the synchronisation.

In general, the algorithm searches the probability space in a binary search fashion in order to bound the optimum mutation probability. Figure 5 depicts the RMSE values achieved in this process in an environment where the receiver is located 50 meters apart from the transmit devices.

4 Modulation and data transmission

When carrier signal components are sufficiently synchronised, transmit devices simultaneously start transmitting their message $m(t)$. The distinct signal components are then superimposed at the remote receiver. Naturally, as synchronisation is not perfect, we expect a considerable



(a) Distributed carrier phase synchronisation with $P_{mut,i} = 0.5$

(b) Distributed carrier phase synchronisation with $P_{mut,i} = 0.875$

Figure 3 Relative phase offset achieved during distributed carrier phase synchronisation processes

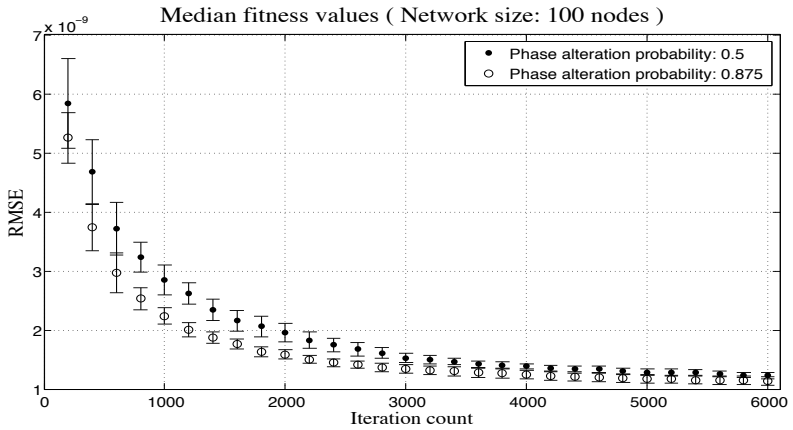


Figure 4 Median RMSE values achieved in the course of the synchronisation

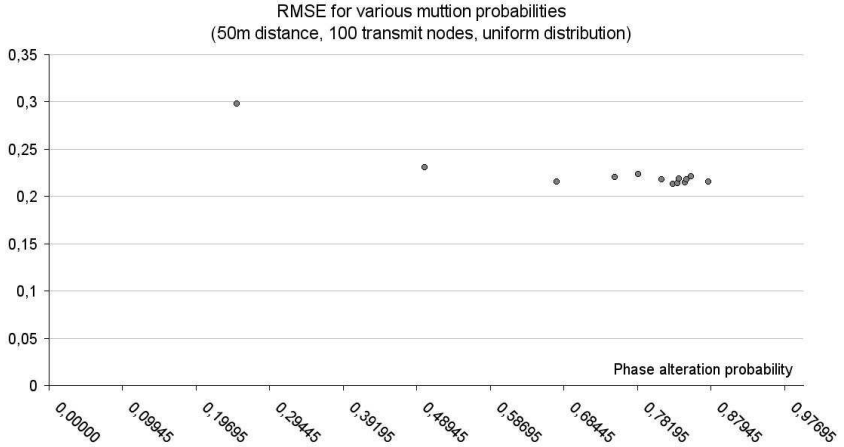


Figure 5 Mean RMSE values after each 10 synchronisations for various phase alteration probabilities $P_{mut,i}$

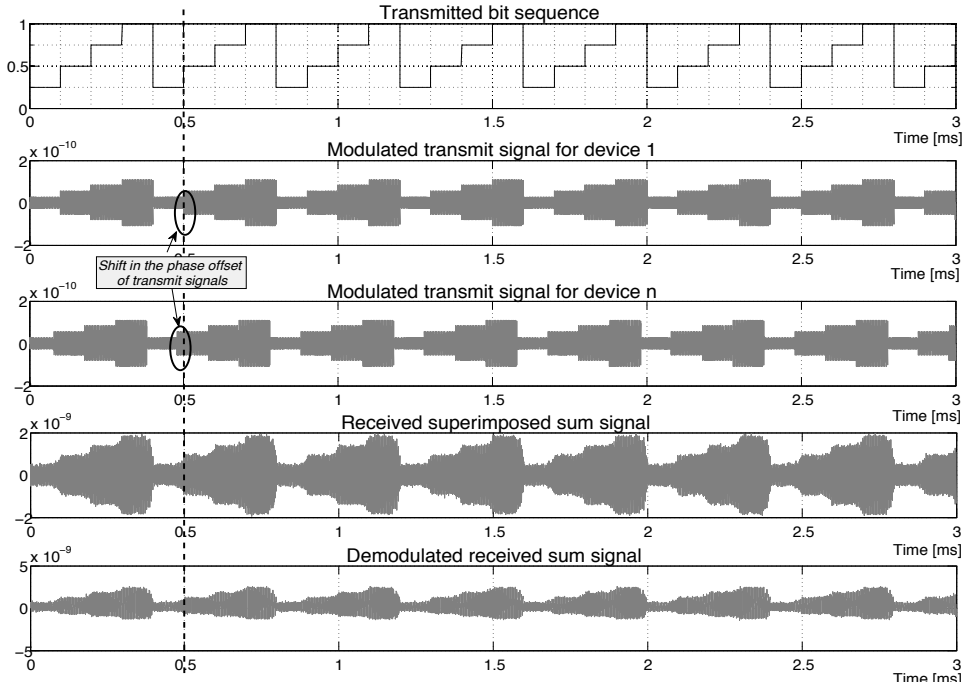


Figure 6 Modulation and demodulation of a simple symbol sequence

bit error rate for this transmission. We study the bit error rate achieved by distributed beamforming devices for various transmission distances and modulation schemes. We again utilise the simulation environment introduced in section 3.

After each complete synchronisation (6000 iterations) all 100 devices transmit an identical signal sequence $m(t)$. We utilise a simple amplitude modulation scheme. At the receiver the BER of the received and demodulated signal is calculated.

After synchronisation is achieved, a message signal $m(t)$ is modulated on the transmit carrier and simultaneously transmitted by all devices. Figure 6 illustrates the process of modulation, transmission and demodulation.

In this modulation scheme, two bits are modulated for one transmitted symbol. We modulated a simple periodical symbol sequence on the transmit carrier of each single node. In the figure, we observe a considerable carrier phase offset of the two nodes depicted. For this environmental setting we observe that the received superimposed signal is improved in its signal strength compared to a single transmit signal. Also, the symbol sequence is clearly visible from the received superimposed signal. Consequently, the bit error rate (BER) for this configuration is low. Figure 7 depicts the BER for three amplitude modulation schemes and for various transmission distances.

In all simulations, 100 transmit devices are utilised to superimpose their carrier signals. In the other two modulation schemes, three and one bits are represented by one transmit symbol, respectively. As expected, we observe that the BER is higher with the higher modulation scheme and with increasing distance. While it is neglectable for the weaker modulation scheme at a distance of 30 meters, the BER becomes significant with increasing distance for all modulation schemes.

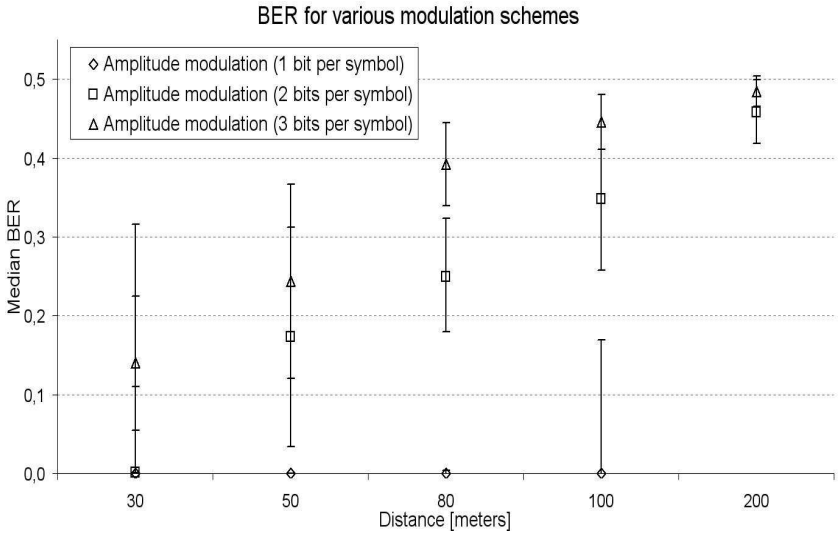


Figure 7 Bit error rate for three amplitude modulation schemes

5 Conclusion

We have introduced an adaptive divide and conquer protocol for data transmission among distributed adaptive beamforming devices that adapts to environmental settings in order to achieve optimum synchronisation between beamforming devices. The protocol builds on a recently proposed computationally cheap randomised iterative beamforming algorithm. We extend the current work by demonstrating the bit error rate for actual data transmission with several modulation schemes.

In mathematical simulations of networks of 100 beamforming devices we have demonstrated that the synchronisation quality is actually improved by the proposed protocol in the long term. The parameters of the synchronisation protocol are then adapted to environmental conditions to achieve the best accuracy under these conditions.

Additionally, we studied various amplitude modulation schemes for beamforming transmissions. We could observe that a collaborative data transmission is possible also at distances of about 200 meters when strong bit guarding schemes are utilised. At shorter distances of about 30 meters the BER observed was neglectable for lower order modulation schemes.

Our studies confirm the feasibility of randomised iterative feedback based carrier synchronisation for beamforming of distributed computation and transmission power restricted devices.

Acknowledgements The authors would like to acknowledge partial funding by the German Academic Exchange Service (DAAD) through the FIT postdoc grant in the project 'Implicit situation awareness of a wireless sensor network utilising channel quality estimations'.

We would further like to acknowledge partial funding by the 'Deutsche Forschungsgemeinschaft' (DFG) for the project "Emergent radio" as part of the priority program 1183 "Organic Computing".

References

1 3GPP. 3rd generation partnership project; technical specification group radio access networks; 3g home nodeb study item technical report (release 8). Technical Report 3GPP TR

25.820 V8.0.0 (2008-03), 2008 March.

2 G. Barriac, Raghuraman Mudumbai, and Upamanyu Madhow. Distributed beamforming for information transfer in sensor networks. In *Proceedings of the third International Workshop on Information Processing in Sensor Networks*, 2004.

3 James A. Bucklew and William A. Sethares. Convergence of a class of decentralised beamforming algorithms. *IEEE Transactions on Signal Processing*, 56(6):2280–2288, June 2008.

4 L.C. Godara. Application of antenna arrays to mobile communications, ii. *Proceedings of the IEEE*, 85(8):1195–1245, 1997.

5 Rayan Merched El Masri, Stephan Sigg, and Michael Beigl. An asymptotically optimal approach to the distributed adaptive transmit beamforming in wireless sensor networks. In *Proceedings of the 16th European Wireless Conference*, 2010.

6 R. Mudumbai, G. Barriac, and U. Madhow. On the feasibility of distributed beamforming in wireless networks. *IEEE Transactions on Wireless communications*, 6:1754–1763, 2007.

7 Raghuraman Mudumbai, D. Richard Brown, Upamanyu Madhow, and H. Vincent Poor. Distributed transmit beamforming: Challenges and recent progress. *IEEE Communications Magazine*, pages 102–110, February 2009.

8 Raghuraman Mudumbai, J. Hespanha, Upamanyu Madhow, and G. Barriac. Scalable feedback control for distributed beamforming in sensor networks. In *Proceedings of the IEEE International Symposium on Information Theory*, pages 137–141, 2005.

9 Raghuraman Mudumbai, J. Hespanha, Upamanyu Madhow, and G. Barriac. Distributed transmit beamforming using feedback control. *IEEE Transactions on Information Theory*, 56(1), January 2010.

10 Raghuraman Mudumbai, Ben Wild, Upamanyu Madhow, and Kannan Ramchandran. Distributed beamforming using 1 bit feedback: from concept to realization. In *Proceedings of the 44th Allerton conference on communication, control and computation*, pages 1020–1027, 2006.

11 Munkyo Seo, Mark Rodwell, and Upamanyu Madhow. A feedback-based distributed phased array technique and its application to 60-ghz wireless sensor network. In *IEEE MTT-S International Microwave Symposium Digest*, pages 683–686, 2008.

12 Stephan Sigg and Michael Beigl. Collaborative transmission in wsns by a (1+1)-ea. In *Proceedings of the 8th International Workshop on Applications and Services in Wireless Networks (ASWN'08)*, 2008.

13 Stephan Sigg and Michael Beigl. Algorithmic approaches to distributed adaptive transmit beamforming. In *Fifth International Conference on Intelligent Sensors, Sensor Networks and Information Processing - Symposium on Theoretical and Practical Aspects of Large-scale Wireless Sensor Networks*, 2009.

14 Stephan Sigg and Michael Beigl. Algorithms for closed-loop feedback based distributed adaptive beamforming in wireless sensor networks. In *Proceedings of the fifth International Conference on Intelligent Sensors, Sensor Networks and Information Processing - Symposium on Adaptive Sensing, Control, and Optimization in Sensor Networks*, 2009.

15 Stephan Sigg, Rayan Merched El Masri, and Michael Beigl. A sharp asymptotic bound for feedback based closed-loop distributed adaptive beamforming in wireless sensor networks. *Transactions on mobile computing*, 2011 (accepted for publication).

16 Stephan Sigg, R.M. Masri, Julian Ristau, and Michael Beigl. Limitations, performance and instrumentation of closed-loop feedback based distributed adaptive transmit beamforming in wsns. In *Fifth International Conference on Intelligent Sensors, Sensor Networks and Information Processing - Symposium on Theoretical and Practical Aspects of Large-scale Wireless Sensor Networks*, 2009.

17 Y. Tu and G. Pottie. Coherent cooperative transmission from multiple adjacent antennas to a distant stationary antenna through awgn channels. In *Proceedings of the IEEE Vehicular Technology Conference*, pages 130–134, 2002.

18 Pramod Viswanath, David N. C. Tse, and Rajiv Laroia. Opportunistic beamforming using dumb antennas. *IEEE Transactions on Information Theory*, 48(6):1277–1294, June 2002.

Web Workload Generation According to the UniLoG Approach

Andrey W. Kolesnikov and Bernd E. Wolfinger

University of Hamburg, Department of Informatics
Vogt-Kölln-Str. 30, 22527 Hamburg, Germany
{kolesnikov,wolfinger}@informatik.uni-hamburg.de

Abstract

Generating synthetic loads which are sufficiently close to reality represents an important and challenging task in performance and quality-of-service (QoS) evaluations of computer networks and distributed systems. Here, the load to be generated represents sequences of requests at a well-defined service interface within a network node. The paper presents a tool (`UniLoG.HTTP`) which can be used in a flexible manner to generate realistic and representative server and network loads, in terms of access requests to Web servers as well as creation of typical Web traffic within a communication network. The paper describes the architecture of this load generator, the critical design decisions and solution approaches which allowed us to obtain the desired flexibility.

1998 ACM Subject Classification C.2.2 [Computer Communication Networks]: Network Protocols – HTTP; C.2.3 [Computer Communication Networks]: Network Operations – Network management; C.2.4 [Computer Communication Networks]: Distributed Systems – Distributed applications; C.4 [Performance of Systems]: Measurement techniques, Modeling techniques, Performance attributes; G.3 [Probability and Statistics]: Distribution functions, Experimental design, Stochastic processes; I.6.5 [Simulation and Modeling]: Model Development – Modeling methodologies.

Keywords and phrases Web workload generation, Web traffic generation, Unified Load Generator, Performance Evaluation, HTTP/1.1

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.49

1 Introduction

Generating load for computer and communication networks in a sufficiently realistic manner represents an important and challenging task. Realistic load generation typically is an indispensable prerequisite to the analysis and prediction of network performance and of quality-of-service (QoS). Moreover, load generation may support functionality testing of network components (on various levels of component utilization) or the testing of the effectiveness of security mechanisms.

Load generation can be required at very different interfaces within a computer network, such as user-/application-oriented interfaces at which sequences of requests have to be generated like requests to a Web server, files to be transferred, sequences of video frames or of digitized voice data to be transmitted, etc. On the other hand, it might also be necessary to generate load for a lower-layer interface within a protocol hierarchy, such as creation of IP packets or of Ethernet frames to be transmitted.

An interesting approach for load generating, from our point of view, is the provisioning of a unified tool which allows one to generate load (i.e. sequences of requests) at an interface which can be chosen by an experimenter dependent on the kind of study he/she is carrying out. This approach has been followed by the authors during the development of the unified

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 49–60

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

load generator `UniLoG` [1]. The basic principle underlying the design and elaboration of `UniLoG` has been to start with a formal description of an abstract load model and thereafter to use an interface-dependent adapter to map the abstract requests to the concrete requests as they are "understood" by the service providing component at the real interface in question. Our earlier research covers the provisioning of adapters for TCP and UDP [2] as well as IP [1] interfaces. Because of the pre-eminent importance of the World Wide Web (WWW) a strong desire emerged to embed the capability into `UniLoG` of generating realistic Web traffic and realistic Web server loads, too.

The following strongly different modeling approaches (MA) are possible and may be useful for various kinds of studies when characterizing HTTP loads and the resulting Web traffic induced by this load:

(MA_1) Realistic description of sequences of requests as they are passed to the HTTP interface $IF_c(HTTP)$ within the Web client. Generating a (realistic) sequence of requests at interface $IF_c(HTTP)$ allows one, with only little expenditure, on the one hand to generate some specific load for a Web server (in terms of client requests of varying complexity) and on the other hand to generate some specific load for the network (in terms of Web traffic of different structure and intensity) [3, 4].

(MA_2) Realistic description of sequences of requests as they are passed, both, in the web client C (as client requests) and in the Web server S (as server responses) to the corresponding HTTP interfaces $IF_c(HTTP)$ and $IF_s(HTTP)$. As server responses are generated as a consequence of client requests, precise modeling here requires coordination of load generation at both interfaces – and this is a tedious task. The concurrent generation of sequences of requests at interfaces $IF_c(HTTP)$ and $IF_s(HTTP)$ replaces the Web server being used in MA_1 by a load generator [5, 6]. Therefore, generation of load here gets much more complicated because a sufficiently realistic model for the Web server is required and load generation in the client and the server have to be coordinated. Moreover, this approach does no longer allow the dedicated loading of a Web server with client requests.

(MA_3) Realistic description of the sequence of IP packets (as observed at the IP interface $IF_s(IP)$ in the Web server) or requests at the TCP service interface ($IF_s(TCP)$ in the server) as they are induced by single or an overlay of client requests to the server. The IP packets or TCP requests result from the transmission of the corresponding server responses. This modeling approach fundamentally differs from MA_1 and MA_2 as it assumes a completely different interface for load description/generation (IP instead of HTTP interface). The approach is useful in cases, when, e.g., streams of IP packets have to be injected into a network in a manner as they would have been induced by single or an overlay of Web server accesses. Most of the currently existing literature on characterization/generation of HTTP loads [7, 8] follows this modeling approach, though, in order to be precise, one would have to consider it as a characterization/generation of IP packet streams. Here also, the dedicated loading of a Web server with client requests is impossible using this approach.

MA_1 is the approach followed by us and presented in this paper, because we want to generate load for the server, too, and not only for the network (which is impossible when using approaches MA_2 or MA_3). Moreover, we want to inject the load directly at the HTTP interface and therefore MA_3 is again not an alternative to the MA_1 approach.

A survey on the literature related with this work follows in Sec. 2. The application of our unified approach to Web workload generation is illustrated in Sec. 3. In Sec. 4, the architecture of the corresponding HTTP adapter along with an algorithm for the allocation

of real requests from a pool of Web sites are explained. The estimation of the relevant workload/traffic characteristics and the construction of a sufficiently large, representative and stable pool of Web sites for load generation are presented in Sec. 5 and Sec. 6. A short summary and an outlook on future work conclude this paper.

2 Related Work

Web workload generators are software tools based either on traces reflecting real Web user sessions or on workload models that are designed and implemented to generate HTTP requests. Floyd and Paxson demonstrated in their study [9] how difficult it is to generate representative Web requests, especially when some particular characteristics in a dynamic Web site should be modeled, and how these characteristics impact on the behaviour of the Web clients.

One of the first studies trying to identify the common characteristics in Web server workloads is the work done by Arlitt and Williamson [10], which used logs of Web server accesses at six different sites (three from university environments, two from scientific research organizations, and one from a commercial Internet service provider). The observed workload characteristics were used to identify the possible strategies for the design of a caching system to improve Web server performance.

Barford and Crovella [4] applied a number of observations of Web server usage to create a realistic Web workload generation tool, called `SURGE` (Scalable URL Reference Generator) which mimics a set of real users accessing a server and generates URL references matching empirical measurements of request and server file size distribution, relative file popularity, embedded file references, temporal locality of reference, and idle periods of individual users. The relevance of these Web workload characteristics as well as their concrete values were identified based on single (nonrecurring) measurements, so that later revisiting done by Williams et al. [11] was required due to emerging Web technologies and a 30-fold increase in overall traffic volume in 2005.

Rolland et al. [8] proposed the open-loop packet-level traffic generator `LiTGen`, which statistically models IP traffic (resulting from Web requests) on a per user and application basis and, in contrary to the recommendations of Paxon and Floyd [9], does not consider such important network and protocol characteristics like RTT, link capacities or TCP dynamics.

`ParaSynTG` [6] is a synthetic trace generator for source-level representation of Web traffic with different characteristics such as document size and type, popularity (in terms of frequency of reference) as well as temporal locality of requests. The tool was designed for the generation of workload traces only (which can be used, e.g., in simulation experiments) and, in the opposite to the design objectives of our Web workload generator `UniLoG`, provides no facilities to generate and to inject real HTTP requests into the network.

`PackMime-NS` [5] is an example for a source-level and `Swing` [7] is an example for a packet-level traffic generation tool, respectively, with a lot of effort spent by the authors on the ability to take into account the network and protocol characteristics (e.g., RTT, link capacities and error rates, dynamic TCP interactions between Web clients and servers). As a consequence, there are not only the Web clients and Web servers that have to be modeled by these solutions, but also the other components of the network under study. Hence, the field of application of these solutions is restricted to network simulation experiments (whereas our solution is targeting performance evaluation studies for real networks and Web servers, too).

A further survey of traffic generation tools (including the solutions for other interfaces, e.g., Ethernet, IP, TCP/UDP) can be found in [12].

3 Application of the UniLoG Approach to Web Traffic Generation

The characteristic behaviour of a user navigating Web sites by means of a browser is presented in Fig. 1 (left). A typical Web site consists of a base object and multiple embedded objects. With the first HTTP request the browser retrieves the base object of the site, parses it and issues subsequent HTTP requests to fetch all embedded objects.

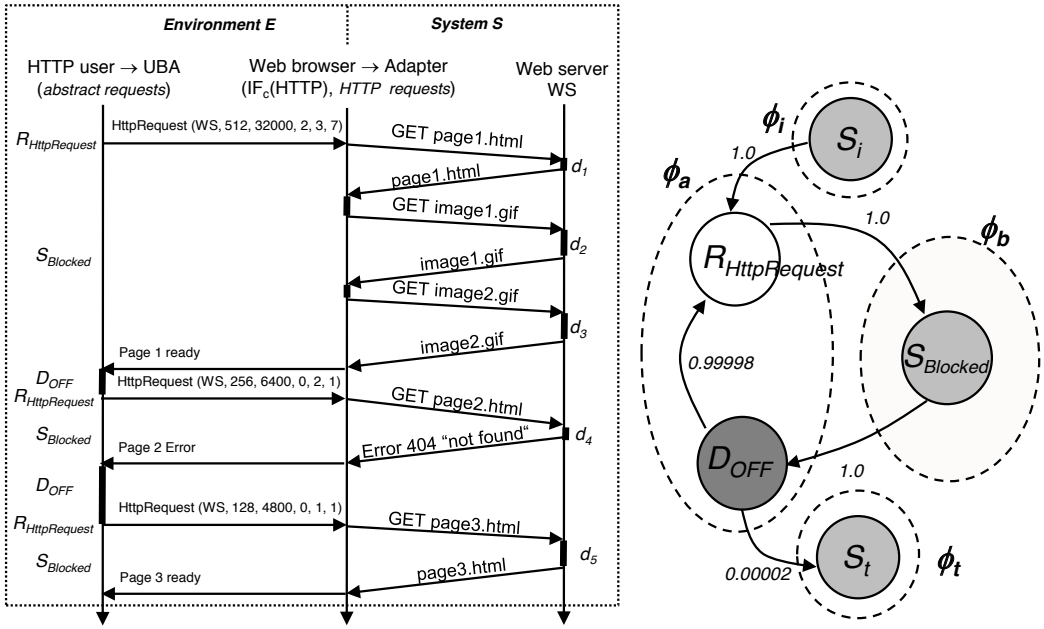


Figure 1 Retrieval of multiple pages and embedded objects from the server `WS` (left) and the corresponding UBA (right).

According to the unified load generation approach presented in [1], a set of HTTP service users which are relevant for the particular modeling task at the interface $IF = IF_c(HTTP)$ in the Web client is considered as an environment E while layers of the protocol stack below the HTTP interface in the Web client, the Web server, and the communication network are considered as being part of the service system S. The workload $L = L(E, S, IF, T)$ is furthermore represented by a sequence of requests $(t_i, r_i), t_i \in T, i = 1, 2, ..., n$ ($t_i \in \mathbb{R}$: arrival time of request r_i at IF, $t_i \leq t_j$ for $i < j$, $n \in \mathbb{N}$) passed by the environment E to the service system S at the interface $IF = IF_c(HTTP)$ during a time interval T.

Each request r_i is characterized by the corresponding abstract request type which is chosen from a set $\mathbb{RT} = \{RT_1, RT_2, \ldots, RT_m\}, m \in \mathbb{N}$, of abstract request types supplied by the experimenter. There is nearly a dozen of different request methods provided by the HTTP/1.1 protocol specification (cf. RFC 2616) for the interaction with the Web server. For the specification of an HTTP workload model, the experimenter can decide to provide different abstract request types for each HTTP request method but he/she would often prefer to define only one single abstract request type (e.g., `HttpRequest`, cf. Fig. 1) addressing HTTP requests in general, and to concentrate on the identification of request parameters which have a significant impact on the workload induced in the server and network. The analysis of the workload characteristics used in a series of studies on Web workload modeling (e.g., [3, 4, 5, 11]) drove the choice of the following request parameters to be provided with our solution by default.

serverName contains either the fully qualified domain name (FQDN) or the IP address of the target Web server and has a direct influence on the generated network traffic matrix. In case the experimenter plans to test a particular Web server or a specific function of a Web service deployed onto it, the exact location of the object to be demanded can be supplied by this parameter (e.g., by means of a URL including a full path and search part, if needed).

replySize is the total amount of response data from the server including the size of all embedded objects. This attribute is not part of the HTTP/1.1 message but it has a significant impact on the load induced by HTTP request.

numberOfEmbeddedObjects specifies the number of embedded objects (e.g. images, links, script objects) in the requested page. HTTP/1.1 allows a single TCP connection to be used for multiple object requests.

complexity is provided to characterize the structural complexity of the page in general and can be specified by means of a complexity class derived from the values of page complexity characteristics `numberOfEmbeddedObjects` and `replySize` which do not consider the induced server load directly.

requestSize is the total amount of data transferred from the client to the Web server, which is affected by the number and the size of requests for the embedded objects in the page. Some of HTTP request headers (e.g. `Accept`, `User-Agent`) may also influence the size of HTTP messages [13].

inducedServerLoad characterizes the amount of time required for the server to respond to the client request and can be specified by means of a server delay class derived from values of particular server delays ($d_1 - d_5$, cf. Fig. 1) induced by requests to the main page and its embedded objects. Client requests may induce further searching, authentication or database retrieval activities on the server side and usually differ significantly in this factor.

The set $\mathbb{RT}$ of abstract request types is constructed by including the relevant request parameters from the list above into the corresponding request types $RT_r = Id(RT_r) \cup \{a_1, a_2, \ldots, a_p\}, RT_r \in \mathbb{RT}, r \in \mathbb{N}, 1 \leq r \leq m$, as abstract request attributes $a_k, k \in \mathbb{N}, 1 \leq k \leq p$, where $p \in \mathbb{N}, p = p(r)$ is the number of attributes in RT_r and $Id(RT_r)$ is its unique name. In this way, the experimenter can define abstract request types with different level of abstraction dependent on the kind of study being carried out. For example, the experimenter can decide to include only one single attribute `inducedServerLoad` into the abstract request type `HttpRequest` in order to analyse the utilization level of some known Web server. In case he/she plans to test a specific function of a Web service/application deployed to that server, the attribute `serverName` containing the full URL of the referenced object is most likely to be added to `HttpRequest`. In order to produce various background loads for the network in terms of Web traffic of different structure and intensity, further attributes like `numberOfEmbeddedObjects` or `replySize` can be included into `HttpRequest`.

For the specification of the possible sequences of requests, the set of relevant HTTP service users is represented by a user behaviour automaton (UBA, cf. Fig. 1). A UBA is an extended finite automaton $U = \{\phi, T_\phi\}$ consisting of the set of macro states $\phi = \{\phi_i, \phi_a, \phi_b, \phi_t\}$ which describe the four typical types of user activity (initialization in ϕ_i, generation of requests in the active macro state ϕ_a, waiting for system reactions in the blocked macro state ϕ_b, termination in ϕ_t) and the set T_ϕ of transitions between these macro states. The macro states are further refined by means of (R)equest-, (D)elay- and (S)ystem-states, which are introduced to model the generation of requests of a particular abstract request type, the

delay times between successive requests and/or system events, and waiting for a certain type of system event to occur, correspondingly (cf. [1]).

Consider a simple UBA for an HTTP user as presented in Fig. 1. The user starts in the S-state S_i of the initialization macro state ϕ_i of the UBA and, after the Internet browser is started completely, becomes active by changing its macro state to ϕ_a. To model the retrieval of Web pages by the user, requests of abstract request type `HttpRequest`(`serverName`, `requestSize`, `replySize`, `numberOfEmbeddedObjects`, `inducedServerLoad`, `complexity`) are generated in the R-state $R_{HttpRequest}$ of ϕ_a. The values of request attributes can be specified by the experimenter by means of constant values, traces of real measurements or various distribution functions. We note that these requests are initially abstract, i.e. they contain attributes (e.g., `replySize`, `inducedServerLoad`) which differ from the attributes of real HTTP messages significantly or do even not exist in HTTP messages at all. Moreover, one `HttpRequest` usually induces more than one real HTTP request to retrieve images, links and script objects embedded in the page as well as objects linked from other servers (e.g., advertisement pop-ups and Web bugs). For example, during the execution of the UBA, an abstract `HttpRequest`(`WS`, `-1`, `32000`, `2`, `-1`, `-1`) represents an instruction at our load generator to invoke a Web page on the server `WS` which contains 2 embedded objects resulting in the total reply size of 32000 byte (a value of -1 means that the corresponding parameter is not used in this request). So, the `UniLoG.HTTP` adapter must issue the corresponding real HTTP requests for the main object and for each of the two embedded objects of the page (cf. Fig. 1).

In the simple UBA presented here, the user resides in the S-state $S_{Blocked}$ of the blocked macro state ϕ_b until the page is retrieved completely (i.e. until the last embedded object is fetched successfully or an error/status code is returned). There are, in general, other possibilities to model this behaviour (e.g., to leave $S_{Blocked}$ immediately after the main object of the page is loaded). After a user-dependent delay time ("think time"), modeled in the D-state D_{OFF}, the next page is accessed by the user (by following a link on the current page or by entering a URL in the address line of the browser directly) and the corresponding `HttpRequest` is generated in $R_{HttpRequest}$. The UBA is executed until the S-state S_t of the termination macro state ϕ_t is reached or the upper limit of the time interval T specified by the experimenter for load generation is exceeded.

It should be noted, that the experimenter can specify advanced Web workload characteristics by means of a UBA, which are not explicitly provided by the set of abstract request parameters. For example, the user think time (which is an important property of Web traffic to capture its bursty nature [4]) can be specified in D-states as interarrival time between subsequent requests `HttpRequest`. Page popularity (defined as relative number of requests made to individual pages on the server) can be specified, e.g., by means of a frequency or Zipf-like distribution for the `serverName` parameter (in this case, containing the full URL of the page to be referenced) of `HttpRequest`. Temporal locality of page requests (referring the likelihood that, once a page has been requested, it will be requested again in the near future) can be characterized by means of a distribution of stack distances [6] for the `serverName` parameter (again, containing the full URL of the page to be referenced) of `HttpRequest` in the request sequence to be generated by our load generator.

4 Allocation of Real HTTP Requests

The Unified Load Generator `UniLoG`, as presented in [1], incorporates a formal automata-based load specification technique and exhibits a distributed architecture to provide a high

degree of flexibility in generating traffic mixes of various structure and intensity for different scenarios. The experimenter can define workload models in form of a UBA, and distribute the parameterized UBA (PUBA) to the load generating nodes (load agents) involved in the experiment, which can be controlled by the experimenter from one central point in the network (management station, cf. Fig. 2). In each load agent, the generator component (GAR) for the PUBA execution and one or many adapters (e.g. `UniLoG.HTTP`) for the allocation of the corresponding real requests are installed.

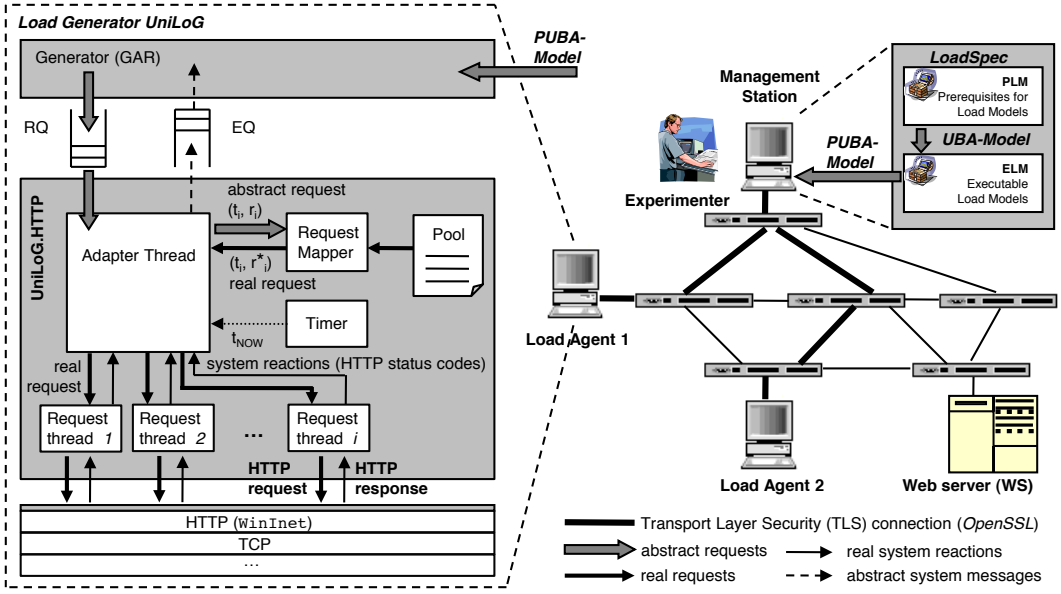


Figure 2 `UniLoG` architecture and the main components of the `UniLoG.HTTP` adapter (left).

The key components of the `UniLoG.HTTP` adapter are presented in the left part of Fig. 2. The main adapter thread consumes abstract requests (t_i, r_i) which are generated by the generator thread (GAR) in R-states of the current PUBA and inserted into the request queue RQ. For each abstract request the adapter invokes the request mapper which is responsible for the allocation of the correspondent real HTTP request from the pool. Each HTTP request in the pool is represented by a tuple $e_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,p}, v^*_{i,1}, v^*_{i,2}, \ldots, v^*_{i,r}), i \in \mathbb{N}, 1 \leq i \leq N$. In such a tuple, $v^*_{i,1}, \ldots, v^*_{i,r}$ denote the values of r real request attributes required to build a request which is conform to HTTP/1.1 (like HTTP request method `requestMethod`, server name `serverName` and port `serverPort`, object name `objectName`, and the optional data `optData` of length `optLength` sent with POST requests). $v_{i,1}, \ldots, v_{i,p}$ denote the corresponding values of p abstract request parameters (defined in Sec. 3), which can be measured manually by the experimenter or estimated by the adapter automatically (see Sec. 5) and used to query the pool for real requests which match as good as possible the current abstract request. Finally, N denotes the pool size, i.e. the number of its elements.

To allocate the best matching real request (t_i, r^*_i), the request mapper starts with the abstract parameter a_1 of the highest priority (the priority of the abstract request parameters can be specified by the experimenter in PUBA) and creates a candidates list consisting of requests from the pool which exhibit the minimal distance between a_1 and $v_{i,1}, 1 \leq i \leq N$. In the next step, the abstract parameter a_2 with the second highest priority is selected and only the requests with the minimal distance between a_2 and $v_{i,2}, 1 \leq i \leq N$ are kept on the candidates list. In case of string parameters, especially for `serverName`, an exact match in

place of the minimal distance candidates is required. This procedure is repeated until only one candidate is left or all of the p abstract request parameters are processed. In case there are more than one real requests left on the candidates list after the last step, one of them is selected randomly.

The request allocation algorithm is flexible in the number and types of abstract request parameters used for the pool query. Its complexity is determined by the number v of required comparisons which yields to $v = N \cdot p$ for the worst case when all N real requests from the pool match each of the p parameters of the current abstract request.

The allocated real request (t_i, r_i^*) is prepared for execution by creating a new request thread and establishing the connection to the Web server, if needed. Thereafter, the adapter thread waits until the specified request injection time t_i is reached and resumes the request thread for execution. HTTP status codes returned by the request thread represent the system messages which are inserted by the adapter thread into the event queue EQ.

As a proof of the presented concept, the first prototype of the `UniLoG.HTTP` adapter for Windows was developed in [14]. Meanwhile, this prototype has been significantly improved to imitate the behaviour of a real browser more closely and, finally, it has been integrated in the `UniLoG` load generator. `UniLoG.HTTP` makes use of the `WinInet` library for the management of Web server connections as well as for the formation and injection of HTTP requests into the network. The choice of `WinInet` is motivated by the fact that it is internally used by the Internet Explorer (IE) browser itself and thus enables the adapter to imitate the behaviour of this browser in a very realistic manner.

For this reason, we applied the same values for the `User-Agent` and `Accept` request header fields as they are used by the IE 7.0 [1] during our `HttpOpenRequest` calls for the formation and injection of HTTP requests in the adapter. Furthermore, similar to [13], a series of `INTERNET_FLAG`s (`NO_CACHE_WRITE`, `RELOAD`, `PRAGMA_NO_CACHE`, `KEEP_CONNECTION`) has been specified in `HttpOpenRequest` calls in order to instruct the adapter to retrieve the requested objects from the original server and not from the local cache or proxy, using, if possible, a keep-alive connection. These measure should help to ensure that the adapter induces Web traffic with characteristics as specified by the values of abstract request parameters in the pool entries.

5 Estimation of Values for Abstract Request Parameters

It becomes apparent, that a sufficiently large pool of Web requests with properly estimated values of their abstract parameters is required to take `UniLoG.HTTP` in operation. Furthermore, the pool should be kept up-to-date to ensure that the characteristics of the Web traffic induced by `UniLoG.HTTP` correspond to the values of abstract parameters specified in the pool entries.

In the measurement approach applied in [15], a series of Firefox browser sessions were traced with `tcpdump` and analyzed to extract various characteristics of worldwide top 500 popular web sites. In this way, all components of the requested page are considered, providing very realistic Web server workload and traffic characteristics. However, this method leads to a sufficient programming effort for combining a series of external tools to fully automate the measurements, whereas the required measurement points can be accessed by us in the `UniLoG.HTTP` source code directly. Furthermore, `UniLoG.HTTP` is aimed at imitating the

[1] These values were extracted from the registry key `HKLM/Software/Microsoft/Windows/CurrentVersion/InternetSettings`

behaviour of the real browser to some extent and is not able yet to interpret all existing types of embedded objects (e.g. JavaScript or CSS objects are loaded but not interpreted). Therefore, we decided to integrate the parameter estimation function for our pool immediately into the adapter and allowed it to be turned on and off by the experimenter. The values of abstract parameters for each pool entry are estimated during the retrieval of the correspondent Web page and its embedded objects from the server according to the following rules.

numberOfEmbeddedObjects (o) the adapter retrieves the main object of the requested page specified by the concatenation of abstract `serverName` and real `objectName` parameters and parses its content to recognize the embedded images, links, and script objects to retrieve them by means of consequent requests from the same server. Objects linked from other servers are not loaded and hence not considered in o at this moment.

replySize (s) is computed as the total amount of data (in kbyte) loaded by the adapter during the `InternetReadFile` calls for the main page and all its embedded objects.

complexity class $C \in \{C_1, \ldots, C_9\}$ for the page is derived from its estimated values o and s using class boundaries o_1, o_2 and s_1, s_2 which can be supplied by the experimenter for the `numberOfEmbeddedObjects` and `replySize` parameters, correspondingly (cf. Fig. 3).

requestSize is calculated as the total amount of data (in byte) sent by the adapter to the server considering requests for the main page, all its embedded objects and the optional data for POST requests. The mean size of GET requests issued by the IE 7.0 browser was estimated to 612 byte in [16].

inducedServerLoad class $D \in \{$`Immediately`, `Fast`, `Slow`, `Annoyingly slow`$\}$ is derived from the delay times $d_i, i = 1, \ldots, o$, induced on the server by requests to the embedded objects of the page. The delay time induced by the main object is not considered to exclude the time spent for potential requests to the DNS server. Each d_i is estimated as a difference between request execution time a_i (cf. Fig. 3) and the server round trip time (RTT_s), estimated either from the TCP handshake in the beginning of the HTTP session or by means of ICMP echo messages exchanged between our adapter and the server. The total server delay d induced by the page is estimated as the sum of d_i in case HTTP pipelining is not used (and d_i can, therefore, not "overlap" on the server) or by the maximum of d_i, otherwise. Finally, the appropriate `inducedServerLoad` class D is chosen considering the boundary values supplied by the experimenter for the server delay time d.

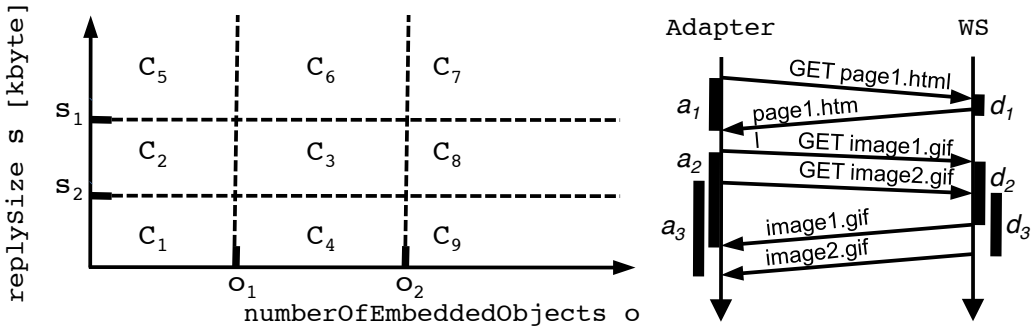


Figure 3 Formation of page `complexity` and `inducedServerLoad` classes.

A series of IE 7.0 browser sessions for a rather small sample of 61 exemplarily chosen home pages were captured and analyzed in [16] using `WireShark` [17] to extract the values

of abstract parameters according to the rules specified above. The IE 7.0 browser was instrumentalized in a manner, that only the object types supported by the adapter were loaded. The results exhibited only marginal difference from the parameter values estimated by the adapter.

6 Construction of the Pool from Popular Web Sites

As already stated in Sec. 5, a sufficiently large, representative and stable pool is indispensable for the generation of realistic and representative Web workloads with `UniLoG`. In this section, we present the construction of a pool populated with home pages extracted from Alexa Web statistics service [18] which ranks Web sites according to their popularity among a large number of users. The representativeness of the Alexa's sample is though restricted to a set of users which were able to install the Alexa's toolbar required to gather the statistics from the users. A total of 1 million most popular sites available for daily download from Alexa's Web site seems to be large enough to construct a comprehensive pool for our load generator, and the restriction to include only the home pages' URLs in the ranking guarantees for some degree of stability in the resulting pool (in respect of reachability of servers and validity of the corresponding pool entries).

The pool was first populated by the top 1000 home pages from the Alexa ranking downloaded from the Alexa Web site on 3 May, 2010, and the values of abstract request parameters were initially set to zero for all pages. The values of real request attributes `serverName`, `serverPort`, `objectName`, `requestMethod`, `optLength` and `optData` were set to the host part of the particular page, `80`, `/`, `GET`, `0` and `NULL`, correspondingly, as the Web sites captured in the Alexa ranking are home pages provided by default on the TCP port 80 by means of a GET request method, and no supplemental data is to be sent to the corresponding Web server by means of POST requests.

Next, `UniLoG.HTTP` was instructed to traverse the pool, retrieve every page and update the values of its abstract request parameters. For this reason, we used the simple UBA presented in Sec. 3, where the values of the `serverName` attribute of every abstract `HttpRequest` to be generated in the R-state $R_{HttpRequest}$ were specified (using the trace parameterization facility of `UniLoG`) to be taken from the column in the pool file, which contains the unique host names of Alexa's home pages. In the adapter, each abstract `HttpRequest` induced a retrieval of the corresponding Web page, whereby the values of its abstract request parameters were estimated and, finally, stored back into the pool.

During the execution of the pool update procedure in May 2010 using the T-Online DSL connection with a maximum of 6 Mbit/s downstream and 640 kbit/s upstream, a total of 979 pool entries were updated. 13 servers were unavailable due to the missing subdomain or Web page to be loaded (as the Alexa ranking contains some pages used only for Webbugs or user tracking). Another 8 sites caused an error during the download by the adapter due to violations of the XHTML or W3C standards in their source code.

A total of 328 servers did not response to ICMP messages making the estimation of the corresponding RTT impossible. The `inducedServerLoad` class for such servers could not be specified ((-1) = n.a., cf. Fig. 4). The mean values of the abstract request attributes `replySize`, `numberObEmbeddedObjects`, `inducedServerLoad`, and `complexity` among all 979 pool entries were estimated to 905.6 kbyte, 38, 2 (delay class "Slow"), and 5 (complexity class C_5), correspondingly. In this exemplarily pool update, we used the threshold values $o_1 = 10$ and $o_2 = 20$ for `numberObEmbeddedObjects` as well as $s_1 = 50$ kbyte and $s_2 = 150$ kbyte for the `replySize` to define the `complexity` class boundaries. For a smaller sample

of Web sites, which were used for testing the implementation of the pool update function presented in Sec. 5, these threshold values provided `complexity` classes with nearly identical number of servers in each class. Applying the same threshold values to Alexa's top 1000 pages emerged that nearly a half of home pages were assigned to the `complexity` class C_7, i.e. these pages contain more than 20 objects and more than 150 kbyte of data (cf. Fig. 4). We note that the experimenter can easily adjust the threshold values in order to obtain `complexity` classes C_1 - C_9 which are suitable for the particular experiment (e.g., classes with uniformly distributed number of servers).

inducedServerLoad d	# servers
-1= n.a. $(d < 0ms)$	328
0 = Immediate $(0ms \leq d < 10ms)$	159
1 = Fast $(10ms \leq d < 30ms)$	66
2 = Slow $(30ms \leq d < 100ms)$	120
3 = Annoyingly slow $(d \geq 100ms)$	306

complexity(numberOfEmbeddedObjects o, replySize s [kbyte])	# servers
$C_1 (o \leq 10 \cap s \leq 50)$	149
$C_2 (o \leq 10 \cap 50 < s \leq 150)$	73
$C_3 (10 < o \leq 20 \cap 50 < s \leq 150)$	65
$C_4 (10 < o \leq 20 \cap s \leq 50)$	20
$C_5 (o \leq 10 \cap s > 150)$	26
$C_6 (10 < o \leq 20 \cap s > 150)$	93
$C_7 (o > 20 \cap s > 150)$	504
$C_8 (o > 20 \cap 50 < s \leq 150)$	44
$C_9 (o > 20 \cap s \leq 50)$	5
n.a. (-1)	0

Figure 4 Number of servers in `inducedServerLoad` and `complexity` classes (cf. [16]).

In this way, the experimenter can create a comprehensive pool of representative Web sites without a lot of effort and in short time. The execution of the pool update procedure immediately before the actual experiment in the particular testbed is a very recommended measure to ensure that the characteristics of Web workloads produced by `UniLoG.HTTP` correspond to the parameter values specified in the pool.

7 Summary and Outlook

In this paper, we presented the application of the `UniLoG` approach to the realistic and rather representative Web workload and traffic generation. Our approach provides different levels of abstraction in the user behaviour models used for load generation. The presented `UniLoG.HTTP` adapter is able to induce Web workloads with characteristics corresponding to the values of abstract parameters of real Web sites from the pool, which is required and provided by our solution. The construction of the sufficiently large, representaive and stable pool of popular Web sites has been presented by us in Sec. 6. The authors hope to have made the next step forward to the combination of Web traffic measurements and generation into a single, coherent approach by this work. The formal load specification technique being used in the `UniLoG` load generator provides for a very precise definition of timing and sequence of resulting requests composing the load being generated, which is indispensable, e.g., to reproduce experiments with a number of predetermined background loads. Moreover, the approach presented in this work is applicable to other interfaces (i.e. TCP, IP, Ethernet) of the protocol stack.

Future work can be directed at the provision of a set of predefined UBA-models for different types and number of HTTP service users. Moreover, the existing adapter prototype can be extended to support a larger set of types for embedded objects or to use a real Web browser (via its remote control interface) for generation and injection of HTTP requests

instead of imitating its behaviour. From the point of view of the authors, these arguments demonstrate the high degree of flexibility and good extensibility of the `UniLoG` approach and support its usage for Web workload generation.

Acknowledgement The authors would like to thank André Gahr and Steffen Jahnke for their valuable contributions to this work in the scope of their diploma theses.

References

1 A. Kolesnikov, M. Kulas, "Load Modeling and Generation for IP-Based Networks: A Unified Approach and Tool Support", Proc. of MMB'2010, Essen, March 15-17, 2010, pp. 91–106.
2 A. Kolesnikov, "Konzeption und Entwicklung eines echtzeitfähigen Lastgenerators für Multimedia-Verkehrsströme in IP-basierten Rechnernetzen", Proc. of Echtzeit'08, Boppard am Rhein, November 27-28, 2008, pp. 91–100.
3 R. Pena-Ortiz et al., "Dweb model: Representing Web 2.0 dynamism", Computer Communications, vol. 32, no. 6, April 2009, pp. 1118–1128.
4 P. Barford, M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", Proc. of SIGMETRICS'98, June 24-26, Madison, 1998, pp. 151–160.
5 J. Cao, W. Cleveland, Y. Gao, K. Jeffay, F. Smith, M. Weigle, "Stochastic Models for Generating Synthetic HTTP Source Traffic", Proc. of INFOCOM'04, Hong Kong, March 7-11, 2004, vol. 3, pp. 1546–1557.
6 R. El Abdouni Khayari, M. Rücker, A. Lehmann, A. Musovic, "ParaSynTG: A Parameterized Synthetic Trace Generator for Representation of WWW Traffic", Proc. of SPECTS'08, Edinburgh, June 16-18, 2008, pp. 317–323.
7 K. Vishwanath, A. Vahdat, "Swing: Realistic and Responsive Network Traffic Generation", IEEE/ACM Transactions on Networking, vol. 17, no. 3, June 2009, pp. 712–725.
8 C. Rolland, J. Ridoux, B. Baynat, "Catching IP traffic burstiness with a lightweight generator", Proc. IFIP NETWORKING'07, Atlanta, Mai 14-18, 2007, pp. 924–934.
9 S. Floyd, V. Paxson, "Difficulties in Simulating the Internet", IEEE/ACM Transactions on Networking, vol. 9, no. 4, August 2001, pp. 392–403.
10 M. Arlitt, C. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications", IEEE/ACM Transactions on Networking, vol. 5, no. 5, October 1997, pp. 631–645.
11 A. Williams, M. Arlitt, C. Williamson, K. Barker, "Web workload characterization: Ten years later", Springer, Heidelberg, 2005.
12 S. Avallone et al., "Performance Evaluation of an Open Distributed Platform for Realistic Traffic Generation", Performance Evaluation, vol. 60, 2005, pp. 359–392.
13 J. Wei, C. Xu, "sMonitor: A Non-Intrusive Client-Perceived End-to-End Performance Monitor of Secured Internet Services", Proc. of USENIX'06, Boston, May 30 - June 3, 2006, pp. 243–248.
14 A. Gahr, "Bereitstellung und Einsatz von Modellen und Werkzeugen zur Erzeugung realitätsnaher synthetischer Webserver-Lasten", Diploma Thesis, University of Hamburg, 2008.
15 J. Charzinski, "Traffic Properties, Client Side Cachability and CDN Usage of Popular Web Sites", Proc. of MMB & DFT 2010, Essen, March 15-17, 2010, pp. 136–150.
16 S. Jahnke, "Last-/Verkehrsmessungen und realitätsnahe Lastgenerierung für Web-Server-Zugriffe", Diploma Thesis, University of Hamburg, 2010.
17 WireShark network protocol analyzer, http://www.wireshark.org (16.09.10).
18 Alexa service, http://s3.amazonaws.com/alexa-static/top-1m.csv.zip (16.09.10).

Improving Markov-based TCP Traffic Classification

Gerhard Münz[1], Stephan Heckmüller[2], Lothar Braun[1], and Georg Carle[1]

1 **Network Architectures and Services,**
Technische Universität München, Germany
{muenz|braun|carle}@net.in.tum.de
2 **Telecommunications and Computer Networks,**
Universität Hamburg, Germany
heckmueller@informatik.uni-hamburg.de

Abstract

This paper presents an improved variant of our Markov-based TCP traffic classifier and demonstrates its performance using traffic captured in a university network. Payload length, flow direction, and position of the first data packets of a TCP connection are reflected in the states of the Markov models. In addition, we integrate a new "end of connection" state to further improve the classification accuracy. Using 10-fold cross validation, we identify appropriate settings for the payload length intervals and the number of data packets considered in the models. Finally, we discuss the classification results for the different applications.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.61

1 Introduction

Traffic classification based on port numbers has reached its limits since a one-to-one mapping between port and application does not exist for many modern network services. Examples are peer-to-peer applications, which often allocate ports dynamically, and web applications, such as streaming servers, which run on top of HTTP on the same port 80 or 443. Classification based on deep packet inspection (DPI) involves pattern matching on packet payload which is computationally intensive. Moreover, DPI is restricted to unencrypted traffic. Therefore, network operators are interested in statistical classification methods which consider traffic characteristics beyond packet contents.

The common assumption of statistical traffic classification is that the communication behavior of an application influences the resulting traffic. Hence, by observing appropriate traffic properties, it should be possible to distinguish applications with different behaviors. In the literature, the application of machine learning techniques to the traffic classification problem is very popular. Examples can be found in the survey paper of Nguyen and Armitage [8]. Similarly, multiple approaches to traffic classification are evaluated and compared in the paper of Kim et al. [6]. Among other approaches, the authors consider various machine learning algorithms with Support Vector Machines being the most accurate. In the article of Este et al. [4], the amount of information carried by various traffic features is investigated. The authors evaluate the *mutual information* of the traffic features and the application type. They conclude that the packet sizes exhibit the highest amount of information concerning the application type for multiple data sets.

In a previous paper, we present a novel TCP traffic classification approach using observable left-right Markov models [7]. This work is motivated by Estevez-Tapiador et al. who deploy observable ergodic Markov models for detecting anomalies in TCP connections [5].

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 61–72

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In contrast to Estevez-Tapiador, we do not consider TCP flags but payload length, flow direction, and position of the first data packets of a TCP connection. Our evaluation shows that taking into account the first four data packets per connection suffices to achieve very good classification results. Evaluating additional data packets allow us to further increases the accuracy. As an advantage to hidden Markov models (HMMs), which have been proposed for traffic classification by Wright et al. [12] and Dainotti et al. [2], the utilization of observable Markov models drastically simplifies the estimation of model parameters as well as the classification of a new connection.

In Section 2 of this paper, we present an improved variant of the Markov-based traffic classifier which uses an additional "end of connection" state to reflect the termination of TCP connections with small numbers of data packets. In Section 3, we apply this approach to TCP traffic which was captured in the Munich Scientific Network and labeled with help of OpenDPI [9], an open-source signature-based traffic classification library. Different parameter settings are compared based on the average accuracy achieved in a 10-fold cross-validation. The most appropriate model parameters are then used to determine the classification results for a realistic traffic mix. All in all, the results confirm earlier findings that our approach yields very high classification accuracies even when dealing with real-world traffic data.

2 Markov-based TCP Traffic Classifier

2.1 High-Level Description

Our traffic classifier considers the packet sequence at the beginning of a TCP connection as the output of an observable Markov model (or Markov chain). In contrast to HMMs used in other traffic classification approaches [2, 12], each packet in the sequence can be directly linked to a specific state in the Markov model. Hence, given a packet sequence, we can directly infer the initial state from the first packet, as well as all further state transitions from the following packets. If the Markov model's initial state and transition probabilities are known, we can calculate the likelihood of the packet sequence.

Our TCP traffic classifier maintains multiple Markov models with identical states but different initial state and transition probabilities. Each of these models reflects the traffic characteristics of a different application. Given the sequence of the first data packets of a TCP connection, the likelihood is calculated for all available Markov models. The classifier then assigns the TCP connection to the application for which the corresponding Markov model yields the maximum likelihood.

2.2 Considered Packet Attributes

The accuracy of the classification depends very much on the considered packet attributes. As a simple example, we could take into account the flow direction of the packets. There are only two possible directions, namely from the client which initiates the TCP connection to server and vice versa. Hence, we can model this attribute in a Markov model with only two states. In fact, the flow direction of the packets is an important aspect of the communication pattern of an application. Nevertheless, this packet attribute alone is not sufficient to accurately distinguish different applications.

In addition to the flow direction, we experimented with further packet attributes, such as the packet length and the TCP flags, and compared the resulting classification accuracies [1]. We found that excellent classification results can be achieved if payload length (equaling the

TCP segment size), direction, and position of the first data packets within a TCP connection are considered in the states of the Markov model [7]. Data packets are those packets which contain at least one byte of payload. All other packets (such as handshake packets and empty acknowledgment packets) are ignored because their transmission is usually triggered by the TCP layer and not by the application.

The payload length of the data packets is quantized into a few intervals in order to map the packet attributes to a reasonably small number of states. We denote these intervals as sets of interval boundaries, i.e. $B_i = \{b_1, \ldots, b_m\}$. Boundary sets with m members lead to a mapping of the packet lengths onto $m+1$ intervals: $[1; b_1], [(b_1+1); b_2], \ldots, [(b_m+1); \infty)$. Another measure to keep the number of states small is to restrict the number of data packets considered in the Markov models. The number and boundaries of the payload length intervals as well as the number of data packets need to be set appropriately to obtain good classification results. We discuss these parameterization issues in Section 3.3.

2.3 New "end of connection" State

In the following, we present an improved variant of our traffic classifier. In addition to the states reflecting packet attributes, we add a new state which is reached if a connection terminates before the maximum number of data packets considered in the Markov models is transmitted. The classification of such already ended connections is useful for collecting traffic statistics, characterizing network hosts, etc. As we will show in Section 3.3, including the "end of connection" (EOC) state leads to an improvement of the classification accuracy.

The new EOC state is highlighted in Figure 1 which displays the example of a Markov model considering the first four data packets of a connection. In the example, the payload length is quantized to four intervals $I_1, \ldots, I_4$; the directions from client to server and server to client are indicated as $C \Rightarrow S$ and $S \Rightarrow C$. Hence, the total number of states is 33. Transitions to the EOC state are possible from the first, second, and third data packet.

2.4 Model Estimation and Storage

Let's assume that $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ is the set of states of the Markov model. Then, the initial state probabilities can be written as vector $\Pi = (\pi_{\sigma_1}, \ldots, \pi_{\sigma_n})$ where π_{σ_i} is the probability that the first packet has the attributes described by σ_i. The transition probabilities form an $n \times n$ matrix $A = \{a_{\sigma_i,\sigma_j}\}$ where a_{σ_i,σ_j} specifies the probability of a transition from σ_i to σ_j.

The initial state and transition probabilities are estimated from labeled training data, i.e. TCP connections for which the application is known. For every application, the probabilities are estimated with the following equations:

$$\pi_{\sigma_i} = \frac{F_0(\sigma_i)}{\sum_{k=1}^{n} F_0(\sigma_k)} \quad ; \quad a_{\sigma_i,\sigma_j} = \frac{F(\sigma_i,\sigma_j)}{\sum_{k=1}^{n} F(\sigma_i,\sigma_k)} \tag{1}$$

$F_0(\sigma_i)$ is the number of initial packets in the training data matching the attributes defined by σ_i. $F(\sigma_i, \sigma_k)$ is the frequency of transitions from packets described by σ_i to packets described by σ_k. In order to obtain good estimates, a sufficiently large set of TCP connections is needed which is representative for the traffic to be classified later.

In our case, only states representing data packets at the first position may be initial states. Furthermore, transitions only occur from one data packet to the next or alternatively to the EOC state. Hence, maintaining an $n \times 1$ initial state probability vector Π and an $n \times n$ transition matrix A is inefficient because most of the elements are zero probabilities

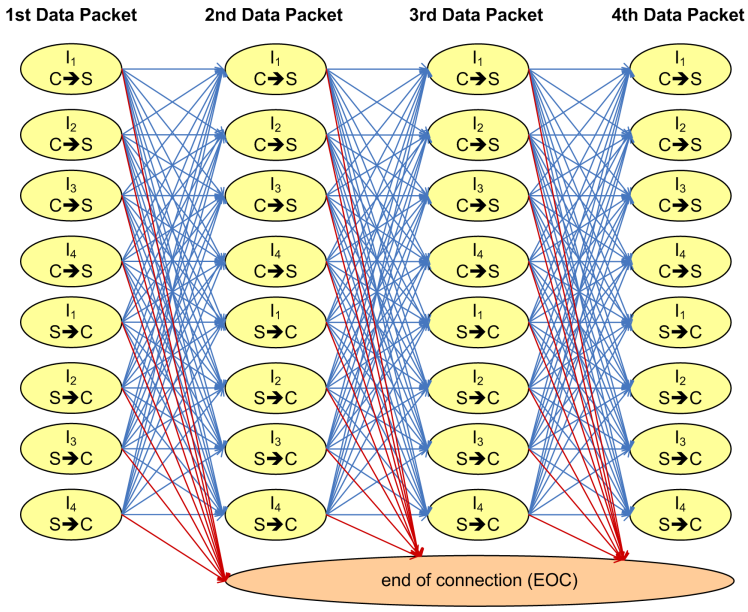


Figure 1 Markov model of the improved classifier

by definition. Due to the regular structure of the Markov model, the probabilities can be stored in a much more memory efficient way. The elements in Π can be limited to the states defining attributes of the first data packet. If l is the number of data packets considered, the transition probabilities can be saved in $(l-1)$ small transition matrices $A_t, t = 1, \ldots, (l-1)$, each including only transitions from packet position t to $t+1$. As an example, the probabilities of the Markov model given in Figure 1 can be stored in a Π vector of length 8 and three 8×9 transition matrices A_t.

Rare initial states or transitions are often not included in the training data. If such an initial state or transition is observed in a TCP connection to be classified, the likelihood of the Markov model drops to zero and disqualifies the associated application. This may lead to wrong classification results. It may also happen that the connection cannot be assigned to any application because the packet sequence has a zero likelihood for all Markov models. For this reason, we replace all zero probabilities of logically possible initial states and transitions by a positive value ϵ which is smaller than any of the estimated non-zero probabilities. Afterwards, we normalize the initial state probabilities and the transition probabilities such that $\sum_i \pi_{\sigma_i} = \sum_j a_{\sigma_i,\sigma_j} = 1$ for all models.

If a TCP connection can only be classified because of multiple zero probabilities replaced by ϵ, it does not really seem to match the traffic characteristics of that application. Therefore, we set a lower threshold h for the model likelihood which corresponds to a small number of ϵ transitions. If this threshold is not exceeded for any application, the connection remains unclassified. Throughout this paper, we use $\epsilon = 10^{-5} = 0.001\%$ and $h = \epsilon^3 = 10^{-15}$.

Concerning the complexity of model estimation, the computation of initial state and transition probabilities using equations (1) requires counting the frequency of initial packet attributes and transitions. The computational effort linearly depends on the number of data packets considered and the number of connections in the training data. Compared to other model types, this is very time-efficient.

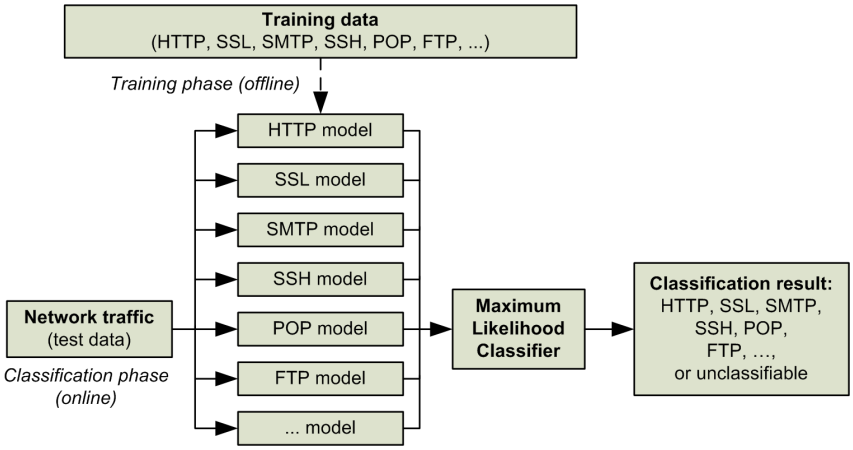


Figure 2 Traffic classification using Markov models

2.5 Classification

During the classification phase, it is beneficial to calculate the log-likelihood instead of the likelihood of the Markov models since this replaces multiplications by additions. Given the first l data packets of a TCP connection $O = \{o_1, o_2, \ldots, o_l\}$, the log-likelihood for this connection is calculated for all Markov models $M^{(k)}$ with $\Pi^{(k)} = (\pi^{(k)}_{\sigma_1}, \ldots, \pi^{(k)}_{\sigma_n})$ and $A^{(k)} = \{a^{(k)}_{\sigma_i,\sigma_j}\}$ using the following equation:

$$\log \Pr\left(O|M^{(k)}\right) = \log\left(\pi^{(k)}_{o_1} \prod_{i=1}^{l-1} a^{(k)}_{o_i,o_{i+1}}\right) = \log \pi^{(k)}_{o_1} + \sum_{i=1}^{l-1} \log a^{(k)}_{o_i,o_{i+1}} \tag{2}$$

If the connection contains less than l data packets (i.e. fewer data packets than considered in the Markov models), the likelihood calculation stops after adding the logarithm of the transition probability to the EOC state. The classifier finally selects the application for which the log-likelihood is the largest. The overall classification process is illustrated in Figure 2. The block diagram shows how the Markov models obtained during the training phase are used to classify new TCP connections.

The computational effort necessary to classify a TCP connection is determined by the computation of the log-likelihood which has to be calculated for every Markov model using equation (2). This effort linearly depends on the number of models and on the length of the state sequence which is bounded by the number of data packets considered in the models.

Other statistical traffic classification methods typically require more complex calculations. This is particularly true for HMMs where emission probabilities have to be considered in addition to transition probabilities.

3 Evaluation

3.1 Training and Test Data

In order to evaluate our traffic classifier, we monitored traffic at the gateway which connects the Munich Scientific Network to the German National Research and Education Network (DFN). For our evaluation, we captured approximately six hours of IPv4/TCP traffic from

Table 1 Application labels provided by OpenDPI

Application label	Connections	Percentage	Distinct server ports
http	1,136,359	64.48%	81 (mainly ports 80, 8080)
ssl	158,331	8.98%	22 (mainly ports 443, 993, 995)
smtp	150,173	8.52%	5 (mainly ports 25, 587)
flash	92,273	5.24%	6 (mainly ports 1953, 80)
ssh	22,087	1.25%	3 (mainly port 22)
http+flash	20,030	1.14%	4 (mainly port 80)
pop	7,390	0.42%	1 (port 110)
ftp	6,232	0.35%	6,035 (73 connections to port 21)
windowsmedia	4,974	0.28%	1 (port 80)
imap	4,367	0.25%	1 (port 143)
dns	2,499	0.14%	1 (port 53)
bittorrent	1,735	0.10%	115 (e.g. ports 8080, 80, 6879)
rdp[a]	1,047	0.06%	1 (port 3389)
other	4,850	0.14%	38 (mainly 80)
unknown	150,013	8.51%	5295 (e.g. ports 10050, 2703)

[a] remote desktop protocol

and to the class B network of our computer science department. Traffic was captured in one chunk on a busy weekday. We modified the intrusion detection system Bro [10] to classify the captured packets into different TCP connections and to extract the corresponding sequences of payload length and flow direction tuples. Individual TCP packets not belonging to any valid TCP connection and TCP connections without any data packet were filtered out. In total, we extracted packet attributes of 1,762,360 TCP connections, each containing at least one data packet.

In addition to Bro, we applied OpenDPI [9] to obtain an application label for each TCP connection. Since signatures can only be found in unencrypted traffic, connections of applications running on top of SSL or TLS are identically labeled *ssl*. Similarly, all connections of applications making use of SSH are labeled *ssh*. OpenDPI was applied to all packets, which sometimes led to the situation that the label returned by OpenDPI changed in the middle of the connection. Most frequently, this happened for TCP connections which were first labeled *http* and later *flash* (20,300 occurrences), which we combined in a new label *http+flash*. Other types of label changes were observed for 1,544 connections which we all assigned to the label *other* (cf. below).

Table 1 shows the number of TCP connections and the number of distinct server port numbers per application label. Applications with less than 1000 connections are not shown individually but summed up as *other*. It is not surprising that 64.48% of the TCP connections are labeled *http* because HTTP traffic dominates in the Internet today. Although most connections are directed to well-known server ports, we observe a certain number of connections with non-standard server ports. Traffic labeled *ftp* mostly includes data connections of FTP sessions. These connections are typically established from the FTP server to an ephemeral port at the FTP client, which is accounted as server port (from the TCP perspective) in the table. In addition, a small amount of FTP control connections is present. 8.51% of all connections could not be classified by OpenDPI and are labeled *unknown*. Almost 28% of these connections are established between two identical endpoints, one of them run-

ning an unknown service on port 10050. More than 11% are directed to port 2703 which is commonly used by Razor servers hosting databases with signatures for spam detection [11].

We restrict our classifier to application with more than 1000 connections in our data. For each of these applications, we randomly select 1000 connections as training data. Although there are more than 1000 connections, we do not include a Markov model for *http+flash* because these connections get easily confused with *http*. While the training data contains an equal number of connections for all applications, the test data shall reflect the original traffic mix. Therefore, we select 2% of the connections of each application as test data, making sure that none of the connections is contained in training and test data at the same time.

3.2 Evaluation Metrics

As evaluation metrics, we calculate *recall* and *precision* for every application k:

$$recall_k = \frac{\text{number of connections correctly classified as application } k}{\text{number of connections of application } k \text{ in the test data}}$$

$$precision_k = \frac{\text{number of connections correctly classified as application } k}{\text{total number of connections classified as application } k}$$

A perfect classifier achieves 100% recall and precision values for all applications.

In order to compare different classifiers with a single metric, we calculate the *overall accuracy*, which is the proportion of correctly classified connections. In the cross-validation case (see Section 3.3), the overall accuracy is calculated for an equal number of connections per application, which means that the overall accuracy is identical to the average recall.

3.3 Parameterization Using Cross-Validation

The performance of the classifier depends on the parameterization, in particular on the number of considered data packets and the payload length interval boundaries. We use 10-fold cross-validation to assess the influence of different parameter settings. For this, the training data is divided into ten disjoint sets. Then, ten classifiers are trained, each one with a different set left out. For each classifier, we use the remaining fold not used for training as test data. To evaluate the overall performance of a parameter setting, we consider the minimum, mean, and maximum classification accuracy resulting from the ten runs.

As a starting point, we use similar interval boundaries as in our previous work [7], namely $B_1 = \{100, 300, 1450\}$. This means that the following four payload length intervals are considered in the states of the Markov models: $[1; 100]$, $[101; 300]$, $[301; 1450]$, $[1451; \infty)$. A preliminary evaluation of B_1 showed deficiencies in the classification of *pop* and *smtp*. By inspection of the applications' payload length distributions, we noticed systematic differences in the payload length of the first data packet: For *smtp*, almost 90% of the first data packets exceed 50 byte whereas, in case of *pop*, less than 20% contain more than 50 byte of payload. Therefore, we add 50 as an additional boundary and evaluate the classifier's performance for $B_2 = \{50, 100, 300, 1450\}$.

The interval boundaries mentioned so far are based on experimental evaluation of the classifier's performance and inspection of payload lengths. Given the huge number of possible boundary combinations, the alternative approach of finding interval boundaries in a systematic and automatic manner is particularly complex. In order to limit the complexity, we use an iterative greedy algorithm which determines the best additional boundary based on an existing set of interval boundaries found in preceding iterations. This is accomplished

Table 2 Cross-validation results for differing number of packets and boundaries

Packets	Boundaries	Overall accuracy (*min/mean/max*)	
		Without EOC state	With EOC state
3 Packets	$B_1 = \{100, 300, 1450\}$	0.8433/0.8508/0.8708	0.8667/0.8764/0.8883
	$B_2 = \{50, 100, 300, 1450\}$	0.8758/0.8938/0.9025	0.8967/0.9096/0.9258
	$B_3 = \{40, 190\}$	0.8442/0.8544/0.8692	0.8433/0.8610/0.8792
	$B_4 = \{40, 190, 380\}$	0.8842/0.8995/0.9067	0.8933/0.9066/0.9183
	$B_5 = \{40, 190, 380, 620\}$	0.9092/0.9187/0.9317	0.9192/0.9287/0.9367
4 Packets	$B_1 = \{100, 300, 1450\}$	0.8583/0.8749/0.8975	0.8842/0.8993/0.9092
	$B_2 = \{50, 100, 300, 1450\}$	0.9158/0.9228/0.9333	0.9267/0.9397/0.9525
	$B_3 = \{40, 190\}$	0.8483/0.8584/0.8700	0.8717/0.8834/0.9000
	$B_4 = \{40, 190, 380\}$	0.9000/0.9163/0.9292	0.9142/0.9309/0.9425
	$B_5 = \{40, 190, 380, 620\}$	0.9208/0.9326/0.9450	0.9358/0.9463/0.9592
5 Packets	$B_1 = \{100, 300, 1450\}$	0.8725/0.8866/0.9000	0.8950/0.9050/0.9183
	$B_2 = \{50, 100, 300, 1450\}$	0.9308/0.9381/0.9458	0.9400/0.9469/0.9625
	$B_3 = \{40, 190\}$	0.8625/0.8760/0.8858	0.8908/0.8977/0.9108
	$B_4 = \{40, 190, 380\}$	0.9217/0.9318/0.9392	0.9408/0.9497/0.9583
	$B_5 = \{40, 190, 380, 620\}$	0.9442/0.9504/0.9567	0.9567/0.9635/0.9717
6 Packets	$B_1 = \{100, 300, 1450\}$	0.8817/0.8914/0.9067	0.8992/0.9102/0.9225
	$B_2 = \{50, 100, 300, 1450\}$	0.9300/0.9407/0.9533	0.9400/0.9491/0.9550
	$B_3 = \{40, 190\}$	0.8725/0.8814/0.8917	0.8900/0.8997/0.9075
	$B_4 = \{40, 190, 380\}$	0.9300/0.9393/0.9500	0.9400/0.9485/0.9575
	$B_5 = \{40, 190, 380, 620\}$	0.9433/0.9503/0.9617	0.9517/0.9625/0.9750

by analyzing the relation of the application labels and the state sequences of the corresponding connections in the training data. We use a concept from decision tree learning, the *Gini impurity index* which measures how "impure" the assignment of state sequences to application labels is (cf. [3]). As a new boundary, we choose the boundary which results in a maximum reduction of impurity in order to achieve a strong differentiation of the applications with respect to the state sequences. Considering state sequences of four data packets, four iterations of this algorithm lead to the following interval boundary sets: $\{190\}$, $B_3 = \{40, 190\}$, $B_4 = \{40, 190, 380\}$, $B_5 = \{40, 190, 380, 620\}$. We restrict ourselves to models with at least two boundaries which turned out to be the minimum number necessary for achieving acceptable classification results.

In the following experiments, we compare the overall accuracy of five boundaries sets $B_1, \ldots, B_5$. Moreover, we assess the influence of the EOC state. The resulting minimum, mean, and maximum overall accuracies are tabulated in Table 2 for 3 to 6 data packets. As can be seen, we achieve a consistent increase in the mean overall accuracy by introducing the EOC state. Detailed inspection of the application specific results shows that there is a particularly strong increase of the mean recall for *dns* and *ssl*. As an example, the mean recall for *dns* increases from 79.3% to 93.5% for boundary set B_1 and 4 data packets. At the same time, the mean recall for *ssl* is improved by 14.4 percentage points reaching 81.7%. An improvement of about 2 percentage points is also observed for *ftp* whenever the EOC state is included. Regarding the other applications, the effect on the recall value is in the range of ± 1 percentage points in most of the cases. Due to the better overall accuracy, we concentrate on models including the EOC state in the following discussion.

The models based on the smallest boundary set B_3 exhibit the worst performance. We conclude that the boundary set should have a minimum size of 3, corresponding to four payload length intervals. Moreover, the parameterizations with interval boundary sets B_2, B_4 and B_5 yield higher overall accuracy compared to the boundary set B_1. In particular, the models based on these boundary sets exhibit better performance in distinguishing the protocols *smtp* and *pop*. Based on the boundary set B_5 and 4 data packets, we achieve a mean recall of 87.1% and 97.8% for *pop* and *smtp*, respectively. In comparison, the model based on B_1 reaches 71.9% and 67.1%, only.

The models based on the interval set B_5 exhibit a slightly higher accuracy than models based on B_2 and B_4. On the other hand, the number of states in the models can be reduced by means of the smaller boundary set B_4. Depending on the scenario, this may compensate for the disadvantages in terms of accuracy. Concerning the number of considered data packets, there is a consistent increase in accuracy from 3 to 5 packets for all parameterizations considered. With 6 packets, the accuracy slightly decreases for B_4 and B_5 wheres a small increase is observed for the other settings. Therefore, we conclude that the best number of data packets depends on the interval boundaries. For the boundary sets reaching relatively higher overall accuracies, i.e. B_2, B_4 and B_5, the strongest increase occurs when considering 4 instead of 3 data packets. Only minor improvements can be achieved by integrating more data packets into these models, which is consistent with our earlier findings [7].

3.4 Classification of Test Data

In the last subsection, we obtained the best overall accuracy with Markov models considering five data packets, the new EOC state, and five payload length intervals with boundaries $B_5 = \{40, 190, 380, 620\}$. With these settings, we train a new traffic classifier using the entire set of training data (i.e. all ten folds). Thereafter, we apply the classifier to the TCP connections in the test data which represent about 2% of the original traffic. As described in Section 3.1, none of the connections in the test data is included in the training data. Connections belonging to one of the twelve applications considered in the Markov models are classified with an overall accuracy of 94.51%, which is 1.8 percentage points smaller than the average accuracy achieved in the cross-validation. The reason is the different traffic mix which now corresponds to the percentages shown in Table 1.

Table 3 shows the recall and precision values of the twelve applications as well as the proportion of unclassified connections, which is negligible in all cases. In addition, the last column indicates the labels assigned to the incorrectly classified connections. All applications are detected with a recall value equal or larger than 92%. The low precision values for *bittorrent*, *dns*, and *windowsmedia* go back to several *http* connections being classified as one of these applications. Several *smtp* connections are classified as *pop*, which explains the low precision value of *pop*.

For comparison, we train and apply another traffic classifier which considers the same number of data packets and the same payload length intervals but does not include the EOC state in the Markov models. Table 4 shows the results based on the same training and test data as used before. As can be seen, the recall value of *dns* significantly decreases from 92% with EOC state to 82% without EOC state. The recall values of *http* and *ftp* decline by about 2.4 percentage points, the one of *ssl* is reduced by 1.1 points. For the other applications, the recall is identical or only slightly decreased. The overall accuracy now is 92.70%, which is about 1.8 percentage points smaller than before. As expected, the influence of the EOC state on the application specific recall values is very similar to what we observed in the cross-validation results for this setting.

Table 3 Classification results of test data for $B_5 = \{40, 190, 380, 620\}$ with EOC state

	Recall	Precision	Unclassified	Most frequently misclassified as
bittorrent	0.9428	0.0906	0.0000	http, ssl
dns	0.9200	0.1776	0.0000	windowsmedia, bittorrent, ssl
flash	0.9777	0.9535	0.0000	http, ssl
ftp	0.9840	0.8145	0.0000	http
http	0.9428	0.9957	0.0008	windowsmedia, ssl, bittorrent
imap	0.9431	0.9880	0.0000	smtp, pop, ssh
pop	0.9729	0.4298	0.0000	smtp
rdp	1.0000	0.9130	0.0000	
smtp	0.9354	0.9975	0.0003	pop
ssh	0.9909	0.9977	0.0000	pop
ssl	0.9406	0.8954	0.0000	bittorrent, http, windowsmedia
windowsmed	1.0000	0.1923	0.0000	

Table 4 Classification results of test data for $B_5 = \{40, 190, 380, 620\}$ without EOC state

	Recall	Precision	Unclassified	Most frequently misclassified as
bittorrent	0.9428	0.0518	0.0000	ssl
dns	0.8200	0.1872	0.0000	ssl, windowsmedia, bittorrent
flash	0.9788	0.8853	0.0000	http, ssl
ftp	0.9600	0.9448	0.0000	http
http	0.9193	0.9948	0.0008	ssl, bittorrent, windowsmedia
imap	0.9431	0.9880	0.0000	smtp, pop, ssh
pop	0.9729	0.4298	0.0000	smtp
rdp	1.0000	0.9130	0.0000	
smtp	0.9357	0.9975	0.0000	pop
ssh	0.9909	0.9977	0.0000	pop
ssl	0.9295	0.8435	0.0000	bittorrent, http, windowsmedia
windowsmed	1.0000	0.1923	0.0000	

In the following, we continue our discussion of the classification results obtained with EOC state. As described in Section 3.1, OpenDPI provided more than one label for several TCP connections. In most of these cases, the first label was *http*, followed by a label related to multimedia content, such as *flash* or *mpeg*. Although we did not include any of these connections in the training data, more than 98% of those contained in the test data are appropriately classified as *http*.

Now, we have a closer look at the traffic which was marked as *unknown* by OpenDPI. For these connections, we can use the server port number as a weak indicator of the application. As already mentioned, a large proportion of *unknown* connections is directed to port 10050 on one host. All of these connections are classified as *rdp*, which is Microsoft's remote desktop protocol. Almost all traffic to port 2703 (Razor) is classified as *pop*. In both cases, there seems to be a strong similarity to the corresponding protocols although the signatures of OpenDPI do not match.

Regarding *unknown* connections to well-known port numbers, the classification results are often as expected. For example, most of the connections to port 143 are classified

as *imap*. About half of the connections to ports 25 and 110 are recognized as *smtp* and *pop*, respectively. 59% of the connections to port 443 (intended for https) and 100% of the connections to port 1352 (Lotus Notes) are classified as *ssl*. Although Lotus Notes is likely to use SSL, OpenDPI did not label any of these connections as *ssl*. 3.3% of the connections labelled *unknown* go to port 80. Interestingly, only 13% of them are classified as *http* while 32% are considered as *ssl* and 12% remain unclassified. A probable explanation for not finding any conformance with HTTP is that port 80 is used by other protocols and applications.

4 Conclusion

In this paper, we presented and discussed a TCP traffic classification approach using observable Markov models. Departing from our previous work [7], we proposed the integration of an additional state which further improves the classification accuracy. We tested and evaluated the improved classifier under real-world conditions using TCP traffic captured in the Munich Scientific Network. By means of cross-validation, we first evaluated various parameterizations and proposed methods to systematically derive the interval boundaries needed for quantizing payload lengths. Thereafter, we demonstrated that our approach is able to classify a representative traffic mix into twelve application classes with very high accuracy.

Our traffic classifier is based on Markov models which are intuitive and easy to understand. As another advantage, the training and classification complexity is low. A time-consuming task is the greedy search for appropriate payload length intervals. Future work will show whether this task actually needs to be repeated for training data from different networks or whether the traffic characteristics across networks are consistent enough to yield good classification results using the same payload length intervals or even the same Markov models.

Regarding the design of our classifier, we still see room for improvements, in particular regarding the mapping of payload length to states. We currently use constant payload length intervals for all packet positions. Instead, the best set of interval boundaries could be determined separately for each packet position, which may lead to better classification results. As a downside, this flexibility increases the complexity of finding appropriate values for all boundaries.

Most kinds of unencrypted traffic can be correctly classified based on port numbers or signatures. Nevertheless, it is useful to deploy our traffic classifier in parallel in order to determine the most likely application for those connections which remain unclassified by DPI. Apart from that, we are currently investigating the classification of encrypted traffic with the goal to distinguish different applications running on top of SSL and TLS. For these kinds of traffic, our Markov-based classifier possibly provides useful information while DPI is only able to detect SSL or TLS records.

Acknowledgments

We gratefully acknowledge support from the German Research Foundation (DFG) funding the LUPUS project in which this research work has been conducted.

References

1 Hui Dai, Gerhard Münz, Lothar Braun, and Georg Carle. TCP-Verkehrsklassifizierung mit Markov-Modellen. In *Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 5. GI/ITG-Workshop MMBnet 2009*, Hamburg, Germany, September 2009.

2 Alberto Dainotti, Walter de Donato, Antonio Pescapè, and Pierluigi Salvo Rossi. Classification of network traffic via packet-level hidden markov models. In *Proc. of IEEE Global Telecommunications Conference (GLOBECOM) 2008*, New Orleans, LA, USA, November 2008.

3 Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

4 Alice Este, Francesco Gringoli, and Luca Salgarelli. On the stability of the information carried by traffic flow features at the packet level. *Computer Communication Review*, 39(3):13–18, 2009.

5 Juan M. Estevez-Tapiador, Pedro Garcia-Teodoro, and Jesus E. Diaz-Verdejo. Stochastic protocol modeling for anomaly based network intrusion detection. In *Proc. of IEEE International Workshop on Information Assurance (IWIA) 2003*, Darmstadt, Germany, March 2003.

6 Hyun-chul Kim, Kimberly Claffy, Marina Fomenkov, Dhiman Barman, Michalis Faloutsos, and KiYoung Lee. Internet traffic classification demystified: Myths, caveats, and the best practices. In *Proc. of ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT) 2008*, Madrid, Spain, December 2008.

7 Gerhard Münz, Hui Dai, Lothar Braun, and Georg Carle. TCP traffic classification using Markov models. In *Proc. of Traffic Monitoring and Analysis Workshop (TMA) 2010*, Zurich, Switzerland, April 2010.

8 Thuy T. T. Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys & Tutorials*, 10(4):56–76, 2008.

9 OpenDPI Homepage. http://www.opendpi.org/, 2010.

10 Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, December 1999.

11 Vipul's Razor Homepage. http://razor.sourceforge.net/, 2010.

12 Charles Wright, Fabian Monrose, and Gerald Masson. HMM profiles for network traffic classification (extended abstract). In *Proc. of Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC) 2004*, pages 9–15, Fairfax, VA, USA, October 2004.

IT Management Using a Heavyweight CIM Ontology

Andreas Textor[1,2], Jeanne Stynes[2], and Reinhold Kroeger[1]

1 **RheinMain University of Applied Sciences**
Distributed Systems Lab
Kurt-Schumacher-Ringer 18, D-65197 Wiesbaden, Germany
{andreas.textor,reinhold.kroeger}@hs-rm.de

2 **Cork Institute of Technology**
Department of Computing
Rossa Avenue, Bishopstown, Cork, Ireland
jeanne.stynes@cit.ie

Abstract

This paper presents an approach for ontology-based IT management based on a heavyweight (formal) ontology using the Web Ontology Language (OWL). The ontology comprises a complete OWL representation of the Common Information Model (CIM) and management rules defined in the Semantic Web Rule Language (SWRL). The ontology not only models the managed system types, but a runtime system dynamically updates model instances in the ontology that reflect values of managed system entities. This allows the evalution of rules that take into account both model and model instances. A reaction module uses the CIM interface of the managed system to invoke CIM methods according to rule evaluation results, thus resulting in automated management. In order to ensure the consistency of the ontology when changes are performed, belief change theory is employed.

1998 ACM Subject Classification C.2.3 Network Operations

Keywords and phrases CIM, OWL, ontology, SWRL, management

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.73

1 Introduction

Knowledge bases grow in size and complexity in every domain. Regardless of the concrete field, the number of data sources and formats that are used in each knowledge base increases constantly. A common goal in this respect is to allow interoperability between different knowledge bases and integration and aggregation of data, as well as the possibility to perform automated reasoning. Established domain models can be used to create a unified view on the knowledge base, but they often have major shortcomings to be able to be used for comprehensive management of the knowledge base: They often have limited support for knowledge interoperability and aggregation, as well as reasoning, and constraints that are part of the models are often expressed using natural language, which renders them less useful for automatic evaluation. Also, the models usually do not deal with instance data that is acquired from the system which the model represents. Unless instance data is kept in a uniform way together with the model, the definition of constraints or rules that affect both model and model instances is not possible.

A formal ontology is a type of knowledge base that can help resolve these issues: While a *lightweight ontology* defines a hierarchy of concepts and a hierarchy of relations, i.e. a

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 73–84

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

description of the concepts and relationships that exist in a domain, a *heavyweight ontology* or *formal ontology* (the terms are often used interchangeably [24]) can contain domain-specific information models, behaviour rules, constraints and axioms, and can also contain the corresponding model instances.

Management of an IT environment is one example of a domain for which management using a formal ontology is desirable. Over time, a number of commercial and free systems for comprehensive management of IT environments have been developed. Usually, the management system is not comprised of a single tool, but consists of a set of different tools. To provide a unified view on the environment and to allow interoperability between multiple management tools and managed elements, several integrated network management models were developed. Notable examples are the OSI network management model (also known as CMIP, the name of its protocol) and the still widely used Simple Network Management Protocol (SNMP). A more recent approach to specify a comprehensive IT management model is the *Common Information Model* (CIM) which is described in more detail in section 2.1. This widely recognized Distributed Management Task Force (DMTF) standard allows consistent management of managed elements in an IT environment. CIM is actively used; for example, the storage management standard SMI-S (Storage Management Initiative Specification) of the SNIA (Storage Networking Industry Association, [19]) is based on CIM.

CIM can be used to create a unified view on the managed system, but suffers from the shortcomings mentioned earlier: It is a *semi-formal* ontology [4, 18] that only has limited abilities for knowledge interoperability, knowledge aggregation and reasoning [16]. To create a formal IT management ontology, two parts are required; the first part being a comprehensive domain model in a suitable format. The second part is a runtime system that administers the domain model and model instances that represent entities of the managed environment. As the runtime system provides means to add and delete model instances to reflect changes in the managed environment, it must take care that adding and deleting facts may not render the knowledge base inconsistent.

In this paper we present an approach for building a formal ontology for IT management based on CIM including a domain model, model instances and rules, a runtime system that manages the ontology, interfaces with the managed system to add and delete model instances, and an engine to evaluate management rules. For the domain model, a conversion of CIM to the *Web Ontology Language* (OWL) is performed. In order to assure the consistency of the knowledge base, *belief change theory* is employed. Section 2 gives an overview of CIM and belief change theory as background, while section 3 describes related work. Section 4 explains the management approach and how the runtime system is constructed. In section 5, implementation details and performance measurements of the approach are discussed. The paper closes with a summary in section 6.

2 Background

2.1 DMTF CIM Overview

This section briefly describes the basic properties of the Common Information Model. CIM defines how managed elements in an IT environment are represented as a set of objects and relationships between them. The model is intended to allow a consistent management of these managed elements, independent of their manufacturer or provider. Unlike SMI (Structure of Management Information, the model underlying the popular SNMP protocol), CIM is an object-oriented model. CIM consists of

- A basic information model called the *meta schema*. The meta schema is defined using the Unified Modeling Language (UML, [13]).
- A syntax for the description of management objects called the *Managed Object Format* (MOF).
- Two layers of generic management object classes called *Core Model* and *Common Model*.

Figure 1 shows the CIM meta schema definition in UML, as shown in the CIM specification [6]. The meta schema specifies most of the elements that are common in object-oriented modelling, namely

- *Classes*, *Properties* and *Methods*. The class hierarchy supports single inheritance (generalization) and overloading of methods. For methods, the CIM schema specifies only the prototypes of methods, not the implementation.
- *References* are a special kind of property that point to classes.
- *Qualifiers* are used to set additional characteristics of Named Elements, e.g. possible access rules for properties (`READ`, `WRITE`), marking a property as a key for instances (using `Key`) or marking a class as one that can not be instantiated. Qualifiers can be compared to Java annotations; some qualifiers also have parameters.
- *Associations* are classes that are used to describe a relation between two classes. They usually contain two references.
- *Triggers* represent a state change (such as create, delete, update, or access) of a class instance, and update or access of a property.
- *Indications* are objects created as a result of a trigger. Instances of this class represent concrete events.
- *Schemas* group elements for administrative purposes (e.g. naming).

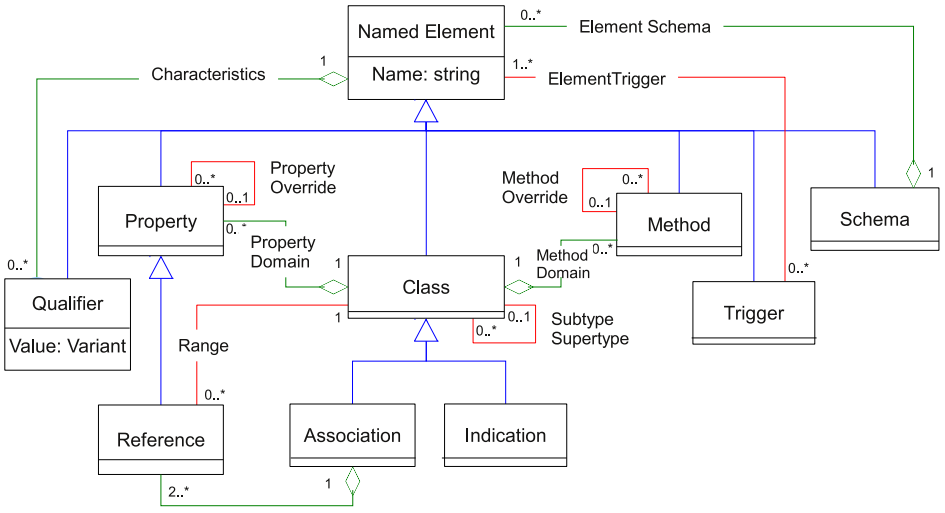


Figure 1 CIM Meta Schema

Properties, references, parameters and methods (method return values) have a data type. Datatypes that are supported by CIM include {s,u}int{8,16,32,64} (e.g. uint8 or sint32), real{32,64}, string, boolean, datetime and strongly typed references (<classname> ref).

In addition to the CIM schema, CIM specifies a protocol, based on XML over HTTP, which is used by CIM-capable network managers to query classes, instances and invoke methods against a so-called CIM object manager (CIMOM).

2.2 Belief change and ontology change

The process of changing beliefs to take into account a new piece of information about the world is called *belief change*. Belief change studies the process an agent has to perform to accommodate new or more reliable information that is possibly inconsistent with existing beliefs. Usually, beliefs are represented as a set of sentences of a logical language. Most formal studies on belief change are based on the work of Alchourrón, Gärdenfors and Makinson (AGM) [2, 1], which is now commonly referred to as the *AGM theory*. The AGM theory distinguishes three types of belief change operations that can be made: contraction, expansion and revision. Contraction is retracting a sentence from a belief set, expansion is adding a sentence to a belief set regardless of whether the resulting belief set is consistent, and revision is incorporating a sentence into a belief set while maintaining consistency. The AGM trio specified postulates for contraction and revision operators which they claim should be satisfied by all rational belief change operators.

As an ontology can be considered a belief base in the sense of the belief change theory, the problems described hold for the change of ontologies as well. In this context, the problem is known as *ontology change*. Belief change theory can not be directly applied to description logics (which are the theoretical foundation of OWL) because it is based on assumptions that generally fail for description logics [15]. However, the authors in [7] show that all logics admit a contraction operator that satisfies the AGM postulates except the recovery postulate. In [17], the authors show that the theory can be applied if the recovery postulate is slightly generalized and to achieve this, replace the recovery postulate with the relevance postulate by Hansson [9].

Another approach to the problem of ontology update is taken in [11], which proposes an ontology update framework where ontology update specifications describe certain change patterns that can be performed. An agent (knowledge worker) "issues a change request by providing a piece of knowledge to be added to or deleted from the ontology". The change request is only accepted when the given ontology update specification accounts for it. An applicable update rule from the update specification is determined and similar to a database trigger, "the change request is acted on accordingly by denying or accepting it but possibly also by carrying out more ontology changes than explicitly requested."

3 Related Work

The idea of applying semantic web technologies to the domain of IT management has been examined in several publications, notably by De Vergara et al., e.g. [3, 8], where a semantic management framework is presented which integrates different definitions of the managed resources, taking into account the semantic aspects of those definitions. In later works they favor the use of OWL for the definition of the management ontology. As a domain model, CIM is often proposed, although the conversion of the native format in which CIM is specified into an ontology is not trivial. In [5] the authors provide a possible mapping for a subset of CIM to OWL and in [16] the authors compare possible conversions of CIM to RDFS (Resource Description Framework Schema) and to OWL. They find that RDFS is unsuitable to express CIM as it does not allow to express constructs such as cardinality restrictions and some CIM qualifiers. Instead, they construct an OWL-based ontology for CIM by using a previously defined mapping of UML. Most CIM concepts (e.g. CIM qualifiers) however, have no mappings to either UML or OWL constructs, so the resulting ontology lacks a large part of the information that is provided in the original model. The authors in [12] introduce a meta-ontology to model CIM constructs that have no direct OWL correspondence, but

they do not describe how this meta-ontology is constructed, and their approach does not specify how several qualifiers and more complex elements, such as CIM methods, can be converted. The approach presented in this paper thus relies on the translation of CIM to OWL described in section 4.2, and in more detail in [20].

To be able to model a heavy-weight ontology for management purposes, several papers examine how SWRL (Semantic Web Rule Language) can be employed. In [8], the authors propose a formal definition of the different management behaviour specifications integrated with the management information definitions. They try to include both behaviour definitions included implicitly in the management information and explicitly in policy definitions in the resulting management ontology. Other works, notably [23] and [22], define a configuration management ontology and augment the ontology with behaviour definitions in SWRL and service definitions in OWL-S (an OWL ontology for describing Web Services).

4 Approach

4.1 Architecture

The goal of the approach presented in this paper is the management of a system using a suitable domain ontology and a runtime system that performs the actual management using the ontology. Although the approach is not limited to IT management (as the domain ontology and interfacing components can be replaced), in this paper it is applied to the domain of IT management. The domain ontology is defined in OWL, as this format is the de-facto standard ontology language and allows the description of ontologies using classes, object properties, data properties, cardinalities etc. As described in section 3, the Common Information Model (CIM) has been examined and found to be an appropriate model for an IT management ontology, however the translation of CIM to OWL has to be performed first.

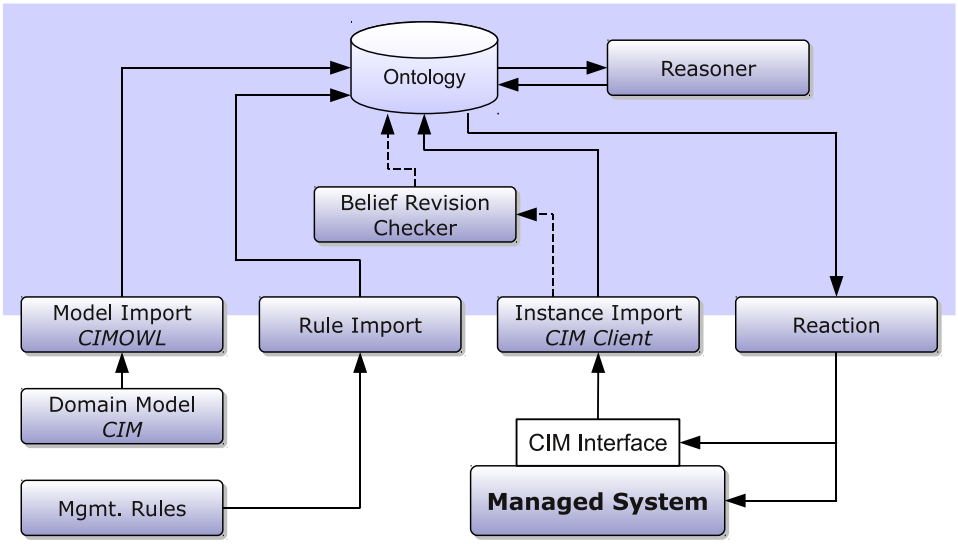


Figure 2 Architecture

Figure 2 shows an overview of the proposed architecture. The grey box at the top represents the runtime system and its components, while the elements at the bottom of the diagram show the parts the runtime system interacts with. The components of the runtime system are described in detail in the following sections.

4.2 Model Import

First of all, the runtime system has a component to import the domain ontology into the runtime ontology (which is shown at the top of figure 2). The domain ontology is a static model that describes the entities in the managed system in detail. This component is responsible for the translation of CIM to OWL. As described in section 3, this translation is not trivial if most or all CIM constructs are to be translated, including CIM qualifiers. When CIM is translated into OWL, the result is named *CIMOWL* here.

Some elements from the CIM schema can be translated in a straightforward way, while other elements, especially qualifiers, can not be directly translated. To solve this problem, the translation approach used here consists of two parts: One part is a CIM meta-ontology, which is statically modelled (i.e. manually) and which consists of super classes, properties and annotations that meta-model CIM constructs which can not be directly translated to OWL. The second part is the CIM schema ontology, which is modelled using OWL, RDFS and CIM meta constructs, and which represents the actual CIM model. It is automatically created by parsing and translating the MOF representation of the CIM schema. One particular part of the translation, the implementation of CIM properties in the OWL model, is given here as an example.

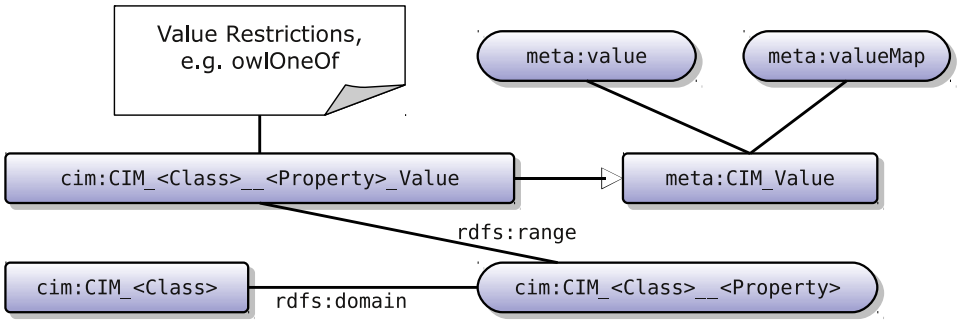


Figure 3 Modelling of properties

To implement CIM properties in the OWL model, a structure as shown in figure 3 is used. In the figure, rectangular boxes represent OWL classes, rounded boxes with one connection represent datatype properties and rounded boxes with two connections represent object properties. Each CIM property is modelled as a combination of an OWL object property and an OWL class. This class inherits from the meta ontology class `CIM_Value` that has two datatype properties, "`value`" and "`valueMap`". To model the semantics from the original `ValueMap` and `Values` qualifiers, the class is restricted using `owl:oneOf` to allow only instances from a list of `CIM_Value` instances (one for each `ValueMap-Values` combination).

The translation of CIM to OWL is described in more detail in [20]. Only constructs that are valid in OWL DL are used; the resulting ontology has the expressiveness of OWL DL, which allows for decidable reasoning.

The translation of the CIM schema has to be performed only once (and whenever the ontology has to be updated, which can be necessary if a new version of the CIM schema is released). CIM schema version 2.23.0 consists of about 1400 MOF files, containing roughly the same amount of OWL classes, and is converted to about 70,000 OWL axioms. The translation result is an OWL file which can be loaded by the runtime system on startup.

4.3 Instance Import

The instance import component is the component of the runtime system reponsible for acquiring up-to-date information from the managed system and incorporating it into the runtime ontology. As such, it has three concerns: The first task is that of a CIM client. As CIM is the domain model of choice, any CIMOM (CIM Object Manager) can be queried for CIM instances. These instances represent concrete entities of the managed system, such as hardware or software components. Queries are executed using the CIM protocol and result in an XML document that describes a collection of instances.

The second task of the instance import component is the conversion of this CIM instance data into the corresponding OWL instances. When performing this conversion, the resulting OWL instances have to be compliant with the OWL representation of the CIM schema that was created in the model import component and described in section 4.2. This means that each CIM instance is mapped to a collection of OWL axioms (for class instances, object properties and datatype properties, as necessary) that represent the same concept. Also, primitive values must be converted accordingly: Each signed and unsigned number type is translated to its XSD equivalent, e.g. `uint8` becomes `xsd:unsignedByte`, `sint32` becomes `xsd:int`, `real32` becomes `xsd:float`, and so on. Translation is mostly straightforward, except for `char16`, which has to be translated into a string, and `datetime`, which has a corresponding XSD data type but which specifies a different lexical representation for the datetime string.

The third task for the import component is to act as a revision operator in the sense of belief change theory (cp. section 2.2). When OWL axioms that are created when reading CIM data from the managed system are to be added to the runtime ontology, they must not render the ontology inconsistent. As the import component normally does not create axioms that perform structural changes on the ontology when added, but only adds or changes values, most of the problems that are studied in the field of ontology change can be circumvented, i.e. a change that leads to an inconsistency does not happen easily. The component must ensure however, that adding values that represent the current state of some part of the managed system does not lead to ambiguities. When an instance for an updated value is to be added, instances of the class the value belongs to have to be removed accordingly. The import component has to make sure at this moment that the adding and removing operations, once performed, result in a consistent state of the ontology, and in unambiguous information.

4.4 Belief Revision Checker

The import component implements an interface for belief revision checks. AGM postulates, updated for ontology change (see section 2.2), are implemented by the belief revision checker component. As it is not possible to create a generic belief revision operator that works for any knowledge base and any new fact, the belief revision checker implements the checks necessary for a given revision operator to verify that it keeps the ontology consistent. Because these checks have a comparably high computational complexity for reasoning, which can slow down the runtime system considerably, the belief revision checker can be switched on and off as necessary. When a new revision operator is installed in the system, the system is run with the revision checker switched on, using representative data, to verify that the operator works as intended (i.e., testing every case of axioms the revision operator adds or updates in the ontology). For production use, the revision checker is then switched off, which reduces the overhead to a minimum (the complexity for adding or removing instances in the ontology).

Listing 1 Sample SWRL rule

```
CIM_DataFile(?f),
CIM_DataFile(?reffile),
CIM_LogicalFile__LastAccessed(?f, ?lastaccvalue),
CIM_LogicalFile__LastAccessed(?reffile, ?reflastaccvalue),
CIM_LogicalFile__Name(?f, ?filenamevalue),
CIM_LogicalFile__Name(?reffile, ?reffilenamevalue),
CIM_LogicalFile__FileSize(?f, ?filesizevalue),
value(?reffilenamevalue, ?reffilename),
endsWith(?reffilename, "/var/reffile"),
value(?filesizevalue, ?size),
value(?lastaccvalue, ?lastacc),
value(?reflastaccvalue, ?reflastacc),
greaterThan(?size, 150000),
lessThan(?reflastacc, ?lastacc) ->
LargeOldFile(?filenamevalue)
```

The instance import component can now continue its work, as its functionality has been verified.

4.5 Specification and Import of Rules

While the model import component provides the static domain model in the ontology, and the instance import component dynamically adds and removes model instances to reflect changes in the managed system, the rule import component loads rules from external OWL files into the runtime ontology. Rules are defined using the Semantic Web Rule Language (SWRL) which can be embedded in an OWL ontology and can take into account both the statically loaded CIM model and the dynamically added model instances. Therefore it is possible to define rules that are triggered depending on the current state of the managed system.

Each set of rules for a certain check or task is accompanied by the definition of classes that are used in the context of these rules to denote the result of the triggered rule. Listing 1 shows a sample SWRL rule, which checks for files that surpass a certain file size (150,000 bytes in this case) and are older than a reference file (`/var/reffile`). As CIM does not natively support the definition of rules, no standard ruleset exists that could be used for testing the system. The sample rule given here was thus devised for this purpose.

`CIM_DataFile(?f)` identifies an instance `f` as belonging to the `CIM_DataFile` class, while `CIM_LogicalFile__FileSize(?f, *)` match for the `FileSize` and `Name` properties of the file, respectively. `CIM_LogicalFile__LastAccessed` matches for the dateTime value at which the file was last accessed. The expression `value(?filesizevalue, ?size)` extracts the numerical value from the `CIM_Value` datatype property that is attached to the file size object property. Using the SWRL builtin `swrlb:lessThan` on the values of the `LastAccessed` fields, it can be determined if the file in question is older than the reference file.

While the classes that start with `CIM_*` are defined in the CIM ontology, the class `LargeOldFile` is defined to accompany the rule. All instances of `CIM_DataFile` that have a `CIM_LogicalFile__FileSize` attribute with a value greater or equal 150,000 and were last accessed before the reference file are defined to also belong to the class `LargeOldFile`. The rule can be extended and used in a use case where large files that are older than a

certain date on a filesystem should be compressed or archived. This example shows the basic approach for the definition of rules. CIM models not only the file system but other aspects of an IT environment as well, so rules covering other areas can be defined similarly.

4.6 Reasoner and Reaction

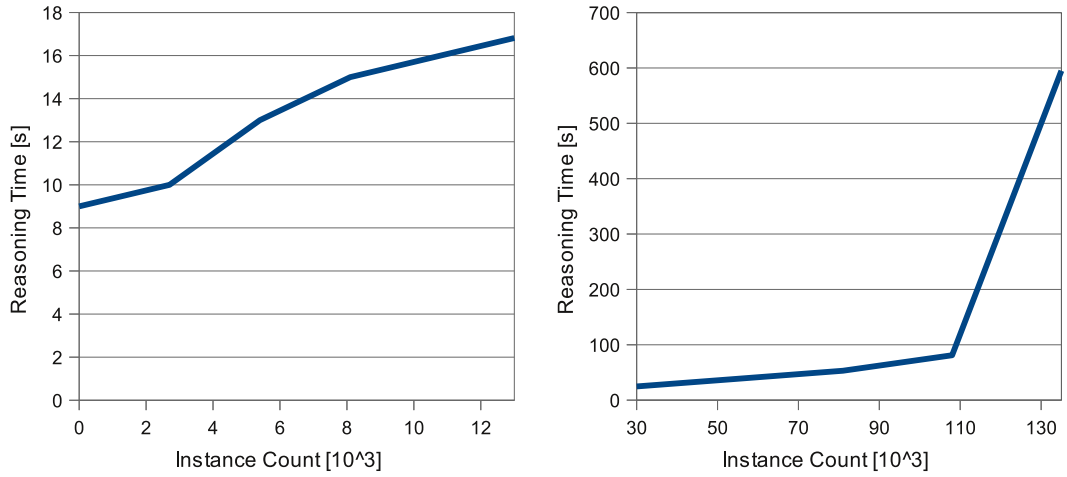
A reasoner is called to evaluate rules on the runtime ontology. For the example from section 4.5, after the reasoning takes place, the reasoner can be queried for instances that belong to the `LargeOldFile` class to receive the results from that rule. After these results have been retrieved, axioms added by the reasoner during rule evaluation are removed from the ontology to reset the state.

A custom module can be installed in the reaction component for a rule or a set of rules, which is configured to watch a certain set of classes (e.g. in the sample case, `LargeOldFile`) and decides how to react when instances of theses classes are found. Reactions can include logging and reporting (e.g. sending SNMP traps to external network managers). More importantly, the reaction module can make use of the CIM interface of the managed system. By invoking CIM methods on the target machine, management decisions can be executed using the existing interface.

5 Implementation and Performance

The translator from CIM to OWL and the runtime system were prototypically implemented in Scala [21], a programming language that integrates object-oriented and functional features and that compiles to Java Byte Code. Java libraries can be used in Scala programs, and OWLAPI 2.2.0 was used for the creation of the ontology. As a reasoner, Pellet 2.2.1 was employed. The CIM client component was custom built.

The conversion of the CIM schema to OWL takes about 35 seconds on an Intel Core 2 Duo with 2 GHz and 2 GB RAM and results in a 12 MB OWL file. Loading the ontology using OWLAPI takes approximately 10 seconds and uses 130 MB RAM on the same computer.



(a) Up to 13,000 instances **(b)** More than 30,000 instances

Figure 4 Reasoning Performance

Figure 4 shows the performance of the runtime system and the time required for reasoning. For this diagram, the computer described above was used and the ontology used for reasoning consisted of two parts: The first part is the CIMOWL ontology with 70,000 axioms, the second part was a varying amount of OWL instances representing `CIM_DataFile`s of a server file system. Each `CIM_DataFile` is represented as a set of 27 OWL instances, which include the instance for the file and one instance for each of the file attributes (file size, modification time, etc.). SWRL rules to categorize files by size (as demonstrated in listing 1) were loaded into the ontology. The test ran on JDK 1.6.0_20. CIM queries were executed on an OpenPegasus ([14]) installation running locally, including a custom built file system instance provider. As the CIM queries take a few milliseconds, their impact on overall performance can be neglected.

In figure 4a the range from 0 to 13,000 instances is shown, which equates to 0 to 500 files. For 2,000 instances and less, the reasoning takes always at least 9 seconds. The graphs show a nearly linear increase in time needed for more instances. Figure 4b shows the limit of the tested system at 130,000 OWL instances, or around 5,000 files. After that, the graph shows a steep rise due to swapping.

The performance results show that the system in its current configuration is not suited for rules that need to be evaluated with a short reaction time (i.e., less than a few seconds). However, a reasonable number of instances for real world applications can be handled. Several changes of the configuration to increase reasoning performances are possible: The reasoning time can be greatly reduced, if only the closure of axioms the rules and instances depend on are loaded into the reasoner. Finding such a closure is not trivial because the corresponding subclass relations and associations have to be taken into account. Another possibility to reduce reasoning time is to reduce the instance count by adding only those file attribute instances relevant to the reasoning result to the ontology.

6 Summary and Future Work

In this paper we presented an approach for management of an IT system, based on the Common Information Model and a heavyweight ontology. Construction of a heavyweight ontology serves multiple purposes: It contains a complete domain model (defined as comprehensively as possible) in a format which is suitable for integration of the managed system with adjacent domains. For example, if processes in an ontology for semantic business process management (see e.g. [10]) refer to physical or logical IT systems, such references can be directly and unambiguously expressed. More importantly, the heavyweight ontology contains model instances that represent the current state of the real world managed system. This allows the definition of behaviour rules that are part of the ontology and reference both model and model instances.

As domain ontology, CIM was chosen, as it is a comprehensive and actively used domain model. As ontology language OWL was applied, as it is the de-facto standard for ontology modelling, and was shown to be the best choice for a translation of CIM. An overview to the approach for the translation of CIM into OWL was given.

A runtime system was presented which is capable of converting the CIM schema into the OWL format and can load the model ontology and statically defined rules in the SWRL format. Moreover, the runtime system acts as a CIM client to gather information about the current state of the managed system and converts this data into corresponding CIM OWL instances. A reasoner that is part of the runtime system is then used to evalute the rules using the model and the current instance data. Reasoning results are used in a reaction

module that can perform arbitrary reactions based each specific rule, in particular issuing CIM requests for calling CIM methods.

By implementing a belief revision operator according to the postulates from belief change theory and adjusted to ontology change, the runtime system component that adds and updates instance data in the ontology makes sure that the ontology stays consistent.

The CIM model is the basis for other standards, such as the storage standard SMI-S (Storage Management Initiative Specification) of the SNIA (Storage Networking Industry Association, [19]), and SMASH (Systems Management Architecture for Server Hardware), a DMTF standard for the unification of the management of data centers. This makes it possible to apply the presented approach directly to ontology-based storage management, where management rules can be specified in SWRL, and ontology-based virtual machine management (for example, VMware server provides a SMASH API). Especially in storage management, usually a large number of rules exist, which can be defined in a coherent manner with the model using the presented approach. As all necessary parts are implemented, application to storage and virtual machine management is investigated in the next step. This investigation will lead to a broader and more detailed evaluation scenario.

Future work also includes the performance optimizations discussed in section 5, especially automatically finding a closure of axioms required for rule evaluation and thus reducing the total OWL instance count. In the long term, the application of the approach to another domain is investigated, Ambient Assisted Living (AAL), where devices and services are combined to provide support for daily life of assisted persons.

References

1 Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50:510 – 530, 1985.

2 Carlos E. Alchourrón and David Makinson. On the logic of theory change: Safe contraction. *Studia Logica*, 44:405–422, 1985.

3 Jorge E. López De Vergara, Antonio Guerrero, Víctor A. Villagrá, and Julio Berrocal. Ontology-Based Network Management: Study Cases and Lessons Learned. *Journal of Network and Systems Management*, 2009.

4 Jorge E. López De Vergara, Víctor A. Villagrá, Juan I. Asensio, and Julio Berrocal. Ontologies: Giving Semantics to Network Management Models. *IEEE Network*, 17(May/June), 2003.

5 Jorge E. López De Vergara, Víctor A. Villagrá, and Julio Berrocal. Applying the Web ontology language to management information definitions. *IEEE Communications Magazine*, 42(7):68–74, July 2004.

6 Distributed Management Task Force. Common Information Model (CIM). `http://www.dmtf.org/standards/cim/`.

7 Giorgos Flouris, Dimitris Plexousakis, and Grigoris Antoniou. AGM Postulates in Arbitrary Logics: Initial Results and Applications, 2004.

8 Antonio Guerrero, Víctor A. Villagrá, Jorge E. López De Vergara, and Julio Berrocal. Ontology-Based Integration of Management Behaviour and Information Definitions Using SWRL and OWL. *Ambient Networks*, 3775:12–23, 2005.

9 Sven Ove Hansson. New operators for theory change. *Theoria*, 55:114–132, 1989.

10 Martin Hepp and Dumitru Roman. An Ontology Framework for Semantic Business Process Management. In *Proceedings of Wirtschaftsinformatik 2007*, Karlsruhe, 2007.

11 Uta Lösch, Sebastian Rudolph, Denny Vrandečić, and Rudi Studer. Tempus fugit - towards an ontology update language. In *6th European Semantic Web Conference (ESWC 09)*, volume 1, pages 278–292. Springer, January 2009.

12 Marta Majewska, Bartosz Kryza, and Jacek Kitowski. *Translation of Common Information Model to Web Ontology Language*, volume 4487 of *Lecture Notes in Computer Science*, pages 414–417. Springer Berlin Heidelberg, Berlin, 2007.

13 Object Management Group. Unified Modeling Language (UML). `http://uml.org/`.

14 OpenGroup OpenPegasus CIM/WBEM Manageability Broker. . `http://www.openpegasus.org/`.

15 Guilin Qi and Fangkai Yang. A survey of revision approaches in description logics. *Lecture Notes In Computer Science; Vol. 5341*, 2008.

16 Stephen Quirolgico, Pedro Assis, Andrea Westerinen, Michael Baskey, and Ellen Stokes. Toward a Formal Common Information Model Ontology, 2004.

17 Márco Moretto Ribeiro and Renata Wassermann. First steps towards revising ontologies. In *Proc. of WONRO'2006*, 2006.

18 Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *ACM SIGMOD Record*, 31(4):12–17, 2002.

19 Storage Networking Industry Association. . `http://www.snia.org/`.

20 Andreas Textor, Jeanne Stynes, and Reinhold Kroeger. Transformation of the Common Information Model to OWL. In Florian Daniel and Federico Michele Facca, editors, *ICWE 2010 Workshops*, volume 6385 of *Lecture Notes in Computer Science*, pages 163–174. Springer Verlag, July 2010.

21 The Scala Programming Language. . `http://www.scala-lang.org/`.

22 Debao Xiao and Hui Xu. An Integration of Ontology-based and Policy-based Network Management for Automation. In *International Conference on Computational Intelligence for Modelling, Control and Automation, 2006*, page 27, 2006.

23 Hui Xu and Debao Xiao. A Common Ontology-based Intelligent Configuration Management Model for IP Network Devices. In *Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC)*. IEEE Computer Society, 2006.

24 Hongwei Zhu and Stuart E. Madnick. A Lightweight Ontology Approach to Scalable Interoperability. *MIT Sloan School of Management Working Paper, 4621-06, CISL Working Paper, No. 2006-06*, 2007. http://hdl.handle.net/1721.1/37307.

TOGBAD-LQ - Using Challenge-Response to Detect Fake Link Qualities

Elmar Gerhards-Padilla, Nils Aschenbruck, and Peter Martini

University of Bonn - Institute of Computer Science 4
Römerstr. 164, 53117 Bonn, Germany
{padilla, aschenbruck, martini}@cs.uni-bonn.de

Abstract

The usage of link quality based routing metrics significantly improves the quality of the chosen paths and by that the performance of the network. But, attackers may try to exploit link qualities for their purposes. Especially in tactical multi-hop networks, routing may fall prey to an attacker. Such routing attacks are a serious threat to communication. TOGBAD is a centralised approach, using topology graphs to detect routing attacks. In this paper, we enhance TOGBAD with the capability to detect fake link qualities. We use a Challenge/Response method to estimate the link qualities in the network. Based on this, we perform plausibility checks for the link qualities propagated by the nodes in the network. Furthermore, we study the impact of attackers propagating fake link qualities and present simulation results showing TOGBAD's detection rate.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.85

1 Introduction

In tactical environments (i.e. military or disaster response scenarios) sensitive data (e.g. soldier positions) is transmitted via insecure links. In addition, there is a high probability of hostile units and the disturbance or eavesdropping of communication may lead to severe consequences, in the worst case even to loss of human life. Thus, secure communication is mandatory in tactical environments.

To enable secure communication, as a first step it is necessary to reliably enable communication. The performance of a network depends on the routing protocol used. Furthermore, the routing protocol's performance heavily depends on the routing metric used. Link quality based metrics have been shown to outperform simple routing metrics like minimum hop-count (e.g. [5, 20, 8]). These metrics should be used to provide high quality routes in the network and by that reliable communication. While it is undoubtedly reasonable to use link quality based routing metrics, it also opens an additional point of attack. An attacker may try to disturb network operation or eavesdrop traffic by propagating fake link qualities.

As mentioned, there is a high demand for security in tactical multi-hop networks. Fortunately, these networks possess a property that can be exploited for security purposes. In such scenarios, there typically exists a command and control structure. The communication necessary for this structure leads to two types of nodes: fully equipped and light-weight nodes. The fully equipped nodes have access to power supply and therefore use more powerful hardware. The light-weight nodes use battery-driven and therefore not so powerful devices. As an example one may think of a hostage rescue scenario, where the fully equipped node is an armoured vehicle, while the light-weight nodes are infantry units. We assume that key material is installed in advance of the tactical mission. Thus, all key material necessary is available at the nodes.

In the following, we consider insider attacks, i.e. attacks of nodes owning valid keys. For example, these attacks may be performed by attackers taking over nodes owning valid keys.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 85–96

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

One way to detect such attacks is to use intrusion detection, or to be more specific anomaly detection. TOGBAD [7, 6], is an anomaly detection approach using topology graphs to counter routing attacks in tactical multi-hop networks. It exploits the structure of tactical multi-hop networks by running the detection instances on the fully equipped nodes.

In its basic version TOGBAD is not able to deal with fake link qualities. Nevertheless, with its detection and communication infrastructure it provides a very good basis for detecting fake link qualities. Thus in this paper, we enhance TOGBAD with fake link quality detection capability. In detail, we use a Challenge/Response method to locally estimate the link qualities in the network. Based on this, we locally perform plausibility checks for the link qualities propagated in each node's neighborhood, send important observations to a central instance and centrally decide whether there is an attack. Furthermore, we present studies on attackers sending fake link qualities and an evaluation of TOGBAD's detection.

The remainder of this paper is structured as follows. We first introduce routing attacks (Sect. 2). Afterwards, we present related work in the field of detection and prevention of routing attacks in multi-hop networks taking into account fake link qualities (Sect. 3). The following section introduces TOGBAD, our approach to detect routing attacks and our recent enhancements to it (Sect. 4). After that, we first introduce our simulation environment (Sect. 5.1) and then show our evaluation concerning the quality of TOGBAD-LQ (Sect. 5.2). Finally, we conclude the paper (Sect. 6).

2 Routing Attacks

In wireless multi-hop networks (e.g. Mobile Ad Hoc Networks (MANETs), Mesh networks), every node of the network may be part of the routing process. Hence, it is quite easy for a node to influence routing in such networks. An attacker sending false routing information can try to attract routes and by doing so gain access to data transmitted in the network. Having gained this access, an attacker may try to achieve different goals: eavesdrop, manipulate or drop traffic. This kind of attacks is already mentioned in [12], [10],[11], and [13]. In this paper, we focus on an attack known as sinkhole attack.

2.1 Sinkhole Attack

The terms black hole, gray hole, and sinkhole attack are used for quite similar attacks. The approach of the attacker is the same for all three attacks. The attacker propagates fake routing information and by that attracts routes. The differences between the three attacks are the attacker's actions after having attracted routes. In a black hole attack, the attacker drops all traffic of the attracted routes. In a gray hole attack, the attacker selectively drops traffic and in a sinkhole attack selectively drops or manipulates traffic. In this work, we consider and detect an attacker propagating fake link qualities. The attacker does this to attract routes. Thus, we detect the basis for all three attacks and for our purpose it is not of interest whether the attacker tries to create a black hole, wormhole or sinkhole. Since the sinkhole attack is the most general one, we will use the term sinkhole attack in the following.

The actual implementation of a sinkhole attack strongly depends on the routing protocol and routing metric used in the network. Many protocols, such as Optimized Link State Routing (OLSR) [4], OLSRv2 [3] and Simplified Multicast Forwarding (SMF) [16], use so called HELLO messages to discover their neighborhood and link qualities. These messages are a good starting point for an attacker to propagate fake link qualities and, by doing so, gain a privileged position in the network.

2.2 Neighborhood Discovery Protocol

The use of HELLO messages to determine a node's neighborhood is standardized in the Neighborhood Discovery Protocol (NHDP) [2]. According to this draft, each node periodically spreads HELLO messages announcing its neighborhood. By these messages, each node learns about its 1-hop and 2-hop neighborhood. In this draft, link quality is only used locally to guarantee a minimum quality of links. The link qualities are not used in signaling nor in a link metric. The neighborhood discovery standardized in the Neighborhood Discovery Protocol is used by a variety of routing protocols like OLSR [4], OLSRv2 [3] or SMF [16]. However, in none of these protocols a link quality based routing metric is integrated, although it has been shown, that link quality based routing metrics lead to significantly improved network performance (cf. [5, 20]).

2.3 Expected Transmission Count

One link quality based routing metric is the Expected Transmission Count (ETX) [5]. Its idea is to minimize the number of packet transmissions required to successfully deliver a packet. It measures two kinds of link qualities to calculate the ETX-value: (1) link quality and (2) neighbor link quality.

The link quality (LQ) depicts the quality of the link in direction from the neighbor node to the node signaling the link quality. Hence, it is measured by the node itself. The neighbor link quality (NLQ) depicts the quality of the link in direction from the node to the neighbor node. Therefore, it is measured by the neighbor node. A correctly behaving node, takes the last link quality signaled by the neighbor as next neighbor link quality for the link to the neighbor. From link quality and neighbor link quality the ETX-value is calculated in the following way: $ETX = \frac{1}{LQ*NLQ}$

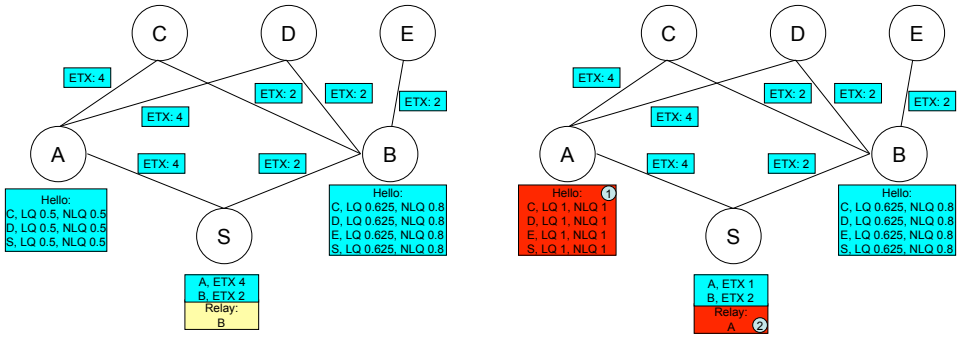
To clarify the signaling of link qualities, we consider a small example. Consider a link between nodes A and B. In the direction A→B 80% of the packets reach node B. In the opposite direction B→A 70% of the packets reach node A. Node A would signal a link quality of 70% and a neighbor link quality of 80%. Node A would know about the neighbor link quality value from node B's last HELLO message. In this message node B would have signaled a link quality value of 80%.

The ETX metric has been shown to significantly increase the network performance in testbeds (cf. [5]) and performs very well under real world conditions (e.g. in various Freifunk networks in combination with OLSR as routing protocol). Thus, in the following we use SMF with NHDP and ETX for routing purposes. To enable the use of ETX together with NHDP, link qualities have to be signaled. Thus, link qualities are added to the HELLO messages. Each node includes for each of its links a link quality and a neighbor link quality value in its HELLO messages.

2.4 Sinkhole in SMF with NHDP and ETX

SMF is a multicast forwarding approach suitable for multi-hop networks. Its basic idea is to forward multicast data using efficient flooding via a selected set of nodes, the relay set. SMF needs three logical components: Neighborhood Discovery Protocol, Relay Set Selection Algorithm, and Forwarding Process.

The Neighborhood Discovery Protocol specifies how nodes determine their neighborhood. Based on this neighbor information, the Relay Set Selection Algorithm chooses the set of relay nodes. On the basis of neighbor information and relay set selection, the forwarding process decides whether an incoming packet is forwarded by the receiving node.



(a) SMF without Sinkhole **(b)** SMF with Sinkhole

Figure 1 Example for SMF

In this paper, we focus on SMF using NHDP (cf. Section 2.2) and ETX (cf. Section 2.3). The neighborhood discovery is done according to NHDP. The relay set selection takes into account link qualities. The neighbor with the highest link quality and reaching at least one 2-hop neighbor not already covered by a relay node, is chosen as relay node. Relays are chosen until all 2-hop neighbors are covered. Each node signals its choice of relay nodes in its HELLO messages. Every node only forwards packets for neighbors that have explicitly chosen it as relay node.

Figure 1a gives a simplified example for SMF with NHDP and ETX. It shows a snapshot of a network after the relay set selection is done. The black lines represent the links in the network. Nodes A and B send correct HELLO messages. Node A propagates neighbors C,D and S with link quality and neighbor link quality 0.5 for each neighbor. This results in an ETX value of 4 for each propagated link. Node B propagates neighbors C, D, E and S with link quality 0.625 and neighbor link quality 0.8 for each neighbor. This leads to an ETX value of 2 for each propagated link. Hence, node S chooses node B as relay node. If node S sends data packets, only node B will forward these packets.

In order to run a sinkhole attack against SMF with NHDP and ETX, an attacker must fake HELLO messages, because they are used to provide the basic connectivity in the network. We consider an attacking node claiming to have high quality links to more neighbors than it actually has in its HELLO messages. In combination, the high quality of the attacker's links and his large number of neighbors lead to a high probability of the attacker being chosen as relay node, while other nodes are not chosen. By doing so, the attacker gains control of routes in the network. The more neighbors the attacking node claims to have and the better link qualities it propagates, the larger the potential impact of the attack. We note: There is a maximum impact achievable by the attacker. Of course, the more an attacker fakes, the larger the possibility of the attacker being detected.

Figure 1b shows the SMF network presented in Figure 1a, but this time node A has been taken over and launches a sinkhole attack. Node A propagates high quality links to nodes C, D, E and S (Label 1). Hence, node S does not choose node B, but node A as relay node (Label 2). Thus, node B does not forward data packets, but the attacker is responsible for forwarding these packets. Therefore, the attacker has gained control over the connection from S to C, D and E.

3 Related Work

There is only very little research done so far on detecting fake link qualities. Thus, in the following we mention not only related work dealing with fake link qualities, but also work at least being aware of link qualities.

[1] deals with securing the Pulse protocol against several attacks. The countermeasures include nonces, packet encryption and authentication. Against insider attacks, Awerbuch et al. use secure loss-rate information. By using cryptographic acknowledgements for each packet traversing a link, a secure loss-rate is determined for each link. These loss-rate information are used as a routing metric. Thus, a link quality based routing metric is introduced. Attacks against this routing metric are not considered in detail.

[15, 14] use intrusion detection to detect sinkhole attacks in wireless sensor networks. Several rules are defined to detect sinkhole attacks when using MintRoute or MultiHopLQI as routing protocol. Alteration of the attacker's link qualities is detected by plausibility checks. Each node acts as watchdog for its neighbors. It compares link qualities from incoming packets to link qualities propagated by other neighbors for the same links. If it detects a significant difference between claims of different neighbors, a fake link quality is assumed. This approach has two major drawbacks. First, without further processing it is only possible to indicate a sinkhole attack, but not to identify which node is launching the attack. Second, the approach requires the presence of watchdog nodes for each link. These watchdogs must be able to hear both nodes reporting a link between them. At least in a network with high mobility this requirement may be difficult to guarantee.

[19] defines a method to detect sinkhole attacks and identify the attacker in wireless sensor networks. It is assumed that data is transmitted to a base station consistently. Statistical methods are used to detect inconsistencies in data from a region. Upon detection of inconsistent data, a sinkhole is assumed. Thus, the base station initiates a procedure resulting in a flow graph for the suspicious region. The root of the biggest tree in the flow graph is assumed to be the attacker. The assumptions of this approach limit it to networks where traffic is transmitted to a base station consistently. Thus, the approach is not applicable to all kinds of multi-hop networks. In addition, fake link qualities are not considered.

[21] describes an algorithm to defend against selective forwarding attacks in wireless mesh networks. The detection happens in two phases. The first phase is threshold based. If for a given source-destination pair of nodes less than threshold packets are received by the destination node, a selective forwarding attack is assumed. The second phase is query based. The source node queries all intermediate nodes on the path to the destination. Based on their feedback, the attacker is determined. The detection threshold for the first phase is calculated according to the ETX metric. However, fake link qualities are not considered in this approach.

[15, 14] are the only approaches taking into account fake link qualities. However, these approaches are tailored to MintRoute and MultiHopLQI in wireless sensor networks, have some drawbacks (described above) and seem not applicable to tactical multi-hop networks. Thus, to the best of our knowledge, there are no approaches detecting fake link qualities in tactical multi-hop networks like our new approach TOGBAD-LQ does.

4 TOGBAD

In this section we present our approach Topology Graph based Anomaly Detection (TOGBAD). TOGBAD is a centralised anomaly detection method against routing attacks in

tactical multi-hop networks. It uses the structure of tactical multi-hop networks by running the detection routines centrally on the fully equipped nodes. Preliminary versions were introduced in [7], [6] and are summed up in Section 4.1. In Section 4.2, we present our new extension TOGBAD-LQ.

4.1 Basic Functioning

TOGBAD utilizes two types of instances corresponding to the two types of nodes present in tactical multi-hop networks. The sensor instances of TOGBAD run on the light-weight nodes of the tactical multi-hop network. These nodes act as watchdogs and periodically send reports to a detection instance. From these reports the detection instances construct a graph modeling the topology of the network. The detection instances run on the fully equipped nodes of the multi-hop network. They perform plausibility checks between the actual topology represented in the topology graph and the topology propagated by nodes in the network. By doing so, the detection instances are able to detect nodes propagating fake topology. However, without enhancements this version of TOGBAD is not able to recognize fake link qualities. For further details concerning the basic functioning of TOGBAD we refer to [7].

4.2 TOGBAD-LQ

The basic idea of TOGBAD-LQ is to use a Challenge/Response method to estimate the link qualities in a node's neighborhood. The nodes include challenges in their messages. Based on these challenges, neighboring nodes create responses. These responses serve as basis for the estimation of link qualities. We will describe the Challenge/Response method in more detail below.

Each node checks the link qualities propagated by its neighbors against its own estimation of the link qualities. Upon detection of a suspicious link quality, a node sends a report to a fully equipped node. At the fully equipped node the reports on suspicious link qualities are aggregated and if the evidence is sufficient the sinkhole attacker is identified.

TOGBAD-LQ can be divided into two parts. The first is the local detection part. It consists of the estimation and checking of link qualities and the checking of neighbor link qualities. The second is the global detection part, where the local detections are aggregated and sinkhole attackers identified.

4.2.1 Local Detection

The local detection consists of two parts: (1) Estimation and checking of link qualities and (2) checking of neighbor link qualities. For (1), each node adds a challenge to its HELLO messages. Upon reception of a HELLO message containing a challenge, the neighboring nodes add a response to this challenge in their next HELLO message. From the number of received responses from a neighbor, a node can estimate the link quality of this neighbor. The number of received responses is used for the checking of link qualities. If the difference between number of received responses and number of expected responses according to the link quality propagated by the neighbor exceeds a threshold, a report is sent to a fully equipped node.

The challenges and responses should be chosen in a way that it is very difficult for an attacker to guess a correct response without the corresponding challenge. In particular, the challenges should be independent of each other. Thus, we use random values as challenges.

We will describe a way to derive responses below. Information from HELLO messages should not be taken as challenges, since successive HELLO messages of the same sender tend to be similar. To minimize the overhead introduced by our approach and to provide tamper-resistance, the responder sends back a hash-based message authentication code (HMAC) calculated from the received challenge. By use of a hash, the overhead is minimized and by using a message authentication code tamper-resistance provided.

The checking of link qualities is done in the following way: Each node maintains a list of challenges sent for a given time frame. Additionally, each node has a list of received responses for each neighbor in the given time frame. A received response is considered correct, if it corresponds to one challenge not already met by the neighbor sending the response from the list of challenges. Let CS be the number of challenges sent in the considered time frame, B the node receiving a HELLO message and performing the check, A the sender of the HELLO message, LQ_A the link quality measured by node A for the Link B→A, LQ_B the link quality propagated by node B, ER the number of expected responses and RR the number of received correct responses to a challenge sent in the time frame.

$$ER := LQ_A * LQ_B * CS$$

$$local\_threshold := ER + \delta$$

A report is sent to a fully equipped node if

$$RR < local\_threshold.$$

The reports consist of a timestamp, the ID of the suspicious node, the ID of the reporting node and the difference between $local\_threshold$ and RR. In the following, this difference is named $diff$. The reports are added to the reports sent for the basic TOGBAD version, minimizing overhead and transmissions needed.

The checking of neighbor link qualities needs no additional signaling. It can be done completely locally. The neighbor link quality propagated by a neighbor is the link quality from a previous HELLO message of the node performing the check. Thus, to defend against fake neighbor link qualities, a node keeps a list of its previously propagated link qualities. To check the plausibility of a neighbor link quality propagated by a neighboring node, a node checks whether this neighbor link quality is present in its list of propagated link qualities for this neighboring node. Note that it is important to keep the lists of previously propagated link qualities short for two reasons: (1) to save resources of the nodes and (2) to reduce the probability of an attacker choosing a valid neighbor link quality. Please note that it is no problem if new nodes join in. New nodes will start transmitting neighbor link qualities for a neighboring node after they received at least one HELLO message advertising a link quality from this node.

4.2.2 Global Detection

The global detection is done at the fully equipped nodes. The approach works period-based. For a defined period the detector collects the reports sent by the watchdogs. At the end of the period, the detector aggregates all reports concerning one suspicious node received in the current period, calculates the mean over the $diff$ values from these reports and generates an alarm for a suspicious node, if

$$mean\_diff > global\_threshold.$$

To clarify our approach, we consider a simplified example (Figure 2). The example consists of three nodes. Node A is the attacker node, node B a benign node and node FE one fully equipped node. For this example, we set $CS := 10$, $RR := 2$ at node B, $\delta := -0.5$, $global\_threshold := 1$ and the link quality for link A-B := 0.5 in both directions. Thus, node B propagates a link quality and neighbor link quality of 0.5 for neighbor A in its HELLO message (Figure 2; Step 1). Additionally, node B adds a challenge to its message. Figure 2; Step 2 shows the subsequent HELLO message of the attacker node A. The attacker fakes the link quality and neighbor link quality and propagates the optimal value for both. Despite faking link quality and neighbor link quality, the attacker correctly includes the HMAC of node B's challenge as response in his HELLO message. Note that it would even simplify the detection of the attack if the attacker would not include responses in its HELLO messages. Upon reception of the attacker's HELLO message, node B first increments its RR to 3. Afterwards, node B checks the propagated LQ and NLQ. In this case, $ER = 5$ resulting in $local\_threshold = 4.5$. Thus, node B sends a report with $diff = 1.5$ to the fully equipped node (Figure 2; Step 3). Additionally, node B checks the NLQ propagated by node A. For this example, we consider a list length of 1 for the previously propagated link qualities. Thus, node B compares the LQ of its last HELLO (0.5) to the NLQ of node A's HELLO (1.0). Since the values are unequal, the fake NLQ is detected. At the fully equipped node, the mean over all reports concerning node A is calculated and compared to the $global\_threshold$. In this case: $mean\_diff(A) = 1.5 > 1 = global\_threshold$ (Figure 2; Step 4). Thus, the fully equipped node generates an alarm for node A.

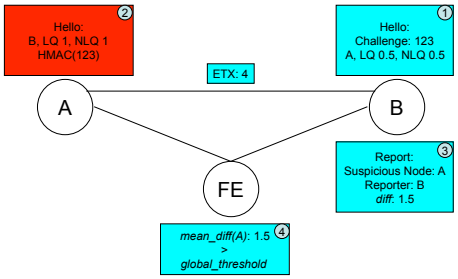


Figure 2 Example for TOGBAD-LQ

5 Evaluation

To evaluate our approach, we conducted simulations. In this section, we first introduce the used simulation environment (5.1) and afterwards present our simulation (5.2) results.

5.1 Simulation Environment

For our simulations, we use the network simulator ns-2 [17] in version 2.33. Since ns-2 does not include an implementation of SMF, we added SMF with NHDP and ETX (cf. Section 2.4).

Since we consider tactical environments, we choose Reference Point Group Mobility (RPGM) [9] as mobility model. In tactical scenarios there typically are two kinds of communication channels present, intra-group and inter-group. The intra-group communication is mainly single-hop as the nodes of a group (at least when using RPGM) stay close together. Typically, the communication between tactical command and the troops in the field is inter-group and therefore multi-hop communication. In the following the inter-group communication channel is depicted as command channel. To guarantee multi-hop scenarios, the RPGM scenarios are generated with the additional criterion that one node at maximum may reach 50% of all nodes within one hop.

The military units of the RPGM scenarios utilize a tactical MANET that is attacked by a sinkhole in the simulations conducted. All nodes are equipped with radio hardware with a maximum communication range of approximately 300m. A combination of log-distance and

ricean fading is used as signal propagation model. All packets are sent with a transfer rate of 11Mbps and are routed by applying the Simplified Multicast Forwarding (SMF) protocol with NHDP and ETX (cf. Sect. 2.4). The modelled military mission consists of several squads of infantry soldiers that are moving in an area of 1000m x 1000m. Each RPGM group consists of 10 nodes, which approximates the size of a military squad. The maximum distance between a unit and its corresponding group center is set to 300m to model a closely operating infantry squad. The total node number is set to 50. 400 replications of length 1000 seconds are done with varying movements and traffic. Additionally, only replications with a non-partitioned network for at least 90% of the simulation time are considered, since we assume that in a real tactical network some kind of topology control is in use.

From the 50 nodes simulated, we uniformly choose one attacker. The attacker sends fake HELLO messages. In these messages he propagates a fake topology and fake link qualities. According to the results of [7], the attacker propagates two-thirds of the total nodes as fake neighbors. Additionally, he sends optimal link qualities and neighbor link qualities for all propagated neighbors (real and fake ones). The attracted traffic is dropped by the attacker. The attacker is active between seconds 100 and 500. The rest of the simulation, the attacker node behaves like a normal one.

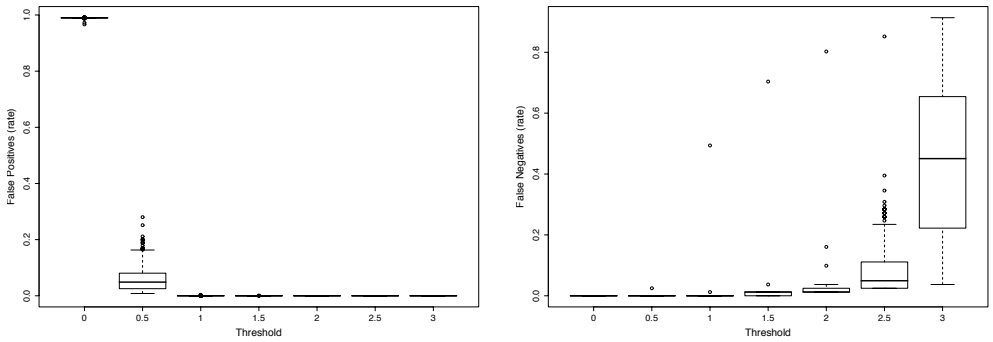
The report interval for TOGBAD should be chosen in dependency of the used HELLO interval. According to [4] we use a HELLO interval of 2 seconds. In [7] a report interval of 5 or 6 seconds was evaluated to work well given the above HELLO interval. In this work, we consider a report interval of 5 seconds. The detection of fake link qualities is done every 5 seconds, accordingly. Consequently, the interval for ETX determination should also be chosen in dependency of the HELLO interval. At least 10 HELLO messages should be evaluated for ETX determination. Thus, we consider an interval of 20 seconds. For plausibility checks of propagated link qualities and neighbor link qualities, only information from packets being considered for ETX determination are of interest. Therefore, each node stores its link qualities and challenges sent for 20 seconds.

5.2 Simulation Results

Concerning our simulation results, we start with an evaluation of a reasonable choice for the *global_threshold*. For this evaluation, every node sends a report to a fully equipped node, if the difference between *local_threshold* and *RR* is greater than zero. Afterwards, we consider the impact of bursty packet losses on the detection rate of our approach.

To find a reasonable choice for the *global_threshold*, we consider the rate of false positives and false negatives of our approach in dependency of the different choices for the *global_threshold*. We choose the *global_threshold* out of the set $\{0; 0.5; 1; 1.5; 2; 2.5; 3\}$. At first, we consider an idealized situation: We assume that the reports of our sensor instances are not hit by packet losses. Thus, we have an optimal information base at the detector instance. The impact of packet losses of the reports is evaluated later (Fig. 4).

Figure 3a shows the rate of false positives over different choices for the *global_threshold*. We use boxplots to show the median, lower and upper quartile. For a *global_threshold* of zero, our approach leads to a median false positive rate of almost 100%. But already the choice of 0.5 for the *global_threshold* reduces this rate to around 10%. A choice of 1 or greater leads to a median false positive rate of 0%. The very high false positive rate for small choices of the *global_threshold* is due to packet losses in the network. Packet losses may lead to small differences between the measured and real link quality. Thus, there are reports with $diff$ values slightly above zero for correctly behaving nodes. Given a *global_threshold* close to zero, these reports lead to a high number of false positives. For more reasonable

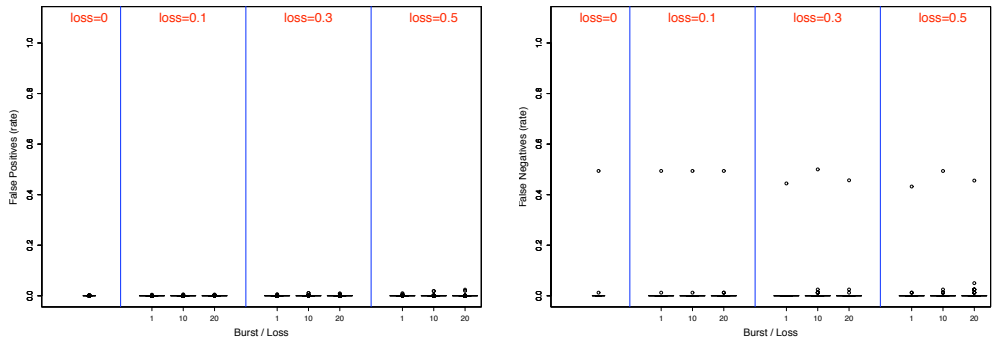


(a) False Positives **(b)** False Negatives

Figure 3 False Positives, False Negatives without packet loss for different global thresholds

choices of the *global_threshold*, these false positives disappear. Figure 3b uses boxplots to illustrate the rate of false negatives over different choices for the *global_threshold*. For small choices of the *global_threshold* the median false negative rate stays at 0%. From a choice of 1.5 the median rate of false negatives slightly increases up to 5% for a choice of 2.5 for the global threshold. The choice of 3 for the global threshold leads to a median false positive of 40%. The link quality of the majority of links in the considered scenarios is around 0.7. Given that the $global\_threshold \geq 3$, with high probability even a node propagating optimal link qualities (1.0) is considered correctly behaving. This leads to the high number of false negatives given $global\_threshold \geq 3$. According to Figure 3 $global\_threshold = 1$ is a reasonable choice. It leads to low false positives and false negatives rates and thus a very good detection rate. Hence, in the following we use a *global_threshold* of 1. There are few outliers in Figure 3b. Given a reasonable choice for the *global_threshold*, two conditions may lead to a high false negative rate: (1)very high link qualities in the direct neighborhood of the attacker node and (2)attacker node nearly exclusively reaches members of its movement group directly. Condition (1) leads to a low difference between the real link qualities and the fake link qualities propagated by the attacker. Condition (2) leads to a small number of watchdogs reporting over the attacker. The outliers visible in Figure 4 are the cases where Conditions (1) and (2) occur simultaneously. Nevertheless, the probability of both conditions occuring simultaneously is very low.

Figure 4 shows the impact of bursty packet losses on the detection rate of our approach. To model the bursty packet losses, we use a two-state Markov chain [18] with states loss and no loss. We vary packet loss rate and average burst length to measure both the impact of increasing packet loss rate and burst length. In Figure 4a the false positive rate is visualized over different loss rate/burst length combinations. Neither with increasing loss rate, nor with increasing burst length the rate of false positives increases significantly. The median of the false positive rate stays at 0% for all combinations. Only the number of outliers increases with increasing loss rate and burst length. Figure 4b illustrates the false negative rate over different loss rate/burst length combinations. Like the false positive rate, the false negative rate is very robust against both, increasing loss rate and increasing burst length. The median false negative rate stays at 0% for all loss rate/burst length combinations. Again, the number of outliers increases with loss rate and burst length. Figure 4 shows that our approach is very robust against packet losses. In tactical scenarios group based movement is to be expected.



(a) False Positives **(b)** False Negatives

Figure 4 False Positives, False Negatives with packet loss for a global threshold of 1

Due to the group based mobility each node has with high probability at least the members of its group as direct neighbors. Thus, -in the considered scenarios with a group size of 10 nodes- for each node there are with high probability at least 9 nodes reporting about its behavior. Even if 50% of the reports of these reporting nodes are hit by packet losses, there is a high probability of at least some of the reports reaching the detection instance. Already a small number of reports is sufficient to reach a very good detection rate.

6 Conclusion and Future Work

In this paper we introduced a new detection capability for fake link qualities to our intrusion detection approach TOGBAD. TOGBAD-LQ uses a Challenge/Response method to estimate link qualities in the neighborhood of nodes. Based on this estimation, first, a local plausibility check is performed. The results of these local plausibility checks are then aggregated at a central detection instance. This approach takes advantage of the characteristics of tactical environments by running the central detection instances on the fully equipped nodes in the network.

We considered an attacker launching a sinkhole attack by propagating fake topology and link qualities. To detect this attack, we evaluated a reasonable threshold for our approach. Given a reasonable choice for the required thresholds, our approach shows very good results. It leads to very few false positives and false negatives and was shown to be very robust against packet losses. In total, it reliably detects fake link qualities in tactical scenarios.

However, there are still questions which have to be examined in the future. An evaluation for a reasonable choice of the local threshold should be performed. Additionally, the overhead introduced by our approach has to be evaluated. Furthermore, the impact and detection of an attacker only slightly increasing its link qualities should be considered.

7 Acknowledgments

This work was supported in part by the State of North Rhine-Westphalia within the B-IT Research School.

References

1 B. Awerbuch, *et al.*, "Secure Multi-Hop Infrastructure Access," *Proc. of Network and Distributed System Security Symposium (NDSS)*, 2005.

2 T. Clausen, *et al.*, "IETF Draft MANET Neighborhood Discovery Protocol," *http://www.ietf.org*, 2009.

3 ——, "IETF Draft The Optimized Link State Routing Protocol version 2," *http://www.ietf.org*, 2009.

4 T. Clausen *et al.*, "RFC 3626 Optimized Link State Routing Protocol (OLSR)," *http://www.ietf.org*, 2003.

5 D. de Couto, *et al.*, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *Proc. of ACM Conference on Mobile Computing and Networking (MobiCom)*, 2003.

6 E. Gerhards-Padilla, *et al.*, "Enhancements on and Evaluation of TOGBAD in Tactical MANETS," *Proc. of the 27th Military Communication Conference (MILCOM)*, 2008.

7 ——, "TOGBAD - An Approach to Detect Routing Attacks in Tactical Environments," *accepted for Wiley Security and Communication Networks*, 2010.

8 M. Gerharz, "Stabile Kommunikation in Dynamischen Ad-hoc-Netzwerken," *GCA-Verlag*, 2006, Doctoral Thesis University of Bonn.

9 X. Hong, *et al.*, "A group mobility model for ad hoc wireless networks," *Proc. of ACM Int. Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 1999.

10 Y.-C. Hu, *et al.*, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," in *Proc. of ACM Int. Conference on Mobile Computing and Networking (MobiCom)*, 2002.

11 ——, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," in *Proc. of Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

12 J. Hubaux, *et al.*, "The Quest for Security in Mobile Ad Hoc Networks," in *Proc. of ACM Int. Symposium on Mobile ad hoc Networking & Computing (MobiHOC)*, 2001.

13 C. Karlof *et al.*, "Secure Routing in wireless sensor networks: attacks and countermeasures," *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, vol. 4837/2008, no. 2–3, pp. 293–315, September 2003.

14 I. Krontiris, *et al.*, "Intrusion Detection of Sinkhole Attacks in Wireless Sensor Networks," *Algorithmic Aspects of Wireless Sensor Networks*, vol. 48372008, 2008.

15 ——, "Launching a Sinkhole Attack in Wireless Sensor Networks; The Intruder Side," *Proc. of IEEE Int. Conference on Wireless and Mobile Computing, Networking and Communication (WIMOB)*, 2008.

16 J. Macker, "IETF Draft Simplified Multicast Forwarding for MANET," *http://www.ietf.org*, 2009.

17 S. McCanne *et al.*, "ns Network Simulator," *http://www.isi.edu/nsnam/ns/*.

18 B. Milner *et al.*, "An Analysis of Packet Loss Models for Distributed Speech Recognition," in *Proc. of the 8th Int. Conference on Spoken Language Processing (INTERSPEECH)*, 2004.

19 E. Ngai, *et al.*, "On the Intruder Detection for Sinkhole Attack in Wireless Sensor Networks," *Proc. of IEEE Int. Conference on Communications (ICC)*, 2006.

20 S. Roy, *et al.*, "High-Throughput Multicast Routing Metrics in Wireless Mesh Networks," *Proc. of IEEE Int. Conference on Distributed Computing Systems (ICDCS)*, 2006.

21 D. Shila *et al.*, "Defending Selective Forwarding Attacks in WMNs," *Proc. of IEEE Int. Conference on Electro/Information Technology (EIT)*, 2008.

Avoiding Publication and Privatization Problems on Software Transactional Memory

Holger Machens and Volker Turau

Hamburg University of Technology, Institute of Telematics
Hamburg, Germany
{machens,turau}@tuhh.de

Abstract

This paper presents a new approach to exclude problems arising from dynamically switching between protected concurrent and unprotected single-threaded use of shared data when using software transactional memory in OO languages such as Java. The approach is based on a simple but effective programming model separating transactions from non-transactional operation. It prevents the application programmer from errors but does not force the software transactional memory library to observe non-transactional access and thereby preserves modularity of the software. A prototypical toolchain for validation and source code instrumentation was implemented as a proof of concept.

Keywords and phrases Software Transactional Memory, Publication, Privatization

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.97

1 Introduction

Due to physical and economical reasons CPU manufacturers have stopped to increase CPU clock speed and now increase the number of cores on a single die instead. As software is constantly getting more complex and consumes more processing power with each new version today's software industry is coerced to move to parallel programming paradigms. Unfortunately parallel programming is a complex and error-prone discipline due to the necessity of data exchange between threads of execution whether by message passing or concurrent access to shared data. Access to shared data was traditionally controlled by locks that may cause deadlocks and do not scale well (cf. [6]). Another approach is the use of transactions as known from databases. Emerging from that discipline transactional memory [6] is currently discussed as a viable alternative for locks to ease parallel programming.

Transactional memory (TM) is a transaction-based concurrency control mechanism for shared non-persistent memory on a single machine. Some prototypes have been developed as hardware transactional memory (HTM) integrated into a CPU, e.g. Sun's Rock processor. Software transactional memory (STM, [11]), as addressed in this contribution, is implemented in software libraries usually providing transactions on heap memory of a process. HTM has not yet reached a productive state, therefore most research focuses on STM.

The underlying principle of software transactions is the repeated execution of a critical section to resolve occurring conflicts in terms of data consistency or overall progress (i.e. deadlocks). Data modifications are committed if no conflict occurred during a transaction and aborted (i.e. discarded) otherwise, resulting in a restart of the transaction. STM can be based on pessimistic or optimistic concurrency control techniques. While pessimistic STMs acquire locks before accessing shared data optimistic STMs do not. Thus, before the latter commits a transaction it checks if the order of concurrently occurred accesses on shared data is result-equivalent with some serial execution of the involved transactions (*serializability*).

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 97–108

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Generally, TM provides the opportunity to easily move to concurrent programming without most of the common problems. It has the potential to push the use of concurrency and thereby prevent the software industry from losses caused by the change of the architecture paradigm on the hardware level. But transactions have fundamental differences compared to traditional locks. Some common patterns to solve tasks in concurrent applications do not apply to transactions and result in errors when used. Thus, to ease the transition to concurrent programming through TM its usability is of higher concern; in particular the prevention from its own known problems.

One class of problems of TM is called the privatization resp. publication problem [12]. Both problems emerge from a strategy used by application programmers to switch between protected shared use and unprotected local use of data and result in inconsistencies when applied with transactions. The currently accepted solution for these problems is to enforce strong isolation, which means that transactions are isolated from modifications of other transactions as well as they are isolated from non-transactional modifications on shared data. This usually requires the STM library to observe non-transactional access to shared data, which either requires modifications of the runtime environment (e.g. the virtual machine) or modifications of the software which uses a library implemented on an STM.

But a library must be self-contained. Considering larger ecosystems, such as the Eclipse platform, where a data model and plug-ins working on that data model are provided by different parties such modifications in plug-ins reduce the attractiveness of the platform. But even those large ecosystems would greatly benefit from deadlock prevention provided by STMs. Firstly it is hard to agree on some locking protocol for objects of the model which prevents deadlocks and secondly the usage of a single lock for the entire model is very inefficient. Therefore, there is a strong need for an alternative solution.

We advocate an imperative style declaring a programming model which prevents common errors when using TM such as privatization and publication. The programming model is enforced by a validation tool and automated code instrumentation is used to achieve a simple and almost transparent API. The programming model is designed for Java STMs and it neither depends on language extensions nor forces the STM library to demand new requirements for the runtime environment or the non-transactional part of the software.

Instead of realizing strong isolation inside the TM system it is enforced by the coding rules of the programming model. All rules of the programming model are derived from the following goal: Transactions must operate in a predefined domain of the software only. Access to the domain is possible via methods of objects only, which automatically start transactions. This completely isolates transactions from non-transactional access logically and physically in terms of software architecture. Thus, it realises strong isolation without the disadvantages in respect to the remaining software system and its environment.

A prototypical implementation of a TM system based on a modified STM library was developed as a proof of concept. Following the main requirement to reduce complexity of concurrent programming it provides a very simple API and a toolchain consisting of a validator checking for violations of the programming model and an automated code instrumentation tool to enable transaction management for declared transactional objects.

Section 2 reviews details on problems resulting from the use of transactions, existing approaches and their pros and cons. Also relevant programming models for STM in Java are discussed. The next section explains the overall concept and the programming model. On this basis the developed prototype and required modifications of the STM library are covered. Section 5 contains a detailed discussion of the consequences on programming and impact in terms of non-functional properties of the STM library induced by the modifications such as

performance and scalability. The paper finishes with a conclusion and a prospective outlook.

2 Pitfalls of Existing Approaches

2.1 Publication and Privatization

As mentioned above, publication and privatization describe strategies of the programmer to implement switching between local unprotected and shared protected access to data objects.

Initially: `data = 42; _private = false;`	
Thread T1	Thread T2

1 atomic {
2
3
4 _private = true;
5 }
6 data = 0;
7
8
9
	1
	2 atomic {
	3 if (!_private) {
	4
	5
	6
	7 val = ++data;
	8 }
	9 }

May this code assign `1` to `val`?	

Table 1 Example of Privatization

Table 1 shows an example implementation of privatization. Thread T1 privatizes variable `data` and signals it to T2 by the flag `_private`. The code works fine with locks but represents a typical problem for transactions when executed in parallel line by line in the given order. Thread T1 expects the data to be private and accesses it non-transactionally but unfortunately thread T2 already tested the data to be public and reads it after the update of T1 in line six. When using conventional locks variable `val` will in T2 never get the value 1, this was the programmer's intention. When using TM instead one of the following may happen:

- The TM system treats the contention caused by the update in line four to be valid because there exists an equivalent serialization of both executions with the same result. This valid serialization is: Execute line seven before line four. It seems valid to the TM system because the access to data in line six is invisible to the transaction in T2 which means line four actually causes no contention.
- The TM system might even consider the contention in line four to be valid and will decide in line nine to rollback all changes. This includes to reset data to the state read at transaction begin, which was 42 and not 0.

A rare but even more obvious privatization problem occurs when trying to synchronize at an empty atomic block as demonstrated in Table 2. This is completely incompatible with transactions because no STM system will ever block a thread which is not accessing any data. But a synchronized block does!

These are just two examples of privatization problems and there are equivalent patterns in usage with publication [4]. Common to these problems is the unprotected use of shared data which is usually invisible to the TM system. The only way to overcome this problem is to protect any access to shared data which is potentially accessed by transactions concurrently. Such a TM system provides strong isolation, i.e. isolation of transactions from modifications of other transactions and from non-transactional modification as well. This is in contrast to weak isolation which isolates transactions from modifications of other transactions only.

Initially: `data = 42 ; _private = false;`	
Thread T1	Thread T2

`1`
`2`
`3 _private = true;`
`4 atomic {} ;`
`5 data = 0;`
`6`
`7`
`8`
	`1 atomic {`
	`2  if(!_private) {`
	`3`
	`4`
	`5`
	`6  val = ++data;`
	`7  }`
	`8 }`

May this code assign `1` to `val`?	

Table 2 Privatization using an empty synchronized block

To enforce strong isolation a TM implementation needs a way to observe non-transactional access to shared data. Regarding Java, this may be achieved by modifications to the runtime environment (e.g. the VM), the use of an agent at the JVMTI or JVMPI interface, or by source or byte code instrumentation. The latter methods work with insertion of code into non-transactional code which accesses the shared data to get notified when certain code lines are executed. In summary, the implementation of strong isolation requires the TM system to either modify the target runtime environment or parts of the program code. Thus, libraries implemented based on the TM system with strong isolation are no longer self-contained.

2.2 Related Problems

A known problem of transactions is caused by its all-or-nothing guarantee which requires rollbacks in case of inconsistent states of data. Rollbacks result in repeated executions of the same code sequence which is incompatible with operations performing data exchange with external entities not supporting transactions (e.g. a text console). A common solution to this problem is the use of so-called irrevocable transactions [9]: A transaction is made irrevocable when hitting a non-transactional method such as an I/O operation. All concurrently running transactions which might get in conflict with the irrevocable transaction are rolled back which guarantees the irrevocable transaction to commit without rollback.

A similar problem is the use of condition-driven synchronization (monitor, condition variables or similar wait/signal primitives) e.g. by calling `wait()` and `notify()` in Java (see Table 3). Usually this scenario is bound to a lock, protecting some variables and an expression based on a subset of those variables representing the condition expected by a waiting thread. Another thread, which changes variables referenced by the expression signals the changes and the waiting thread is resumed. The condition is checked again and either the loop gets restarted or the condition is met.

Waiting Thread	Signaling Thread

`1 synchronized (<lock>) {`
`2  while (<condition>) {`
`3  wait();`
`4  }`
`5 }`
	`1 synchronized (<lock>) {`
	`2  <condition> = true;`
	`3  notify();`
	`4 }`

Table 3 Blocking synchronization

The inherent problem of this scenario is that the waiting thread must release the lock to allow the signaling threads to change the variables referenced in the expression. There are several proposals to provide an equivalent mechanism for transactions (cf. [5], [2], [1]) but they all are complex, reintroduce the danger of lifelocks or deadlocks, or are of restricted use.

2.3 Programming Models of STMs

The most prominent way to declare code sections to be transactional is the use of `atomic` blocks (see Listing 1) as in the TM systems ATOMOS [2] and AtomJava [7].

Listing 1 Use of `atomic` blocks

```
1 atomic {
2   // transactional code goes here ...
3 }
```

Appropriate code to begin, control and end a transaction inside a block are transparently added by a compiler or by dynamic code instrumentation, thus rollback or commit is in most cases invisible to the application programmer. This approach requires an extension of the language causing problems with existing tools such as IDEs and compilers. Alternatively, transaction handling may be explicitly programmed by the application programmer using calls into the TM system. A similar approach is to provide transactional code sections to the STM system through an object with a designated method (e.g. `call()`) as in DSTM2 [8].

Another prominent object-oriented programming model for STM which provides transparent handling of transactions is to declare so-called transactional objects which support transactions. Transactional objects define transactional code sections in their methods and transactional data as member variables of the object. Programmers are still allowed to use even non-transactional access to transactional objects. There are examples such as Deuce [10] and Multiverse [13] which are based on this model and use annotations to declare objects and methods to be transactional (see Listing **??**).

Listing 2 Example for transactional objects in Multiverse label

```
1 @TransactionalObject
2 class MyTxObject {
3   private int x;
4   @TransactionalMethod
5   public int incX() {
6     return ++x;
7   }
8 }
```

Atomicity of transactional objects is just an optional feature as long as the underlying STM library does not enforce strong isolation. Member variables may be globally visible and accessed by non-transactional operations, and transactional methods are able to access non-transactional data.

The first approach proposing so-called type-level enforcement of strong isolation was published by Harris et al. when introducing composable memory transactions [5] as an extension for Concurrent Haskell. Concurrent Haskell requires operations performing I/O actions to be explicitly declared as IO actions. With the proposed extension by Harris et al. the programmer can also define so-called STM operations. These require a running transaction and are restricted to use non-I/O operations only. Thus, strong isolation is enforced through the type system.

3 Proposed Programming Model

When looking at the problems mentioned in Section 2 it seems to be non-trivial to use TM and it was no surprise when the question appeared whether TM was just a research toy [3]. In databases those problems simply do not occur, because transactional data and transactions are completely separated from non-transactional operations. This implicitly provides strong isolation in terms of TM and it even prohibits calls to non-transactional operations and use of blocking statements. Correspondingly, the fundamental and generic concept of our programming model is to move TM and associated transactional code into a closed part of the system which is separated from non-transactional code.

In order to reduce the complexity in concurrent programming our programming model addresses the following requirements:

- Avoidance of privatization and publication problems: To eliminate these problems the programming model should support strong isolation.
- Preserve software modularity: To reduce impact on the runtime environment and to keep software modular, strong isolation should be enforced through the programming model rather than through the STM library as explained in Section 2.1.
- Simple API: The application programmer should be free from explicitly controlling concurrency by calling methods of the STM library.
- No changes to the programming language's syntax: The syntax should remain unmodified to preserve compatibility with existing tools. Therefore, language extensions such as usage of new keywords (e.g. `atomic`) are excluded.

3.1 Overall Concept

As mentioned above, the generic fundament of the programming model presented here requires to move transactions and transactional data into a logically separated *transactional domain* of the software. The transactional domain is meant to guarantee consistency by restricting access to contained data and operations to transactions only.

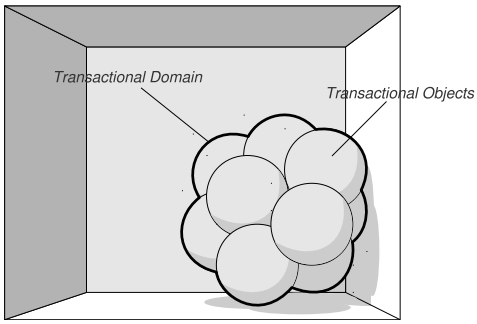


Figure 1 Transactional domain

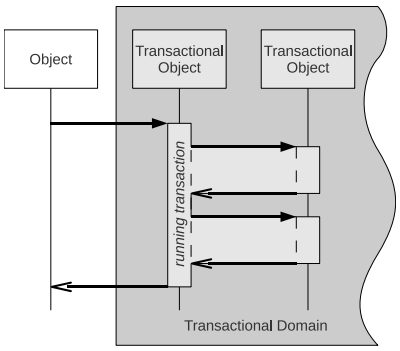


Figure 2 Implicit transactions

To achieve such a guarantee for Java the transactional domain is build up from transactional objects as depicted in Figure 1. Each transactional object is responsible to ensure guaranteed data consistency for its member variables. Each transactional object therefore offers no member variables but transactional methods only and each such transactional method transparently starts a transaction if not already running (see Figure 2). This way transactional data inside of transactional objects is accessed via transactional methods only which guarantees the consistency inside of them.

The interface of the transactional domain consists of the public transactional methods offered by the transactional objects. Crossing the boundaries of the transactional domain without starting a transaction is made impossible.

3.2 Coding Rules

Transactional objects are instances of transactional classes which are declared by implementing the empty interface `Transactional`. A class declared to be transactional will get all of its methods and constructors turned into transactional methods. Compared to annotations the usage of interfaces provides type-safety even for a development environment without the toolchain of the STM (e.g. when using a library with transactional objects). APIs of the STM can refer to objects implementing the interface `Transactional` while annotations need runtime checks or a tool for extended type checks at compile time.

A transactional class has to fulfill the following rules to guarantee strong isolation:

- Instance variables of transactional objects must be of primitive type, of type `String`, or references on transactional objects. This rule even prevents a programmer to put consistency on risk through methods returning references on internal non-transactional data or to keep temporary non-transactional data between two method calls. In Java primitives are returned by value and strings are constants. This makes modifications of both kinds of internal data impossible.
- Non-`final` instance variables of transactional classes must be `private`. By doing so, only methods - which start a transaction if necessary - can access internal variables.
- Inside of transactional methods any access such as method calls or variable access to external non-transactional objects is prohibited. That way it is impossible to break up strong isolation and get access to inconsistent data inside a transactional method. Usage of temporary local non-transactional objects in transactional methods (e.g. on the process stack) are allowed.
- Parameters of a transactional method must be of primitive type, of type `String` or transactional. Parameters provided to a method might be modified in the transaction and therefore need to support transactions as well. For all method parameters of primitive type, Java provides a call-by-value semantics. Thus, the method is working on its own copy and needs no concurrency control. The same applies to objects of class `String`.
- Arrays as parameters of transactional methods are prohibited because they do not support transactions. Arrays have to be replaced by a generic transactional wrapper class provided by the STM and supporting access to an underlying array.
- Base classes of transactional classes and sub-classes of transactional classes or interfaces must be transactional as well. This prevents a programmer from calling non-transactional methods of the base class or sub-class when expecting guaranteed consistency. It also allows to declare any implementation of an interface to be transactional. Methods of the super class `Object` are overridden by transactional methods or prohibited when final.
- Type arguments of transactional generic classes must be either of type `String` or transactional. Therefore, a transactional generic class declaration must have at least one transactional type bound (e.g. `<T extends Transactional>`) or a type bound of type `String`. Generics allow to implement type-safe transactional versions of standard interfaces such as `Collection`, `Set`, `List` or `Map`.

Use of non-transactional irrevocable operations is implicitly prevented by the rules. Additional rules are required to guarantee deadlock-freedom. This is not addressed here but is needed to meet the requirement of a simple API. Therefore, explicitly locking or

suspending of threads inside transactional objects using `wait`, `notify`, `notifyAll`, or keyword `synchronized` is prohibited. There are already approaches to support blocking transactions and irrevocable transactions (see Section 2.2) but those are not addressed in this paper.

3.3 Ramifications on Software Development

The proposed programming model prevents a programmer from errors concerning privatization or publication because it is impossible to use shared transactional data in an unprotected local fashion. In contrast to other approaches the proposed programming model primarily expects the data to be declared transactional and not only the critical code sections which leaves the responsibility to avoid those problems at the application code.

The programming model guarantees consistency for individual transactional objects only. To achieve consistency of an operation over a set of transactional objects a programmer still needs to move the operation into a transactional method. Appropriate rules, to force a programmer to do this could consist of an access restriction on transactional objects to transactions and the requirement for explicit declaration of intended non-transactional access to transactional objects. But both rules are at odds with the requirement to enable the implementation of self-contained libraries with transactional objects.

Currently the only way to privatize data and to get the performance advantage of unprotected data access is to create a corresponding non-transactional class to receive a copy of the data of a transactional object. This method will be applicable on certain but not all use cases because copying may produce a lot of overhead and narrows the intended speedup. There exist other feasible techniques to support privatization and publication through the STM system (cf. [12]). Those techniques have smaller per-call overhead than transactions and very low initial overhead compared to copying. Thus, a preferable alternative is to provide such a mechanism and offer an appropriate simple API for the following operations:

- **privatize:** Switches visibility mode of the transactional object to private, i.e. moves the object into the local area of a thread. Exactly one thread gets the owner of the object and has exclusive access to it. Transaction management on this object is turned off but the correct ownership of accessing threads is still checked. A thread trying to access a privatized object of another thread causes a runtime exception to occur.
- **check visibility:** Allows every thread to identify the current owner of the object, e.g. to check if the data is currently privatized. This is the only operation of a privatized object available to threads not owning the object.
- **publish:** Switches visibility mode of the transactional object back to public and returns it into the shared area. The method is available to the current owner of the object as long as the object is private. It turns on transaction handling for this object.

All three operations should support transaction handling. This allows for example to perform the visibility check and the intended access to the data in a transaction.

Privatization and publication follow a general pattern which is supported by these operations: The visibility of an object is indicated by a flag which must be considered by all threads prior to an access to a potentially privatized object. A thread can neither privatize, publish or access an object owned by another thread. To prevent programmers from any error in usage with this pattern each access violation according to this pattern leads to a runtime exception. There are other patterns which rely on the modification of the reference on the shared object. When the reference on a privatized object is redirected to a new object or null or has been removed from a list, the object is simply no longer reachable for other threads. To get the performance advantage in this case the programmer still needs to use

the privatization operation with the modification of the reference. Thus, combined with transactions these operations are enough to support all use cases addressed by privatization.

By the restrictive rules of the programming model, a programmer is forced to decouple transaction enabled code from unprotected code. Thereby a programmer merges transactional data and related code which should be protected by transactions in the transactional domain. Each operation on transactional data will be delegated to transactional methods inside the transactional domain. In larger ecosystems, such as the Eclipse platform, there will exist even a three level architecture as depicted in Figure 3. A lower level will provide simple transactional objects with just getter and setter methods representing the shared data model (data objects). The programmer of a specific software package for the ecosystem will create some additional transactional objects offering transactional methods for complex operations of its specific application on top of this level (data access services). The non-transactional application logic operating on the transactional shared data model will be on the highest layer (application objects). Thus, there will be three layers separating basic data objects from some application specific data access service objects and the application logic on top.

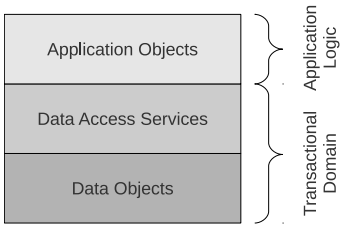


Figure 3 Sample application

Figure 4 Toolchain for the compilation

4 Prototypical Toolchain

To support the designed programming model a prototypical toolchain has been developed (see Figure 4). It consists of a validation tool which checks the given source code for conformance with the coding rules (*Validation*) and an instrumentation tool which injects code for transaction handling in transactional objects (*Instrumentation*) prior to compilation.

Source code that does not pass the validation is not given to the instrumentation. An advantage of the separation of both tools is that the instrumentation tool can be easily exchanged with an instrumentation tool using another instrumentation style such as byte code instrumentation or supports another STM library.

The usage of AtomJava, which supports source code instrumentation, allows the debugging of instrumented classes. The following modifications and extensions were incorporated in AtomJava to enforce the guaranteed strong isolation and modularity of transactional objects.

- The STM library now supports transactional constructors keeping the order of constructor calls in respect to the inheritance dependencies.
- The initialisation expressions of Java member variables are executed before a constructor is called. Those expressions may contain method calls to other objects. Because a transactional object must guarantee consistency throughout its life cycle, the initialisation of member variables of a class was made transactional as a whole.
- Initialisation of static variables of a class takes place at class loading time. To guarantee consistency of those initialisations too, they are now put into one transactional method executed at class loading time.

- AtomJava replaced the main thread by an instance of `AThread` requiring an instrumentation of method `main`. This provides fast access to thread context information but it requires modifications in the environment. The dependency on class `AThread` was removed and a thread context object stored in thread local storage has been introduced.
- AtomJava uses a special strategy for lock handling to implement deadlock detection. A thread holding locks keeps them as long as no other thread signals its need for it even when the transaction has already been finished. The owner of the lock constantly polls for lock requests of other threads and releases them on demand. This required instrumentation of any source code which uses the STM and is in conflict with the modularity requirement. Therefore, all locks are made available immediately after a transaction has completed and the instrumentation of calling source code is discarded.

5 Evaluation of the Prototype

The compliance with the functional requirements such as avoidance from privatization and publication, consistency of class and object instantiations, has already been tested during development. Of major interest for the evaluation was the impact of required modifications to the STM library even with this first not optimized prototype. A second interest was to show the difference of source code instrumentation and byte code instrumentation which actually provides more possibilities for optimizations.

The evaluation was performed on a shared memory system with two Intel Xeon Quadcore X5365 processors at 3.00GHz, 16GB DDR RAM memory, a Linux Debian operating system with 64 bit kernel in version 2.6.26-2 supporting SMP and a Sun Java VM version 1.6.0_21 also supporting 64 bit using the incremental garbage collector to get more precise results.

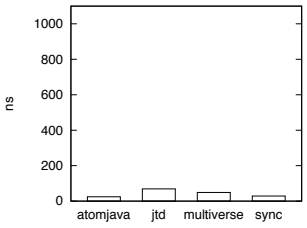


Figure 5
Method calls

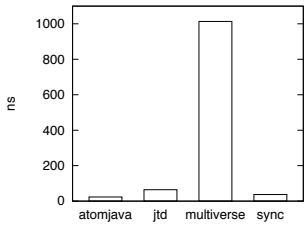


Figure 6
Object instantiations

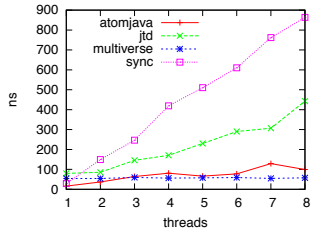


Figure 7
Hotspot behaviour

Measurements were made over 10^7 iterations of a loop over a single test unit (method or constructor call) which have been repeated one hundred times to check the precision of the calculated average. Measured constructors and methods contained a read and a write operation on a four byte integer instance variable. All measurements were performed for instrumentations with AtomJava (*atomjava*), our modified version (*jtd*), Multiverse with byte code instrumentation (*multiverse*) and for comparison with `synchronized` versions of methods and constructors (*sync*) which provides similar abstraction. The difference of the calculated average to the average duration of the same method or constructor without concurrency control provided the average overhead of a concurrency control mechanism.

Figure 5 and 6 show the average overhead of method calls and object instantiations over ordinary calls or instantiations without concurrency control. As expected the modified version shows a significant additional overhead compared to the original AtomJava instrumentation but it is still similar to the overhead produced by Multiverse and even lower in case of

transactional object instantiations.

When the number of threads concurrently calling a method is increased the impact of the access to thread local storage is getting more obvious (see Figure 7). This is the main difference to the original version of AtomJava. While Multiverse shows a nearly constant behaviour the overhead of the modified AtomJava version increases with each additional thread but is still far less than the overhead of the synchronized version.

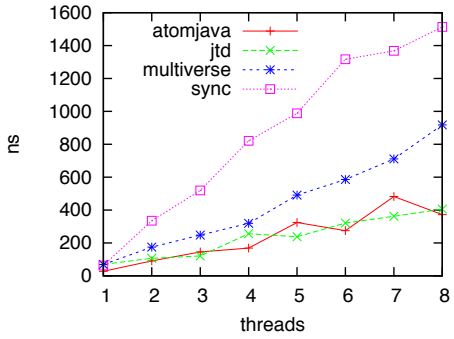


Figure 8 Load distribution 1:10

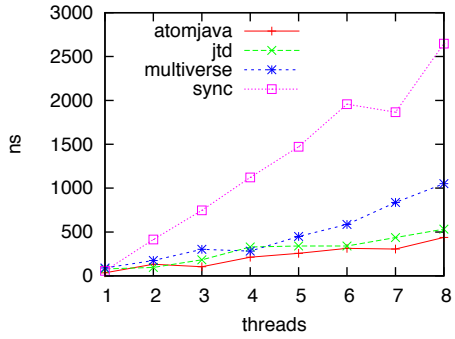


Figure 9 Load distribution 10:1

STM is known to have problems with so-called hotspot objects, which are rapidly accessed by many threads. Therefore, another measurement was performed where the amount of operations a thread performs inside and outside the instrumented method was synthetically controlled by iterations of a loop with read/write operations (approximately ten Java byte code operations) on a shared four byte integer variable. Figure 8 shows an example where the load inside the instrumented method is ten times higher than the load the thread spent outside of the method. In this case the modified version performs better because the effort to access thread local storage is lower compared to the total work inside the transaction. It is even better than Multiverse and has nearly the same performance as the original AtomJava STM system. When increasing the work outside the method as depicted in Figure 9 the overhead of the instrumentation by the modified version rises again due to the same reason of the increased access to the thread local storage.

6 Summary and Future Prospectives

This paper proposes an alternative programming model for Java STMs which prevents a programmer from errors with privatization and publication in use of STM. The fundamental concept is to keep transactions and transactional data in a closed domain. The model relies on source code instrumentation to provide a simpler API. The concept keeps libraries using STM self-contained preventing an impact on software which uses the STM-based library and the target runtime environment. Ramifications on the software development process are discussed and a proof of concept is given by a prototypical implementation of a toolchain consisting of a validation tool and an instrumentation tool based on AtomJava. The evaluation of the prototypical instrumentation tool shows a significant overhead due to the necessary modifications to the original instrumentation tool which is still lower as the overhead produced by Java monitors.

Existing STM systems have been predominantly developed with the focus on performance and transaction throughput and already reached acceptable quality compared to traditional locks. But the complexity of the APIs is still high and significantly reduces the advantages

over traditional locks such as deadlock-freedom. The work presented here is an attempt towards a simpler API preventing a programmer from errors while using an STM system. The overall goal of this work is to ease the move to parallel programming. But there are still open issues left which are considered in the design of the programming model and will be addressed in subsequent work. The issues include support for privatization and publication by the STM system as described above, blocking transactions, irrevocable transactions, and long running transactions. For all concepts elaborated approaches already exist and it may be possible to improve them in order to achieve simpler error-preventing APIs.

References

1 A.-R. Adl-Tabatabai et al. Compiler and runtime support for efficient software transactional memory. In *PLDI '06: Proc. ACM SIGPLAN Conf. on Programming Language Design and Implementation*, pages 26–37, New York, USA, 2006. ACM.

2 B. D. Carlstrom et al. The Atomos transactional programming language. *SIGPLAN Not.*, 41(6):1–13, 2006.

3 C. Cascaval et al. Software transactional memory: Why is it only a research toy? *Queue*, 6(5):46–58, 2008.

4 V. Menon et al. Practical Weak-atomicity Semantics for Java STM. In *SPAA '08: Proc. Annual Symp. on Parallelism in Algorithms and Architectures*, pages 314–325, New York, USA, 2008. ACM.

5 T. Harris, S. Marlow, S. P. Jones, and M. Herlihy. Composable memory transactions. In *PPoPP '05: Proc. 10th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pages 48–60, New York, USA, 2005. ACM.

6 M. Herlihy and J. E. B. Moss. Transactional memory: Architectural support for lock-free data structures. *SIGARCH Comput. Archit. News*, 21(2):289–300, 1993.

7 B. Hindman and D. Grossman. Atomicity via source-to-source translation. In *MSPC '06: Proc. Workshop on Memory System Performance and Correctness*, pages 82–91, New York, USA, 2006. ACM.

8 M. Herlihy; V. Luchangco and M. Moir. A flexible framework for implementing software transactional memory. *SIGPLAN Not.*, 41(10):253–262, 2006.

9 A. Welc; B. Saha and A.-R. Adl-Tabatabai. Irrevocable transactions and their applications. In *SPAA '08: Proc. 20th Annual Symp. on Parallelism in Algorithms and Architectures*, pages 285–296, New York, USA, 2008. ACM.

10 G. Korland; N. Shavit and P. Felber. Noninvasive java concurrency with deuce stm - an extensible java stm framework. poster. In *SYSTOR 2009: The Israeli Experimental Systems Conference*, 2009.

11 N. Shavit and D. Touitou. Software transactional memory. In *PODC '95: Proc. 14th Annual ACM Symp. on Principles of Distributed Computing*, pages 204–213, New York, USA, 1995. ACM.

12 M. F. Spear, V. J. Marathe, L. Dalessandro, and M. L. Scott. Privatization techniques for software transactional memory. In *PODC '07: Proc. 26th Annual ACM Symp. on Principles of Distributed Computing*, pages 338–339, New York, USA, 2007. ACM.

13 P. Veentjer. Multiverse - Software Transactional Memory for Java, 2009. `http://multiverse.codehaus.org`.

A Resilient and Energy-saving Incentive System for Resource Sharing in MANETs*

Holger Teske, Jochen Furthmüller, and Oliver P. Waldhorst

Institute of Telematics,
Karlsruhe Institute of Technology
Zirkel 2, 76131 Karlsruhe, Germany
holger.teske@student.kit.edu, furthmueller|waldhorst@kit.edu

Abstract

Despite of all progress in terms of computational power, communication bandwidth, and feature richness, limited battery capacity is the major bottleneck for using the resources of mobile devices in innovative distributed applications. Incentives are required for motivating a user to spend energy on behalf of other users and it must be ensured that providing these incentives neither consumes much energy by itself nor allows for free-riding and other types of fraud. In this paper, we present a novel incentive system that is tailored to the application scenario of energy-aware resource-sharing between mobile devices. The system has low energy consumption due to avoiding the use of public key cryptography. It uses a virtual currency with reusable coins and detects forgery and other fraud when cashing coins at an off-line broker. A prototype-based measurement study indicates the energy-efficiency of the system, while simulation studies show its resilience to fraud. Even in scenarios with 75 % of fraudulent users that are colluding to disguise their fraud only 3.2 % of them get away with it while the energy overhead (about 3 %) for the incentive system is still moderate

Keywords and phrases incentive system; resource sharing; fraud detection

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.109

1 Introduction

Current sales figures[1] indicate a significant increase in the popularity of smart phones and evidently show a trend towards feature-rich mobile devices. Besides offering computing and storage resources almost comparable to desktop PCs ten years ago, such devices offer a variety of other resources, including different communications capacities like 3G, WiFi, and Bluetooth, as well as sensors for position, acceleration, light, and temperature. Combining the resources provided by multiple devices enables new and exciting applications. These are typically observed as a natural subset of pervasive computing [25] and find increasing interest in many other disciplines of distributed computing, e. g., in Grid computing [10] and service overlays [5]. Example applications range from pooling capacities of the cellular connections of multiple devices to speed up downloads [3] to people-centric sensing exploiting the sensors of thousands of smart-phones [8].

Unfortunately, despite of the growth in resource variety, processor speed, memory size, and communication bandwidth, battery capacity remains the limiting factor for realizing the vision described above [18]. Providing resources for applications running on remote devices may consume a significant amount of energy, limiting the operating time of a mobile device for the owner's personal use. In fact, mechanisms are required to motivate device owners—that are not known to each other in general and, thus, do not pursue a common goal—to spend energy on behalf of others. Such mechanisms can be provided by *incentive*

* This research is supported by the "Concept for the Future" of Karlsruhe Institute of Technology within the framework of the German Excellence Initiative.

[1] http://www.gartner.com/it/page.jsp?id=1306513

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 109–120

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

systems. These systems could recompense the energy spent for serving a remote resource request, and allow to use the refund in turn to recompense others for using their resources.

Many incentive systems for motivating cooperation among users have been proposed with different application scenarios in mind, e. g. MilliCent [20], NetPay [9], and Micromint [24]. However, most of them cannot be used to motivate resource sharing among mobile devices, since they either require trusted hardware, connections to a central broker or other third parties on each interaction that requires a refund, or utilize refunds that cannot be reused without opening the door for fraud. An even more important drawback when it comes to providing incentives for spending energy is that most systems consume lots of energy by themselves, e. g., by requiring the use of public key cryptography on each payment, contradicting the primary goal of the incentive system.

In this paper, we present an energy-efficient and resilient incentive system tailored to the use-case of recompensing the energy required for resource sharing. For achieving energy-efficiency, the system works entirely without public key cryptography. Nevertheless, it is based on a virtual currency that can be exchanged between participants without communication with a central broker or any other third party on each payment. An amount earned by providing resources for a remote device can be re-used for consuming remote resources from other devices. Multiple-spending of coins is persecuted by posterior fraud detection when coins are cashed by an off-line broker.

We illustrate the main benefits of our incentive system—energy-efficiency and resilience—in in-depth performance studies. By prototype-based measurements we show that using our system increases the energy consumption only marginally compared to a resource-sharing system that does not provide incentives. Furthermore, in a simulation study considering different fractions of malicious users, we show that fraud can be detected with high probability. Even in a worst case scenario with 75% of all participants being colluding malicious users only 3.2% of them get away without being punished for their fraud for a short period of time. These results clearly illustrate that the proposed system can successfully recompense energy for servicing remote resource requests with low energy-consumption by itself and high resilience to fraudulent users.

The remainder of this paper is organized as follows. Section 2 reviews existing incentive systems with focus on the design goals energy efficiency and resilience. The concepts of our incentive system are described in Section 3. Section 4 presents the mechanisms that are applied for fraud detection when cashing coins at the off-line broker. Evaluation results with respect to the energy consumption and the probability of fraud detection are provided in Section 5. Finally, concluding remarks are given.

2 Requirements and Related Work

Subsequently we present a reference scenario to show the requirements that we hold necessary to be fulfilled: Imagine a group of tourists gathering around a point of interest. Assuming that their mobile devices advertise the resources they can provide in a MANET a camera may add location tags to taken pictures by obtaining GPS readings from a nearby navigation device. Beyond offering remote access to a single resource, a device can offer remote services that combine multiple resources. For example, exploiting its GPS and WAN links a PDA can offer a service for location-tagging and gallery-upload of a picture.

Providing such services is expensive in terms of energy. That is why there must be some kind of incentive system that recompenses users who provide services to other users. Based on the described scenario we see the following requirements for such an incentive system:

- *No need of trusted hardware:* Incentive systems that depend on trusted hardware modules are applicable only on a subset of the currently widespread mobile devices. The more requirements an incentive system makes with respect to the using devices the smaller the group of potential users will become.

- *No need of always available central infrastructure:* Long-living connections to central infrastructure components via wireless WAN technology considerably decrease the battery lifetime of mobile devices [14]. The incentive system itself should not be a major consumer of energy itself.
- *No involvement of third party:* Including a third party in every payment increases the communication overhead considerably. Either an internet connection is needed every time a user wants to consume or provide a service or the involved party must be nearby in the very same MANET. Both assumptions are to strong restrictions for our use case.
- *Vendor-independent currency:* Currency that can be used only to pay a specific vendor restricts the usability in scenarios in which every participant is a potential resource provider *and* consumer.
- *Reuse of currency possible:* Currency that can be earned and spent several times allows participants to achieve liquidity by providing services. So there is a real incentive to provide resources and services.
- *No need of public key cryptography:* Creation and verification of digital signatures as well as de- and encryption of data using public key cryptography is a demanding task with respect to CPU capacity.
- *Fraud detection possible:* In case malicious users commit fraud it is a desirable property of an incentive system, that the fraud can be detected and the initiator can be tracked down.

There is plenty of related work on accounting and creating incentives in decentralized networks. According to the classification presented by Obreiter and Nimis [12] incentive schemes can be categorized into trade based and trust based systems. Examples and discussion of trust based systems can be found in [16], [6]. As good reputation only gives good return in settings with stable cooperation patterns [12] and our use case assumes flexible and often changing cooperation patterns our approach is a trade based incentive system. Subsequently we will explain in which aspects our approach differs from other trade based incentive systems.

Buttyán and Hubaux [7] present an incentive system that stimulates cooperation in MANETs by introducing a so called *nuglet counter*. This counter is incrementend whenever foreign data is forwarded and decremented whenever own data is sent. As a positive counter is required to send own data, there is a strong incentive to behave cooperatively. To avoid users to manipulate this counter is is secured by a tamper resistant security co-processor. Although this approach might be used for other resources than forwarding capacity as well, the need for a trusted hardware module on each participating device contradicts our first requirement.

Liebau et al. [19] propose a token-based accounting system for P2P-systems. Although they provide a concept for accounting without central infrastructure their scenario differs in an important point: As we focus on resource sharing on *mobile* devices it is most important to us not to spend to much energy on the incentive system itself. Otherwise the accounting and incentive system would not encourage people to share resources at all. As we want to build an incentive system that is tailored especially to the needs of battery driven mobile devices, we try to go without the means of public key cryptography. According to Rivest et al. [24] the usage of public key cryptography for securing electronic currency is rather expensive in terms of CPU load and thus also in terms of energy: They estimate that the computing time needed for verification of an RSA signature is 100 times higher than computing a hash function, generating an RSA signature takes even 10000 times longer than computing a hash function.

Other payment schemes that apply public key cryptography are P-Pay [26], Sprite [27], and the work by Blaze [4] and Abdelkader [1]. Another important difference is the value of a token. The tokens proposed by Liebau et al. do not have any intrinsic value whereas each coin that we use in our approach represents a fixed value. How this specific value can be determined is discussed in Section 3.2.

There are incentive systems that do not make heavy use of public key cryptography. However, some of them do not offer a currency that can be used for several subsequent payment operations. Millicent [20], NetPay [9], and Micromint [24] for example assume that the means of payment are redeemed after a single payment. In our use case this would mean that each participant must either possess a larger amount of electronic currency to maintain liquidity for a sufficient period of time or that he must get small amounts of electronic currency more frequently.

Bocek et al. propose a private and shared history-based incentive mechanism (PSH). However, their approach is based upon an assumption that is fundamentally different from ours: Bocek et al. assume that resource interests in the resource sharing system are asymmetric, i. e. that an entity that provides a resource to another entity is not interested in the resources that this one is sharing. In contrast we do not take that assumption but hope that resource consumers also offer resources that are attractive to resource providers. Besides, PSH also makes use of public key cryptography, which we do not for reasons that are explained above.

In PeerMint [13] a decentralized accounting system for P2P applications is provided. In contrast to our approach it tries to accomplish a system without any central component whereas we rely on a central broker although this component of our system may be reachable only casually. In order to guarantee high scalability and robustness it is built upon a structured P2P overlay network. As we are considering use cases with only a small number of users we do not base our approach on such a system that comes along with a considerable communication overhead.

Another distinguishing feature is the scale of the incentive system. As our reference scenario that was described before shows we are considering resource sharing in small MANETs. This means that the cost for implementing large scale solutions as proposed for example in [17] and [6] is not adequate.

3 An Incentive System for Resource Sharing in MANETs

Our resilient and energy-saving incentive system for resource sharing in MANETs uses electronic currency to compensate resource providers for their efforts. We use the MicroMint system [24] to create such "digital coins". A coin consists of a certain number of values that create a hash collision when used as input for a certain hash function. It is very hard in terms of computing time (and thus energy) to create such collisions but easy to verify if certain values create such collisions. This means it is hard to create coins (even on strong computing machines), but easy to check, whether a coin is valid (even on poorly equipped mobile devices). There are a couple of techniques to vary the effort that is needed to create valid coins. That way it is possible to make it unprofitable if not impossible to forge a substantial amount of coins. The coins are generated on a central system, that has plenty of resources compared to the mobile devices that are supposed to use the coins.

Coins that are constructed following the MicroMint approach are actually not made for multiple payments. They are valid only for a specified amount of time and they are supposed to be used only once. However, we see the option of using earned coins for consuming services as a major motivation to provide services. Furthermore, redeeming the coins after using them only one time would increase communication with the broker and would therefore mean higher energy expenses. Thus, we developed a system that allows to use MicroMint based coins in multiple payments. As this opens up several opportunities to commit fraud for malicious users we also present a technique to track down fraudulent users. Thus, we explore the design space that is spanned by the trade-off security vs. energy consumption.

In the remainder of this section we will clarify the assumptions that our work is built upon before we give an overview on the entire system.

3.1 Assumptions

Our incentive system and the technique for tracking attempted and committed fraud relies on several assumptions that are reasonable with respect to the use case presented in Section 1.

- Participants can be identified: Tracking down and punishing malicious users requires a way to identify each user. In a lot of incentive systems this is achieved by digital signatures that employ public key cryptography. As explained before, it is one of our explicit goals not to use such techniques because of their high demand of computing capacity and energy. Still we assume, that every participating device can be identified. This can be done for example using pre-distribution of random keys like proposed e. g. by Ramkumar et al. [21].
- There is a central, trusted component: This component that is called broker subsequently does not need a continuing connection to the resource providers and consumers, but at least once in a validity period of the used coins. The broker is not involved in any payment operation between a service provider and a service consumer. It only delivers new coins at the beginning of a validity period and redeems coins at the end of a period.
- There is a common schedule: All participants that want to take part in the fraud detection system contact the broker within a certain time frame. This could be the first day of each month for example. Participants that do not meet this deadline still can redeem their coins, however they must accept not be recompensated if the coin has been redeemed by a malicious user before.
- There are means to invoice the transactions: Just in case that the virtual currency is supposed to be bound to a real currency (which is not necessarily the case) we further assume that the broker is able to bill and withdraw the according amount of money from the participants account. Just as it is done for example by a cell phone provider. The actual arrangement of this business process is out of scope for this paper.
- Attacker model: Furthermore, we assume that malicious users cannot forge coins that pass the validity checks run by each device before accepting a coin. This assumption is reasonable as Rivest et al. [24] provide techniques that make the necessary efforts for forging coins many times higher than the costs that the legitimate issuer of the coins has to bear. Instead we consider malicious users that try to spend original coins several times. We examine both cases, a single fraudulent user and colluding fraudulent users. A single fraudulent user can try to conceal the fraud by forging transaction logs whereas a group of colluding fraudulent users can even come to an agreement to incriminate a third party of having committed the fraud. We assume such colluding fraudulent users to act in fixed groups. We think this is reasonable as the most likely appearance of such a colluding group is a person that owns serveral mobile devices that she configured to commit fraud.

3.2 Overview

The architecture of our system consists out of three major components: broker, service consumer and service provider. Mobile devices might either act as both service consumer *and* service provider or incorporate only one of the two functions. At least once in the validity period of a coin (that might be a month, e. g.) service providers and consumers need a connection to the broker. This can be done either through a wireless WAN interface or when the mobile devices eventually gain access to the internet via WiFi-LANs. Then the mobile devices can get new valid coins or redeem the ones that they earned for providing services. Note that this requires only a sporadic connection between broker and participating devices. Each coin represents a fixed, small monetary value. This could be 0.1 cent, for example.

Every time a mobile device wants to offer a service, an adequate fee for using the service is also published within the service advertisement. As we want to recompense service providers for their energy expenses, the price to pay for using a service should be related to its energy cost. However, the value of energy for the user of a mobile device varies depending on usage

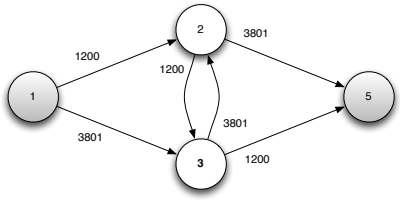


Figure 1 Transactions between honest users. Node one and five standing for the broker's dispensing and redeeming part.

scheme, remaining battery charge and chances to recharge the battery. There are several approaches in related work how the price of a service can be adjusted with regard to those variable conditions, e. g. in [3].

As the price of a service is published within the service advertisement, a service consumer can base the choice of provider on the announced service cost. On requesting the service, the service consumer transfers the specified number of coins to the service provider. If the service provider fulfills its obligation the service consumer marks the corresponding coins as spent. If the service provider does not meet its duty the service consumer adds the provider's id to a blacklist. In case of a successful payment both participants create transaction logs. A transaction log contains the unique id of all coins involved in the payment and the ids of service provider and consumer. These transaction logs can be used for the posterior fraud detection presented subsequently in Section 4.

4 Posterior Fraud Detection

According to the attacker model presented in Section 3.1 only multiple-spending of coins will discussed in this work.

The only chance of detecting the reuse of already spent coins at user-level is, if the coin already exists in the local coin set and a user wants to pay this user with the same coin. In all other cases, fraud can only be detected by the broker after all coins have been redeemed. Therefore every client saves a transaction log for received and spent coins including the involved user-IDs. The transaction log is handed over to the broker when redeeming the coins. If the broker detects that one or more coins are redeemed by more than one user, he uses a graph built from the collected transaction logs. This graph is used for detecting the fraudulent users. The nodes thereby represent the users, the edges the transfer of coins (identified by their coin-IDs) between these users (as shown in Figure 1, e. g.).

Honest users, redeeming all received coins at the end of the validity period, should have the same amount of incoming to outgoing edges, as shown in Figure 1. In this case we call all edges verified as for all of them two congruent transaction log entries can be found. Only fraudulent users or users trading with fraudulent users will show an imbalance or have unverified transactions, where only one log entry for the payment can be found.

As a fraudulent user is willing to hide his fraud, he will likely manipulate his own transaction log by either keeping only one of the entries for the multiple-spent coin or adding bogus entries telling he received the coin multiple times. This manipulation leads to notcongruent transaction log entries as shown in Figure 2. We call these unverified edges.

The broker scans the transaction logs for these unverified transactions and marks the involved users. A naïve approach would be to only mark the users with outgoing unverified edges as cheaters. Then two or more users could conspire to draw the broker's attention on another user as shown in Figure 3. In this example user 3 could hand over a coin received from user 2 to node 4. Both colluding fraudulent users could delete the transaction log for this transaction and create transaction logs that indicate that they received the coin from user 2.

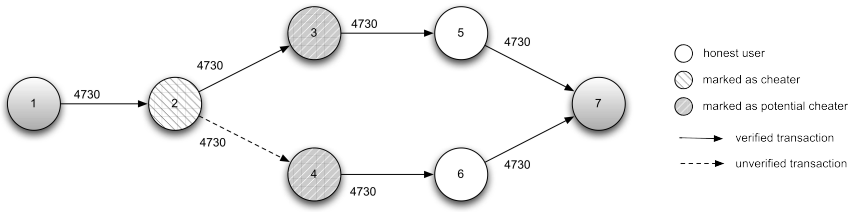


Figure 2 Transactions including a cheaters (node 2) double spending of a coin. Node one and seven depict the broker that is dispensing and redeeming the coins.

Algorithm 1 explains the tracking algorithm that is used to face the challenge of colluding fraudulent users. If an unverified edge is detected, the user with the outgoing unverified edge *and* all users receiving the involved coin from this user are marked as possibly fraudulent. The user with the outgoing edge hereby gets a score of two, the users with the incoming edge receive a score of one (line 5 and line 10). Considering the example from Figure 3 user 2 gets a score of 2, users 3 and 4 get a score of one each. These scores were chosen with respect to the thresholds used in Algorithm 2. A user must occur at least two times spending a duplicated coin or three times receiving a duplicated coin before he is considered a malicious user.

Thereafter the set of the receiving users (in our example this is {3,4}) is put into a list of potential conspiratorial groups. If the group has been added before, its group score is increased (lines 14 - 19). The score of the users and the group score is used to determine whether a user is treated as fraudulent user or as victim of a conspiracy. How this is done is shown in detail in Algorithm 2. A fraudulent user who spends a coin several times without forging transaction logs will be caught as he has more verified outgoing transactions than incoming transactions for the very same coin (lines 22 - 27). He gets a score of 3 (line 25) which is higher than the COIN-FRAUD-THRESHOLD (=2).

Algorithm 1 Algorithm to detect fraud

```
 1: for all coins in multipleRedeemedCoins as coin do
 2:     for all unverified transactions of coin as ut do
 3:         source = ut.source
 4:         if source has more than 1 outgoing transactions then
 5:             setScore(source, coin.id, 2)
 6:             possibleConspirators = new List
 7:             for all outgoing transactions of source as ot do
 8:                 receiver = ot.receiver
 9:                 if receiver has at least one outgoing transaction && getScore(receiver) == 0 then
10:                     setScore(receiver, coin.id, 1)
11:                     possibleConspirators.add(receiver)
12:                 end if
13:             end for
14:             if suspectedGroups.contains(possibleConspirators)) then
15:                 incrementGroupscore(possibleConspirators)
16:             else
17:                 suspectedGroups.add(possibleConspirators)
18:                 setGroupscore(possibleConspirators, 1)
19:             end if
20:         end if
21:     end for
22:     for all verified transactions of coin as vt do
23:         source = vt.source
24:         if source has more outgoing than incoming transactions for coin then
25:             setScore(source, coin.id, 3)
26:         end if
27:     end for
28: end for
```

The process of actually resolving the fraud is described in Algorithm 2: The marking of multiple users as possible fraudulent requires the adoption of thresholds for the scores to

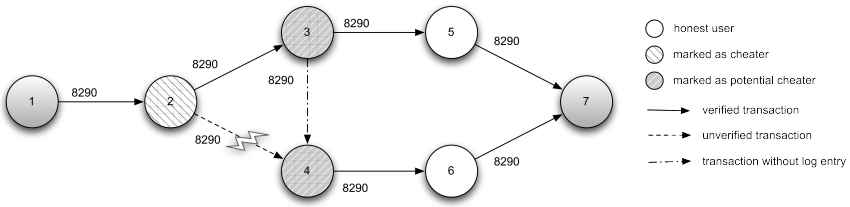


Figure 3 Colluding cheaters (node 3 and 4) blame an honest user for the double spending by forging history entries.

avoid bringing innocent users to justice. First, only if the score for one coin is above the COIN-FRAUD-THRESHOLD (=2) for the user, he is regarded when reallocating the costs of the excessive redemption (line 8). This means a user must be involved at least two times into suspicious transactions before he is charged for a fraud. Users with a low score are likely to be innocent while the probability of a fraudulent user is higher the more he is being involved in unverified transactions for the coin.

Nevertheless a user whose score for all the coins is below the threshold could still be fraudulent by just double spending each coin and hereby keeping a low score for each coin. So the total score over all unverified transactions also has to be below a second threshold, TOTAL-FRAUD-THRESHOLD (=4), before the user is marked as innocent (line 8). The values for both thresholds provided good results, however, future work might show that there are better settings. Generally higher values for the thresholds bear the risk of an increased number of false negatives whereas lower thresholds increase the number of false positives.

After the broker's detection of who of the users is fraudulent, he uses the scores of these users to reallocate the values of the excessive redemption and debits their user accounts. The amount a convicted user is held liable for depends on his score. So users that are more likely to be cheaters have to pay more.

The broker maintains a list where he keeps a record of all frauds committed by the users. When a user reaches a predefined limit he will be blocked out of the system. The blocked users are regularly distributed to all other users, so a blocked user can not trade anymore with the broker or any other users.

Note that Algorithm 1 can be easily parallized. This is important as checking the transaction logs might become a bottleneck of the system. Future work will show whether and how it is possible, to parallize or speed up Algorithm 2.

5 Evaluation

The evaluation of our approach was twofold: First, we examined how much energy is spent for the actual payment. Related work states clearly, that systems based on hash functions are cheaper than for example RSA signatures. Rivest et al. explain that hash functions are about 100 times faster than RSA signature verification [24]. Still, it remains to show that the energy cost of the incentive system is small compared to the resource sharing framework itself. For this reason we did measurements to determine the additional energy overhead caused by our incentive system.

Second, we analyzed how good the mechanism to detect fraudulent users actually works. Both are critical issues with respect to user acceptance. The energy measurements were done using a prototype whereas we did a simulative evaluation of the fraud detection mechanism.

5.1 Energy Consumption

The prototype [11] for measuring the energy overhead that is caused by the incentive system consists out of two Nokia N810 devices. We implemented a resource sharing framework based on OSGi [2, 23] and R-OSGi [22]. Resources and services that are supposed to be shared

Algorithm 2 Algorithm to resolve fraud

```
1: //remove possibleConspirators from suspectedGroups if groupscore is below threshold
2: cleanSuspectedGroups();
3: for all coins in multipleRedeemedCoins as coin do
4:    conspiracyDetected = false
5:    totalScore = 0
6:    fraudulentUsers = list of users with score > 0 for this coin
7:    for all users in fraudulentUsers as user do
8:       if getScore(user, coin.id) > COIN-FRAUD-THRESHOLD || getScoreForAllCoins(user) >
      TOTAL-FRAUD-THRESHOLD then
9:          score = getScore(user, coin.id)
10:         if user is in suspectedGroups then
11:            score = (number of possibleConspirators groups the user appears in for coin) * 3
12:            conspiracyDetected = true
13:         else
14:            if conspiracyDetected && score < COIN-FRAUD-THRESHOLD then
15:               fraudulentUsers.remove(user) // Delete innocent user when conspiracy is detected
16:               score = 0
17:            end if
18:         end if
19:         totalScore += score;
20:         //User is below thresholds
21:      else
22:         fraudulentUsers.remove(user) // Delete fraudulent users below threshold
23:      end if
24:   end for
25:   doubleRedeems = number how often this coin has been redeemed
26:   ratio = doubleRedeems / totalScore
27:   for all fraudulent users of coin as user do
28:      valueToPay = round((getScore(user, coin.id) * ratio) + accRoundingError);
29:      accRoundingError = ((getScore(user, coin.id) * ratio) + accRoundingError) - valueToPay;
30:   end for
31: end for
```

as well as the broker and the software for doing the actual payments are implemented as remotely accessible OSGi bundles. One of the devices runs a simple service that changes the characters of a word from upper- to lowercase and vice versa. The service is advertised in a wireless ad hoc network. The other N810 device acts as service consumer. It requests the service and sends the needed number of digital coins to the service provider. The service provider checks the validity of the payment and if there is no reason to complain it provides the service. We used an SNMD [15] to measure the current drawn from the battery during the process of providing the service and checking the validity of the coins. In each run of the experiment the service was requested, provided and payed for 100 times. We conducted 10 runs of this experiment. The measured energy cost does not include the cost for redeeming the coins. However, this occurs rather infrequently (e.g. once a month) compared to the actual payment operations and thus it can be neglected.

As a reference point we repeated the entire experiment without doing any payment. Figure 4 shows the electric charge that was drawn from the battery in these two experiments. In average, checking the payment for valid coins adds an overhead in terms of energy of about 3%. This is mainly due to a slightly longer runtime (in average 4.54%). Note, that the relative overhead will become much smaller for services that consume more energy than our modest dummy service. So the 3% overhead can be seen as an upper bound for the overhead that is caused by our incentive scheme. The fact that there are even some runs with payment that consume less energy than some of the runs without payment shows, that the overhead of the payment system in fact is very small. Obviously it can be outweighed by other factors that are not related to the incentive system, e. g.background processes scheduled by the operating system or necessary retransmissions due to a busy WiFi channel.

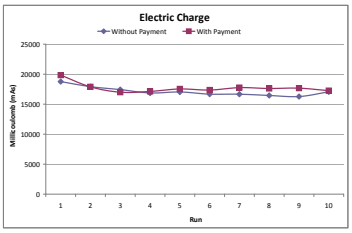


Figure 4 Comparison of energy consumption with and without the incentive system

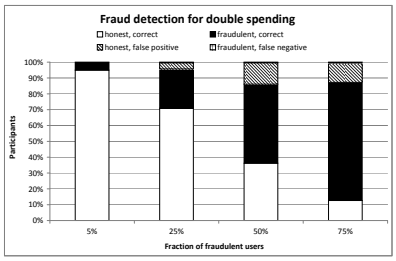


Figure 5 Detection rate of single fraudulent users.

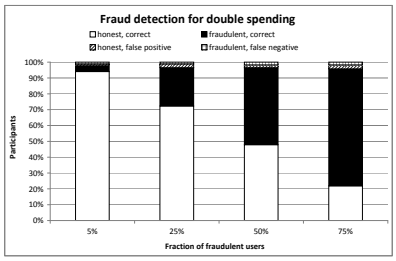


Figure 6 Detection rate of conspiratorial fraudulent users.

5.2 Fraud Detection

The effectiveness of our technique for posterior fraud detection was analyzed in simulations. The simulations were done with an event based simulator written in Java. In each run 100 participants did 1000 payments in total with a randomly chosen transaction partner. The fraction of fraudulent users was set to 5, 25, 50 or 75 percent. In each configuration we did 10 runs with different random seeds. We did the entire experiment twice, once for the attacker model of a single malicious user that tries to use his coins more than once, and a second time for a more powerful attacker that is able to cooperate with other malicious users. Such collusive participants can try to spend coins multiple times and both blame a third party, that is actually innocent.

After each run the broker did its graph based fraud detection using the transaction logs provided by the service providers and consumers. The results of the fraud detection process were compared to the actual behavior of the users and each decision that was taken by the broker was classified to be one of the following:

- honest, correct: An honest participant was correctly detected as an honest user.
- fraudulent, correct: A fraudulent participant was correctly detected as a fraudulent user.
- honest, false positive: An honest participant was wrongly detected as a fraudulent user.
- fraudulent, false negative: A fraudulent participant was not detected.

Figure 5 shows the results in case that fraudulent users act on their own and do not collude. Our key finding from this result is that the number of malicious participants that are not detected by the broker is close to zero (0.6 % in average). As malicious users hardly have any chance to get away without being punished there is only little appeal to behave badly.

Figure 6 shows even better results for groups of collusive cheaters. Fraudulent users colluded in teams of two, spent coins multiple times and blamed a third, innocent user for it.

They are detected, because they appear several times as a suspicious group, according to the assumption presented in Section 3.1. The broker and its detection mechanism work pretty well. Less than 5 % of the users are classified wrong, even in the worst case scenario with 75% of fraudulent users. A substantial amount of wrong classifications is due to colluding malicious users that did not spend a coin twice but were involved in blaming an innocent user. Such users are often detected as cheaters and this decision is considered to be a false positive, here. If this was not the case the rate of wrong detections would even be lower (2.4 % of all users in the case of 75 % fraudulent users). Relating to the number of fraudulent users this means that 3.2 % of the cheaters get away.

6 Conclusion and Outlook

In this paper we argued that resource sharing systems for MANETs lack an incentive system tailored to this specific use case. Our approach provides the means to recompense service providers for the energy they spent on service provision. As neither public key cryptography nor communication with a third party is needed on each payment we could eliminate two major drivers of energy costs. This results in only very little additional energy consumption ($< 3\,\%$) caused by our incentive system as shown by prototype measurements. Although the abandonment of powerful cryptographic tools and the reuse of currency create opportunities for abuse we showed through simulations that almost all malicious users can be tracked down by our system for posterior fraud detection by an off-line broker. In a worst case scenario with 75 % of the users being cheaters only 3.2 % of them get away without being caught. This proves that it is possible to build an incentive system that suites the needs of resource sharing in MANETs: being cheap with respect to its energy consumption and still effective on preventing abuse.

In the near future we plan to compare the energy consumption of our incentive system to solutions that apply digital signatures for payment transactions. Further we want to investigate on how to further decrease the number of false positives in our fraud detection system.

References

1 M. Abdelkader, N. Boudriga, and M. S. Obaidat. Secure Grid-Based Multi-Party Micropayment System in 4G Networks. In *ICE-B*, pages 137–148, 2007.

2 OSGi Alliance. *OSGi Service Platform, Release 3*. IOS Press, Inc., Fairfax, 2003.

3 G. Ananthanarayanan, V. Padmanabhan, L. Ravindranath, and C. Thekkath. Combine: Leveraging the Power of Wireless Peers through Collaborative Downloading. In *Proc. 5th International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM New York, 2007.

4 M. Blaze, J. Ioannidis, and A. D. Keromytis. Offline Micropayments without Trusted Hardware. In *Proc. 5th International Conference on Financial Cryptography*, pages 21–40. Springer-Verlag, 2002.

5 R. Bless, C. Hübsch, C. Mayer, and O. Waldhorst. SpoVNet: An Architecture for Easy Creation and Deployment of Service Overlays. In Anand R. Prasad, John F. Buford, and Vijay K. Gurbani, editors, *Advances in Next Generation Services and Service Architectures*. River Publishers, 2011. *To appear*.

6 T. Bocek, M. Shann, D. Hausheer, and B. Stiller. Game theoretical analysis of incentives for large-scale, fully decentralized collaboration networks. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*. , pages 1–8, 2008.

7 L. Buttyán and J.-P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad hoc Networks. *Mobile Networks and Applications*, 8(5):579–592, 2003.

8 A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The Rise of People-Centric Sensing. *IEEE Internet Computing*, 12:12–21, 2008.

9 X. Dai and J. Grundy. NetPay: An Off-line, Decentralized Micro-Payment System for Thin-Client Applications. *Electronic Commerce Research and Applications*, 6(1):91–101, 2007.

10 J. Furthmüller and O.P. Waldhorst. A Survey on Grid Computing on Mobile Consumer Devices. In N. Antonopoulos, G. Exarchakos, M. Li, and A. Liotta, editors, *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing*, pages 313–363. IGI-Global, 2010.

11 J. Furthmüller, S. Becker, and O.P. Waldhorst. An Energy Manager for a Mobile Grid. Demo on MobiSys09, Krakow, Poland, 2009.

12 *A Taxonomy of Incentive Patterns - The Design Space of Incentives for Cooperation.* In M.P. Singh G. Moro, C. Sartori, editors, *Agents and Peer-to-Peer Computing* volume 2872 of *Lecture Notes in Computer Science*, pages 89–100. Springer Berlin / Heidelberg, 2003.

13 D. Hausheer and B. Stiller. PeerMint: Decentralized and Secure Accounting for Peer-to-Peer Applications. In R. Boutaba, K. Almeroth, R. Puigjaner, S. Shen, and J.P. Black, editors, *NETWORKING 2005*, volume 3462 of *Lecture Notes in Computer Science*, pages 40–52. Springer Berlin / Heidelberg, 2005.

14 H. Haverinen, J. Siren, and P. Eronen. Energy Consumption of Always-On Applications in WCDMA Networks. In J. Siren, editor, *Proc. IEEE 65th Spring Vehicular Technology Conference (VTC)*, pages 964–968, Dublin, Ireland. IEEE Vehicular Technology Society, 2007

15 A. Hergenröder, J. Wilke, and D. Meier. Distributed Energy Measurements in WSN Testbeds with a Sensor Node Management Device (SNMD). In M. Beigl and F.J. Cazorla-Almeida, editors, *Workshop Proceedings of the 23th International Conference on Architecture of Computing Systems*, pages 341–438, Hannover, Germany. VDE Verlag, 2010.

16 S. Kaune, K. Pussep, G. Tyson, A. Mauthe, and R. Steinmetz. Cooperation in P2P Systems through Sociological Incentive Patterns. In K. Hummel and J. Sterbenz, editors, *Self-Organizing Systems*, volume 5343 of *Lecture Notes in Computer Science*, pages 10–22. Springer Berlin / Heidelberg, 2008.

17 S. Kotrotsos, P. Racz, C. Morariu, K. Iskioupi, D. Hausheer, and B. Stiller. Business Models, Accounting and Billing Concepts in Grid-Aware Networks. In O. Akan, P. Bellavista, J. Cao, F. Dressler, D. Ferrari, M. Gerla, H. Kobayashi, S. Palazzo, S. Sahni, X.(S.) Shen, M. Stan, J. Xiaohua, A. Zomaya, G. Coulson, A. Doulamis, J. Mambretti, I. Tomkos, and T. Varvarigou, editors, *Networks for Grid Applications*, volume 25 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 27–34. Springer Berlin Heidelberg, 2010.

18 K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi. Battery-driven system design: a new frontier in low power design. In A. Raghunathan, editor, *Proc. 7th Asia and South Pacific and the 15th Int. Conference on VLSI Design Design Automation Conference Proceedings ASP-DAC 2002*, pages 261–267, Bangalore, India. Center for Embedded Computer Systems, 2002.

19 N. Liebau, V. Darlagiannis, A. Mauthe, and R. Steinmetz. Token-Based Accounting for P2P-Systems. In W. Brauer, P. Müller, R. Gotzhein, and J. Schmitt, editors, *Kommunikation in Verteilten Systemen (KiVS)*, Informatik aktuell, pages 16–28. Springer Berlin Heidelberg, 2005.

20 M.S. Manasse. The Millicent Protocols for Electronic Commerce. In *Proceedings of the 1st conference on USENIX Workshop on Electronic Commerce - Volume 1*, pages 9–9. USENIX Association, New York, 1995.

21 M. Ramkumar and N. Memon. An efficient random key pre-distribution scheme. In *Global Telecommunications Conference (GLOBECOM). IEEE*, volume 4, pages 2218 – 2223 Vol.4, 29. 2004.

22 J. Rellermeyer, G. Alonso, and T. Roscoe. R-OSGi: Distributed Applications Through Software Modularization. In *Proc. ACM/IFIP/USENIX 8th International Middleware Conference*, pages 1–20. 2007.

23 J. Rellermeyer and G. Alonso. Concierge: A Service Platform for Resource-Constrained Devices. In *Proc. EuroSys 2007 Conference*, Lisbon, Portugal. ACM, 2007.

24 A. Shamir R.L. Rivest. *Security Protocols*, chapter PayWord and MicroMint: Two simple micropayment schemes, pages 69–87. Springer Berlin/Heidelberg, 1997.

25 D. Saha and A. Mukherjee. Pervasive Computing: A Paradigm for the 21st Century. *IEEE Computer*, 36(3):25–31, 2003.

26 B. Yang and H. Garcia-Molina. PPay: Micropayments for Peer-to-Peer Systems. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 300–310, Washington D.C., USA. ACM, 2003.

27 S. Zhong, J. Chen, and R. Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. *Proceedings of IEEE INFOCOM*, pages 1987—1997, 2002.

Supporting Cooperative Traffic Information Systems through Street-Graph-based Peer-to-Peer Networks

Jedrzej Rybicki, Benjamin Pesch, Martin Mauve, and Björn Scheuermann

Institute of Computer Science, Heinrich Heine University Düsseldorf
{rybicki,mauve,scheuermann}@cs.uni-duesseldorf.de,
benjamin.pesch@uni-duesseldorf.de

Abstract

In this paper we present a novel peer-to-peer system specifically designed to support the unique properties of traffic information systems. We discuss important design decisions, such as update strategies and algorithms for dynamic vehicular route planning. Our system is then assessed using a combination of network (OverSim/OMNeT++) and road traffic (SUMO) simulators. We discuss the network load required by our system and show the benefits—in terms of travel time savings—that users can expect from it.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.121

1 Introduction

Traffic information systems (TIS) provide navigation units with information about current traffic conditions and thus enable dynamic routing decisions. Research in this area has investigated all flavors of communication as a basis for TIS applications, ranging from VANET-style direct communication [9, 10, 22] to infrastructure-based approaches using either a centralized server [19] or a peer-to-peer network [17]. The focus of existing work, however, has mainly been on the technical feasibility of the proposed solutions, e. g., regarding the constrained bandwidth and connectivity of VANETs. Feasibility of a technical solution is certainly an important point. But, as far as market introduction is concerned, it is even more important to show that such a system, when in place, can really bring benefits to its users.

Given this background, the contribution of this paper is threefold. We first introduce a novel peer-to-peer network structure that is particularly well suited to managing traffic information data. For our approach we rely on infrastructure-based cellular communication, like for example UMTS. We therefore assume that an IP-based communication channel is present between the cars. Car navigation units participating in our system use this channel to set up a fully distributed overlay network, over which they cooperatively share information about the current traffic situation. As a second contribution, we extend this peer-to-peer structure by a publish/subscribe mechanism to handle updates efficiently, and show how dynamic routes can be calculated efficiently. Finally, and potentially most importantly, we investigate the benefit—i. e., the travel time reduction—that a user can expect.

The remainder of this paper is structured as follows: we first review related work on both cooperative traffic information systems and peer-to-peer networks in Section 2. Following this, in Section 3, we introduce a novel peer-to-peer structure that has been specifically designed for storing and retrieving TIS information. We then show how the handling of updated information can be realized efficiently by means of a publish/subscribe paradigm in Section 4. The evaluation of the performance of the proposed peer-to-peer structure and an assessment of the benefits for participants of our system will be presented in Section 6.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 121–132

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Related Work

2.1 Cooperative Traffic Information Systems

Traffic information systems are systems for collecting and providing information about the current traffic situation in a road network. They constitute the basis for dynamic car navigation, i. e., navigation avoiding traffic jams and selecting the fastest route with regard to current traffic conditions.

Pfoser et al. [13] used historical floating car data (FCD) extracted from recorded movements of a taxi fleet as a basis for their analysis. The data were aggregated over small time intervals. The work is based on the assumption that there exists a strong correlation between historical data (already collected and analyzed) and future traffic conditions: knowing the current situation and historical data it is then possible to estimate the traveling times in the future. However, this assumption does not necessarily hold. This was observed, for instance by Yang et al. [23]. They likewise used FCD collected by taxis and treated them as if they were current information in a traffic information system. They demonstrated that the routes originally chosen by the taxi drivers were sub-optimal in most of the cases. Further, the authors also determined that historical information about traveling speed often deviates widely from real-time data. This motivates the use of a real-time traffic information system.

In order to gain real-time traffic information, communication is required. In cooperative traffic information systems, the data are collected by the system users themselves: cars make observations on current traffic conditions and share them. Ideas to realize such systems span a wide range. On the one hand, fully distributed approaches based on local, immediate wireless communication via vehicular ad-hoc networks (VANETs) were discussed, including, e. g., SOTIS [22] or TrafficView [10]. Such approaches, however, inevitably suffer from the inherent scalability constraints [18] and limited connectivity [9] of VANET communication.

Therefore, alternatives based on mobile Internet access and centralized servers [19] have been proposed. The main drawback of this approach is that significant resources in a central location are required, and that the central server constitutes a bottleneck or even a single point of failure. Furthermore, all the data that is cooperatively collected by the users are put in the hands of a central (and typically commercial) operator—a fact that might not be desirable and that contradicts the cooperative nature of the application. By using a peer-to-peer overlay it is possible to overcome these issues and to preserve the benefit of decentralization provided by VANETs, while at the same time making use of the good network service of infrastructure-based (cellular) communication. We first put this idea forward in [16]. Subsequently, there has been a first implementation: in [17], where we proposed to adjust the CAN distributed hash table (DHT) [14] to store and retrieve traffic data more efficiently. In the paper at hand we go a significant step further: we propose an overlay that has been specifically tailored to the application's key space structure and look-up pattern.

2.2 Peer-to-Peer Networks

Due to their inherent scalability, peer-to-peer networks are commonly viewed as a solid basis for many distributed applications. The work in this area often focuses on optimizing the independent retrieval of data associated with a single looked-up key—the classical setting of a DHT. However, a traffic information system typically generates updates and queries regarding multiple keys that are related to each other. Our work therefore focuses on how to design peer-to-peer networks that exploit the application-level knowledge about these dependencies in order to store and retrieve the data more efficiently.

Figure 1 Minimal bounding box containing planned route (Route 66 from Chicago to L.A.).

Among the subjects discussed in the peer-to-peer community, range queries are probably closest to what we do in our work. The general idea of range queries is to efficiently retrieve all values that are associated with a range of keys. This is a hard problem, since most peer-to-peer networks use hashing to map keys to peers. Hence, without any specific support, a range query would result in independently querying all key values in that range. A good overview of search methods typically employed in peer-to-peer networks including a discussion of range queries is given in [15].

The problem we face in our work is quite different from requesting all values in a given range. We need to query keys where we know—on the application level—how the keys are related to each other, i. e., how the individual road segments are connected in the street graph. The difference can be best shown through an example. Imagine a traffic information system relying on two-dimensional range queries and a driver which wants to drive from Chicago to Los Angeles along the famous Route 66. Before the journey she wants to be sure that the current traffic conditions are sufficiently good to enjoy the trip, so she generates a query for the traffic information system to retrieve all relevant measurements. With a typical interval-based range query system, this would encompass all the information for the two-dimensional "range" between Chicago and L.A. (Fig. 1). This would include a huge amount of data that are not relevant, like parallel routes, routes heading in the opposite direction, and so on. Our work, in contrast, takes full advantage of the information provided by the road network itself to query only the required data effectively and efficiently.

3 Peer-to-Peer Network for Traffic Information Systems

As a basis of our work we assume that each participating navigation unit has access to a street map that provides information on road segments and how they are connected to each other. A road segment is a part of a road delimited by two consecutive junctions. Each road segment is uniquely identified by a globally known ID, which will be used as a key in the peer-to-peer network. In the following, we thus use the terms "key" and "road segment" interchangeably.

Traffic information applications differ in some significant aspects from most "classical" peer-to-peer applications like file sharing. In particular, this holds for the very specific request and update pattern. Individual overlay nodes access information about a limited, contiguous part of the road network between their position and the planned destination. This characterizes the subset of the key space a given overlay node is interested in, and thus the interrelation between the queried keys. An important observation that constitutes the basis of our approach is that the organizational structure of the data in a traffic information system is a graph: the street map. Information about the current traffic situation is always

associated with a road segment, i. e., with an edge in this graph. The technical challenge tackled here is to store this graph in a distributed fashion in such a way that typical queries can be served quickly.

3.1 Overlay construction

In our design, each peer—that is, each car navigation unit currently participating in the system—is responsible for one part of the road network, i. e., for one sub-graph. This sub-graph need not be close to the car's current geographical position (very much in contrast to the situation in VANETs, immediate physical proximity of communication partners brings little to no benefits in cellular networks); in particular, the responsibility zones do not change due to the cars' movement, but only when peers join or leave. A peer stores the measured data for the road segments it is responsible for, and makes them available for other peers.

Initially, the overlay structure consists of only one peer which is responsible for the whole street map. In order to join the overlay, a new peer obtains the address of at least one active peer by some arbitrary bootstrapping mechanism, and sends a join request to a randomly picked peer. The contacted peer splits its sub-graph and hands over one half to the newly arriving node, along with the stored data for the respective keys.

In the next section we will describe how a graph can be divided between peers. For the moment let us assume that there exists an algorithm which takes a graph as an input and bisect it into two disjoint and roughly equally sized partitions. The further development of the peer-to-peer structure is a sequence of join and leave operations of the peers, during which sub-graphs are split and re-joined. Each peer maintains overlay links to those peers managing a partition connected to the peer's own sub-graph in the street graph, i. e., to those peers responsible for neighboring roads. These overlay links are used for routing queries: in essence, queries follow the structure of the road network, which is resembled in the overlay. The information on the relevant neighbor peers, to which a newly arriving node can then set up overlay connections, is exchanged during the join operation.

3.2 Graph partitioning

In order to realize an overlay as outlined above, we need a mechanism to partition the street network graph and to dynamically assign a sub-graph to each overlay node. Our guideline for this process is to keep keys preferably within one partition if they are often requested together: in our application, cars request information about contiguous routes, so we should keep connected road segments close together in the overlay to speed up typical queries. Consequently, this boils down to a graph partitioning problem: dissecting the graph into a number of (almost) equally sized disjoint sub-graphs, while cutting as few edges as possible.

Given a graph $G = (V, E)$ with n vertices V, edges E, and weights $w_{i,j} > 0$ for $(i, j) \in E$, the general graph partitioning problem is to divide the original graph into k (approximately) equally sized subsets $V_1, \ldots, V_k$ such that

1. $\forall 1 \leq p \leq k : |V_p| \simeq n/k$,
2. $\forall 1 \leq p, q \leq k : p \neq q \Rightarrow V_p \cap V_q = \emptyset$, and
3. $\bigcup_{p=1,\ldots,k} V_p = V$,

and the *total edge cut*

$$T := \sum_{(i,j)\in E, i\in V_p, j\in V_q, p\neq q} w_{i,j}$$

is minimal. Here, we will only need the case $k = 2$, i. e., bisecting into two partitions.

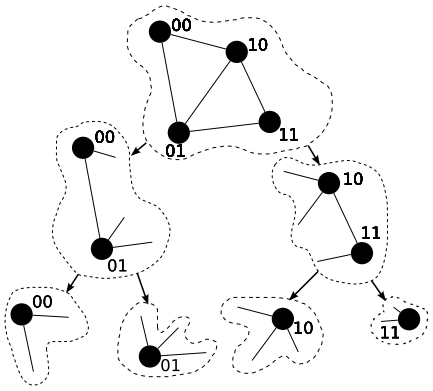


Figure 2 A street network graph with four nodes and its partitioning tree.

It is known that optimal partitioning is NP-hard. Unfortunately this holds true even if all edge weights are equal, if k is small (even if only a bisection of the graph is sought), and if the maximum vertex degree is limited [2]. However, a number of heuristics for finding good graph partitions have been proposed, on which we can build. Here, we use geometric partitioning techniques [5] and graph growing methods [6]. Both algorithms can be refined with an algorithm devised by Kernighan and Lin [7]. For more details we refer the reader to the respective publications.

The tree of graph partitions that is traversed as nodes join and leave the network is exemplarily depicted in Fig. 2 for a graph with four nodes in total. Each internal node in the tree has two child nodes resulting from bisecting the corresponding sub-graph. This hierarchical partitioning of the road network is calculated in advance and is stored with the road map data, so that pre-computed bisections of all sub-graphs are readily available to all peers. Therefore, it is not necessary to perform the complex computations for the above mentioned heuristics on the (computationally limited) peers while the system is running.

3.3 Requesting data for a single, arbitrary key

A peer that queries a given key (road segment) first locates it on the street map. It then uses a shortest path algorithm on the local street map data to find the shortest path from any node in one of its neighbors' sub-graphs to the sought-after segment.[1] This identifies the neighbor that minimizes the remaining distance (in terms the road network graph) to the destination key. This neighbor is selected as the next hop for the query. The look-up message transmitted to that neighbor includes the sought-after key and the address of the node initiating the query. The same process is then repeated by each peer receiving the query, until the destination key is reached. Therefore, the query essentially travels across the road network graph on its way to the peer responsible for the sought-after key. Once that peer has been reached, it replies directly to the query originator. A requesting peer is provided with a record that describes the current traffic condition on the queried road segment. This record is generated by the peer that is responsible for the road segment, based on the information that has previously been uploaded by other peers. It could, for instance, characterize the average value and the gradient of the driving speeds recently reported for the segment.

[1] This requires only one run of Dijkstra's algorithm (or of another shortest path algorithm).

3.4 Requesting a set of correlated keys

Finding the data for a single, arbitrary key is an important function of the overlay; however, as argued before, the efficient processing of queries for sets of consecutive keys is much more important, since it clearly dominates the request pattern. The specific strength of our overlay is the handling of such queries.

In order to obtain information on a set of consecutive road segments, the query originator first needs to locate a peer responsible for one of the keys. This peer—it might be the one that is responsible for the starting point of a queried route—can be found as described in the preceding subsection. The list of all searched segments is included in the query that is sent to this first node. Because the employed graph partitioning algorithms preferably keep related segments (that are often requested together) within one partition, the first node will often already be able to provide information on more than one segment. The remaining request—with the segment IDs that it cannot answer itself—is forwarded to the respective neighbors. These answer their "share" of the request and then proceed analogously until all the requested information has been retrieved. The query thus, in some sense, follows the queried route through the overlay. Since our overlay is built in such a way that consecutive road segments are either handled by the same peer or by peers between which a direct connection exists, each hop in the overlay yields information on at least one of the queried keys—and typically more than one.

3.5 Leave and recovery

When it comes to maintaining the overlay we stay close to concepts presented so far in the peer-to-peer community. In particular, we adapt the respective mechanisms from the CAN overlay [14] for join, leave, and recovery from node failures. They are well understood and, though originally designed for a hierarchical subdivision of a Cartesian space, they can quite easily be adapted to a hierarchically bi-partitioned graph. We only outline the mechanisms here, because the details are equivalent to what has been discussed for CAN.

For failure detection, peers exchange periodic hello messages with their neighbors. If a peer misses a number of periodic hellos from one of its neighbors, a node failure is assumed and a recovery strategy is utilized to restore a consistent overlay structure. Leaving peers and recoveries from peer failures can be handled directly if there is a "sibling peer" that is responsible for the full other half of the sub-graph from which the leaving peer's responsibility zone had previously been obtained through splitting. In that case, the sub-graphs can easily be merged, handing over the responsibility to the remaining sibling. If there is no sibling node (that is, if the sibling sub-graph has been further partitioned), a leave request is sent to the smallest known neighbor. It will become responsible for both partitions for a short time and will try to resolve the merging either by accepting a join or by employing a defragmentation mechanism.

4 Publish/Subscribe

The basic functionality of a traffic information system is to provide information on the current conditions on a given route. However, the situation on a given route is dynamic. In order to keep up-to-date regarding the traffic situation, the relevant data have to be updated periodically. If periodic requests are used, this could be a waste of resources: when the situation does not change, redundant information is transmitted. We mentioned the idea to use a publish/subscribe approach as an alternative in prior work [16], but until now we did

not investigate it in detail. In a publish/subscribe-based system, instead of repeating the request, the participants subscribe once for the selected keys. They are then actively informed by the overlay if relevant changes occur. We will now describe how we have implemented this idea for a TIS. For a general discussion of using publish/subscribe in peer-to-peer networks, we refer the reader to [4].

Using the taxonomy from [4], we use a peer-to-peer content-based publish-subscribe system with a hierarchical multicast topology. Peers implicitly subscribe for road segments when they request data about them while performing the route planning. Subscription request are thus regular query packets, and they are directly answered with the requested data. In addition to this immediate response, though, the responsible peers also keep track of a list of all peers that subscribed for the segment. Thus, each peer is a so-called broker for the data concerning the road segments it manages.

The broker of a road segment monitors the incoming data and determines if the traffic conditions on that segment have changed significantly. This is the case when the travel time needed to traverse the road segment changes by more than a given factor compared to the value reported previously (in our evaluation we used a deviation of more than 10 %).

To reduce the burden of informing all subscribers, the broker divides the group of peers which shall be informed into several equally sized subgroups. Only one representative out of each such group is directly informed. The messages, beside the actual notification, also contain the addresses of further peers which shall be informed. Each representative will then forward the notification to those peers. We consider more elaborate multicast schemes to be an orthogonal direction of research compared to our own work. We plan to use them when they turn out to be beneficial to the overall performance of our system.

For the subscription management we employed a soft-state approach: subscriptions have limited time-to-live after which they are removed from the system. When a broker leaves the network it hands over the list of subscriptions similarly to the way it hands over the traffic data it was responsible for.

5 Dynamic Vehicle Routing

The process of dynamic routing of vehicles encompasses two phases. First, an initial route to the planned destination has to be found. Second, while the vehicle follows this route, updates on the traffic condition may cause it to change.

5.1 First routing decision

As far as classical navigation is concerned, routing algorithms like Dijkstra's shortest path algorithm are used. Unfortunately, though, these algorithms require information on all edge weights. While the peer-to-peer traffic information system can provide the navigation unit with data on any road segment it asks (or subscribes) for, the retrieval of all available measurements each time a routing decision has to be made is not feasible. Thus a challenge is to select and subsequently request only the data crucial for the routing decision. For that purpose we developed two strategies.

In the first one, the navigation unit determines the static (not considering the current traffic conditions) shortest path to the planned destination. Thereafter the information about the current traffic condition on this route is requested and the local weights for that route are updated accordingly. The static values in the map are based on the maximum allowed speed, so these travel times are lower bounds for the real-time values. The returned information can thus only increase the estimated travel time along the calculated route. Hence, the routing

is started again after the information has been received, now with static travel times for those route segments not yet queried (those that have not been on the static shortest path), and with real-time travel times for those route segments that have been on the calculated route. If a new shortest path is found, the respective missing data is requested. The process is repeated until the shortest path contains only segments for which information is available. We call this the greedy approach. Because the static travel times are lower bounds, this algorithm is guaranteed to find the route with the currently lowest travel time based on real-time data—but it may happen that a huge amount of data needs to be requested in many iterations until the algorithm terminates. In practice, it is therefore advisable to limit the number of iterations (here, we use at most five rounds).

An alternative strategy defers the communication until the set of segments that should be queried is fully known. First a set of the static n fastest routes are determined by means of a modified Dijkstra Algorithm [3] (here, we use $n = 5$). For these routes the dynamic data is requested. As soon as the data are available, the fastest route out of that subset is selected.

5.2 Keeping in touch with the development of the situation

Since the traffic conditions change over the time, the navigation units need to update their information in order to check if the original route decision is still optimal. This is done either by periodically requesting the data for the current route (in our simulations, every three minutes) or by subscribing for the data and waiting for notifications as described in the previous section. When new data becomes available, the route has to be adjusted. This is done in a similar manner as the first routing decision described above. In order to avoid an overreaction, we do the following: if a better alternative route is found when the car is already driving, it will not necessarily change its route. Only when the expected travel time savings exceed some predefined threshold (we use 10 % here), the current route is changed.

6 Evaluation

6.1 Simulation setup

We implemented the proposed algorithms and protocols in the peer-to-peer simulator OverSim [1], which we coupled with the road traffic simulator SUMO [8]. The latter was responsible for generating car movements on a real street map of the city of Düsseldorf, extracted from the OpenStreetMap project [12]. We focus on the city scenario as there are number of working solutions for a dynamic navigation on highways. Our main motivation for using a fully-fledged road traffic simulator like SUMO was to obtain realistic query and update patterns. SUMO also offers a convenient way of interacting with the simulation through its TraCI interface [21]: not only is it possible to get information about the current position of a given car, its planned route, etc. but also to change the routes of individual cars. We extended this interface by implementing some new functions. In particular we integrated the extended Dijkstra's algorithm, as described in Section 5, in SUMO. The results of the routing can now be fetched via TraCI. We coupled cars in SUMO with OverSim peers: when a participating car sets off on its journey we created a new peer, upon arrival at its destination the peer left the overlay. It is possible to limit the number of cars accessible via TraCI by defining a penetration ratio. We used 20 % penetration in our simulations, resulting in up to 1 000 peers present in the simulation. The simulation time was 2 400 seconds of road traffic for each simulation run. At the beginning of the simulation, all initially present cars will join the overlay almost at the same time. Furthermore, at the very beginning of the simulation,

there will be no measurements available in the system. Such a situation will obviously not occur in the real world, where cars join and leave continuously. We therefore removed the initial seconds of simulation time from our evaluations, giving the system time to reach a stable state and gather enough data for biased routing decisions. Stable state does not mean that the set of participating peers is constant, but rather that the overall number of peers stays more or less the same: new peers join the network and others leave the network as soon as they reach the planned destination.

The communication between the peers was realized in OverSim (via the underlying network simulator OMNeT++ [11]). As soon as a car traversed a road segment it contributed a measurement of its travel time along that segment. Such a strategy is often utilized in traffic information systems, but may lead to the following problem: a traffic jam is only reported after the relevant road section has been completely traversed by a participating car. This could potentially take a long time, depending on the road condition and the length of the road segment. We therefore also used triggered updates: each car estimates the travel time along each segment it is entering, and if the traversal already took much more time before the segment is left, severe congestion is inferred and a triggered update is sent.

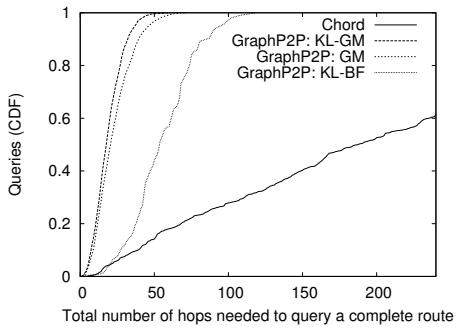
6.2 Graph-based peer-to-peer overlay

We first compared the performance of the graph-based peer-to-peer structure with the general-purpose DHT Chord [20] using periodic requests. The network performance of both systems was measured by the number of overlay hops needed to answer a query completely, as shown in a cumulative distribution plot in Fig. 3a. It can be seen that a graph-based peer-to-peer network improves the hop-count significantly, yet its performance depends on the graph partitioning algorithm employed. The best choice is the algorithm based on geometric mesh partitioning [5] combined with the Kernighan-Lin algorithm [7] (labeled KL-GM in the plot). Simple graph-growing partitioning combined with Kernighan-Lin (KL-BF) performed much worse, but still significantly better than Chord. An important result of these simulations is that a general-purpose DHT tuned to handle independent queries for single keys cannot keep up with overlays preserving the structure of the data.

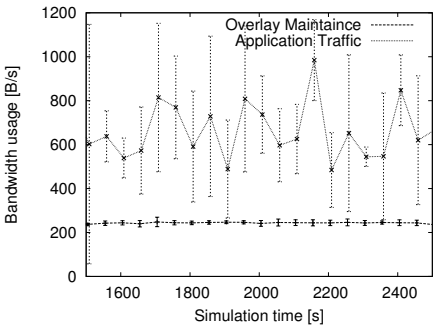
We were also interested in the bandwidth usage of our approach in order to see whether it can be realized over existing mobile access technologies. Fig. 3b shows the per-peer bandwidth required for maintaining the overlay and transporting the application data when the alternative routes approach with $n = 5$ alternatives is used. The error bars show 95 % confidence intervals. It can be seen that, on average, about 800 bytes per second are required—this can easily be achieved even with GPRS.

6.3 Dynamic Routing

Next, we estimated the potential benefits offered by a traffic information system. We specifically compared the results achieved by different routing algorithms: greedy and alternative routes as described in Section 5 versus routing decisions taken without information about the current traffic conditions. Again, periodic updates were used. The results of our experiments are depicted in Fig. 4a, which shows the cumulative distribution of the relative travel time changes: by which fraction are travel times reduced or increased if drivers follow the recommendation, compared to their travel time if routing based on static information is used? As the plots show, both the greedy and the alternative routes algorithm yield significant improvements for many of the drivers, in relation to the travel times with static routes. The differences between the two request algorithms, though, are negligible.

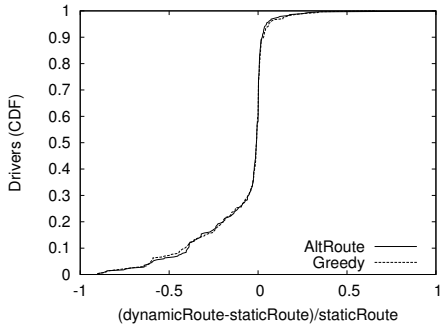


(a) Number of hops to answer a complete query.

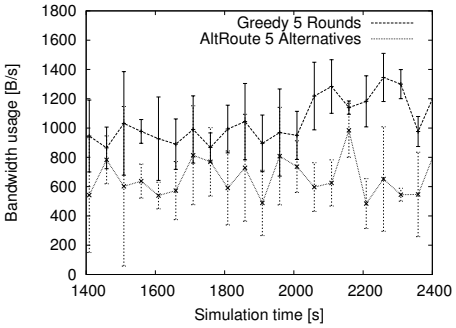


(b) Bandwidth usage (AltRoute 5 alternatives).

Figure 3 Performance of the street graph-based overlay.



(a) Relative travel time changes.

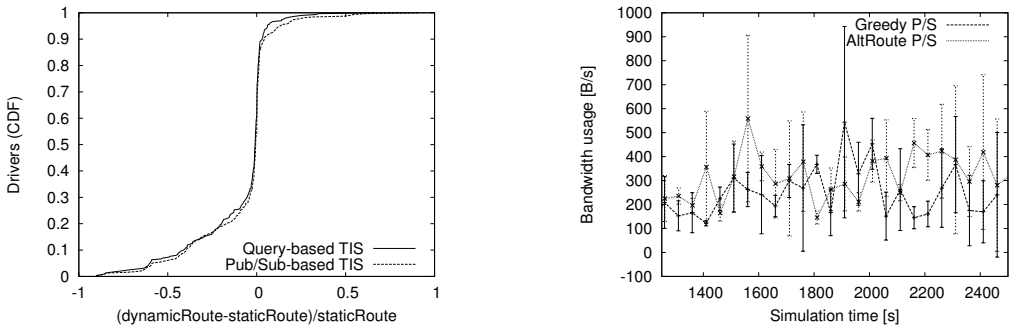


(b) Total bandwidth usage of the application.

Figure 4 Comparison of different vehicle routing algorithms.

However, the greedy algorithm causes significantly higher bandwidth usage in the network. To illustrate this we summarized the size of all packets sent by the application in each simulation second. The load over time plot including 95 % confidence intervals is presented in Fig. 4b. The greedy algorithm requests more data than the alternative routes approach because the measured travel times provided by the cars are typically significantly worse than the static values, so each iteration in the greedy algorithm generates a route that is largely disjoint from the previously calculated ones, and requests information on it. The routes produced by the alternative routes algorithm, on the other hand, were often very similar to each other, and thus resulted in requesting information about a lower number of keys.

We were also able to observe an interesting interrelation between the changes in the road traffic due to the TIS and the overlay structure: due to the dynamic routing decisions, many formerly unused road segments were traversed. This, in turn, increased the fraction of road segments about which data was available in the system substantially. As a consequence, we are convinced that any simulation environment which does not include the feedback loop between application and network (i. e., re-routing of vehicles based on the collected information) is unlikely to capture the full complexity of this environment.



(a) Relative travel time changes (greedy). **(b)** Bandwidth usage.

Figure 5 Comparison of publish/subscribe and query-based traffic information systems.

6.4 Publish/Subscribe

The integration of publish/subscribe aimed at reducing the bandwidth usage in network. The difference in bandwidth usage between systems using periodic queries and publish/subscribe is depicted in Fig. 5b. Especially in combination with the greedy algorithm to determine routes, a substantial difference can observed. These bandwidth savings, however, go hand in hand with a slight degradation in route quality. This is due to the fact that the traffic conditions on a segment have to change significantly before this change is reported to the subscribers. For our choice of the threshold the impact can be seen in Fig. 5a. It should be noted that this is a trade-off: for both periodic requests and publish/subscribe, bandwidth can be saved by accepting less accurate road traffic information.

However, an interesting general effect which we observed is that there is a stabilizing property of the traffic information system, which in turn reduces the bandwidth used for notifications: as more vehicles follow the recommendations, traffic jams are reduced and the road network is better utilized, so that less notifications are necessary.

7 Conclusions

In this paper we presented a novel graph-based peer-to-peer network which takes into account the special properties of traffic information systems. We also reported on first efforts to use a publish/subscribe scheme to reduce the overall network traffic. By employing dynamic routing of vehicles we were able to investigate the benefit that a user can expect from using this system and were able to identify interesting effects regarding the interplay between the overlay structure and the real-world traffic that is influenced by the application.

Acknowledgments

The authors are grateful to the German Research Foundation (DFG) for funding this research.

References

1 Ingmar Baumgart, Bernhard Heep, and Stephan Krause. OverSim: A flexible overlay network simulation framework. In *Global Internet*, pages 79–84, May 2007.

2 Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is NP-hard. *Inf. Process. Lett.*, 42(3):153–159, 1992.

3 Eugene Inseok Chong, Sanjeev Maddila, and Steve Morley. On finding single-source single-destination k shortest paths. In *ICCI '95*, pages 40 – 47. IEEE, July 1995.

4 Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

5 John R. Gilbert, Gray L. Miller, and Shang-Hua Teng. Geometric mesh partitioning: implementation and experiments. In *IPPS '95*, pages 418–427, April 1995.

6 George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

7 Brian W. Keringhan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(1):291–308, February 1970.

8 Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. SUMO (Simulation of Urban MObility): An open-source traffic simulation. In *MESM '02*, pages 183–187, September 2002.

9 Christian Lochert, Björn Scheuermann, Christian Wewetzer, Andreas Luebke, and Martin Mauve. Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System. In *VANET '08*, pages 58–65, September 2008.

10 Tamer Nadeem, Sasan Dashtinezhad, Chunyuan Liao, and Liviu Iftode. TrafficView: traffic data dissemination using car-to-car communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3):6–19, July 2004.

11 OMNeT++. http://www.omnetpp.org.

12 OpenStreetMap. http://www.openstreetmap.org.

13 Dieter Pfoser, Sotiris Brakatsoulas, Petra Brosch, Martina Umlauft, Nektaria Tryfona, and Giorgos Tsironis. Dynamic travel time provision for road networks. In *GIS '08*, 2008.

14 Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *SIGCOMM '01*, pages 161–172, August 2001.

15 John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: search methods. *Comput. Netw.*, 50(17):3485–3521, 2006.

16 Jedrzej Rybicki, Björn Scheuermann, Wolfgang Kiess, Christian Lochert, Pezhman Fallahi, and Martin Mauve. Challenge: Peers on wheels – a road to new traffic information systems. In *MobiCom '07*, pages 215–221, September 2007.

17 Jedrzej Rybicki, Björn Scheuermann, Markus Koegel, and Martin Mauve. PeerTIS - A Peer-to-Peer Traffic Information System. In *VANET '09*, pages 23–32, September 2009.

18 Björn Scheuermann, Christian Lochert, Jedrzej Rybicki, and Martin Mauve. A fundamental scalability criterion for data aggregation in VANETs. In *MobiCom '09*, pages 285–296, September 2009.

19 Christoph Sommer, Armin Schmidt, Yi Chen, Reinhard German, Wolfgang Koch, and Falko Dressler. On the feasibility of UMTS-based traffic information systems. *Ad Hoc Networks*, 8(5):506 – 517, 2010.

20 Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, Marinus Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, 2003.

21 Axel Wegener, Michal Piorkowski, Maxim Raxa, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubeaux. TraCI: An interface for coupling road traffic and network simulators. In *CNS '08*, pages 155–163, April 2008.

22 Lars Wischhof, André Ebner, Hermann Rohling, Matthias Lott, and Rüdiger Halfmann. SOTIS – a self-organizing traffic information system. In *VTC '03-Spring*, pages 2442–2446, April 2003.

23 Yang Yang, Xu Li, Wei Shu, and Min-You Wu. Quality evaluation of vehicle navigation with CPS. In *GLOBECOM '10*, 2010.

Distributed Probabilistic Network Traffic Measurements

Alexander Marold*[1], Peter Lieven[2], and Björn Scheuermann[2]

1 **University of Duisburg-Essen**
Institute for Experimental Mathematics
alexander.marold@iem.uni-due.de
2 **Heinrich Heine University Düsseldorf**
Mobile and Decentralized Networks Group
{lieven,scheuermann}@cs.uni-duesseldorf.de

Abstract

Measuring the per-flow traffic in large networks is very challenging due to the high performance requirements on the one hand, and due to the necessity to merge locally recorded data from multiple routers in order to obtain network-wide statistics on the other hand. The latter is non-trivial because traffic that traversed more than one measurement point must only be counted once, which requires duplicate-insensitive distributed counting mechanisms. Sampling-based traffic accounting as implemented in today's routers results in large approximation errors, and does not allow for merging information from multiple points in the network into network-wide total traffic statistics. Here, we present Distributed Probabilistic Counting (DPC), an algorithm to obtain duplicate-insensitive distributed per-flow traffic statistics based on a probabilistic counting technique. DPC is structurally simple, very fast, and highly parallelizable, and therefore allows for efficient implementations in software and hardware. At the same time it provides very accurate traffic statistics, as we demonstrate based on both artificial and real-world traffic data.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.133

1 Introduction

In this paper, we consider the problem of measuring *network-wide flow sizes* in high-speed network environments. We are interested in determining how many distinct packets of each given flow traversed a network during some time period. These data are relevant, for example, in the context of network usage monitoring, network layout optimization, or data traffic accounting. Gathering such statistics is not straightforward, though, if the considered network has a non-trivial topology: if there is no single, central point in the network that will "see" all the traffic, it is necessary to take measurements at multiple different points to obtain a complete picture. As packets may enter and leave the network at different points, and may take different paths through it, a packet may traverse multiple measurement points. Then, the question arises how to merge the individual routers' measurements without counting packets that have been observed at multiple routers more than once.

The reasons that make this problem so challenging are twofold. First, solving it requires distributed duplicate-insensitive counting, which is algorithmically challenging: if router A observed 500 packets of flow x, and router B saw 800 packets of this flow, how can we determine how many *distinct* packets of this flow have traversed the network? Did all of the 500 packets at A also traverse B, or did B just see a subset of them? Or are these two

* The work described herein was done while Alexander Marold was with the University of Düsseldorf.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 133–144

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

sets of packets completely disjoint, so that there were a total of 1,300 distinct packets of flow x the network? Second, it must be taken into account that traffic measurements in high-speed network environments are a highly performance-critical task. For a single 40 Gbps link, the available time for processing a minimum-size IP packet is only 12 ns. Switching and routing equipment with a potentially large number of high-speed ports cannot provide enough processing power and memory bandwidth to maintain exact per-flow statistics—at least not at reasonable cost [14, 21, 29, 22]. As Gilder's law states, network bandwidth increases more rapidly than processing power, so the situation is getting worse and worse.

Here, we propose a technique which is able to perform distributed, duplicate-insensitive traffic accounting with extremely low effort per processed packet. Our technique—termed Distributed Probabilistic Counting (DPC)—uses probabilistic techniques to obtain approximate flow sizes. To this end, DPC builds upon ideas developed in our Probabilistic Multiplicity Counting (PMC) algorithm first described in [22]. However, while PMC allows for high performance traffic accounting at a single measurement point with very high accuracy, it makes heavy use of random numbers and, as a result, does not allow to merge traffic statistics from multiple routers in a duplicate-insensitive manner. DPC avoids these random decisions entirely and is able to perform duplicate-insensitive merging of distributed measurements. This ability is the key distinguishing feature of DPC.

The probabilistic approach employed in DPC deliberately waives perfect accuracy and accepts an (adjustable) estimation error. In turn, it becomes possible to perform distributed duplicate-insensitive measurements with minimal computational and storage requirements: DPC is able to record information on a passing-by packet by setting only one single bit in a bit field. This operation can be performed in constant time, without loops or conditional branches in the code, and with write-only memory access. It is therefore ideally suited for pipelined and parallelized implementation, and can also easily and efficiently be realized in hardware. We will first focus on a basic DPC variant which can count the *number* of packets, and then also discuss how to measure the total *size* of the packets in a flow.

The remainder of this paper is structured as follows. We first discuss related work in the area of traffic monitoring in Sec. 2. Subsequently, we derive the Distributed Probabilistic Counting algorithm step-by-step from its basic building blocks in Sec. 3. We evaluate DPC with both artificial network traffic patterns and based on real-world network traces in Sec. 4, before we conclude this paper with a summary in Sec. 5.

2 Related Work

Due to its high practical relevance, the topic of flow measurement has attracted a lot of attention in recent years, and many approaches have been proposed. However, they typically cannot be used for duplicate-insensitive multipoint measurements. This fundamental constrain applies to deterministic techniques like [27, 26, 23], and likewise to packet sampling based network usage data generation and collection approaches like Cisco's NetFlow [2], IPFIX [29], sFlow [18], Sampled NetFlow [3], Sketch-Guided Sampling [20] or ANLS [15]. The well-known "Sample-and-Hold" approach [9] and its successors [25, 19, 6] reduce the storage overhead at a single measurement point, but can again not easily be generalized to duplicate-insensitive multipoint measurements. The same holds true for hash-based and (pseudo-)randomized traffic accounting techniques including Spectral Bloom Filters [4], Count-Min Sketches [5], Counter Braids [23], MRSCBF [7], and also for our own prior work PMC [22].

The lack of techniques to merge traffic statistics from multiple measurement points into a network-wide per-flow total has previously been recognized as a problem. Nevertheless,

there is only one existing approach that is able to fill this gap: hash-based sampling [11]. This technique builds upon existing sampling techniques as employed in, e. g., Sampled NetFlow. These standard techniques either record every n-th packet passing by, or they randomize the process and sample each packet with probability $1/n$. Based on the sampled subset of packets at a measurement point, the total amount of data is estimated. However, since only a (random) subset of the packets is considered, duplicate-insensitive merging of observations is not possible: different measurement points will draw different sample subsets of the passing-by packets, so one cannot reliably decide if two measurement points have seen entirely different sets of packets, or whether the same set of packets passed by and the nodes have just drawn different sample subsets from it.

Hash-based sampling overcomes this problem by using a hash value over the packet contents as a basis for the sampling decision. For instance, a packet might be sampled if its hash value modulo n is zero, which results in one out of n packets being sampled in expectation. This has the benefit that a packet will either be sampled at all measurement points it traverses, or at none of them, so that sampled packet sets from distinct measurement points can be compared. However, this approach exhibits a number of drawbacks. First, it shares the well-known granularity drawbacks of all sampling-based approaches [8, 17, 1, 9]. Moreover, in order to allow for duplicate-insensitive merging, it is necessary to collect and store additional payload data for each sampled packet, which implies significant storage requirements at the measurement points. Our proposed algorithm DPC is structurally so simple that it can process each packet, so sampling is not required. It does also not need to collect any per-packet information—and still allows for duplicate-insensitive merging of multiple measurement points' data.

3 The Distributed Probabilistic Counting Algorithm

Distributed Probabilistic Counting (DPC) builds upon the concept of FM sketches [10] to allow for efficient duplicate-insensitive counting. On this basis, it is able to process a traversing packet in $O(1)$ time. In the course of this section, we will first recapitulate the essential prerequisites for understanding DPC, which in particular includes a brief outline of the key ideas behind FM sketches. Subsequently, we discuss how distributed network flow size measurement can be accomplished on this basis.

Before we dive into the details of the algorithms, though, it is helpful to concretize the notion of a "flow" that we are going to use throughout this paper. By a flow we denote a subset of the packets in the network that are characterized by some common criterion. Each packet belongs to exactly one flow. For instance, a flow might consist of all the packets originating from the same source address, belonging to the same protocol, or traveling between the same pair of communicating hosts or subnets. The techniques proposed in this paper are independent from the specific criterion used to characterize the flows, as long as it is possible to determine the flow from the packet with little effort. For most practical purposes, determining the flow ID of a given packet does not incur more effort than to extract one or two header fields. This is trivial in both software and hardware. In the following, we may therefore assume that the flow ID of each processed packet is known.

3.1 FM sketches

FM sketches have been introduced by Flajolet and Martin to estimate the number of distinct elements in a multiset. Determining this number exactly requires $\Omega(n \log n)$ time for a multiset of size n; FM sketches provide a good estimate in linear time, i. e., with constant

effort per element. Before we step into the details of their application to network flow measurement, we briefly summarize the key concepts of FM sketches; a much more detailed description and analysis can be found in Flajolet and Martin's original publication [10].

An FM sketch is based upon a bit field $S = s_1, \ldots, s_w$, $w \geq 1$ and a hash function h_1 with geometrically distributed positive integer output, where the probability that $h_1(x) = i$ ($i \geq 1$) for any randomly picked element x equals $P(h_1(x) = i) = 2^{-i}$. The bit field S is initialized to zero. When an FM sketch is used to estimate the cardinality of a multiset M, each element $x \in M$ is hashed using h_1. Each output of h_1 is interpreted as an index in S, and the corresponding bit $s_{h_1(x)}$ is set to one (regardless of its current value). This leads to a bit pattern emerging in S, which, due to the geometric distribution of h_1, will typically have many 1-bits on the left hand side and many 0-bits on the right hand side.

Flajolet and Martin found that a good estimate for the number of distinct elements can be obtained from the length of the uninterrupted, initial sequence of ones in S, i. e., from

$$Z(S) := \min\{i \in \mathbb{N}_0 \mid s_{i+1} = 0\}. \tag{1}$$

The estimate C is calculated from Z as

$$C = \frac{2^Z}{\varphi} \qquad \text{where} \qquad \varphi \approx 0.77351. \tag{2}$$

The variance of $Z(S)$ is quite significant, and thus the approximation is not very accurate. This can be improved by using multiple sketches in parallel to represent a single value, essentially trading off accuracy against memory. The respective technique is called Probabilistic Counting with Stochastic Averaging (PCSA) in [10]. With PCSA, each element or packet is first mapped to one of the sketches by using a uniformly distributed hash function h_2, and is then added to this (and only this) sketch.

If m sketches are used, denoted by $S_1, \ldots, S_m$, then the estimate for the total number of distinct items added is given by

$$C(S_1, \ldots, S_m) := m \cdot \frac{2^{\sum_{i=1}^{m} Z(S_i)/m}}{\varphi}. \tag{3}$$

One can identify a PCSA set with an $m \times w$ matrix, where each row is a standard FM sketch. Upon addition of an element, a uniformly distributed hash function h_2 selects one row, and a second, independent, geometrically distributed hash function h_1 picks one column. The bit located at these coordinates in the matrix is then set to one.

For a sufficiently large number of elements, PCSA yields a standard error of approximately $0.78/\sqrt{m}$ [10]. Increasing m thus results in a higher estimation accuracy. Note that increasing m also increases the total number of 1-bits in the PCSA matrix for a given, fixed multiset; this trait will be important for understanding DPC's parameter tradeoffs discussed in our evaluation. For very small element counts in the order of m or below, there are well-known initial inaccuracies. These can be partially alleviated by switching to a different evaluation method, based on "hit counting" [28], when Z is small. For details, we would like to refer the reader to the respective discussion in [22].

FM sketches (and likewise PCSA matrices) can be merged to obtain the total number of distinct elements added to any of them by a simple bit-wise OR. Combining the FM sketch with all elements of set A and the FM sketch with all elements of set B using bit-wise OR produces a FM sketch that is identical to the sketch of set $A \cup B$. Elements present in both A and B will obviously not be counted twice, since the respective bit will always have value 1 in both sketches. This trait—termed duplicate insensitivity—is an essential key to the operation of DPC, and one key reason why FM sketches are an ideal basis for distributed traffic measurement.

3.2 FM sketches for flow accounting

We will now consider the case of measuring traffic data in a very much simplified case, where there is only one single flow f in the network. Subsequently, we will then turn to the problem of handling an arbitrary number of flows in the network in parallel.

We consider the set P_f of all packets from f. Of course, we can only measure traffic that has traversed at least one measurement point; we will therefore assume that there are enough measurement points in the network to observe each packet at least once (e. g., all routers or, for transit traffic, all entry/exit nodes). Consequently, each measurement point $m \in M$ has observed a subset of $P_{f,m} \subseteq P_f$, such that

$$P_f = \bigcup_{m \in M} P_{f,m}.$$

In general, the subsets $P_{f,m}$ will not be disjoint. Due to the duplicate insensitivity property of FM sketches discussed above, though, the overlap between the observed packet sets of distinct measurement points is not an issue if FM sketches are used to determine packet counts: the sketches from multiple measurement points can then be merged using bit-wise logical OR, and packets that have been recorded at multiple measurement points will only be counted once. Bringing this idea to reality, though, poses a number of challenges.

The first question that arises is what to use as the input for hash functions h_1 and h_2. There are two conflicting aims: one the one hand, distinct packets should yield different inputs to the hash functions—if this does not hold, the packets are essentially considered identical, and the duplicate insensitivity property of FM sketches will result in an underestimation of the flow size. From this perspective, it appears desirable to use the maximum available amount of information—that is, the complete packet—as the hash input. On the other hand, though, processing packets at the measurement points is highly performance critical. Therefore, it is reasonable to keep the amount of data that needs to be hashed as small as possible. There is clearly a tradeoff.

In PMC, the problems associated with deciding what to hash are overcome by introducing random decisions in the algorithm. This, however, comes at the cost of sacrificing the duplicate insensitivitiy property of FM sketches. We explicitly need duplicate insensitivity for distributed measurements in DPC, so the solution employed in PMC is not an option here. However, we can take the work in the context of hash-based sampling into account, where a similar tradeoff exists. It has been explored in that context by Henke et al. [12], who found that using a subset of the network and transport header fields as the hash function input is enough. They propose a solution which includes different transport header fields depending on the used transport layer protocol. In the context of DPC, this is undesirable, as it requires complex decisions and conditional branches in the algorithm, which contradicts DPC's aim to stay algorithmically very simple, in particular avoiding conditional branches. We therefore opt for a solution which includes a fixed subset of the bytes in each packet.

In order to investigate such a solution, we performed experiments to see how much data is necessary to distinguish packets. As the data basis we used 17 hours of real-world traffic captured in the network of FH Salzburg and made available in the MOME database [16]. This is the same data set that has also been used by Henke et al.. We extract varying amounts of data from the packets in this trace: the IP header (ignoring those fields that are modified by intermediate routers) plus an increasing number of bytes from the IP payload. We then determined in each case how many distinct packets resulted in identical extracted byte strings, and thus in identical inputs to h_1 and h_2.

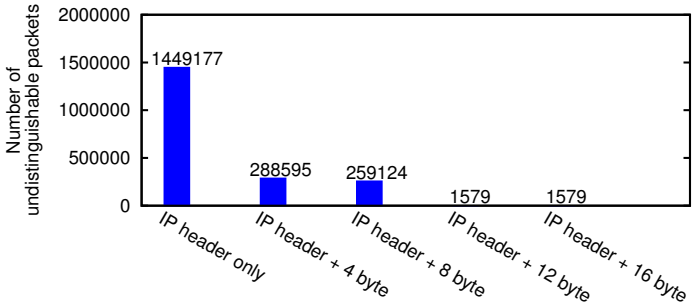


Figure 1 Distinguishability of packets based on varying amounts of IP payload.

Figure 1 shows how many of the packets in the MOME Salzburg traces were indistinguishable from a previously seen packet, based on comparing the non-volatile fields of the IP header plus a varying initial fraction of the IP payload (0–16 bytes). The traces contain a total of ca. 12.6 million packets. Using the IP header alone is clearly insufficient, as this leads to huge numbers of non-distinguishable packets. As expected, when more bytes from the IP payload are taken into account, the accuracy increases. The steps to using 4, 8, and 12 bytes of IP payload in addition to the IP header result in significant improvements. Then, however, the number of undistinguishable packets has become very low already, and increasing the amount of data taken into account further to 16 bytes does not yield further benefits. Therefore, we use the IP header plus the first 12 bytes of IP payload as the input to hash functions h_1 and h_2 in the following.

3.3 Measuring all flows in parallel

So far, we have discussed how an FM sketch can be used to measure the number of packets in an individual flow. This alone, however, is of limited utility: even if each router used an FM sketch in the way discussed above for each individual flow, this would require to locate the FM sketch for the respective flow in the router's memory whenever a packet is to be recorded. This in turn would require using some lookup data structure to find the corresponding FM sketch for each processed packet—which clearly incurs far too much overhead.

We therefore take up the idea of *virtual matrices* from PMC [22]: there, the information about individual flows is not stored explicitly and separately. Instead, all bits describing all flows are mapped to one single, unstructured bit field B of size l bits. The parameter l can be chosen arbitrarily and is typically in the order of a few megabits. This mapping is performed using another (uniform) hash function H, which maps a tuple of a flow ID plus the row and column coordinates of a PCSA matrix entry to one of the bits in B, i. e.,

$$H : \mathcal{F} \times \{1, \ldots, m\} \times \{1, \ldots, w\} \to \{1, \ldots, l\},$$

where $\mathcal{F}$ is the set of possible flow IDs and m and w are the dimensions of the virtual PCSA matrices. This is visualized in Figure 2. If the bit at row i and column j in the virtual PCSA matrix of flow f is to be accessed, $H(f, i, j)$ yields the corresponding index in B.

Recording information about packets in a router using DPC consequently works as follows: initially, all positions in B are set to zero. Whenever a packet from flow f is encountered, it is hashed using hash functions h_1 and h_2 (using the IP header plus 12 bytes of IP payload, as discussed above). This yields the row and column indices—denoted by i and j, respectively—in f's virtual matrix. Then, $H(f, i, j)$ is evaluated and the resulting bit in B is set to one.

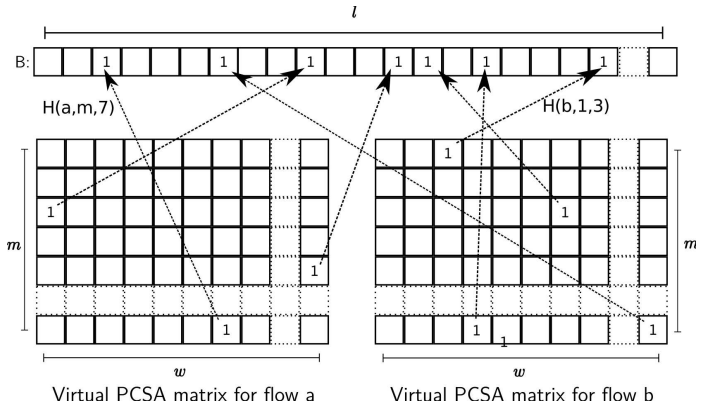


Figure 2 Mapping of virtual matrix entries to B [22].

All these operations can be implemented very efficiently in both software and hardware. In particular, if hash functions are chosen carefully, implementations without conditional branches are possible, which is ideal for using the algorithm on pipelined CPU designs. Moreover, DPC inherits the write-only property of FM sketches: counting a packet with DPC does not require reading from memory—instead, only one single bit needs to be written to B. The address of this bit can be determined from the packet alone, without looking up any information from memory. In particular, there is no need for a complex lookup data structure. This, too, makes the algorithm a perfect match for highly performance critical tasks like network traffic monitoring: in contrast to other approaches like, e. g., per-flow counters in combination with hash-based sampling, DPC's per-packet effort is constant.

Of course, it may (and will) occur that multiple positions from the same or different virtual matrices are mapped to the same bit in B. This must be taken into account when an estimate is extracted from a virtual matrix: it may happen that a value of one is read from bit position $H(f, i, j)$, even though the addressed entry (i, j) in the virtual matrix of flow f has never been set to one during the measurement period—simply because $H(f, i, j)$ is equal to $H(f', i', j')$ and the bit at position (i', j') in the virtual matrix of flow f' *has* been set to one. We call this a "false positive bit".[1]

In [22], we have argued that the probability that this happens is identical to the *fill rate* p of bit field B, i. e., to the fraction of bits in B that are set to one. The fill rate p is trivial to determine from B, and can be used to adjust Flajolet and Martin's estimation formula accordingly. Due to space limitations it is not possible to repeat the reasoning in detail here, but the key result from [22] is that it suffices to substitute Flajolet and Martin's constant φ in (2) or (3), respectively, by a value φ_p which takes the bit field fill rate p into account. This φ_p can be obtained as the following limit

$$\varphi_p = \lim_{n\to\infty} 2^{E[Z(n,p)]}/n \tag{4}$$

where

$$E[Z(n,p)] = \sum_{k=1}^{w} k \cdot (q_k(n,p) - q_{k+1}(n,p)) \quad \text{with} \quad q_k(n,p) = \prod_{i=1}^{k} \left[1 - (1-2^{-i})^n \cdot (1-p)\right].$$

[1] Note that the inverse case of a "false negative bit" cannot occur, since bits are never reset to zero during one measurement interval.

The convergence in (4) is so fast that simply evaluating the formula for a sufficiently large value of n (e. g., $n = 10^5$) suffices to obtain a practically usable value for φ_p.

Consequently, after a measurement period (e. g., one day) is over, the bit fields B can be collected from all measurement points in the network. They can then be merged in a duplicate-insensitive manner by a bit-wise OR operation. For each flow f of interest, it is then possible to extract the virtual PCSA matrix by iterating over the tuples (f, i, j) for all virtual matrix entries (i, j), evaluating H for each of them, and thereby re-constructing f's virtual PCSA matrix. This matrix can then be evaluated using Flajolet and Martin's methodology, with the modification of using φ_p in the role of φ as discussed above. The result is a duplicate-insensitive estimate for the total number of packets from flow f observed in the entire network. By combining not all bit fields from all measurement points, but only some of them, it is of course likewise possible to obtain per-flow statistics for different parts of the network or for individual measurement points—based on the very same raw data.

3.4 Determining flow sizes instead of packet counts

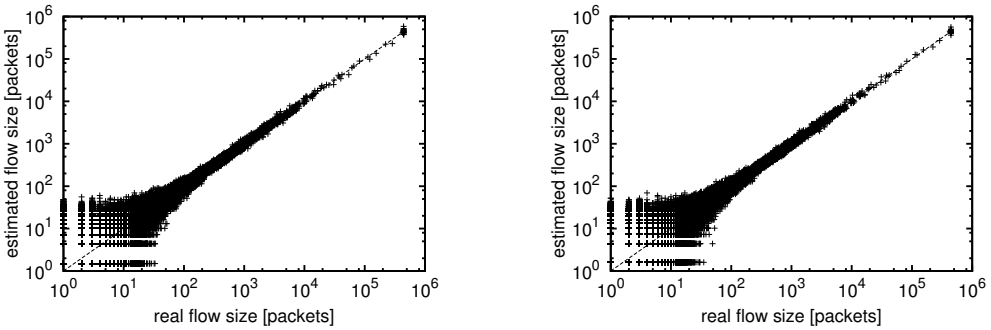
In some applications, including billing of customers of an ISP, it is not only interesting to count the number of packets in a flow. Instead, the total size of the flow in bytes is required. To solve this, [22] suggests to draw a random integer in the range $[0, \mathrm{MTU} - 1]$ for each packet. The packet is counted if and only if this random number is less than the size of the packet. This essantially leads to probabilistically counting "MTU equivalents" instead of packets. The flow size estimate is obtained by multiplying the estimate with the MTU.

This cannot be done in the same way for DPC, though: the random decision may lead to a packet being counted at one measurement point, but not at another. Duplicate-insensitive merging as discussed above is then not possible. However, following the spirit of hash-based sampling, we can substitute the random decision by a hash-based one, where instead of the random number a (uniform) hash value over the packet is used (with a hash function that is independent from h_1, h_2). A bit in B is set if and only if the packet size exceeds this hash value modulo the MTU. Then, a packet will either lead to a bit in B being set at all measurement points, or at none of them, avoiding the above discussed problem. We therefore obtain a variant of DPC that can perform distributed flow size measurements.

4 Evaluation

We evaluated DPC by simulating artificial network flows in a small and static topology on the one hand, and by applying it to traffic traces from a real university network on the other hand. This provides insights into the performance of the algorithm both under controlled conditions and when applied to real-world network traffic.

Let us first consider real-world network traffic. We use the freely available traces from FH Salzburg [16] that have been employed above in Sec. 3.2 already. These traces were captured at a mirror port of a 4 Mbps link using tcpdump and were anonymized with tcpdpriv. They contain TCP, UDP, and ICMP traffic without payload. These traces are from one single measurement point only; we can therefore use them to compare DPC's results to PMC, but we cannot assess DPC's distributed measurement capabilities. PMC is structurally similar to DPC, and has been shown to greatly outperform sampling-based traffic accounting and other bitfield-based techniques in [22]. Here, we use both PMC and DPC with a bit field size l of 4 MBit, $m = 64$ rows per virtual matrix, and SHA1 for hashing (using some of the bits to construct h_1, others for h_2).



(a) PMC. **(b)** DPC.

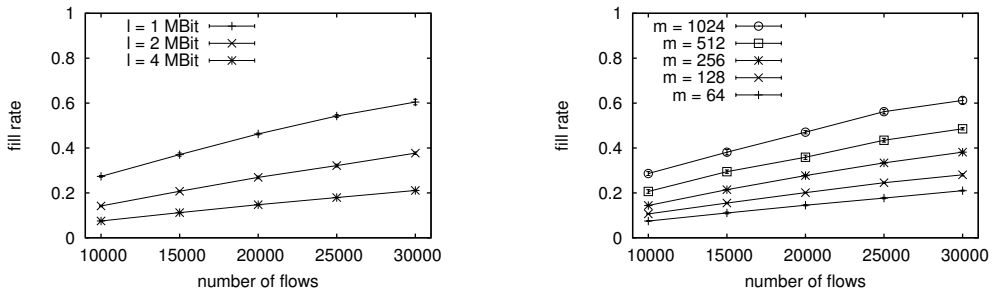
Figure 3 PMC and DPC flow size estimation accuracy on the FH Salzburg traces.

In Figure 3, each data point represents one flow. The x axes show the true flow sizes, the y axes the corresponding estimates (note the log scales). For a perfect estimation, all points would therefore lie on the line of identity (here indicated as a dashed line). The subfigures show the results for DPC and PMC, respectively, if applied to the same network traffic.

Both algorithms produce accurate results at least for flows of non-negligible size; they do not differ in their accuracy. This becomes clear if we consider the standard error. When calculating the standard error in our setting, there is one caveat, though: large deviations between correct flow size and estimate are visible only on the left hand side of the plots. There, the absolute flow sizes are very small, so an estimation error of only a few packets already causes large relative deviations. This expected effect is due to the inherent inaccuracies of FM sketches for small estimates, which can be compensated only to a certain extent as discussed in Sec. 3.1. If not handled appropriately, these large relative deviations for very small flows dominate the standard error, and do not allow to appropriately judge the estimation quality for (practically much more relevant) flows of non-negligible size. We therefore consider the standard error for those flows which exceed a size of m packets (as discussed in Sec 3.1, this is the point where the inaccuracies vanish). While the standard errors without this adjustment are 5.879 (for PMC) and 5.854 (for DPC) in this experiment, they drop to 0.1669 (for PMC) and 0.1673 (for DPC) when only flows of size $> m$ are taken into account. Whichever of the two variants is considered, it becomes clear that the accuracy differences between DPC and PMC observed here are vanishingly small and statistically not significant.

Since the central intention behind DPC is to allow for duplicate-insensitive measurements, we now turn towards a more detailed assessment of DPC in a distributed environment. To this end, we use simulated network traffic in a structurally simple, artificial network with multiple measurement points. This network consists of four routers A–D in a fully connected topology. Each of the four nodes runs one instance of DPC. We generated a varying number of flows in the network, with Pareto-distributed flow sizes (shape parameter $\alpha = 0.5$) to model the Internet traffic flow size distribution [24, 13]. We randomly picked one out of six possible routes for each flow, all with equal probability.[2] We then generated the packets for each flow (with random payload) and recorded them in all traversed DPC bit fields. Since the order and exact timing of packet arrivals at the router has no influence at all on

[2] The configured routes were: route 1: all packets A→D; route 2: all packets 3→4→2; route 3: 50 % of the packets 1→2, 50 % 1→3; route 4: 50 % of the packets 2→4, 50 % 3→4; route 5: 50 % of the packets 1→2, 50 % 3→4; route 6: all packets 1→2→3→4.



(a) Varying l (fixed $m = 64$). **(b)** Varying m (fixed $l = 4\,\text{MBit}$).

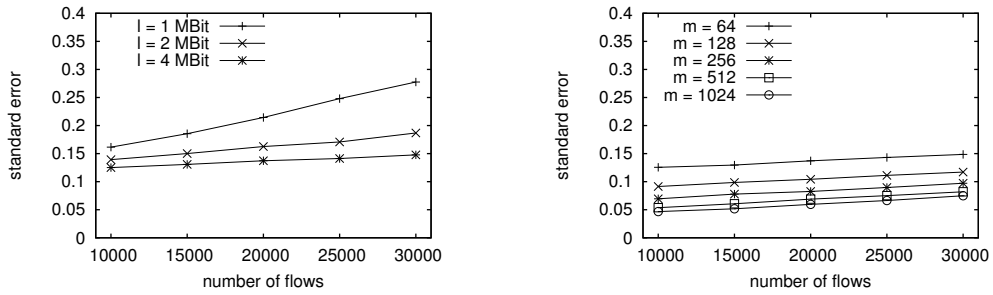
Figure 4 Impact of the number of flows on the fill rate, for varying l and m.

the resulting bit field, it is not necessary to employ a full-scale network simulator for these experiments. At each node, the result was a DPC bit field, containing information about all packets encountered by the respective router. The bit fields from all routers in the topology were subsequently combined for evaluation. Based on this combined bit field, the flow size for each flow was estimated.

The fill rate p plays a central role for the estimation accuracy of DPC. The more bits in B are set, the more false positive bits will occur, which in turn increases the estimation error. We are therefore interested in how fast p grows with an increasing number of recorded flows, and performed a corresponding evaluation. Figure 4 shows the results for different values of m and different bit field sizes l, as the number of flows increases. The figure includes 95 % confidence intervals; the error bars are so small that they are barely visible, though. As one would expect, using a larger bit field results in lower fill rates.

Figure 5 shows that lower fill rates directly translate into more accurate estimates, if other parameters remain unchanged. It shows the resulting standard errors for the same parameter combinations as above (for the reasons stated before, only flows of size $> m$ are taken into account; this figure does not include error bars, since the y values already *are* a quantification of the error). These insights imply that it is possible to increase the accuracy by simply increasing the size of the available memory—or by decreasing the measurement interval duration, as a lower total number of sampled packets, too, is an obvious means to reduce the fill rate of the bit field. Note that the used bit field size has no influence on the computational effort for processing one packet, so the accuracy can be increased without incurring higher processing effort. This trait stands in sharp contrast to sampling-based techniques, where a higher fraction of packets need to be sampled to achieve higher accuracy.

The impact of the number of rows in the virtual PCSA matrix (parameter m) is slightly more complex. As m inherently limits the estimation accuracy of the underlying FM sketches, it is, at some point, necessary to increase m to achieve higher accuracies. Just like using a larger bit field, increasing m does not increase the effort for processing a packet. As stated in Sec. 3.1, though, a higher value of m results in a higher number of 1-bits per virtual matrix, and thus in turn increases the fill rate. This effect is visible in Figure 4b. It causes a higher number of false positive bits and thus reduces the accuracy benefits obtained by increasing m in the first place. Despite this inherent tradeoff, finding a good value for m is not too difficult: as Figures 4 and 5 show, the impact of this parameter is again just as one would expect, and all degradations are very graceful. Depending on the accuracy requirements, values of m in the range between 32 and 1024 seem reasonable in practice, where for higher values it is advisable to use correspondingly larger bit fields (or shorter measurement intervals).



(a) Varying l (fixed $m = 64$). **(b)** Varying m (fixed $l = 4\,\text{MBit}$).

Figure 5 Impact of the number of flows on the standard error, for varying l and m.

5 Conclusion

In this paper, we have introduced Distributed Probabilistic Counting (DPC), an algorithm for obtaining flow size estimates in an entire network. With DPC, routers in the network can independently measure the network traffic during a measurement period. Their local observations—represented in an unstructured bit field—can then be merged in a duplicate-insensitive manner to yield statistics for the network in total. DPC is based on probabilistic counting techniques to perform this task with minimal effort per processed packet. We have shown that DPC yields accurate flow sizes estimates using both simulated network traffic patterns and real-world traffic traces.

References

1 Baek-Young Choi and Supratik Bhattacharyya. Observations on Cisco sampled NetFlow. *SIGMETRICS Performance Evaluation Review*, 33(3):18–23, 2005.

2 Cisco Systems. NetFlow. http://www.cisco.com/web/go/netflow.

3 Cisco Systems. Sampled NetFlow. http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/12s_sanf.html.

4 Saar Cohen and Yossi Matias. Spectral bloom filters. In *SIGMOD '03: Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 241–252, June 2003.

5 Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms*, 55:29–38, 2004.

6 Xenofontas Dimitropoulos, Paul Hurley, and Andreas Kind. Probabilistic lossy counting: An efficient algorithm for finding heavy hitters. *SIGCOMM Computer Communications Review*, 38(1), 2008.

7 Ruan Donghua, Lin Chuang, Chen Zhen, Ni Jia, and Peter D. Ungsunan. Handling high speed traffic measurement using network processors. In *ICCT '06: Proceedings of the International Conference on Communication Technology*, pages 1–5, November 2006.

8 Nick Duffield, Carsten Lund, and Mikkel Thorup. Charging from sampled network usage. In *IMW '01*, pages 245–256.

9 Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 21(3):270–313, 2003.

10 Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, October 1985.

11 Christian Henke, Carsten Schmoll, and Tanja Zseby. Empirical evaluation of hash functions for multipoint measurements. *SIGCOMM Computer Comm. Review*, 38(3):39–50, 2008.

12 Christian Henke, Carsten Schmoll, and Tanja Zseby. Evaluation of header field entropy for hash-based packet selection. In *PAM '08: Proceedings of the Passive and Active Measurement Conference*, pages 82–91, April 2008.

13 Félix Hernández-Campos, J. S. Marron, Gennady Samorodnitsky, and F. D. Smith. Variable heavy tails in Internet traffic. *Elsevier Performance Evaluation*, 58(2+3):261–261, 2004.

14 Nicolas Hohn and Darryl Veitch. Inverting sampled traffic. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 222–233, October 2003.

15 Chengchen Hu, Sheng Wang, Jia Tian, Bin Liu, Yu Cheng, and Yan Chen. Accurate and efficient traffic monitoring using adaptive non-linear sampling method. In *INFOCOM '08: Proceedings of the 27th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 26–30, April 2008.

16 Information Society Technologies (IST) Programme of the European Union. Traffic measurement database MOME. http://www.ist-mome.org/, March 2010.

17 InMon Corp. sFlow accuracy & billing. http://www.inmon.com/pdf/sFlowBilling.pdf.

18 InMon Corp. sFlow version 5. http://sflow.org/sflow_version_5.txt.

19 Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems*, 28(1):51–55, 2003.

20 Abhishek Kumar and Jun Xu. Sketch guided sampling – using on-line estimates of flow size for adaptive data collection. In *INFOCOM '06: Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies*, April 2006.

21 Abishek Kumar, Jun Xu, and Jia Wang. Space-code bloom filter for efficient per-flow traffic measurement. *IEEE Journal on Selected Areas of Communications*, 24(12):2327–2339, December 2006.

22 Peter Lieven and Björn Scheuermann. High-speed per-flow traffic measurement with probabilistic multiplicity counting. In *INFOCOM '10: Proceedings of the 29th Annual Joint Conference of the IEEE Computer and Communications Societies*, March 2010.

23 Yi Lu, Andrea Montanari, Balaji Prabhakar, Sarang Dharmapurikar, and Abdul Kabbani. Counter braids: A novel counter architecture for per-flow measurement. In *SIGMETRICS '08: Proceedings of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 121–132, June 2008.

24 Bruce A. Mah. An empirical model of HTTP network traffic. In *INFOCOM '97: Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies*, April 1997.

25 Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB '02: Proceedings of the 28th International Conference on Very Large Data Bases*, pages 346–357, August 2002.

26 Sriram Ramabhadran and George Varghese. Efficient implementation of a statistics counter architecture. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2003.

27 Devavrat Shah, Sundar Iyer, Balaji Prabhakar, and Nick McKeown. Analysis of a statistics counter architecture. In *HOTI '01: Proceedings of the 9th Symposium on High Performance Interconnects*, August 2001.

28 Kyu young Whang, Brad T. Vander-Zanden, and Howard M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Transactions on Database Systems*, 15(2):208–229, June 1990.

29 Tanja Zseby, Maurizio Molina, Nick Duffield, Saverio Niccolini, and Frederic Raspall. Sampling and filtering techniques for IP packet selection. RFC 5475, March 2009.

A Feasibility Check for Geographical Cluster Based Routing under Inaccurate Node Localization in Wireless Sensor Networks*

Hannes Frey[1] and Ranjith Pillay[2]

1 **University of Paderborn, Germany,** hannes.frey@uni-paderborn.de
2 **Amrita Vishwa Vidyapeetham, Kollam, India,** ranjith@arl.amrita.edu

Abstract

Localized geographic single path routing along a wireless network graph requires exact location information about the network nodes to assure message delivery guarantees. Node localization in practice however is not exact. Errors ranging from several centimeters up to several meters are usual. How to perform localized routing in practice when such errors are prevalent?

In this work we look at a promising routing variant which does not completely overcome this practical problem but which mitigates it. The concept does away with trying to find node positions as precise as possible but allows inaccuracies from the very beginning. It partitions the plane by a regular mesh of hexagons. The only information which is of interest is in which cell of that partitioning a node is located in. Using this node embedding, a virtual geographic overlay graph can then be constructed.

To find the node positions we apply three variants of multidimensional scaling, two of them being a node localization approach which has been well studied in the context of sensor networks and one which we apply here for the first time in that context. Using the location information we get from these localization approaches we embed the nodes into the clusters their location falls into. We define two graph metrics to assess the quality of the overlay graph obtained by the embedding. Applying these two metrics in a simulation study, we show that cluster based routing is an eligible approach to support localized geographic routing when location errors are prevalent.

1998 ACM Subject Classification C.2 [Computer-Communication Networks]: Network Protocols – *Protocol verification*

Keywords and phrases Geographical clustering, face routing, planar graph routing, localization, multi dimensional scaling, simulation study

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.145

1 Introduction

We consider geographic routing along geographical clusters which are resulting from a regular hexagon tiling of the plane. With that approach a node does not need to know its exact location. It is sufficient that each node locates itself in the geographical cluster it is actually located in.

As we briefly describe in Section 2, in conjunction with routing, geographical clustering has been considered to improve robustness of localized routing methods. In these routing

* This work was funded in part by the Erasmus Mundus External Cooperation Window (EMECW) of the European Commission.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 145–156

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

approaches, it is assumed that nodes already know their exact physical location (e.g. by means of GPS). This, however, is not a realistic assumption.

In this work we consider geographical clustering under the assumption that nodes can perform erroneous range measurements with their immediate neighboring nodes. In Section 3 we apply multidimensional scaling, one of the well known localization techniques applied in sensor networks, to estimate node locations based on range measurements. We apply three different multidimensional scaling variants, a non-metric, a classical, and a probabilistic one. The latter variant has not been used in sensor networks so far. Using the position estimates we get from multidimensional scaling, we then embed the nodes in hexagonal geographical clusters. To reduce the number of nodes placed in a wrong cluster we propose a simple heuristic to align the hexagonal grid based on anchor node positions.

The proposed approach is a heuristic and the quality depends on the number of anchor nodes and the extent of the range measurement errors. In Section 4 we define two quality metrics, the connectivity of an overlay graph one can construct over the geographical clusters and the consistency of the view of each node on the cluster structure. Both metrics are of interest when geographical cluster based localization is applied in combination with geographic routing.

With these metrics we performed a simulation study on the quality of geographical clustering under range measurement errors. In Section 5 we first derive a model to express range measurement errors which is used in our simulations. The results of the simulations are then summarized in Section 6. In Section 7 we finally conclude the findings and draw some open directions which could be followed in the sequel.

2 Related Work

2.1 Multidimensional Scaling

Multidimensional Scaling (MDS) [2, 15, 12] is a method used to depict the spatial structure, normally in 2 or 3 dimensions of distance-like data between points in space. MDS aims to find a geometric representation of the data, such that the distances between data points fit as close as possible to the given dissimilarity (or similarity) information.

In the domain of sensor networks MDS has been used to determine node positions. In this work we consider the following well known centralized MDS localization method[1]. Nodes which are neighbors in the wireless network determine their mutual distances and communicate these distances to one central node. This node collects these distance values in a so called dissimilarity matrix and then completes this matrix by the shortest paths between all possible node pairs. The matrix then contains an estimate of the distance between all possible node pairs. The dissimilarity matrix is used as MDS input to determine node coordinates. After that the central node communicates the coordinates back to the network nodes.

Depending on the MDS method applied two kind of node localization can be considered: range free and range based. In range based localization, inter nodal distance measurements are determined through different ranging techniques, viz. RSSI, time of arrival (TOA), or time difference of arrival (TDOA). The goal is to estimate the true Euclidean distance between the neighboring nodes. Whereas, in range free localization Euclidean distances are not considered. The only information available is whether two nodes are connected or not.

[1] Distributed methods of multidimensional scaling based localization can be found in the literature as well but are not subject of study of this work.

Range-free localization based on MDS can be found in [12]. Starting with the given network connectivity information, the shortest path in hop counts between each possible pair of nodes is determined. So called non-metric multidimensional scaling is then applied which allows a dissimilarity matrix just consisting of the hop counts. The resulting coordinates are shifted, rotated and/or reflected about an axis so as to get the anchor nodes (nodes with known location information) as much as in their right locations. The result is a placement of nodes which compares quite well with the nodes' true physical locations.

Range-based localization based on MDS can be found in [11]. The starting point is the Euclidean distance measurements between neighboring pairs of nodes through some range measurement technique. The missing distances are completed by the path lengths of the Euclidean shortest paths between all node pairs. The dissimilarity matrix is then fed into so called classical metric MDS which expects a dissimilarity matrix of Euclidean interpoint distances.

A third MDS variant which we consider in this work is probabilistic MDS. It hitherto has not been explored for wireless sensor localization. Unlike the other two MDS methods, probabilistic MDS [8] can take as input a variance value associated with each distance measurement along with the distance (dissimilarity) matrix. This makes probabilistic MDS more feasible to use in cases where the standard deviation of a ranging measurement technique is known beforehand. It was shown in [8] that probabilistic MDS performs better over classical MDS in node positioning when variances are associated with distance measurements.

2.2 Geographical Cluster Based Routing

Geographical cluster based routing was originally introduced in [5]. It is a combination of routing and topology control based on a virtual geographic overlay graph. To support correct routing, the overlay graph may not contain intersecting links. The original publication described a way to remove intersecting links which was then further refined in a follow up publication [3]. Moreover, the basic idea of geographical cluster based routing was one year later rediscovered in [14]. The publication describes a slightly different way to remove intersecting links. In [13], a follow up of that publication, the same algorithm was simulated with another routing metric. Besides these two works in that year one more algorithm was introduced in [9] which is an alternative to remove intersecting overlay links.

While these works focus on removal of intersecting links, one plain alternative approach is preventing intersecting links in the first place such that no intersections have to be removed later on. This is the variant considered in [4] which forms also the object of investigation of this work. The details of that approach are as follows: As assumed in all geographical cluster based routing variants, nodes have to know their physical location on a two dimensional plane. The plane is partitioned by a regular mesh of hexagons. Each hexagon defines a geographical cluster. Using its location information each node can assign itself to the cluster it is located in. We denote such assignment as $v_1 \in S$. Moreover, $C(v)$ denotes the cluster, a node v is assigned to.

Using the cluster assignment, the geometric overlay graph considered here is then constructed as follows. The centers of the clusters which contain at least one physical network node form the nodes of the overlay graph. The links between the clusters are defined by the physical links. Two clusters C and D are connected if they share a hexagon boundary and if there exists at least one pair of nodes $v \in C$ and $w \in D$ which are connected in the physical network graph. We term these links *short links*, since these overlay graph links are the ones with the smallest Euclidean length.

It is easy to see that the resulting overly graph is always planar. However, simplicity of

the scheme comes at a cost. In sparse networks the overlay graph might be disconnected. However, if the network is dense enough – for instance in sensing covered networks with the double range property defined in [16] – then connectivity of the network can always be assured [4].

Routing along the planar overlay graph is done according to a combined greedy face routing strategy. Whenever possible a message is forwarded to a cluster which is closer to the destination cluster compared to the cluster the message is currently residing in. If no such cluster is found, right hand or left hand rule is applied to traverse a sequence of clusters until a cluster again closer to the destination cluster is found where greedy routing can be resumed. Under right hand or left hand rule a message is forwarded to the overlay graph link which in clockwise or counterclockwise is lying next to the last overlay link visited by the message.

3 Subject of Study

As it will be detailed in Section 5 we assume that nodes which are neighbors in the wireless network determine their mutual distances using RSSI based ranging. In the way already explained in Section 2 we then apply multidimensional scaling (MDS) on one central node to determine the node locations. We consider the three mentioned MDS variants: the non-metric, the metric (also termed classical), and the probabilistic one. The node locations we get out of these MDS variants are then used to determine the clusters the network nodes are located in and each node in the network is then informed by the central node about the cluster it is located in.

This basically explains our subject of study. However, two aspects need further explanation: the way the central node determines exactly the cluster a node is located in, and the way the central node computes the additional variance input required by the probabilistic MDS variant.

To determine the cluster a node is located in, we consider three anchor nodes which already know their positions. In our evaluation we select from the random node set the three nodes which are farthest from each other as anchor nodes. We then align the hexagonal grid with a heuristic to get all the anchor nodes as close to the cell centers. Starting with an initial alignment of the grid, for each anchor node, the hexagon it is located in and the six neighboring hexagons are determined. Then, for each anchor a and for each hexagon h of the seven hexagons determined for a, the grid is shifted such that a is the center of h. Then the grid is rotated around a to get the other anchor nodes closest to the center of one of the seven hexagons determined for them. Among all iterations the grid placement which minimizes the placement error is chosen, while the placement error is the sum of the squared errors of the distance of the anchor nodes from their cell centers.

The variance input for probabilistic MDS is determined with the following heuristic. The actual Euclidean distances between anchor nodes is known a priori. From the disparity matrix fed into MDS we also know the estimated erroneous distances between the anchor nodes. For the three anchor node pairs, the statistical variance of the erroneous distances from the actual true Euclidean distances is then computed.

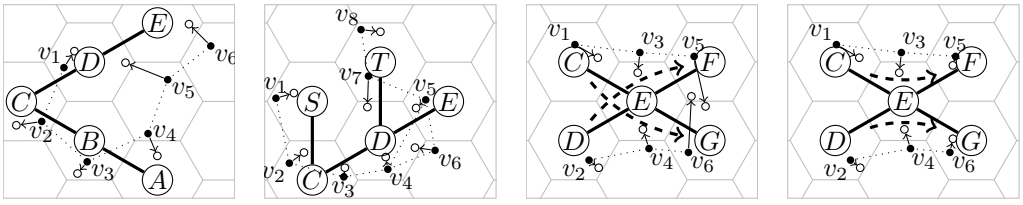
4 Evaluating Cluster Based Localization

If all network nodes are assigned to the right cluster they are physically located in, then the scheme described in [6] supports guaranteed message delivery from one cluster C to another

cluster D; of course, provided that there exists at least one path in the overlay graph which connects C and D and provided that all nodes exchange adjacent clusters they see with all neighbor nodes which are in the same cluster. If we consider, however, that nodes may be wrongly assigned to clusters they are not physically located in, then any localized routing scheme following the links connecting neighboring clusters may fail although the underlying physical network is connected. We observe two phenomena which might occur. One is that the overlay graph might be disconnected and the other one is that the nodes in the same cluster might have an inconsistent view on the adjacent clusters. Both the phenomena and the evaluation metrics we use to account for them are detailed in the next two subsections.

4.1 Number of Partitions

Consider the example depicted in Figure 1a. The true node locations are depicted with solid dots. These are pointing to the determined node locations which deviate from true locations due to localization errors. In the example, the network nodes are connected by a network link – the dotted lines – if the Euclidean distance between the nodes' true locations are less or equal than the cluster diameters. This results in the depicted physical connected network over $v_1, \ldots, v_6$ and the depicted overlay graph over the clusters $A, \ldots, E$. Obviously, the overlay graph is connected. However, sending a message from A to E for instance, following the links will not be successful. A message can get from A to B via the physical link v_4v_3, from B to C via v_3v_2, from C to D via v_2v_1. However, there is no physical link which supports sending a message directly from cluster D to E.



(a) Network partitions. **(b)** Inconsistent views. **(c)** Critical case. **(d)** Benign case.

Figure 1 Potential failure sources in localized, cluster-based routing.

The example shows that just checking connectivity of the overlay graph defined by the cluster centers and the short links one can draw between the cluster centers is not sufficient to see if localized routing along this structure is always successful. To account for such phenomena in our empirical study, given an overlay graph we count the *minimal number of connected cluster partitions* we can construct over the set of clusters. Thereby, a given set of clusters $\mathcal{S}$ is termed *connected* if for any two clusters $C, D \in \mathcal{S}$ there exists a path $v_1v_2 \ldots v_k$ in the physical network such that $C(v_1) = C$, $C(v_k) = D$, and $C(v_i)$ and $C(v_{i+1})$ are either the same or have a hexagon edge in common.

In the network depicted in Figure 1a communication between any pair of clusters in $\{A, B, C, D\}$ is possible based on the path $v_1v_2v_3v_4$. However, no communication is possible between a cluster in $\{A, B, C, D\}$ and cluster E. Thus, in this example we count two connected partitions, one partition is $\{A, B, C, D\}$ and the other one is $\{E\}$.

4.2 Number of Inconsistent Clusters

Consider now the example depicted in Figure 1b. There is a direct connection from cluster S to C via the physical link v_1v_2, from cluster C to D via v_3v_4, from D to E via v_4v_5, from

E back to D via v_5v_7, and then from D to T via v_7v_8. Thus, the clusters S, C, D, E, and T form a single partition. However, the success of planar graph routing based on that overlay graph depends on the actual selection of the physical nodes located in those clusters.

Suppose starting at node v_1 a message is to be sent from cluster S to T. Since S has no adjacent cluster closer to T, face routing has to be employed. This leads to routing along the clusters C, D, and E using the physical network path $v_2v_3v_4v_5$, for instance. Once the message arrives at v_5 in cluster E, the next hop decided at v_5 may either be v_4, v_6, or v_7. If v_5 decides on v_4 or v_6 the message will traverse the clusters D and C back to S and will eventually be dropped at S since nodes v_4 and v_6 know about adjacent clusters C and E but not about T. If in contrast v_5 decides on v_7 the message will travel as well back from cluster E to cluster D, however, at cluster D the message arrives at a node which has a direct connection into cluster T. Thus, the message will eventually reach its destination cluster.

The example shows that in case of location errors it is not sufficient to exchange information about adjacent clusters within a cluster. In the example the node sets $\{v_4, v_5\}$ and $\{v_7\}$ are forming two partitions of the physical nodes in cluster D which can not communicate without resorting to relaying nodes outside of cluster D.

If the overlay graph is connected in the sense defined in the previous subsection and if none of the involved clusters are hosting partitions of the physical nodes then face routing along the overlay links will always find a path to the destination cluster. This follows immediately from the correctness proof in [7]. Thus, testing for clusters with physical node partitions is a reasonable metric for measuring if routing errors might occur with cluster based localization under location errors.

The routing failure depicted in Figure 1b originates from the fact that the inconsistent view of the nodes in cluster D leads to intersecting overlay graph links. In the example, the link DE exists twice. It exists due to the connection of v_5 with v_7 and due to the connection of v_5 with v_4, v_6. Besides intersection due to such edge doubling, intersections may also occur in the way depicted in Figure 1c. In this example the physical network path $v_1v_3v_5$ produces the overlay path CEG, while the path $v_2v_4v_6$ results in the overlay path DEF. The physical network nodes v_3 and v_4 located in cluster E are forming two partitions in that cluster. A message traveling along the edge CE according to the left or right hand rule will always be forwarded along the edge EG, while a message traveling along DE will always be forwarded along the edge EF. Thus, the overlay paths CEG and DEF are basically forming two intersecting links CG and DF. It is well known that face routing in case of such intersecting links may fail. We term this as an inconsistent view and more so, an inconsistent view with duplicate overlay graph links as exemplified in Figure 1b a *critical inconsistent view*.

Besides the critical ones we can also observe what we denote *benign inconsistent views* which will not lead to a routing failure. In the example in Figure 1d the physical nodes v_3 and v_4 in cluster E are forming two partitions in that cluster as well. The difference however with the case in Figure 1c is that the links $\{EC, EF\}$ seen by v_3 and the links $\{ED, EG\}$ seen by v_4 can be interpreted as face boundary links of two faces which are touching each other in one single point. A message traveling along the overlay edge DE according to the left or right hand rule will always be sent to the overlay edge EG next. It will thus never travel inside the upper area of the face having CE and EF on its boundary. The same applies to a message traveling along the overlay edge CE. It will never travel inside the lower area of the face having DE and EG on its boundary.

5 Model Assumptions

5.1 Modeling RSSI Measurements

We consider ranging based on RSSI[2] measurements. Neighboring nodes exchange a sequence of data packets and compute an average over the RSSI values they get for each data packet. We assume that the number of exchanged data packets is sufficient such that fading and interference will average out. In that case the measured average RSSI between two node pairs at different locations but same transmitter receiver separation may still vary due to shadowing and multipath propagation caused by surrounding environmental clutter. We use the well established log-normal shadowing [10] to model this effect. Expressed in dB the signal attenuation $PL(d)[\text{dB}]$ encountered when sender and receiver are separated by a distance d is then given by $PL(d)[\text{dB}] = \overline{PL}(d_0)[\text{dB}] + 10n\log_{10}\left(\frac{d}{d_0}\right) + X_\sigma$, where n is the path loss exponent, $\overline{PL}(d_0)[\text{dB}]$ is the average path loss in free space for reference distance d_0, and X_σ is a zero-mean Gaussian distributed random variable with standard deviation σ.

Given a transmitter output power $P_t[\text{dBm}]$ and signal attenuation due to log-normal shadowing over a transmitter receiver separation d, the average over the RSSI values at the receiving node can then be expressed in dBm as $RSSI(d)[\text{dBm}] = P_t[\text{dBm}] - PL(d)[\text{dB}]$.

5.2 Modeling Ranging Errors

Once a node has determined the RSSI average for a given transmitter, the value has to be mapped to a distance value. We consider the technique of inverting the average large-scale path loss formula. Given path loss exponent n, reference distance d_0, path loss at reference distance $\overline{PL}(d_0)[\text{dB}]$, and transmitter receiver separation d, the theoretical average large-scale path loss $\overline{PL}(d)[\text{dB}]$ is $\overline{PL}(d)[\text{dB}] = \overline{PL}(d_0)[\text{dB}] + 10n\log_{10}\left(\frac{d}{d_0}\right)$. Thus, given transmitter output power $P_t[\text{dBm}]$ the average RSSI value $\overline{RSSI}(d)[\text{dBm}]$ not taking shadowing into account amounts $\overline{RSSI}(d)[\text{dBm}] = P_t[\text{dBm}] - \overline{PL}(d)[\text{dB}]$. By inverting this function we can estimate the transmitter receiver separation $d(x)$ from measured RSSI averages x by $d(x) = d_0 \cdot 10^{\frac{P_t[\text{dBm}] - \overline{PL}(d_0)[\text{dB}] - x}{10n}}$. Substituting x with the expression $RSSI(d)[\text{dB}]$ we use to model measured RSSI averages under shadowing then eliminates $P_t[\text{dBm}]$ and $\overline{PL}(d_0)[\text{dB}]$, yielding the erroneous distance estimate $\tilde{d}$ for a given transmitter receiver separation d as $\tilde{d} = d_0 \cdot 10^{\frac{10n\log_{10}\left(\frac{d}{d_0}\right) + X_\sigma}{10n}} = d \cdot 10^{\frac{X_\sigma}{10n}}$.

6 Evaluation

We present a simulation study on the quality of the MDS based geographical clustering methods as described in Section 3. We consider how localization errors at node level affects correctness of geographical clustering. Moreover, using the metrics derived in Section 4, we express the feasibility of the resulting geographical clustering for geographic routing. The simulations were done with MATLAB using the built in function for non-metric and classical MDS. In the following we term those functions NMDS and CMDS, respectively. In addition, PROSCAL [8] was used to perform probabilistic MDS.

Every depicted measurement point is the average of 100 independent simulation runs. In each simulation run the network nodes are placed uniformly random distributed on a

[2] The received signal strength indicator (RSSI) expresses the power received at the receiver. Many wireless transceivers provide an RSSI value for each received packet and usually a mapping from RSSI values to radio frequency level in dBm can be found in the data sheets.

square field. Then for each node pair the path loss between two nodes is randomly selected according to the log normal model. Connectivity of the wireless network is then tested by checking if all nodes can communicate directly or via intermediate hops. Two nodes are defined as connected if the average RSSI for transmission between the two nodes is on or above the nominal receiver sensitivity. Non-connected networks are discarded.

6.1 Simulation Parameters

For choosing evaluation parameters, we consider TmoteSky sensor nodes which are equipped with the Chipcon CC2420 transceiver as a reference architecture and we consider the empirical average RSSI value estimates for CC2420 transceivers listed in [1]. The latter reports (among others) results of an indoor measurement with one CC240 transmitter and one CC2420 receiver. From standard formulas (see [10]) a reference distance $d_0 = 1\text{m}$ and path loss at reference of $\overline{PL}(d_0)[\text{dB}] \approx 40[\text{dB}]$ can be determined. Applying the least squared error method[3] described in Example 4.9 of [10] on the empirical mean RSSI values listed in table 2 of [1] we determine a path loss exponent $n \approx 2.7$ and a standard deviation $\sigma \approx 3.3$ for the log normal shadowing model.

We conduct simulations with 50 nodes. The simulations are adjusted to path loss exponent 2.7 and vary the log normal standard deviation around the estimated value 3.3. The cluster diameter is adjusted to 100 which is close to the average maximum communication range of the nodes given by the CC2420 nominal receiver sensitivity of $-94[\text{dBm}]$.

To adjust the node density we keep the number of nodes fixed and vary the size of the rectangular simulation field. Given k nodes, average maximum communication range d, and a square field length l, under uniform node distribution the expected neighbors δ in a node's average maximum communication range can (despite boundary effects) roughly be expressed as $\delta = (k-1) \cdot \frac{\pi d^2}{l^2}$. Thus, for a desired network density δ we estimate the square field length by solving for l.

6.2 Results

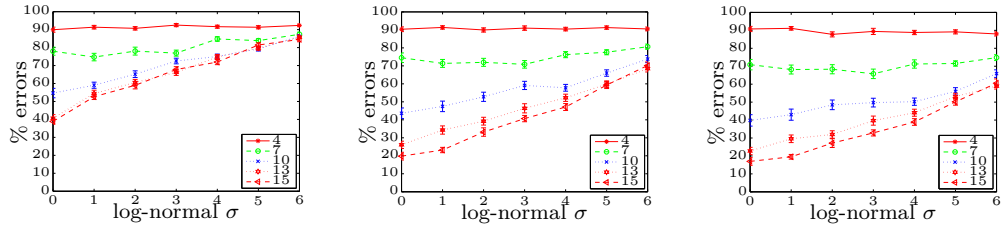
6.2.1 Assignments to Wrong Clusters

The first question we look at is how far the physical node position errors of the MDS variants propagate into the geographical clustering structure. We consider the Geographical clustering of nodes using the physical node positions determined by MDS and then embedding the node into the cluster structure as described in Section 3. We consider the clusters the nodes are placed according to that method and the actual clusters with respect to their true physical positions. For the latter we use the same alignment and orientation of the hexagonal grid computed by the method from Section 3.

Figures 2a, 2b, and 2c show depending on σ of the log normal path loss, the percentage of nodes which are placed into a cluster they are physically not located in. Of course, a higher σ means a higher variance in distance measurement errors which then negatively affects the MDS precision and consequently clustering as well.

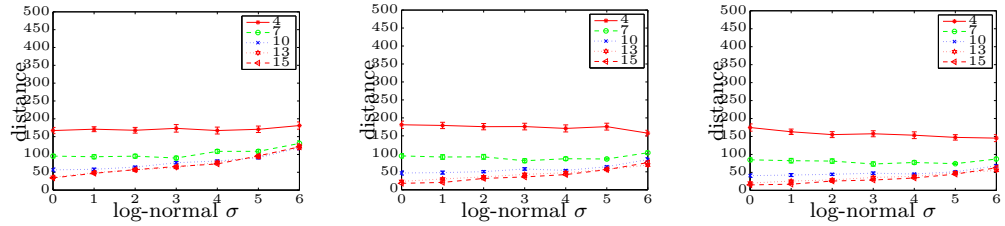
What can be seen as well is how improved quality of physical location information positively affects the assignment of nodes into the right cluster: clustering based on probabilistic

[3] The method finds the path loss exponent n which minimizes the sum over the squared distances between path loss due to log normal shadowing and the empirical values. Having found n, σ is determined by the statistical variance of the empirical and the mean values given by the log normal model.



(a) non-metric **(b)** metric **(c)** probabilistic

Figure 2 Percentage of node placement errors over log-normal σ for different node densities.



(a) non-metric **(b)** metric **(c)** probabilistic

Figure 3 Average distance from original cell over log-normal σ for different node densities.

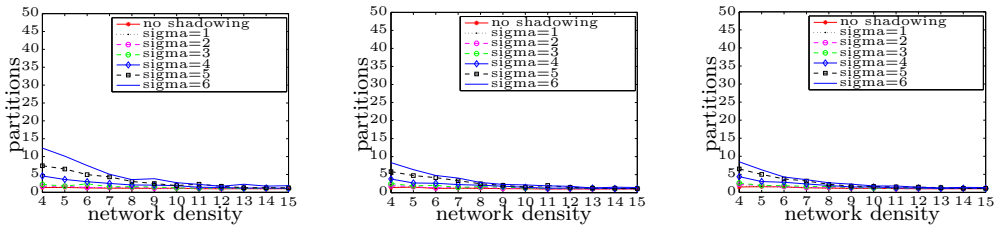
MDS performs slightly better than the one based on classical MDS, and classical MDS performs visibly better than the non-classical one.

Finally, for all methods, increasing the network density reduces the wrong assignment of nodes to clusters. This can be explained by the fact that completing the distance matrix by computing shortest paths among all node pairs has larger distance estimate errors when the network is sparse. Of course, the shortest path length may deviate from the Euclidean distance between the nodes. In the case where the network density tends to infinity, the shortest path becomes the true Euclidean distance between the nodes, and MDS would produce error free node locations, and no node placement errors would occur. In our measurement we went to a network density up to 15 which of course is not an infinite network density, so still results in node placement errors.

6.2.2 Closeness to the Correct Cluster

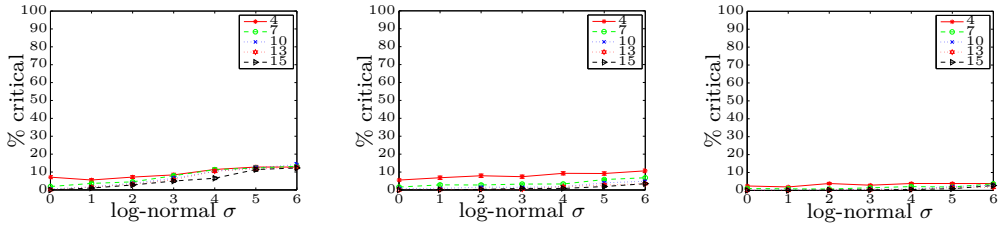
Figures 2a, 2b, and 2c show that particularly in sparse networks almost all nodes are located in a wrong cluster. So the question arises if the considered MDS based geographical clustering makes sense at all if used to reflect the actual geographical positions of nodes. To answer this question in Figures 3a, 3b, and 3c we consider the average Euclidean distance of the center of cluster the node is assigned by the considered MDS based geographical clustering methods and the center of the cluster the node is actually physically located in. Here again we use the alignment and rotation of the hexagon grid resulting from the MDS based geographical clustering methods to decide the cluster a node is physically located in.

We observe that though many assignments to wrong clusters may exists, on an average the deviation from the actual clusters are just local fluctuations. Of course, we consider a cluster diameter of 100. Thus, the Euclidean distance between the centers of two neighboring hexagons is given by $\frac{\sqrt{3}}{2} \approx 87$. The numbers depicted on the y-axis of the Figures 3a, 3b, and 3c show that on an average error distances range within a few hops from the true cluster



(a) non-metric **(b)** metric **(c)** probabilistic

Figure 4 Network partitions over network density for different log-normal σ values.



(a) non-metric **(b)** metric **(c)** probabilistic

Figure 5 Percentage of critical inconsistent clusters over log-normal σ for different node densities.

in low density networks. In high density networks the value is clearly below 87 which means that the combined MDS and clustering methods often hit the right cluster.

The intuition behind the observation in Figures 3a, 3b, and 3c is that the applied MDS strategies do not find a perfect node embedding which reflects the true physical node locations but of course, for all variants it is highly unlikely that a node suddenly happens to be located in another area of the network than its neighboring nodes.

6.2.3 Support of Localized Routing

When considering the Figures 2a, 2b, and 2c, also the question arises if MDS based geographical clustering makes sense at all to be used in combination with geographical cluster based routing. To asses the quality of the overlay graph we apply the metrics which we defined in Section 4. Figures 4a to 5c show though there are local fluctuations in the right cluster assignment, the resulting structure is in fact surprisingly a robust one for geographical cluster based routing.

The average number of network partitions depicted in Figures 4a, 4b, and 4c is about 1 for all MDS variants when σ of the log normal shadowing is low. Only in low network densities performance visibly degrades for all schemes when σ increases. At this, the frequency of partitions is the highest for non-metric MDS, is better for classical MDS and, is the best for the probabilistic one. This is again in accordance with the fact that probabilistic MDS results in node positions closest to the physical node positions, which in terms of precision is then followed by classical MDS and then by non-metric MDS which considers only hop count but not the Euclidean distance.

The same trend can be observed in Figures 5a, 5b, and 5c for the percentage of clusters where nodes have an inconsistent critical view. Within the considered parameter range the fraction of clusters with such inconsistencies is around 10% for non-metric and classical MDS in the worst case. Probabilistic MDS achieves even better results in those worst case parameter settings with a fraction of clusters with inconsistencies never exceeding 5%.

7 Conclusion

Geographical cluster based routing is an interesting research track to make the conceptual idea of localized routing robust to the stochastic vagaries of real wireless networks. In this work we were interested in the effects of location errors which occur when information about node positions is determined by the wireless network infrastructure itself. In conclusion: The percentage of nodes assigned to wrong clusters can be high under large localization errors. However, in our study we observed that even for very large localization errors, the number of network partitions and inconsistent clusters is low. Thus, though message delivery is not guaranteed, geographical cluster based routing is feasible under inaccurate node localization. Here, among the three variants observed, geographical clustering with probabilistic MDS turned out to be the best variant.

In this work we investigated a centralized localization scheme. Though it seems curious at the first look to apply a centralized scheme first which already learns all nodes and then resort to a localized routing method later, this actually makes sense. The wireless links may change over time while the node positions are not (in a static wireless sensor network). Once nodes are assigned to geographic clusters, localized routing further requires only local maintenance to adapt to network dynamics such as changing links, and node failures.

We derived a model for location errors based on a well established path loss model and we derived the model parameters based on previous work which cited testbed results. This gives us high confidence that our conclusions actually hold for real sensor network deployments too. However, to get to the final conclusion that geographical cluster based routing in conjunction with certain localization scheme makes sense, it will be of high interest to complement our results with testbed experiments.

In this work we defined graph metrics to asses the quality of the geographical cluster based overlay graph with respect to localized routing. We selected this approach to have results independent of a particular routing algorithm. In fact, the results of the network partitions and critical inconsistent views depicted in our plots are worst case statements. In case of more than one overlay graph partition, two randomly chosen communication partners may happen to be in the same partition and routing is still successful. And, though there are clusters with critical views in the overlay graph, for two randomly chosen communication end points the path followed by a concrete face traversal algorithm may happen to visit only clusters with no or with benign inconsistent views. Even when a cluster with a critical inconsistent view is visited this does not necessarily mean that routing fails; there is only a chance that it may fail. An interesting future research direction is to complement our findings with measurements where actual localized geographic routing variants are applied over the constructed overlay graphs. Compared to the already promising low fraction of network partitions and critical inconsistent views observed in this work, we expect even better outcomes with respect to the success rate of those routing protocols.

Finally, it is important to note that if we allow several non connected node partitions in one cluster (as it may happen under localization errors as described before) the geographical cluster based routing approach as described, does not assure that a node in a certain cluster can be reached. In the case that there are more than one node partitions in one cluster, a message may eventually reach its destination cluster but happen to arrive in any of the partitions. If we are just interested in reaching a particular cluster, this is not an issue. If we are interested in reaching a particular node and when the node is not in the partition where the message arrived, a recovery has to take place to eventually reach the right partition in that cluster. The latter problem opens an interesting track of future research.

References

1 K. Benkic, M. Malajner, P. Planinsic, and Z. Cucej. Using RSSI value for distance estimation in wireless sensor networks based on ZigBee. In *Proceedings of the 15th International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 303–306, 2008.

2 Ingwer Borg and Patrick J.F. Groenen. *Modern Multidimensional Scaling – Theory and Applications*. Springer Series in Statistics, 2005.

3 Hannes Frey. Geographical cluster based routing with guaranteed delivery. In *2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2005)*, Washington, DC, USA, November 7–10 2005.

4 Hannes Frey and Daniel Görgen. Geographical cluster based routing in sensing-covered networks. In *Proceedings of the 2nd International Workshop on Wireless Ad Hoc Networking (WWAN 2005)*, pages 885–891, Columbus, Ohio, USA, June 6–9 2005. IEEE Computer Society.

5 Hannes Frey and Daniel Görgen. Planar graph routing on geographical clusters. *Ad Hoc Networks, Special issue on Data Communication and Topology Control in Ad Hoc Networks*, 3(5):560–574, September 2005.

6 Hannes Frey and Daniel Görgen. Geographical cluster based routing in sensing-covered networks. *IEEE Transactions on Parallel and Distributed Systems: Special issue on Localized Communication and Topology Protocols for ad hoc Networks*, 17(4), September 2006.

7 Hannes Frey and Ivan Stojmenović. On delivery guarantees of face and combined greedy face routing in ad hoc and sensor networks. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (Mobicom)*, Los Angeles, USA, September 2006.

8 David B. MacKay and Joseph L. Zinnes. A probabilistic model for the multidimensional scaling of proximity and preference data. *Marketing Science, Special Issue on Consumer Choice*, 5(4), 1986.

9 Sumesh J. Philip, Joy Ghosh, Hung Q. Ngo, and Chunming Qiao. Routing on overlay graphs in mobile ad hoc networks. In *Proceedings of the IEEE Global Communications Conference, Exhibition & Industry Forum (GLOBECOM'06)*, 2006.

10 Theodore S. Rappaport. *Wireless Communications: Principles & Practice*. Prentice Hall, 2002.

11 Yi Shang and Wheeler Ruml. Improved MDS-based localization. In *Proceedings of the 23th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, number 4, pages 2640–2651, 2004.

12 Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P. J. Fromherz. Localization from connectivity in sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(11):961–974, nov 2004.

13 Hector Tejeda, Edgar Chavez, Juan A. Sanchez, and Pedro M. Ruiz. Energy-efficient face routing on the virtual spanner. In *Proceedings of the 5th International Conference on AD-HOC Networks & Wireless (ADHOC-NOW'06)*, pages 101–113, 2006.

14 Hector Tejeda, Edgar Chavez, Juan A. Sanchez, and Pedro M. Ruiz. A virtual spanner for efficient face routing in multihop wireless networks. In *Proceedings of the IFIP TC6 11th International Conference on Personal Wireless Communications*, pages 459–470, 2006.

15 W.S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrica*, 17:401–419, 1952.

16 Guoliang Xing, Chenyang Lu, Robert Pless, and Qingfeng Huang. On greedy geographic routing algorithms in sensing-covered networks. In *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc '04)*, pages 31–42, Roppongi Hills, Tokyo, Japan, May 24–26 2004.

An Arbitrary 2D Structured Replica Control Protocol

Robert Basmadjian and Hermann de Meer

University of Passau, Department of Computer Network and Communication
Innstrasse 43, 94032, Passau, Germany
{basmadji,demeer}@fim.uni-passau.de

Abstract

Traditional replication protocols that logically arrange the replicas into a specific structure have reasonable availability, lower communication cost as well as system load than those that do not require any logical organisation of replicas.

We propose in this paper the A2DS protocol: a single protocol that, unlike the existing proposed protocols, can be adapted to any 2D structure. Its read operation is carried out on any replica of every level of the structure whereas write operations are performed on all replicas of a single level of the structure. We present several basic 2D structures and introduce the new idea of obtaining other 2D structures by the composition of several basic ones.

Two structures are proposed that have near optimal performance in terms of the communication cost, availability and system load of their read and write operations. Also, we introduce a new protocol that provides better performance for its write operations than those of *ROWA* protocol while preserving similar read performance.

1998 ACM Subject Classification Fault tolerance, Distributed systems.

Keywords and phrases Replication, Performance attributes, Reliability, Availability, Load.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.157

1 Introduction

In large distributed systems, data is replicated to provide a high level of *fault-tolerance* and to improve the *system performance*. However when replication is used, data becomes susceptible to inconsistency problems. Therefore, a replica control protocol (RCP) is required to "synchronize"concurrent read (query) and write (update) operations on replicated data. To ensure *one-copy equivalence*[1], a read and a write operation to two different replicas of the same data should not be allowed to execute concurrently. *Quorum systems*[2] are used by these protocols which serve as a basic tool of achieving one-copy equivalence.

Given the importance of the topic, several replication protocols have been described in the literature [1 − 11]. They differ according to various parameters such as the number of replicas involved in a given operation (henceforth referred to as their *communication cost* and ranges between 1 and total number of replicas n), their ability to tolerate replica failures (also termed as their *availability* and ranges between 0 and 1), as well as the *load* (ranges between 0 and 1) they impose on the system. Also, these protocols can be classified into two categories: those that arrange logically replicas of the system into a particular structure

[1] The fact of existing several replicas of a data should be abstracted as if there exists only a single replica.
[2] A quorum system is defined as a set of subsets of replicas called quorums having pair-wise non-empty intersections.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 157–168

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

[3 – 11] henceforth called structured RCPs, and those that do not impose any structure on the replicas [1, 2] which we call non-structured RCPs. The difference between the two categories is that the former has lower communication cost and system load than the latter, whereas the latter has better availability than the former for read and write operations.

The motivation of this paper is to ask whether it is possible to provide a single protocol that can be applied to any 2D structure and to afford optimal performance in terms of the communication cost, availability and system load of its read and write operations.

We answer with the affirmative by proposing the A2DS protocol: a protocol which assumes that replicas are logically arranged into any 2D structure based on its width w and height h. Read operations are carried out on any single replica at every level of the structure whereas write operations are performed on all replicas of any single level of the structure. We provide several basic 2D structures and give each one's performance in terms of the communication cost, availability and system load of its read and write operations. We also introduce the new idea of obtaining other 2D structures by the composition of several basic ones. For *mostly-read* systems, we propose a structure that has the same performance for its read and write operations as *ROWA* protocol [1]. When read and write operations happen in *equiprobable frequencies*, we introduce two structures that provide near optimal performance for both operations. Among the structured RCPs, the former has the best combined read and write costs of $2\sqrt{n}$ and its operations induce the best combined loads of $\frac{2}{\sqrt{n}}$, whereas the latter has a cost of w for its write operations which is lower than those of the existing structured RCPs and its operations induce near best combined loads of $\mathcal{O}(\frac{2}{\sqrt{n}})$; yet preserving comparable availability for its read and write operations. Finally, we propose a new protocol that we call ***R**ead**T**wo **W**rite**M**ajority* which provides better performance for its write operations than those of the well known *ROWA* protocol [1] while still maintaining similar performance for its read operations. The rest of the paper is organized as follows: Section 2 provides a brief overview of the related work. Our protocol is introduced in section 3. We give a general comparison among the structured RCPs in section 4. Section 5 shows the conclusion.

2 Related Work

For a system of n replicas, the well known *ReadOneWriteAll* (*ROWA*) protocol [1] has a read cost of 1 and a write cost of n. Its read operations are highly fault-tolerant (an availability of 1) and induce on the system a load of $\frac{1}{n}$. On the other hand, write operations have a very low availability (reaches to 0) as an update cannot be performed in the presence of a single replica failure or network partitions; they impose to the system the highest load of 1. The *Majority Quorum Consensus* (*MQC*) protocol [2] has read and write communication costs of $\frac{n+1}{2}$ for an odd-sized number of replicas n and imposes to the system a load of at least 0.5. It tolerates replica failures for read and write operations at the expense of increased read costs with respect to those of *ROWA*. However, both *ROWA* and *MQC* protocols have a high communication cost as well as system load, and are classified as non-structured RCPs.

By arranging replicas of the system into a logical structure, it is possible to reduce the communication cost as well as system load further. Several protocols have been proposed in the literature which make use of quorum systems and assume that the replicas are organized logically into a specific structure: *finite projective plane* [3] , a *grid structure* [4] and [5], or a *tree structure* [6], [7], [8], [9], and [10]. The load of these protocols was studied in [5] and it was proven that for a system of n replicas, the least induced load is $\frac{1}{\sqrt{n}}$ for any operation.

For a system of $n = t^2 + t + 1$ replicas ($t > 0$), the Finite Projective Plane (FPP) protocol

[3] has read and write costs of $\frac{1+\sqrt{4n-3}}{2}$ and induces on the system a load of $\frac{1+\sqrt{4n-3}}{2n}$. The major drawback of this protocol is that when the number of replicas $n > 100$, the availability of its read and write operations deteriorates gradually as it was shown in [12].

The Grid Quorum (GQ) protocol [4], for a system of n replicas, has a read cost of $\sqrt{n}$ and a write cost of $2\sqrt{n}-1$. Its read and write operations are fault-tolerant and induce on the system a load of $\frac{1}{\sqrt{n}}$ and $\frac{2\sqrt{n}-1}{n}$ respectively. Note that all these results are based on the fact that replicas are arranged logically into a square grid. For a system of $n = 2d^2 + 2d + 1$ replicas ($d > 0$) arranged logically into a percolation grid, the read and write operations of the Paths Quorum System (PQS) protocol [5] have a minimum communication cost of $\sqrt{2n-1}$, are highly available, and induce on the system a load of $\frac{\sqrt{2n-1}}{2n}$.

In general, the tree-structured RCPs have a tight trade-off between the communication cost and the load induced by their operations: a low cost results in inducing a high load and vice versa. In [11], the *Arbitrary Tree* protocol was introduced and it was shown that its write operations only induce on the system a load of $\frac{1}{\sqrt{n}}$ with a cost of $\sqrt{n}$, which is lower than state-of-the-art tree-structured RCPs, while preserving comparable write availability. On the other hand, its read operations only induce a cost of $\sqrt{n}$ which is lower than previously proposed tree-structured RCPs with comparable load and availability. In this paper, we adopt load related definitions, notations and propositions of section 2.1 of [11] as well as the system model of section 2.2 of [11].

3 Our Protocol

Given a *replication-based* system of n replicas, we organize them logically into any 2-dimensional structure of height $h > 0$. More precisely, let $R(i,k)$ denote the i^{th} replica of the k^{th} level of the 2D structure where the orientation is taken from left to right and top to bottom respectively such that $i \in [1, m_k]$ and $k \in [0, h]$, where m_k denotes the total number of replicas at level k.

3.1 The operations

We construct the set of read quorums $\boldsymbol{R}$ and write quorums $\boldsymbol{W}$ by respecting the definition 2.3 of [11] on *Bi-coteries*. Furthermore, we assign separate strategies of picking read and write quorums using the definition 2.4 of [11] on *strategies*. More precisely, let w_{read} and w_{write} denote a strategy for picking read quorums of $\boldsymbol{R}$ and write quorums of $\boldsymbol{W}$ respectively where w_{read} and w_{write} are defined in subsequent sections. The availability computations are carried out by taking the assumption that every replica is *independently available* with a probability $p = 1 - q > \frac{1}{2}$, where $q \in [0, \frac{1}{2}[$ denotes the failure probability. In the rest of this section, we use $h > 0$ to denote the height of the structure, whereas we use d and e to denote respectively the *minimal* and *maximal* number of replicas among the levels of the 2D structure such that $d = \min\{m_k \mid \forall k : 0 \leq k \leq h\}$ and $e = \max\{m_k \mid \forall k : 0 \leq k \leq h\}$.

3.1.1 Reads

A read operation takes place by accessing all the members (replicas) of a read quorum $R_j \in \boldsymbol{R}$ and retrieving the data of the replica whose timestamp has the highest *version number*. In case several members of such a quorum R_j have the same highest version number, then the data of the replica (among replicas with highest version number) whose timestamp has the smallest *identifier* (*RID*) is fetched. A read quorum R_j is constructed by having as

its members any single replica of every level of the 2D structure:

$$R_j = \{R(i,k) \mid \forall k : 0 \leq k \leq h \ \wedge \ \exists i : 1 \leq i \leq m_k\}$$

▶ FACT 3.1.1. Let $\boldsymbol{R}$= $\{R_1, R_2, \ldots, R_j\}$ be the set of read quorums such that every read quorum R_j is constructed as explained above. Then the number of read quorums (size) of $\boldsymbol{R}$ is denoted by $m(\boldsymbol{R}) = \prod_{k=0}^{h} m_k$.

In order to compute the load of the system induced by this read operation, a strategy $w_{read} = \sum_{j=1}^{m(\boldsymbol{R})} w_{read,j} = 1$ is taken that picks each read quorum $R_j \in \boldsymbol{R}$ with a probability of $w_{read,j} = \frac{1}{m(\boldsymbol{R})}$ where $j \in \{1, \ldots, m(\boldsymbol{R})\}$. The read operation of our protocol has:

$$\text{A communication cost of } RD_{cost} = 1 + h$$

$$\text{An availability of } RD_{av}\ (p) = \prod_{k=0}^{h}(1-(1-p)^{m_k}) \tag{3.1}$$

$$\text{An optimal system load of } \mathcal{L}_{RD} = \frac{1}{d} \tag{3.2}$$

The proof of the *optimality* of the system load can be found in the *Appendix* of [11].

3.1.2 Writes

A write operation, after retrieving (from the replicas of a read quorum) the highest version number of the data to be modified and then incrementing it by one, accesses all the members of a write quorum $W_j \in \boldsymbol{W}$ in order to update their data with a new value and timestamp (the new *version number* along with the write quorum's first replica's *identifier*).

A write quorum W_j is constructed by taking as its members all replicas of any single level of the 2D structure:

$$W_j = \{R(i,k) \mid \exists k : 0 \leq k \leq h \ \wedge \ \forall i : 1 \leq i \leq m_k\}$$

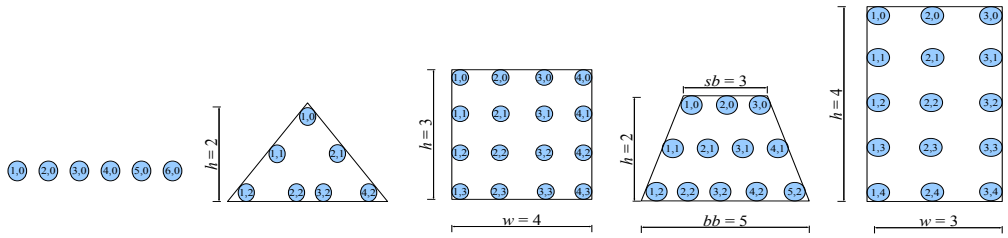
▶ FACT 3.1.2. Let $\boldsymbol{W} = \{W_1, W_2, \ldots, W_j\}$ be the set of write quorums such that every write quorum W_j is constructed as explained above. Then the number of write quorums (size) of $\boldsymbol{W}$ is denoted by $m(\boldsymbol{W}) = 1 + h$.

In order to compute the load of the system induced by this write operation, a strategy $w_{write} = \sum_{j=1}^{m(\boldsymbol{W})} w_{write,j} = 1$ is taken that picks each write quorum $W_j \in \boldsymbol{W}$ with a probability of $w_{write,j} = \frac{1}{m(\boldsymbol{W})}$ where $j \in \{1, \ldots, m(\boldsymbol{W})\}$. The write operation of our protocol has a minimum communication cost of d, a maximum cost of e and an *average* cost of $WR_{cost} = \sum_{k=0}^{h} m_k \times w_{write,k}$. Hence such a strategy w_{write} of picking write quorums induces a communication cost of $\frac{n}{1+h}$. This operation has an availability of:

$$WR_{av}\ (p) = 1 - WR_{fail}\ (p) \tag{3.3}$$

where $WR_{fail}\ (p) = \prod_{k=0}^{h}(1-p^{m_k})$ and it imposes an optimal system load of:

$$\mathcal{L}_{WR} = \frac{1}{1+h} \tag{3.4}$$



(a) Line **(b)** Triangle **(c)** Square **(d)** Trapezoid **(e)** Rectangle

Figure 1 An example of our basic structures for $n = 6, 7, 16, 12$ and 15 replicas respectively.

The proof of the *optimality* of the system load can be found in the *Appendix* of [11].

3.1.3 The expected system load computations

The loads imposed by read and write operations of (3.2) and (3.4) respectively are computed by assuming that all replicas of the system are fail-free. The load of the system induced by these operations becomes higher as replicas of the system start to fail one after another. In order to compute the *expected load* assuming that replicas are available with a probability $p > \frac{1}{2}$, we use the following two equations expressed in terms of (3.1), (3.2), (3.3), and (3.4):

$$\mathbb{E}\mathcal{L}_{RD} = RD_{av}(p) \times (\mathcal{L}_{RD} - 1) + 1 \quad (3.5)$$

$$\mathbb{E}\mathcal{L}_{WR} = WR_{av}(p) \times \mathcal{L}_{WR} + WR_{fail}(p) \times 1 \quad (3.6)$$

We can notice from (3.5) and (3.6) that the expected load computations largely depend on the availability of the operations: as the availability of the operations is high, the *expected load* becomes close to the *fail-free load* and thus the system is classified as *stable*.

3.1.4 The intersection of read and write quorums

In this section, we demonstrate that our system is a *bi-coterie* i.e. any read quorum $R_j \in \boldsymbol{R}$ has a *non empty intersection* with any write quorum $W_j \in \boldsymbol{W}$. The proof is by induction on the height h of the structure:
Basis Step: Trivial for a structure of height $h = 0$, because all replicas are placed at one and only one level.
Induction Hypothesis: Assume that it holds true for a structure of height $h > 0$.
Induction Step: Consider a 2D structure of height $h' = h+1$. Since any read quorum $R_j \in \boldsymbol{R}$ intersects with any write quorum $W_k \in \boldsymbol{W}$ such that $0 \leq k \leq h$ (*induction hypothesis* step), then it holds true because the fact of adding one new level $i = h + 1$ does not prevent any read quorum $R_j \in \boldsymbol{R}$ to have a non-empty intersection with any write quorum $W_k \in \boldsymbol{W}$ such that $0 \leq k \leq h$. On the other hand, since any read quorum $R_j \in \boldsymbol{R}$ contains a replica from the *new level* i and the *new write quorum* W_i contains all replicas of the level i, then any read quorum $R_j \in \boldsymbol{R}$ has a non-empty intersection with this write quorum W_i. Hence by induction, our protocol guarantees *non-empty intersection* of read and write quorums.

3.2 Basic structures

In this section, we introduce various basic 2D structures and give each one's characteristics in terms of the communication cost, availability and load of its read and write operations.

3.2.1 Straight line

This is a special case of our 2D structures ($h = 0$) where n replicas of the system are arranged logically into a *straight line* (see Figure 1(a)). Its read operation has a cost of 1, an availability of $1-(1-p)^n$ and induces on the system a load of $\frac{1}{n}$. The write operation of this structure has a cost of n, an availability of p^n and imposes to the system a load of 1. Note that such a structure has the same characteristics of *ROWA* [1] and therefore is most appropriate for *mostly-read* systems because it favors read operations over write ones.

3.2.2 Triangle

The illustrated structure of Figure 1(b) is constructed by arranging logically $n = 2^{h+1} - 1$ replicas into a *triangle* of height $h > 0$ such that $m_k = 2^k$ at every level $k \in [0, h]$. Its read operation has a cost of $\log(n+1)$, an availability of $\prod_{k=0}^{h}(1-(1-p)^{2^k})$ and imposes on the system a load of 1. The write operation has a cost of $\frac{n}{\log(n+1)}$, an availability of $1-\prod_{k=0}^{h}(1-p^{2^k})$ and induces on the system a load of $\frac{1}{\log(n+1)}$. The major drawback of this structure is that its read operations always induce the highest load of 1 and these operations are poorly available i.e. if the replica at level 0 fails, then no read operations can take place.

3.2.3 Square

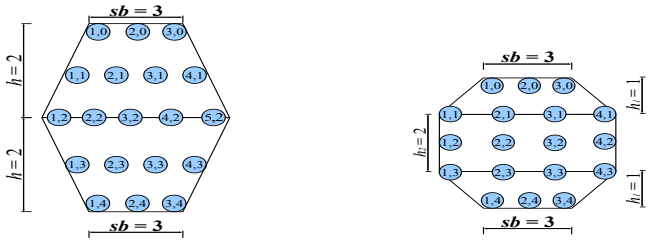
The replicas of this structure (see Figure 1(c)) are arranged logically into a *square* of height $h > 0$ and width $w = h + 1$ such that the number of replicas $n = w \times (h + 1)$. Its operations have a cost of $\sqrt{n}$, and induce on the system a load of $\frac{1}{\sqrt{n}}$. The read and write operations have an availability of $(1-(1-p)^{\sqrt{n}})^{\sqrt{n}}$ and $1-(1-p^{\sqrt{n}})^{\sqrt{n}}$respectively. The major drawback of this structure is that when $n > 25$, its write operations become poorly available. Thus, such a structure is most appropriate for systems having highly available replicas: $p \geq 0.90$.

3.2.4 Trapezoid

The structure of Figure 1(d) is obtained by arranging logically $n = sb \times (1 + h + \sum_{i=1}^{h} \frac{i}{sb})$ replicas into a trapezoid of height $h > 0$, small base $sb > 1$ and big base $bb = sb + h$ such that $m_k = sb + k$ at every level $k \in [0, h]$. Its read operation has a cost of $\frac{-2sb+1+\sqrt{4sb^2-4sb+1+8n}}{2}$, an availability of $\prod_{k=0}^{h}(1-(1-p)^{(sb+k)})$, and imposes to the system a load of $\frac{1}{sb}$. The write operation of such a structure has a cost of $\frac{2sb-1+\sqrt{4sb^2-4sb+1+8n}}{4}$, an availability of $1-\prod_{k=0}^{h}(1-p^{(sb+k)})$, and induces a system load of $\frac{2sb-1+\sqrt{4sb^2-4sb+1+8n}}{4n}$. In order to obtain satisfactory results for both read and write operations at the same time, we fix $sb = 2$.

3.2.5 Rectangle

The replicas of this structure (see Figure 1(e)) are organized logically into a *rectangle* of height $h > 0$ and width $w > 1$ such that the number of replicas $n = w \times (h + 1)$. The read operation of this structure has a cost of $\frac{n}{w}$, an availability of $(1-(1-p)^w)^{h+1}$, and induces a



(a) Hexagon **(b)** Octagon

Figure 2 An example of our composed structures for $n = 19$ and 18 replicas respectively.

system load of $\frac{1}{w}$. Its write operation has a cost of w, an availability of $1-(1-p^w)^{h+1}$, and imposes to the system a load of $\frac{w}{n}$. To obtain satisfactory results for both operations, we set $h > w$ such that $w = 3$ for $15 \leq n \leq 21$, $w = 4$ for $24 \leq n \leq 48$, $w = 5$ for $50 \leq n \leq 85$, $w = 6$ for $96 \leq n \leq 152$, $w = 7$ for $154 \leq n < 252$ and $w = 8$ for $256 \leq n < 408$.

3.3 Composed structures

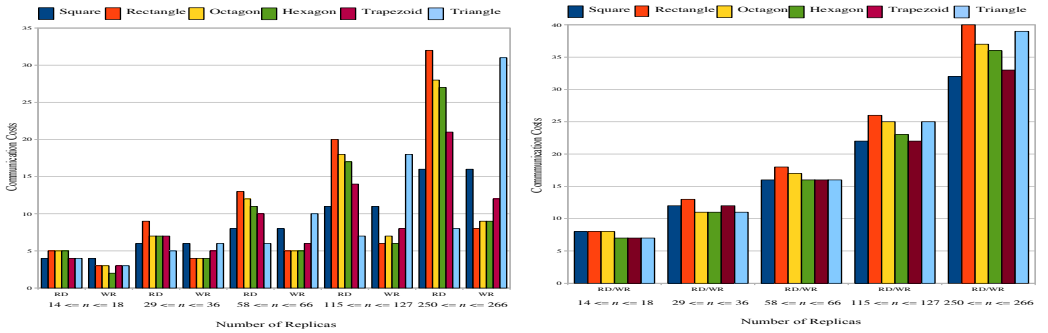
In this section, we introduce two composed structures and give the communication cost, availability and system load of their read and write operations.

3.3.1 Hexagon

The replicas of this structure (see Figure 2(a)) are arranged logically into a *hexagon* composed of two symmetric trapezoids, each of height $h > 0$ and small base $sb > 1$, joined at their corresponding big bases $bb = sb + h$ such that $n = h \times (2sb-1) + sb + 2 \times \sum_{i=1}^{h} i$. The read operation of this structure has a cost of $1-2sb+2\sqrt{sb^2-sb+n}$, an availability of $\left(\prod_{k=0}^{h-1}(1-(1-p)^{(sb+k)})\right)^2 \times (1-(1-p)^{bb})$, and induces on the system a load of $\frac{1}{sb}$. Write operations have a cost of $\frac{n\times(2\sqrt{sb^2-sb+n}+2sb-1)}{4n-1}$, an availability of $1-\left(\prod_{k=0}^{h-1}(1-p^{(sb+k)})\right)^2 \times (1-p^{bb})$, and imposes a system load of $\frac{2\sqrt{sb^2-sb+n}+2sb-1}{4n-1}$. In order to obtain satisfactory results for both operations, we set the small base $sb = 3$ for both trapezoids whenever $n \geq 30$.

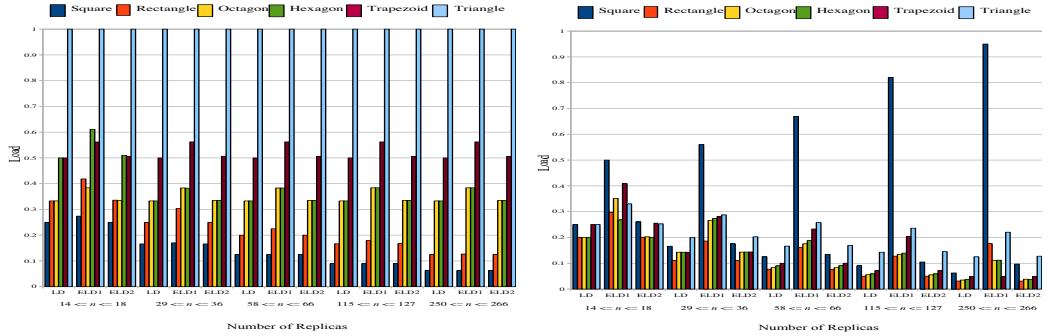
3.3.2 Octagon

This structure is obtained by organizing logically $n = sb\times(1+2h_1+h_2)+h_1\times(h_2-1)+2\times\sum_{i=1}^{h_1} i$ replicas into an *octagon* composed of two symmetric trapezoids, each of height $h_1 > 0$, small base $sb > 1$ and big base $bb = sb+h_1$, which are joined to each other by means of a rectangle of height $h_2 > 0$ and width $w = bb$ (see Figure 2(b)). The read operation of such a structure has a cost of $\frac{n-h_1\times(h_1+h_2)}{sb}$, an availability of $\left(\prod_{k=0}^{h_1-1}(1-(1-p)^{(sb+k)})\right)^2 \times \prod_{k=0}^{h_2}(1-(1-p)^{bb})$, and imposes a system load of $\frac{1}{sb}$. Its write operation has a cost of $\frac{n\times sb}{n-h_1\times(h_1+h_2)}$, an availability



(a) Separate costs **(b)** Combined costs

Figure 3 The communication costs of read (RD) and write (WR) operations.



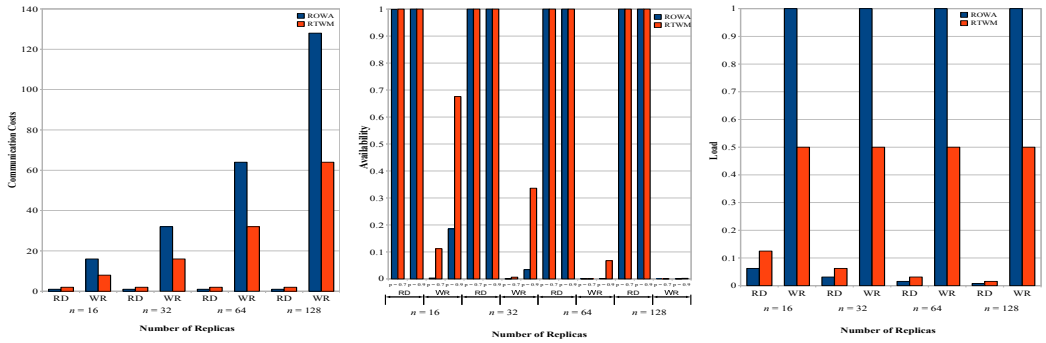
(a) Read operation **(b)** Write operation

Figure 4 The system (LD) and expected system (ELD) loads for $p = 0.7$ and $p = 0.9$.

of $1 - \left(\prod_{k=0}^{h_1-1} (1 - p^{(sb+k)}) \right)^2 \times \prod_{k=0}^{h_2} (1 - p^{bb})$, and induces on the system a load of $\frac{sb}{n - h_1 \times (h_1 + h_2)}$. To obtain satisfactory results for both operations, we fix $sb = 3$.

3.4 Comparison

When the number of replicas $n > 20$ (see Figure 3(a)), the *Rectangle* structure of width w has the highest cost of $\frac{n}{w}$ for read operations and the least cost of w for write operations. On the other hand, the *Triangle* structure has the fewest cost of $\log(n+1)$ for read operations and the worst cost of $\frac{n}{\log(n+1)}$ for write operations. Figure 3(b) demonstrates that the structures have quite comparable combined read and write costs. However when $n > 200$, the difference in combined costs of *Rectangle*, *Octagon*, *Hexagon* and *Triangle* structures becomes evident with respect to those of *Square* and *Trapezoid*. The expected system load computations of Figure 4 are carried out by using (3.5) and (3.6) such that we set $p = 0.7$ for *ELD1* and $p = 0.9$ for *ELD2*. We can observe in Figure 4(a) that the *Square* structure has the least system load of $\frac{1}{\sqrt{n}}$ for read operations and such a load diminishes as the number of replicas n increases. Also, the expected system loads of this structure are close to the fail-free system loads due to the high availability of its read operations. The *Triangle* structure has the highest fail-free and expected system loads of 1 due to the fact that the single replica at level 0 participates in every read operation. The *Rectangle* structure (see Figure 4(b)) has



(a) Communication cost **(b)** Availability **(c)** Load

Figure 5 The *RTWM* vs *ROWA* protocols.

the least system load of $\frac{w}{n}$ for write operations. Also, this structure has the least expected system loads either when $p = 0.7$ or $p = 0.9$ except for certain cases. The *Triangle* structure has the highest system load of $\frac{1}{\log(n+1)}$, as well as the highest expected system load when $p = 0.9$ for any number of replicas n. Note that, the *Square* structure has the worst expected system loads when $p = 0.7$ due to the low availability of its write operations.

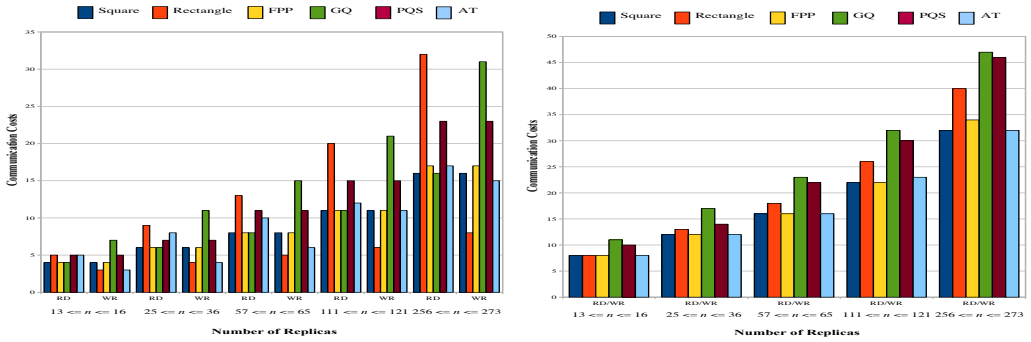
3.5 The RTWM protocol

In order to circumvent the drawbacks (see section 2) of write operations of the *ROWA* [1] protocol and yet to preserve the advantages of its read operations, we propose a protocol which we call *ReadTwoWriteMajority*. More precisely, for an even-sized number of replicas n, we arrange the replicas logically into the *Rectangle* structure of section 3.2.5 such that $w = \frac{n}{2}$ and $h = 1$. For an odd-sized n, we organize the replicas logically into the *Trapezoid* structure of section 3.2.4 such that $sb = \frac{n-1}{2}$, $h = 1$ and $bb = \frac{n+1}{2}$. The read operation of the *RTWM* protocol has a cost of 2, an availability of $\left(1 - (1-p)^{\frac{n}{2}}\right)^2$ if n is even, otherwise $\prod_{k=0}^{1}(1-(1-p)^{(sb+k)})$, and induces a system load of $\frac{2}{n}$ if n is even, otherwise $\frac{2}{n-1}$. On the other hand, its write operation has a cost of $\frac{n}{2}$, an availability of $1 - \left(1 - p^{\frac{n}{2}}\right)^2$ if n is even, otherwise $1 - \prod_{k=0}^{1}(1-p^{(sb+k)})$, and imposes a system load of $\frac{1}{2}$.

The *RTWM* protocol (see Figure 5) has much fewer communication cost (half), better availability and smaller system load (half) for its write operations than those of the *ROWA* [1] protocol, still preserving the advantages of read operations of this latter where our protocol has one communication cost higher than that of *ROWA*, and has comparable availability and system load especially for large number of replicas n.

4 General Comparison

In this section, we provide a general comparison among the *most relevant* existing structured as well as our 2D structured RCPs in terms of the communication cost, availability and system load of their operations. The "Square" configuration is set up based on the structure of section 3.2.3 such that $h = 3, 5, 7, 10$, and 15. The "Rectangle" configuration is studied using



(a) Separate costs **(b)** Combined costs

Figure 6 The communication costs of read (RD) and write (WR) operations.

the structure of section 3.2.5 such that $h = 4, 8, 12, 19$, and 31. The "FPP" configuration is examined by taking the protocol of [3] such that the number of replicas $n = t^2 + t + 1$ where $t = 3, 5, 7, 10$ and 16. The "GQ" configuration is considered by studying the protocol of [4] such that $n = 16, 36, 64, 121$, and 256 replicas. The "PQS" configuration is set up based on the protocol of [5] such that $n = 2d^2 + 2d + 1$ where $d = 2, 3, 5, 7$, and 11. Finally, the "AT" configuration is studied by considering the Algorithm 1 of [11] such that $n = 15, 31, 63, 127$, and 255 replicas.

4.1 Communication cost

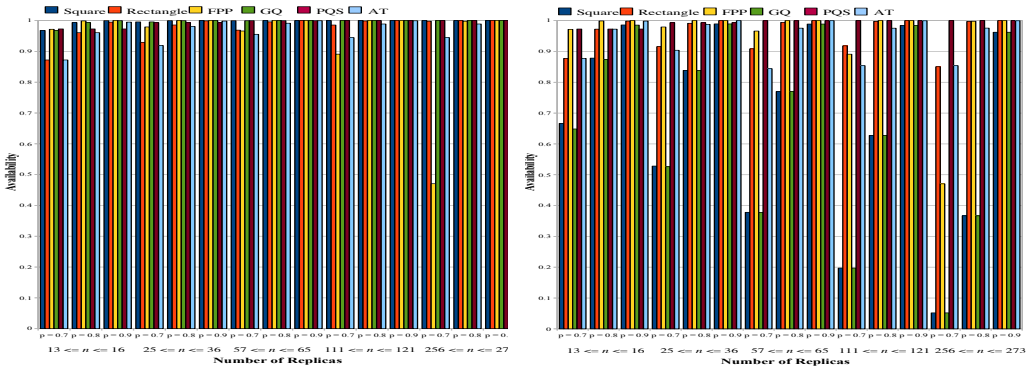
The "Rectangle" of width w has the highest cost of $\frac{n}{w}$ for read operations and the least cost of w for write ones (see Figure 6(a)). The configurations "Square", "FPP", and "GQ" have the fewest cost of $\sqrt{n}$ for read operations whereas "GQ" has the worst cost of $2\sqrt{n} - 1$ for write operations. We can observe in Figure 6(b) that the configurations "Square", "Rectangle", "FPP" and "AT" have quite comparable combined read and write costs. Also, "GQ" and "PQS" have the highest combined costs and that the difference in their combined costs becomes evident with respect to those of the other configurations when $n > 100$.

4.2 Availability

All the configurations have similar availability for read operations when $p = 0.8$ and 0.9 (see Figure 7(a)). When $p = 0.7$, the configurations "Square", "GQ" and "PQS" have quite better read availability than "FPP" and "AT". Note that the read availability of "Rectangle" ameliorates gradually as the number of replicas n increases. Figure 7(b) illustrates that all the configurations have similar write availability when $p = 0.8$ and 0.9 except for "Square" and "GQ". Both of these configurations have the worst write availability which degrades gradually with increasing n. "PQS" has the best write availability when $p = 0.7$ especially for large number of replicas n. Note that the read and write availability of "FPP" degrade gradually when $n > 100$ and $p = 0.7$ due to the non-Condorcet property as shown in [12].

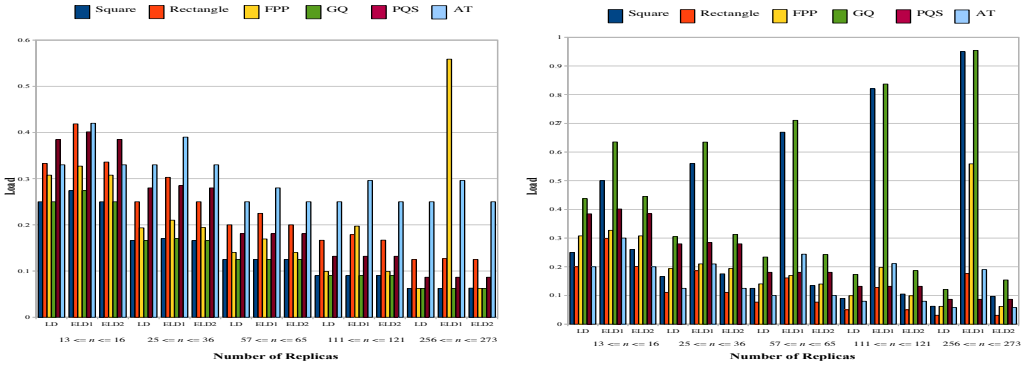
4.3 (Expected) system loads

The expected system load computations of read and write operations are carried out by using (3.5) and (3.6) respectively such that $p = 0.7$ for *ELD1* and $p = 0.9$ for *ELD2*. The "Square" and "GQ" have the least system load of $\frac{1}{\sqrt{n}}$ for read operations and such a load



(a) Read operation **(b)** Write operation

Figure 7 The availability of the operations for $p = 0.7, 0.8$ and 0.9.



(a) Read operation **(b)** Write operation

Figure 8 The system (LD) and expected system (ELD) loads for $p = 0.7$ and $p = 0.9$.

diminishes as the number of replicas n increases (see Figure 8(a)). Also, the expected system loads of both of these configurations are close to the fail-free system loads due to the high availability of their read operations. "AT" has the highest read system load of 0.25 when $n > 24$. Also, this configuration has the highest expected system loads for $24 < n < 200$. Note that when $n > 200$ and $p = 0.7$, the "FPP" configuration induces the highest expected system loads due to the fact that its read availability deteriorates gradually when $n > 100$.

Figure 8(b) illustrates that "Rectangle" of width w has the least system load of $\frac{w}{n}$ for write operations. Also, this configuration has the least expected system loads either when $p = 0.7$ or $p = 0.9$ except for certain cases. The "GQ" configuration has the highest write system load of $\frac{2\sqrt{n}-1}{n}$ which is quite similar to that of "PQS". Also, the former configuration has the highest expected system loads for any number of replicas n. It is worthwhile to note that the configuration "Square" has quite comparable expected system loads with respect to those of "GQ " when $p = 0.7$ due to the low availability of its write operations.

5 Conclusion

In this paper, we proposed a structured replica control protocol that, unlike the previous proposed ones, can be implemented using any logical 2D structure of height $h > 0$. We

presented two structures that provide near optimal performance for their read and write operations. The former has the best combined read and write costs of $2\sqrt{n}$ and induces the best combined read and write loads of $\frac{2}{\sqrt{n}}$, whereas the latter has a cost of w for its write operations which is lower than those of the existing structured RCPs and induces near best combined read and write loads of $\mathcal{O}(\frac{2}{\sqrt{n}})$; yet preserving comparable availability for its read and write operations. We also proposed a new protocol which provides better performance for its write operations than those of the well known *ROWA* [1] protocol while still maintaining comparable performance for its read operations.

6 Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme in the context of the FIT4Green project (grant agreement no. 249020) and the EuroNF Network of Excellence (grant agreement no. 216366).

References

1 Bernstein, P.A., Goodman, N.: An algorithm for concurrency control and recovery in replicated distributed databases. ACM Transactions on Database Systems, 9(4), 596-615 (1984)
2 Thomas, R.H.: A majority consensus approach to concurrency control for multiple copy databases. ACM Transactions on Database Systems, 4(2), 180-209 (1979)
3 Maekawa, M.: A $\sqrt{n}$ algorithm for mutual exclusion in decentralized systems. ACM Transactions on Computer Systems, 3(2), 145-159 (1985)
4 Cheung, S.Y., Ammar, M.H., Ahamad, A.: The grid protocol: a high performance scheme for maintaining replicated data. IEEE Transactions on Knowledge and Data Engineering, 4(6), 438-445 (1990)
5 Naor, M., Wool, A.: The load, capacity, and availability of quorum systems. SIAM Journal on Computing, 27, 214-225 (1998)
6 Agrawal, D., El Abbadi, A.: The tree quorum protocol: an efficient approach for managing replicated data. Proceedings of the sixteenth international conference on Very Large Databases, 243-254 (1990)
7 Agrawal, D., El Abbadi, A.: An efficient and fault-tolerant solution for distributed mutual exclusion. ACM Transactions on Computer Systems, 9(1), 1-20 (1991)
8 Choi, S.C., Youn, H.Y., Choi, J.S.: Symmetric tree replication protocol for efficient distributed storage system. Proceedings of the International Conference on Computational Science, 474-484 (2003)
9 Koch, H.: An efficient replication protocol exploiting logical tree structures. The 23rd annual International Symposium on Fault-Tolerant Computing, 382-391 (1993)
10 Kumar, A.: Hierarchical quorum consensus: a new algorithm for managing replicated data. IEEE Transactions on Computers, 40(9), 996-1004 (1991)
11 Bahsoun, J.P., Basmadjian, R., Guerraoui, R.: An arbitrary tree-structured replica control protocol. The 28th International Conference on Distributed Computing Systems, 502-511 (2008)
12 Peleg, D., Wool, A.: The availability of quorum systems. Inform. and Comput, 210-223 (1995)
13 Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of ACM, 228-234 (1979)

Short Papers

Resolving Conflicts in Highly Reactive Teams

Hendrik Skubch, Daniel Saur, and Kurt Geihs

Distributed Systems
Universität Kassel
Wilhelmshöher Allee 73, Kassel, Germany
{skubch, saur, geihs}@vs.uni-kassel.de

Abstract

In distributed cooperation frameworks for completely autonomous agents, conflicts between the involved agents can occur due to inconsistent data available to the agents. In highly dynamic domains, such as RoboCup, it is often neccessary to accept a certain level of conflicts in order to decrease reaction time of single agents, instead of relying on error-free, but extensive communication. However, this may lead to situations in which unresolved conflicts linger and cause cooperation to break down completely. In our cooperation framework, ALICA, we designed and implemented a simple but effective approach to detect these cases and resolve them quickly through a bully algorithm.

Keywords and phrases Agents, Multi-agent, cooperation, time critical, conflict resolution

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.170

1 Introduction

Cooperation between autonomous agents acting in highly dynamic domains under time constraints is an important and difficult challenge. As it becomes feasible to tackle increasingly challenging scenarios using autonomous robots, such as search and rescue or exploration, completely distributed approaches to teamwork become more important. In this light, central points of failure need to be eliminated, and adaptability and robustness increased.

One prominent scenario in which such conditions are examined is the Middle Size League of RoboCup[1], where teams of five autonomous robots compete against each other. The RoboCup domain has several key features common to many real world problems: *Noisy data:* Data obtained from the sensors is very noisy due to the high speeds at which objects move and unforseen confusing objects in the background, e.g., in the audience. *Incomplete information:* The sensor range of a single robot covers only a fraction of the field's size, hence important objects such as the ball are sometimes not observed by any robot. *Wireless Communication:* 802.11 wireless is used for communication, which is an unreliable, potentially crowded medium, so packet loss and latency are relevant. *Dynamic Environment:* The game is very quick, as robots move with up to $6m/s$, i.e., $20cm$ per decision cycle (30 Hz). *Full Autonomy:* During a game, there is absolutely no human interaction with the playing robots, with the exception of the referee starting and stopping the game.

Hence, both highly reactive acting and coherent teamwork are needed. However, these two goals interfere with each other, as reactivity forbids communication before acting, and coherent teamwork requires communication, given noisy and incomplete sensor information.

Using RoboCup as a test scenario, we developed a modelling framework for cooperative agents, called ALICA [8], which uses a completely distributed and autonomous way of decision making. Although each robot informs its team about its actions and integrates received

[1] www.robocup.org

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 170–175

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

information into its decision making process, due to timing constraints a robot cannot afford to wait for messages from its team members before it decides and acts. While this approach guarantees swift reactions, it can cause inconsistent decisions. This is most pronounced if the sensory data are very noisy, or even diverge completely. In most cases, these inconsistencies are momentarily and are solved within a few decision cycles [8], however in some cases, the problem can persist and cause the teamwork to break down for a significant period of time. Requiring sensory data to be of sufficient quality to avoid this problem is unrealistic. Instead, we deem a more explicit handling of the conflicts in question necessary. In this paper, we present a simple method extending the teamwork approach of ALICA, which solves the problem of persisting conflicts.

In the next Section, we briefly discuss related work, focussing on approaches in similar domains. In Section 3, we explain the fundamental notions of teamwork within ALICA. Section 4 explains how persisting conflicts can be detected and handled in detail. Finally, Section 5 shows evaluation results on real robots and concludes our discussion.

2 Related Work

Plenty of teamwork approaches use a central processing unit to either guarantee consistent data, or to make the team-relevant decisions. Among them, several are employed in RoboCup [6, 1, 2]. In these approaches, perceived data is sent from the robots to a central processing unit which either returns commands or fused sensory data, thus guaranteeing consistency. These approaches are robust and easy to implement, however they lack the flexibility to scale up to larger teams or to teams operating in unreliable communication environments with real-time constraints. Furthermore, a central processing unit always presents a single point of failure.

Other teamwork approaches, such as STEAM [9], rely on explicit communication before acting. In particular an extension to STEAM, CONSA (COllaborative Negotiation System based on Argumentation) [5, 4], allows the involved agents to present arguments to each other when confronted with a conflict.

Although the requirement of explicit communication is partially alleviated by a decision theoretic estimation of the risk of not communicating, we deem this approach not suitable for very dynamic domains. In particular, in domains such as robotic soccer, the situation changes too fast for argumentation to take place. Moreover, it is unclear how local data can be used in an argument, when sensor fusion has already failed to avoid the conflict.

Some approaches, such as [3], follow a related line of thought, by basing argumentation on belief revision operators, focussing on how arguments are understood and what kind of agreements can be achieved. These approaches are not suitable for fast paced domains, where it is more important to act immediately than it is to make the ideal choice.

3 Teamwork in ALICA

The central notion within ALICA are *plans*. Intuitively, a plan is a recipe, describing how a certain goal can be achieved by a set of robots. More specifically, a plan contains a non-empty set of finite automata, each labelled with a *task*. If a team of robots is to execute a plan, the robots have to allocate themselves to these tasks and hence each executes one finite automaton. A task allocation is defined by a set of believes of the form $\mathrm{In}(a, p, \tau, z)$, denoting that agent a takes on task τ within plan p, and currently inhabits state z.

Figure 1 shows a simple plan, with the two tasks *Attack* and $Defend$. *Attack* contains

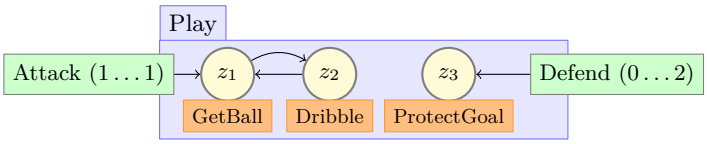


Figure 1 A simple ALICA plan

two states, one for obtaining the ball, one for dribbling towards the opponent's goal. $Defend$ contains only a single state, in which agents should protect the goal. Each task can be executed by multiple agents, restricted by a cardinality, denoted $\xi(p, \tau)$. Here, exactly one robot must execute the task *Attack*, while up to two can execute the task $Defend$.

The task allocation problem is subject to several restrictions, namely said cardinalities, potential preconditions the plan might have, formulated in first order logic, and role restrictions, which can forbid robots with certain roles to take on specific tasks. For instance, a robot designated as goalie should not take on the task *Attack*.

Although these restrictions potentially rule out plenty of allocations, there are usually a vast number of valid allocations for a given situation and plan. For each plan, a utility function maps the currently believed situation, including a potential allocation, onto the real numbers. Each robot chooses the task allocation which maximises this function. The utility of the plan can depend on arbitrary conditions reflected in a robot's belief base, ranging from the roles a robot is assigned to highly dynamic properties such as distances to other robots, the ball, or the goal. Hence, each participating robot calculates the allocation individually.

In ALICA, plans occur grouped in *plan types*, such that each plan within such a plan type represents an alternative solution to the given problem. In order to execute a plan type, a single plan together with a task allocation needs to be picked by the executing robots. Thus, task allocations for different plans are evaluated and compared, adding a further dimension to the task allocation problem.

Accommodating the dynamic environment, each robot constantly recalculates the best achievable allocation and adopts it if a small hysteresis criteria is met. This is called *reallocation*. In the example above, a suitable utility function could maximise if the robot closest to ball takes on the task *Attack*. If an opponent kicks the ball, the robots can switch tasks immediately. Calculating an allocation and acting upon it is done completely autonomously by each robot. Since the involved beliefs are noisy, conflicts arise. ALICA assumes that the robots communicate periodically their chosen tasks, thus most conflicts are dealt with automatically by reallocation. If however, their beliefs regarding the decision-relevant data do not tend to converge, these conflicts can persist. For example, our current ball detection algorithms tend to underestimate small distances and overestimate large distances. Thus, if two robots are very close to the ball, each will perceive it as being closer to itself. In this situation, a persistent conflict can occur.

4 Dealing with Conflicts

Our approach to deal with persisting conflicts is divided in two parts: Detection and reaction. In the following, we will present an indicator of such conflicts and afterwards discuss a simple solution based on bullying to resolve these conflicts. Firstly, we discuss how conflicts can be detected. The decisions which are subject to potential conflicts are called *task allocations*.

▶ **Definition 1** (Task Allocation). (Slightly simplified from [7]) A task allocation for a plan type pt is a conjunction of the form $\mathrm{In}(a_1, p, \tau_1, z_1) \wedge \mathrm{In}(a_2, p, \tau_2, z_2) \wedge \ldots \wedge \mathrm{In}(a_n, p, \tau_n, z_n)$,

such that plan p belongs to pt. A task allocation C for a plan p is valid wrt. belief base $\mathcal{F}$ iff

$$\mathcal{F} \cup C \models (\forall \tau \in \text{Tasks}(p))(\exists A)(\forall a)\,(a \in A \leftrightarrow (\exists z) In(a, p, \tau, z)) \wedge \tag{1}$$

$$(\exists n_1, n_2)\,\xi(p, \tau) = (n_1, n_2) \wedge n_1 \leq |A| \leq n_2 \tag{2}$$

$$(\forall \tau \in \text{Tasks}(p))(\exists n_1, n_2)\,\xi(p, \tau) = (n_1, n_2) \wedge n_1 \leq |A| \leq n_2 \tag{3}$$

$$\mathcal{F} \cup C \models \text{In}(a, p, \tau_1, z_1) \wedge \text{In}(a, p, \tau_2, z_2) \rightarrow \tau_1 = \tau_2 \wedge z_1 = z_2 \tag{4}$$

$$\mathcal{F} \cup C \models \text{Pre}(p) \tag{5}$$

$$\mathcal{U}_p(\mathcal{F} \cup C) \geq 0 \tag{6}$$

Where $\text{Tasks}(p)$ denotes the set of tasks of plan p, $\xi(p, \tau)$ the cardinality of a plan-task pair and $\text{Pre}(p)$ indicates the precondition of p.

Since each agent maintains its own task allocation, it is possible for the team to disagree on the allocation. In this case, the team is in *conflict*. Each agent periodically broadcasts all plan-task-state triples it executes, enabling the team to detect inconsistent allocations. We call these messages *plan messages*. When an agent receives a plan message, it updates its allocation accordingly, trusting the sender completely.

While this treatment of conflicts enables the team to react swiftly [8], it can cause reoccurring conflicts. If the underlying data is inconsistent within the team, it is possible for two or more agents to consider conflicting allocations as optimal. Reallocation can then cause them to continuously inform each other about the outcome of their decisions, while reallocating the respective other agents. This can go on until an external event stops the loop or the data becomes consistent. In the remainder, the specific state an agent inhabits given a plan and task is irrelevant, we will thus write $\text{In}(a, p, \tau)$ instead of $\text{In}(a, p, \tau, z)$.

▶ **Definition 2** (Conflict). Two allocations C_1 and C_2 are in conflict, if and only if they are for the same plan type pt, and, for some agent a, $\text{In}(a, p, \tau) \in C_1$, but $\text{In}(a, p, \tau) \notin C_2$. Two agents a_1 and a_2 are in conflict wrt. plan type pt iff they believe in conflicting allocations. We say $\text{In}(a, p, \tau)$ is a cause of the conflict. Note, there can be multiple causes for a conflict.

▶ **Proposition 1.** *If two agents a_1 and a_2 are in conflict wrt. plan type pt, caused by $\text{In}(b, p, \tau)$, then there is a conflict between one of them and agent b.*

This property guarantees that the robot, which needs to change its actions, is in conflict with at least one robot and thus has a chance to detect it. In order to devise a detection mechanism, we examine how allocations are modified over time. An agent's allocation for a plan type pt can change due to: *Reallocation*: the agent adopts a new allocation to improve the utility. *Messages*: the agent receives a message informing it about the state of another wrt. pt. *Leaving*: the agent leaves the plan and thus no longer tracks its allocation. *Deletion*: the agent has not received a message from another agent for a predetermined time. In this case the agent is removed from the team and thus from all task allocations.

Deletion is done to account for agents breaking down. We will not consider the deletion event further, as it can be seen as an empty plan message. We will also not consider agents leaving a plan, this event terminates the local agent's tracking of the plan and is visible to other agents by message events. These are able to detect a conflict arising by an agent entering and leaving a plan repeatedly. Conflicts which endure for a longer period of time cause a temporal pattern to appear in allocations believed by the involved agents. We therefore define formally how allocations change over time.

▶ **Definition 3** (Allocation Event). An allocation event e is a tuple $(\vartheta^+, \vartheta^-)$, consisting of allocation additions ϑ^+ and allocation subtractions ϑ^-, both sets of atoms of the form $\text{In}(a, p, \tau)$, such that $\vartheta^+ \cap \vartheta^- = \emptyset$.

▶ **Definition 4** (Allocation Event Composition). Let $e_1 = (\vartheta_1^+, \vartheta_1^-)$ and $e_2 = (\vartheta_2^+, \vartheta_2^-)$ be two task allocation events. Then their composition is defined as:

$$e_1 \circ e_2 \stackrel{def}{=} \big((\vartheta_1^+ \setminus \vartheta_2^-) \cup (\vartheta_2^+ \setminus \vartheta_1^-), (\vartheta_1^- \setminus \vartheta_2^+) \cup (\vartheta_2^- \setminus \vartheta_1^+)\big)$$

This composition allows to reason about results of multiple events independently of the allocations they apply to. Note, allocation events form an abelian group under event composition. If an agent a receives a plan message from another agent b, this causes an allocation event for every plan type agent a is executing. There are four cases to distinguish:

- a agrees with b on its allocation, hence the allocation event is empty and can be ignored.
- a does not believe that b participates in the plan type, but b does, this yields the allocation event $(\{\mathrm{In}(b, p, \tau)\}, \emptyset)$ for some p and τ.
- a believes that b executes a different plan or task than b actually does, yielding the event $(\{\mathrm{In}(b, p, \tau)\}, \{\mathrm{In}(b, p'\tau')\})$ such that $p \neq p' \vee \tau \neq \tau'$.
- a wrongly believes that b participates in the plan type, causing $(\emptyset, \{\mathrm{In}(b, p', \tau')\})$.

By Proposition 1, these are the only cases we need to consider. If a now reallocates and the composition of both events contains no statement about b, a cycle occurred.

▶ **Definition 5** (Allocation Cycle). Let e_m be a non-empty allocation event due to a plan message sent by agent b to agent a, and let e_r be the next allocation event after a receives the message. If $e_m \circ e_r = (A, B)$ such that $(\forall p, \tau)\, \mathrm{In}(b, p, \tau) \notin A \cup B)$, a cycle occurred.

Hence, a cycle occurs, if an agent reverts its allocation with respect to a message by reallocation. An agent can easily monitor its allocation events and thus detect such cycles. Of course, a cycle does not necessarily entail a conflict, it might well be that the cycle just reflects a very quickly changing situation. However, if multiple cycles occur subsequently, this becomes more and more unlikely to be a proper reaction to the changing conditions.

Given this detection scheme, we can devise a reaction to arising conflicts. The cycle length depends on the frequency with which the robots communicate and deliberate. In our case, messages are sent every $100ms$ and the deliberation loop takes $30ms$. Thus, the duration of two subsequent cycles is in average $115ms$. It is improbable that during this time frame the situation requires the team to change its allocation back and forth twice. Thus, we set a limit $n \geq 2$ on the number of subsequent cycles which may occur before an agent acts upon them.

In order to resolve the detected conflicts, we use a bully algorithm, where one agent forces the others into the allocation it deems best wrt. to the relevant utilities. The bullying agent assumes authority on the corresponding task allocation for a fixed time period. This time interval should be large enough to overcome the cause of the conflict, yet as small as possible so the team can return to its more dynamic and adaptive state as quickly as possible. Once the interval passed, all agents resume normal operation. An agent detecting n subsequent cycles will trigger the bully algorithm by broadcasting its allocation, triggering all involved agents to either respond or assume the proposed allocation. The leader is determined using unique ids for each robot. Only those agents which execute the corresponding plan type participate, not the whole team. Thus, the conflict is dealt with locally.

5 Evaluation & Conclusion

We evaluated our solution on a team of real robots, with both lab experiments and observations of the behaviour during real tournament situations. In all our experiments, we set the number of subsequent cycles to four and the bullying duration to two seconds, as mentioned above.

Fighting for the ball: In cases, where two robots moved very close to the ball, persistent conflicts become most apparent. Previously we observed cases, in which a conflict persisted for several minutes. The presented approach reliably detected and solved this conflict.

Phantom ball: Another typical scenario where conflicts can occur, is if at least one robot of the team detects a phantom ball far from the actual ball position. This can be due to an object looking similar to the ball appearing in the audience (e.g., an orange t-shirt), or an actual second ball on the field. In this case, depending on where the phantom ball is seen, the conflict will cause either two robots to pursue the ball, or none at all. Again, the conflict could be reliably detected and solved, however the solution to the conflict is arbitrary, as the team will react to the hypothesis of the robot with the highest id.

Tournament situations: We employed our approach during real tournament situations at RoboCup German Open 2010. Although we could not gather enough data for a rigorous statistical analysis, it can be said, that during one hour of playing, bullying occurred on average twelve times. In all cases, there was an actual conflict, which needed to be solved. In one third of the cases, the solution to the conflict was incorrect, due to the bullying robot not being correctly localised. In the remaining cases, the solution was acceptable and lead to teamwork being reinstated quickly.

In conclusion, our simple, yet efficient conflict resolution method deals well within teams of agents, extending the ALICA framework for cooperation. During real matches, the robotic team was able to detect and repair conflicts quite reliably.

The drawback of our approach is that the solution to a conflict is only as good as the data, the bullying robot bases its decision on. Hence, the robot with the best data should take precedence, which could be indicated by confidence values of the sensory information. However, since these confidence values vary over time and are possibly similar among the participating agents, this bears the danger of being another source for disagreement. Argumentation based methods, as in [5], might help to identify which team member should take lead, using confidences, or geometric arguments such as line-of-sight.

References

1 J. J. T. H. de Best, D. J. H. Bruijnen, R. Hoogendijk, R. J. M. Janssen, K. J. Meessen, R. J. E. Merry, M. J. G. van de Molengraft, G. J. L. Naus, and M. J. C. Ronde. Tech United Eindhoven Team Description 2010. Technical report, University of Eindhoven, 2010.

2 Nau Lau, Luís Seabra Lopes, Gustavo A. Corrente, and Nelson Filipe. Multi-robot team coordination through roles, positionings and coordinated procedures. In *IROS*, 2009.

3 Benedita Malheiro and Eugenio Oliveira. Argumentation as distributed belief revision: Conflict resolution in decentralised co-operative multi-agent systems. In *EPIA '01*, 2001.

4 Zhun Qiu and Milind Tambe. Flexible Negotiation in Teamwork (Extended Abstract). In *AAAI FALL Symposium on Distributed Continual Planning*, 1998.

5 Zhun Qiu and Milind Tambe. Towards Argumentation-based Collaborative Negotiation: A Preliminary Report. In *Int. Workshop on MAS*. Mass. Institute of Technology, 1998.

6 H. Rajaie, U.-P. Käppeler, O. Zweigle, K. Häussermann, A. Tamke, A. Koch, B. Eckstein, F. Aichele, D. DiMarco, A. Berthelot, T. Walter, and P. Levi. 1. rfc stuttgart,overview of hardware and software. Technical report, University of Stuttgart, 2010.

7 Hendrik Skubch, Michael Wagner, Roland Reichle, and Kurt Geihs. A modelling language for cooperative plans in highly dynamic domains. *Mechatronics*, 2010.

8 Hendrik Skubch, Michael Wagner, Roland Reichle, Stefan Triller, and Kurt Geihs. Towards a comprehensive teamwork model for highly dynamic domains. In *ICAART*, 2010.

9 M. Tambe. Towards flexible teamwork. *Journal of AI Research*, 7:83–124, 1997.

A Privacy-Preserving Social P2P Infrastructure for People-Centric Sensing

Michael Dürr and Kevin Wiesner

Mobile and Distributed Systems Group, Institute for Informatics
Ludwig-Maximilians-Universität München, Munich, Germany
{michael.duerr,kevin.wiesner}@ifi.lmu.de

Abstract

The rapid miniaturization and integration of sensor technologies into mobile Internet devices combined with *Online Social Networks* allows for enhanced sensor information querying, subscription, and task placement within *People-Centric Sensing* networks. However, PCS systems which exploit knowledge about OSN user profiles and context information for enhanced service provision might cause an unsolicited application and dissemination of highly personal and sensitive data.

In this paper, we propose a protocol extension to our OSN design *Vegas* which enables secure, privacy-preserving, and trustful P2P communication between PCS participants. By securing knowledge about social links with standard public key cryptography, we achieve a degree of anonymity at a trust level which is almost good as that provided by a centralized trusted third party.

Keywords and phrases People-Centric Sensing, Online Social Networks, P2P, Privacy, Trust

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.176

1 Introduction

The increasing spread of powerful mobile Internet devices like smartphones or tablets, accompanied by their rich set of integrated sensing facilities, already allow for numerous mobile and context-enriched applications. Induced by those technical innovations, People-Centric Sensing (PCS), a recently emerging research area in the field of Wireless Sensor Networks, attracts increasing attention. PCS focuses on the collection of user-generated sensor data as well as its application-oriented aggregation and utilization at Internet-scale. In contrast to traditional sensor networks, PCS heavily builds on sensor information which is generated by a user's personal sensing environment [3, 5, 7]. Triggered by the rocketing number of *Online Social Network* (OSN) users and the imminent integration of several mobile sensor devices into personal sensing environments, recent research attempts to combine PCS environments and OSNs [1, 8]. Knowledge derived from social graphs and OSN user profile information can significantly improve the process of sensor information querying, sensor data subscription, and sensor task placement.

A well-known problem of PCS environments and sensor data context information generation emerges from the insufficient compliance with user privacy and security demands. Prevailing concerns stem from the possibility that any other subscriber could infer user profiles from continuous tracking information in order to e.g. perform unsolicited advertisement or even personal attacks. Despite recent efforts to incorporate privacy and security features into PCS systems [3, 5, 7] and OSNs [2, 4], present architectures still do not provide adequate precautions to fulfill all related privacy and security demands. On the one hand, experiences from the operation of location-based services have shown that a widespread utilization of such systems suffers from inadequate anonymization. The same problem arises in association with PCS networks as users contribute sensing data comprising personal information which

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 176–181

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

must not be publicly available. On the other hand, integrity of sensor data originating from PCS networks must be guaranteed. At present, the only possibility considered secure consists in the application of a trusted third party for the generation and assignment of public key certificates for each PCS network participant. However, a trusted third party cannot guarantee user anonymity as there is at least one party which is able to map users, their messages, and their content to one and the same identity.

In order to avoid the need for a trusted third party, we consider a protocol extension for our P2P-based OSN architecture *Vegas* which allows for secure and privacy-preserving information querying inside PCS networks.

In the remainder of this paper section 2 gives a short overview of Vegas and details the proposed protocol extension. Section 3 discusses our extension focusing on performance, privacy, and security aspects. Work which is closely related to our approach is presented in section 4. Section 5 concludes the paper.

2 System Architecture

In the following, we review parts of our OSN architecture developed in previous work before we present our P2P-based overlay query routing extension which allows for privacy-preserving sensor task, subscription, and query distribution.

2.1 Vegas Design Principles

Our proposed P2P design Vegas builds on an architecture currently developed at the Ludwig-Maximilians-University Munich. The architecture focuses on privacy and security aspects within a P2P-based OSN. Vegas complies with a set of requirements that we consider inevitable for a secure and privacy preserving OSN. These requirements encompass a user's *informational self-determination*, *strong trust relationships* between friends, anywhere and anytime *profile availability*, and transparent *mobility support* [6].

Vegas represents a highly restrictive OSN as it does not support communication between participants that are not directly connected by an edge of the underlying social graph. This restriction is motivated by a problem we termed *social network pollution*. To give but a few examples of social network pollution, present OSNs offer the possibility for search operations on its social graphs, provide unsolicited friendship recommendations, and offer support for non-authorized linkage of a friend's friends. This leads to a multitude of unwanted friendship establishments i.e. links in the social graph which not necessarily represent a real friendship. Although this design choice disallows some appealing and beneficial applications, we consider this fact as an acceptable trade-off in order to guarantee a high degree of privacy and security.

2.2 Vegas Operation

In a nutshell, Vegas functions as follows: Any two users A and B who maintain a friendship in Vegas own a public key pair i.e. a *link-specific* key pair for each other. As user A holds a unique key pair $K^{-}_{A\to X_i}/K^{+}_{A\to X_i} (i \in 1,\ldots,n)$ for each of his n friends $X_1,...,X_n$, a key pair represents nothing else than a directed edge in the overall social graph. For the remainder of this paper, we always mean link-specific keys when we talk about keys and key pairs. The notion of a key $K^{[-|+]}_{A\to B}$ means that this key is a private/public key generated by A for exclusive communication with B. A utilizes B's public key $K^{+}_{B\to A}$ to encrypt messages intended for B. In addition, A's public key $K^{+}_{A\to B}$ is included into each message in order to map the originator of a message to its content. It should be stressed that, for the purpose

of signing, the application of A's private key $K^-_{A\to B}$ is restricted to messages addressed to B. Since a key pair represents an edge of the social graph, removal of such an edge – which equals to the cancellation of a friendship – is performed by the deletion of the matching key pair. This allows for easy key revocation as the application of keys is limited to two friends.

In order to allow for delay-tolerant offline communication, we build on an asynchronous message exchange scheme based on a concept presented in [13]. Vegas relies on well known services like email, SMS, or instant messaging which can be exploited to simulate the *exchanger* component. An exchanger represents the abstract concept of a message queue which may be used to transmit messages or any other kind of content. Any two Vegas friends A and B are aware of one or more such exchanger addresses of each other. In case A wants to send a message to B, A applies B's public key $K^+_{B\to A}$ to encrypt the message content. After signing the message with $K^-_{A\to B}$, A sends this message to one of B's exchangers. Now B can fetch this messages and identify sender A through his attached public key. It should be stressed that B is the only user that knows about the mapping of the included public key to the identity of user A. In case A considers the mapping of a public key to B's identity to be compromised, A can trigger a key refresh operation in order to replace all former key pairs shared with B.

2.3 Vegas Extension for Sensor Querying

Vegas represents a highly restrictive P2P OSN as it disallows communication between users that do not share an edge in the underlying social graph. Since P2P-based PCS applications require multi-hop (i.e multi-edge) communication we developed a protocol extension which allows for the application of Vegas for highly anonymous and trustful sensor information communication. In this paper we focus on the utilization of Vegas for sensor querying i.e. we provide an answer to the question *how to figure out one or more OSN users that a) support the discovery of sensor information, b) allow for sensor subscription, and c) facilitate sensor task placement.* Figure 1 illustrates this forwarding process in detail. Information from the OSN domain gives access to all properties and preferences from all OSN users that represent a friend. For instance, in case a user A wants to figure out other OSN users which could support querying sensor-related information, A can issue a sensor query to friends $X_1, \ldots, X_m$ ($1 \leq m \leq n$, n denotes the total number of A's friends) whose profiles match A's query content. Dependent on the kind of available sensors and user defined access policies a user X_i ($i \in 1, \ldots, m$) can respond to such a sensor query. To benefit from X_i's social relationships, we introduce a mechanism which allows for trusted and privacy-preserving forwarding of queries to users that do not share an edge with the query originator. Queries are tuples of the form `(operation, sensing properties, task properties)`. An example query for measuring the temperature in a certain area might look like this:

```
Q(i) = (put_task,(type=temp,frequency=1),(location={x,y},radius=r,time={t1,t2}))
```

At first, A attempts to locally match all profiles of his friends with the constraints of request `Q(i)`. Considering the request `Q(i)` a matchmaking process could be based on one or more heuristics that operate on the properties *location* and *time*. This restricts positive matches to users who not only support the requested sensor type and its properties but also visit places close to the provided location with a certain frequency and during certain periods of time. Since we do not limit requests to a predefined set of properties it could also be necessary to apply reasoning operations to reduce/increase the set of positive matches. After A finished the refinement process, he sends a request message `Q(i)` to all friends X_i ($i \in 1, \ldots, m$) whose profiles indicated a match. This message includes an exchanger addresses of A. A recipient,

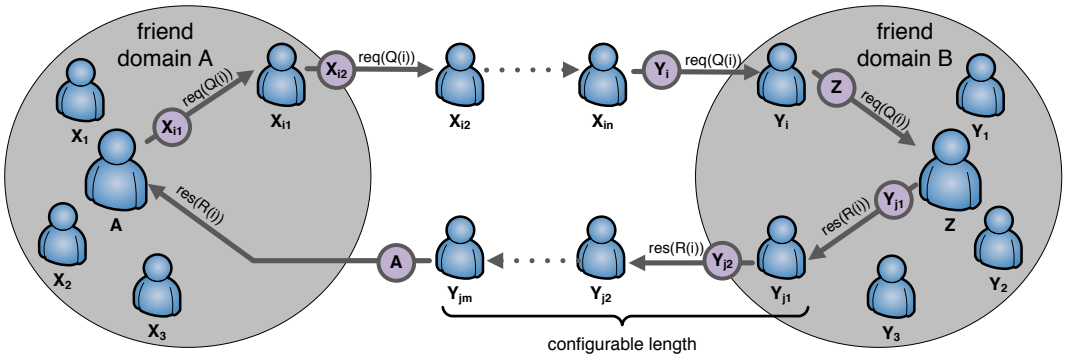


Figure 1 Schematic illustration of a query routing path. Messages are labeled with the corresponding exchanger utilized for transmission. Sending a message via exchanger T_i includes message encryption and signing with the corresponding key pairs.

in this example X_{i1}, who does not fulfill the requirements of `Q(i)` may decide to forward the query to all his friends whose profiles indicate a potential match for `Q(i)`. The decision whether to respond and forward a query depends on user preferences (either set by policies or manual interaction). This process repeats until `Q(i)` cannot be forwarded any longer or arrives at user Z who is able to setup the specified task. As we attach an exchanger address and a temporary public key of sender A to `Q(i)`, Z can send an encrypted response `R(i)` to A notifying him about a successful task setup. Due to our privacy demands, we enable Z to anonymize his identity by relying `R(i)` via one of his randomly selected friends Y_{j1}. To increase the degree of anonymity, we allow for a dynamic configuration of the number of edges `R(i)` has to traverse before it may reach the originator of the request. Dependent on the degree of anonymity, Y_{j1} does not directly respond to A but randomly selects one of his friends Y_{j2} as the next relay, and so far. Therefore, even multiple, successive responses are not delivered by the same user concealing a relation among those messages.

3 Discussion

Our approach represents a trade-off between support for efficient query routing inside PCS networks and the assurance of a high degree of anonymity and trust. This section provides some considerations regarding these aspects.

3.1 Routing Aspects

P2P networks can be separated into structured and unstructured systems. While unstructured systems suffer from query routing inefficiency due to their flooding approach, structured networks are susceptible to churn. The idea to infer query routing paths from OSN user profile information has recently been proposed for Pastry [10]. Our approach does not utilize the concept of DHTs but allows for simple forwarding based on preferences and capabilities of OSN participants. In essence, our design represents a trade-off since the application of OSN knowledge is expected to perform better than flooding but worse than DHT-based query routing. It should be mentioned that our proposed extension focuses on delay-tolerant networks. Although one could imagine scenarios which suffer from such a best effort approach, an asynchronous implementation is motivated by our demand for a high degree of anonymity and trust. In addition, this restriction can help to obtain a larger result set as users can forward requests even in case a recipient is currently offline.

3.2 Privacy Aspects

Although Vegas profiles are publicly available, by default, they are always encrypted with a friend's link-specific public key. Furthermore, all messages sent between two friends A and B are encrypted based on the corresponding public key. As long as direct messaging is limited to friends, Vegas adheres to all requirements listed in [6]. The proposed query extension for Vegas introduces a routing process which involves OSN users that do not share an edge with the query originator. However, our stringent privacy requirements are still fulfilled: Considering the query path of a sensor information request, the only information disclosed by the originator is one of probably multiple exchanger addresses. Dependent on the desired degree of anonymity, a sensing node may decide on the number of utilized exchangers as well as the frequency for exchanger refreshes. To increase the anonymity of a responding user, our extension allows him to specify the minimum number of edges a message has to pass before it can arrive at the originator of the request. Certainly, our extension cannot guard against protocol attacks like routing, storage, sybil, and eclipse attacks that all structured P2P networks suffer from. For instance, it takes no effort to replace an exchanger address to forward responses to another user. However, assuming trustful friendships, our extension cannot be exploited for security attacks targeting at message integrity or disclosure of user identities.

3.3 Trust Aspects

Vegas does not support unsolicited friendship establishment. Due to the introduction of multi-edge query routing, this stringent restriction becomes relaxed as private information disseminates over the boundary of the personal friendship domain. However, even in case a user observes hostile behavior of his friends, it necessitates a simple key pair deletion to cancel any association with such a friend. The degree of trust achieved by our design is not equivalent to that offered by a single trusted third party. However, compared to the application of a *web of trust* solution, our extension allows for much better trust relations between users that do not share an edge in the social graph. In summary, our trust model must be considered as a serious alternative to a trusted third party solution as it achieves complete informational self-determination in a decentralized way.

4 Related Work

A lot of research has been published with concepts focusing on privacy, security, and integrity aspects in PCS networks. The key idea of PriSense [11] is to slice data and send it via several different nodes, instead of sending it at once and directly to a server. Thereby, it conceals the data sensed by the individual users; however, it is still known which users participated. Our solution conceals the identities of participating users to allow for a real anonymous participation. Another approach is to add noise to sensed data, which can be subtracted in a later stage. In PoolView [7] measured time-series data is superimposed with a noise model, which is shared within the community. It preserves recently sensed values as well as their changes with time. Instead of perturbing measurements, AnonySense [3] employs several anonymization components, such as Tor and MIX networks, to obscure a user's identity. k-anonymity is ensured through a registration authority. An orthogonal approach is to actively engage users to decide on *what to reveal to whom*, and to learn what kind of data can be shared without compromising the user's privacy [12]. However, this concept simply retains sensitive data which results in a smaller amount of useful information.

Due to our fully anonymous communication, our solution also allows for sharing of sensitive data. An important research field represents privacy protection in OSNs. Some related work already proposes the application of P2P concepts to allow for privacy-preserving social networking [4, 2]. OSNs have also been combined with PCS networks but merely to enable new applications [8, 9] and not to enhance privacy. To the best of our knowledge, our approach is the first that uses an OSN to ensure privacy and trust in PCS networks.

5 Conclusion and Future Work

We proposed a protocol extension to our OSN design Vegas which achieves a high degree of anonymity and trust for delay-tolerant PCS networks. Our solution does not rely on a trusted third party but achieves trust by exploiting knowledge about the social graph. As our extension circumvents the spread of social network pollution and therefore complies with our requirement for perfect informational self-determination, we achieve a higher degree of trust than with a decentralized solution like the web of trust. In our future work we intend to perform several user studies to gain knowledge about the applicability of our approach in real world settings. In parallel, we will conduct intense simulations of typical protocol attacks identify the requirements for important system parameters like the minimum number of friends necessary to achieve a sufficient degree of anonymity and trust.

References

1 A. Beach, M. Gartrell, X. Xing, R. Han, Q. Lv, S. Mishra, and K. Seada. Fusing mobile, sensor, and social data to fully enable context-aware computing. In *Proc. of HotMobile '10*, pages 60–65. ACM, 2010.

2 S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta. PeerSoN: P2P social networking - early experiences and insights. In *Proc. of SocialNet '09*, 2009.

3 C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. AnonySense: Privacy-aware people-centric sensing. In *Proc. of MobiSys '08*, pages 211–224. ACM, 2008.

4 L. A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *Proc. of WONS'09*, pages 133–140. IEEE, 2009.

5 T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. PRISM: platform for remote sensing using smartphones. In *Proc. of MobiSys '10*, pages 63–76. ACM, 2010.

6 M. Dürr, M. Werner, and M. Maier. Re-Socializing Online Social Networks. In *Proc. of CPSCom'10*. IEEE, Dec 2010.

7 R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proc. of SenSys '08*, pages 281–294. ACM, 2008.

8 I. Krontiris and F.C. Freiling. Integrating people-centric sensing with social networks: A privacy research agenda. In Proc. of SESOC 2010, 2010.

9 E. Miluzzo, N. Lane, S. Eisenman, and A. Campbell. CenceMe – Injecting Sensing Presence into Social Networking Applications. In *Smart Sensing and Context*, volume 4793 of *LNCS*, pages 1–28. Springer Berlin / Heidelberg, 2007.

10 A.-K. Pietiläinen, E. Oliver, J. LeBrun, G. Varghese, and C. Diot. MobiClique: middleware for mobile social networking. In *Proc. of WOSN '09*, pages 49–54. ACM, 2009.

11 J. Shi, R. Zhang, Y. Liu, and Y. Zhang. Prisense: privacy-preserving data aggregation in people-centric urban sensing systems. In *Proc. INFOCOM'10*, pages 758–766. IEEE, 2010.

12 K. Shilton. Four billion little brothers?: privacy, mobile phones, and ubiquitous data collection. *Commun. ACM*, 52(11):48–53, 2009.

13 M. Werner. A privacy-enabled architecture for location-based services. In *Proc. of MobiSec '10*, 2010.

Optimization-based Secure Multi-hop Localization in Wireless Ad Hoc Networks*

Sander Wozniak[1], Tobias Gerlach[2], and Guenter Schaefer[1]

1 **Telematics and Computer Networks Research Group**
Ilmenau University of Technology, Germany
sander.wozniak@tu-ilmenau.de, guenter.schaefer@tu-ilmenau.de

2 **Operations Research and Stochastics Research Group**
Ilmenau University of Technology, Germany
tobias.gerlach@tu-ilmenau.de

Abstract

The problem of localizing nodes without GPS based on a small fraction of anchor nodes which are aware of their positions is considered to be an important service for applications in wireless ad hoc networks. With an adversary trying to mislead nodes about their estimated locations, several approaches aiming to defeat attackers by means of robustness instead of cryptographic measures have been proposed in the past. Nevertheless, these robust techniques focus on single-hop based localization. Hence, we investigate the impact of employing the well-known Least Median of Squares (LMS) algorithm in the context of the multi-hop based DV-hop approach. We argue that in this case LMS is no longer able to meet its requirements. We examine the source of this behavior and show that LMS leads to more accurate results when using the median to obtain average hop lengths in DV-hop. Furthermore, we investigate the feasibility of performing lateration using the l_1-norm instead of the typically employed l_2-norm, as well as the possibility of enhancing the robustness of LMS using lateration based on the l_1-norm. Contrary to our expectations, the l_1-norm only results in a slight, neglectable advantage compared to the computationally less expensive l_2-norm lateration.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.182

1 Introduction

Wireless Ad Hoc Networks offer a wide variety of applications ranging from environmental monitoring to intrusion detection or battlefield surveillance. An important service for these applications is the localization of the participants without relying on GPS. Thus the problem of localizing nodes using only a small fraction of anchor nodes which are aware of their positions has gained much attention from researchers in the past. While these mechanisms usually assume cooperative behavior among the participants, certain applications demand the deployment of nodes in an adversarial environment. In order to prevent an adversary from misleading nodes about their locations, a variety of secure localization schemes have been presented the last few years [5]. Yet most of these approaches require single-hop communication between nodes and anchors to conduct distance measurements. In contrast, multi-hop based schemes only rely on a few anchors to measure the distance between nodes and anchors. A well-known example is the DV-hop approach, where anchors broadcast small beacon messages holding their locations [3]. Nodes receiving such a message increment a contained hop count value and broadcast the adjusted message, assuming it provides a

* This work is supported by the DFG Graduiertenkolleg 1487 *(Selbstorganisierende Mobilkommunikationssysteme für Katastrophenszenarien).*

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 182–187

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

shorter path to the respective anchor. Once a node has received the messages from at least three anchors, it is able to perform lateration to obtain its coordinates using the according set of references (x_i, y_i, d_i), where (x_i, y_i) is the position of the anchor and d_i the respective measured distance. In order to obtain a distance measurement d_i, the number of hops of the shortest path to the respective anchor is multiplied with an average hop length value. Anchors estimate this hop length by calculating the sum of the euclidean distances to the other known anchors and dividing it by the sum of the number of hops of the shortest paths to the according anchors. Then, the hop length is broadcast in a separate message or piggy-backed with a beacon message sent out at regular intervals. Finally, nodes receiving these estimates from several anchors calculate the mean to obtain an aggregated hop length.

Based on these observations, this work provides the following contributions: First, we investigate the influence of employing the Least Median of Squares (LMS) approach in the multi-hop based DV-hop scheme. We show that LMS is unable to defeat a basic attack in the originally described DV-hop algorithm. Second, we show that estimating an average hop length using a slightly modified technique based on the median enables LMS to again provide robustness against this attack. Finally, we investigate the feasibility of the l_1-norm in contrast to the widely-used l_2-norm, as well as the possibility of enhancing the robustness of LMS by employing the l_1-norm.

2 Linear Least Squares Lateration

Given the set of N references (x_i, y_i, d_i), a node would ideally reside at the point of intersection of at least three circles with center (x_i, y_i) and radius d_i. Hence, assuming no distance measurement errors, it would be sufficient to find this point of intersection by solving a system of non-linear circle equations [2]. In reality, however, these circles usually do not intersect at a specific location (i.e. the system of equations is not solvable). In this case, a *least squares* approach minimizing the sum of residue squares can be used to estimate a position. However, this involves solving a non-linear optimization problem, which is usually considered too expensive as it requires methods of global optimization. Therefore, the result of the non-linear least squares approach is approximated by using the *Linear Least Squares* (LLS) technique which is based on the following non-linear optimization problem [2]:

$$\begin{pmatrix}\hat{x}\\ \hat{y}\end{pmatrix} = \arg\min_{\mathbf{x}} \|\mathbf{Ax}-\mathbf{b}\|_p \qquad \mathbf{A}\in\mathbb{R}^{(N,2)},\ \mathbf{x}\in\mathbb{R}^2,\ \mathbf{b}\in\mathbb{R}^N \tag{1}$$

Here, $\|\cdot\|_p$ is the l_p-norm ($p \geq 1$ is a parameter which may be chosen to fit a specific application) and $\mathbf{Ax} = \mathbf{b}$ is the matrix form of a system of linear equations. This system of equations is obtained by subtracting the mean of all left and right parts of the system of non-linear circle equations from each equation according to [2]. Furthermore, it is easy to see that $(\hat{x}, \hat{y})^T$ is a solution of $\min_{\mathbf{x}} \|\mathbf{Ax}-\mathbf{b}\|_p$ if and only if it is a solution of $\min_{\mathbf{x}} \|\mathbf{Ax}-\mathbf{b}\|_p^p$.

Usually, the l_2-norm ($p = 2$) is used to estimate a location. This is due to the fact that, for $p = 2$, a location can be estimated by simply solving a system of linear equations using QR-factorization for example. Nevertheless, while employing the l_2-norm ($p = 2$) is considered to be the most feasible approach, it is known to be vulnerable to malicious references forging the location or the distance to an anchor [2].

The LLS approach may also be applied to fit a function to a given set of data points. In this case, compared to the l_2-norm, the l_1-norm is generally less vulnerable to outliers contained in the data, e.g. caused by measurement errors. Therefore, we are interested in whether using the l_1-norm instead of the l_2-norm might increase the robustness of lateration

against attackers. Employing the l_1-norm, the position of a node is estimated by solving the following optimization problem:

$$\begin{pmatrix}\hat{x}\\ \hat{y}\end{pmatrix} = \arg\min_{\mathbf{x}} \|\mathbf{Ax}-\mathbf{b}\|_1 = \arg\min_{\mathbf{x}} \left\{\sum_{i=1}^{N} |\mathbf{a}_i\mathbf{x}-b_i|\right\} \quad (2)$$

This problem can be formulated as a linear optimization problem by introducing a vector $\mathbf{h} = (h_1, h_2, \ldots, h_N) \in \mathbb{R}^N$ of auxiliary variables $h_i \geq 0$, $\forall i \in \{1, \ldots, N\}$:

$$\underset{\mathbf{x}}{\text{MIN}} \left\{\sum_{i=1}^{N} |\mathbf{a}_i\mathbf{x}-b_i|\right\} \Leftrightarrow \underset{\mathbf{x},\mathbf{h}}{\text{MIN}} \left\{\sum_{i=1}^{N} h_i \;\middle|\; -\mathbf{h} \leq \mathbf{Ax}-\mathbf{b} \leq \mathbf{h}\right\}$$

Hence, we obtain:

$$\underset{\mathbf{x},\mathbf{h}}{\text{MIN}} \left\{ (\mathbf{0}^T, \mathbf{1}^T) \begin{pmatrix}\mathbf{x}\\ \mathbf{h}\end{pmatrix} \middle| \begin{pmatrix}\mathbf{A} & -\mathbf{E}\\ -\mathbf{A} & -\mathbf{E}\end{pmatrix} \begin{pmatrix}\mathbf{x}\\ \mathbf{h}\end{pmatrix} \leq \begin{pmatrix}\mathbf{b}\\ -\mathbf{b}\end{pmatrix} \right\}$$

where $\mathbf{E}$ is the identity matrix of dimension N, $\mathbf{0} = (0,0)^T \in \mathbb{R}^2$ and $\mathbf{1} = (1, 1, \ldots, 1)^T \in \mathbb{R}^N$. Depending on the norm, we refer to the respective estimation technique as l_1-LLS or l_2-LLS. While l_1-LLS is computationally more expensive than l_2-LLS, a potential increase in robustness might justify solving a simple linear optimization problem.

3 Threats and Countermeasures

There are a variety of attacks which aim at deceiving nodes about their locations [5]: *Impersonation attack*, *sybil attack*, *wormhole attack* and *location reference attack*. It should be noted here that while a Denial-of-Service attack (e.g. jamming) might also disrupt the process of localization, an adversary is usually assumed to try to unnoticeably mislead nodes about their whereabouts. Several mechanisms aiming to defeat one or more of the above mentioned threats have been proposed. They can be divided into *prevention*, *detection* and *filtering* (i.e. robust) techniques [5]. In the past, several robust location estimation schemes have been proposed. Within this work, we only consider the well-known Least Median of Squares (LMS) filtering approach [2], focusing on possible advantages of performing lateration using the l_1-norm instead of the commonly used l_2-norm, as well as its behavior in the context of the multi-hop based DV-hop scheme. In addition, we only focus on the colluding location reference attack where a number of malicious anchors broadcast beacon messages with false coordinates. These coordinates are shifted into a certain common direction away from the true positions. This threats is also known as *false beacon location attack* and very popular in terms of evaluating the robustness of location estimation schemes [2, 5].

LLS employing the l_2-norm is not robust against outliers [2]. Li et al. therefore propose to minimize the median instead of the sum of residue squares based on the method described in [4]. Finding the exact solution of this non-linear optimization problem is computationally expensive. Thus, the authors present the following algorithm as an approximate solution [2]:

1. Randomly draw $M = 20$ subsets of size 4 from the set of given references.
2. Estimate a location for each subset $j = 1, \ldots, M$ using l_2-LLS and calculate the median of the estimation residuals r_{ij}^2 to each anchor $i = 1, \ldots, N$.
3. Define $m = \arg\min_j med_i \left\{r_{ij}^2\right\}$ (least median of all medians of each subset).
4. Calculate $s_0 = 1.4826(1 + \frac{5}{N-2})\sqrt{med_i r_{im}^2}$.
5. Assign a weight w_i to each reference, where $w_i = 1$ if $|\frac{r_i}{s_0}| \leq 2.5$ or 0 otherwise.
6. Compute a weighted least squares of all given references using weights w_i. This corresponds to estimating a position using l_2-LLS with only the references with weight $w_i = 1$.

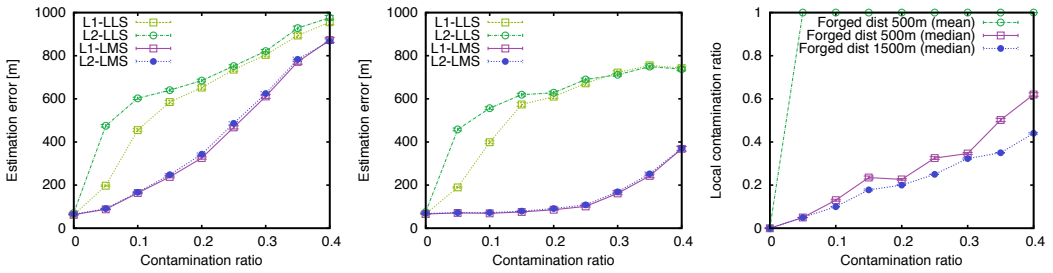
4 Evaluation

To evaluate the impact of employing LMS in DV-hop, as well as investigating the suitability of using the l_1-norm instead of the l_2-norm, we implemented DV-hop using OMNeT++ (http://www.omnetpp.org/) as follows: Each anchor broadcast a beacon message with a sequence number containing its location in a total of 3 rounds separated by intervals of roughly 60 seconds to provide the network with enough time to distribute the messages. So far, we did not incorporate a mechanism to limit flooding, since this might result in different contamination ratios at different nodes. Anchors receiving beacons from other anchors calculated a hop length estimate following either the original DV-hop approach or the median variant and included this information in the message to be sent out in the next round. Additionally, nodes kept the most recent hop length estimate announced by the corresponding anchor. Finally, after finishing the 3 rounds, nodes estimated their locations by choosing a hop length from their list of available estimates and running each of the location estimation schemes on their respective set of references. When selecting a hop length, nodes employed the mean or the median according to the technique currently in use by the anchors.

For communication among nodes, we incorporated the radio model provided by the MiXiM framework (http://mixim.sourceforge.net/). Apart from evaluating the original LMS approach using l_2-LLS in step 2 and 6 of the algorithm which from now on we will refer to as l_2-*LMS*, we also consider a new variation of LMS employing l_1-LLS which we refer to as l_1-*LMS*. We implemented and compared l_1-LLS, l_2-LLS, as well as both LMS variants. In order to obtain a location using l_1-LLS by finding a solution to (2), we employed lpsolve (http://lpsolve.sourceforge.net/). Furthermore, regarding l_2-LLS, we used lapack++ (http://lapackpp.sourceforge.net/) to estimate a position by solving the respective system of linear equations with QR-factorization. We randomly placed 300 nodes and $N = 20$ anchors on a 1000 m $\times$ 1000 m field using the uniform distribution. The field size was chosen according to a density required to prevent the partitioning of the network and to the applied transmission power of 110 mW, which roughly corresponds to an interference range of 140 m. In our scenario, the mean number of incoming connections at a node is about 12, which is a stable value above the critical threshold of 9 determined by Langendoen and Reijers [1] with a mean of about 7 hops on the shortest paths between nodes and anchors.

Regarding the false beacon location attack where anchors announce a false position, out of N anchors, $\lceil N \cdot \epsilon \rceil$ were randomly selected to be malicious. The contamination ratio ϵ was varied from 0 (no attacker) to 0.4 in steps of 5%. Malicious anchors forged their location by adding a vector defined by a common direction and specific length to their actual coordinates. The length of this vector which is from now on referred to as *forged distance* was set to 500 m and 1500 m to examine different strengths of the attack. To measure the influence of the attacking anchors, we used the *mean estimation error* which corresponds to the mean of the euclidean distance between the actual and the estimated location over all nodes. Furthermore, it should be noted here that the following figures all show the mean of 30 repetitions including the confidence intervals at a confidence level of 99 %.

We first evaluated the robustness of the location estimation schemes when employed in the original DV-hop approach. Figure 1a shows the estimation error for a forged distance of 1500 m. Here, l_2-LLS shows the expected non-robust behavior, suffering from an increasing number of malicious anchors. Furthermore, according to our assumptions, for $\epsilon < 0.2$, l_1-LLS is able to provide a decrease of the error compared to l_2-LLS. However, contrary to our initial expectations, the difference between the estimation error of l_1-LLS and l_2-LLS becomes neglectable for $\epsilon \geq 0.2$. This may be based on the fact that the system of non-linear circle



(a) Original DV-hop approach **(b)** DV-hop using median **(c)** Impact of malicious hop lengths

Figure 1 Evaluation of LMS in DV-hop (false beacon location attack).

equations is linearized by subtracting the mean of all equations according to [2]. Therefore, malicious references may still be able to influence the resulting system of equations $\mathbf{Ax} = \mathbf{b}$, preventing the l_1-norm from providing a clear advantage for an increasing forged distance.

To understand why LMS is unable to provide the expected robust behavior as shown in figure 1a, it is necessary to explain the effect of attackers exploiting honest nodes and anchors to support the attack. Malicious anchors increase the euclidean distance computed at benign anchors, while the number of hops between the anchors remains the same. Therefore, when estimating the hop length by summing up the euclidean distances and dividing it by the sum of the number of hops, an attacker is able to cause benign anchors to announce increased hop length estimates. Computed at a benign anchor, while still being influenced by malicious distance measurements, we call such a hop length estimate *polluted*. Consequently, in the original DV-hop algorithm, with the number of hops of the shortest paths being multiplied with the hop length estimate, a polluted hop length resulting from forged anchor locations affects all distance measurements obtained at a node. With the median being able to ignore outliers up to 50 %, it seems reasonable to aggregate hop length estimates using the median in order to only incorporate references from other benign anchors (which should be a majority). Zeng et al. shortly state this assumption and propose to employ the median when aggregating hop lengths at the anchors [6]. However, they do not mention the effect of benign anchors increasing the strength of the attack. Furthermore, while they provide no evaluation, we are able to confirm their assumption according to the estimation error shown in figure 1b. Here, anchors and nodes aggregate the respective hop lengths using the median, enabling LMS to yield its expected robust behavior. This decreases the estimation error from over 800 m at $\epsilon = 0.4$ in the original DV-hop to about 400 m when using the median. It should be noted here that the assumption of benign anchors supporting the attack is also confirmed by the estimation error of the non-robust l_2-LLS approach decreasing from roughly 1000 m (figure 1a) at $\epsilon = 0.4$ to about 750 m (figure 1b) when using the median in DV-hop.

Accordingly, while employing the l_1-norm reduces the estimation error for $\epsilon < 0.2$, LMS does not benefit from using l_1-LLS (figure 1a and 1b). This may be based on the fact that for $\epsilon < 0.2$, the original l_2-LMS approach is already able to filter out the majority of malicious references. Hence, in terms of LMS, we conclude that employing l_1-LLS instead of the computationally less expensive l_2-LLS approach does not provide a clear benefit.

In order to obtain a better understanding of the actual filtering ability of the median regarding polluted hop lengths resulting in an increased strength of the attack, we investigated the *local contamination ratio*. At a node, the local contamination ratio describes the ratio of malicious references among all given references. To obtain this ratio, we tagged each beacon message with a contamination field in our simulation. Hence, this field allowed to determine

whether a beacon contained a false location or a polluted hop length. While the local contamination ratio usually corresponds to the global contamination ratio ϵ, transforming a number of hops to a distance by multiplying it with a polluted hop length can result in a local contamination ratio of 1 (i.e. all references are affected by the attack). Figure 1c shows the mean local contamination ratio over all nodes. According to our expectations, when using the original DV-hop approach, all references are either malicious or affected by a polluted hop length at a forged distance of 500 m. In contrast, using the median allows to filter the majority of polluted hop lengths. For a forged distance of 500 m, the median is unable to filter all polluted hop lengths due to noise in the benign hop lengths. However, with an increasing forged distance of 1500 m, the median is able to almost filter out all polluted hop lengths. This may be based on the fact that in this case polluted hop lengths are larger than the noise among the benign hop lengths. We therefore conclude that employing the median should be considered mandatory in DV-hop.

5 Conclusions and future work

In this work, we evaluated the robustness of LMS when employed in the multi-hop based DV-hop approach. We showed that LMS is already unable to provide the expected robust behavior for a simple anchor-based attack. However, when estimating a hop length with a slightly modified approach based on the median, LMS shows the expected robust behavior. Thus, in terms of secure localization, using the median based technique in DV-hop should be considered mandatory. Furthermore, we employed the l_1-norm instead of the l_2-norm to perform lateration. Contrary to our expectations, the l_1-norm which is typically more robust against outliers, only provided a slight, neglectable benefit when employed in LMS. We assume that this behavior is based on the subtraction of the mean of all equations from each equation (i.e. the linearization of the system of equations) according to [2]. Therefore, we recommend using the computationally less expensive l_2-LLS approach.

In our future work, we aim at providing an extensive comparison of the performance of a wider variety of robust location estimation techniques employing the median-based DV-hop algorithm in a three-dimensional, multi-hop based environment.

References

1 K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: A quantitative comparison. *Computer Networks*, 43(4):499 – 518, 2003. Wireless Sensor Networks.

2 Z. Li, W. Trappe, Y. Zhang, and Badri Nath. Robust statistical methods for securing wireless localization in sensor networks. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 91 – 98, April 2005.

3 D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 5, pages 2926 –2931 vol.5, 2001.

4 P.J. Rousseeuw and A.M. Leroy. *Robust regression and outlier detection*. Wiley-IEEE, 2003.

5 Y. Zeng, J. Cao, J. Hong, S. Zhang, and L. Xie. Secure localization and location verification in wireless sensor networks: A survey. *The Journal of Supercomputing*, pages 1–17, 2010. 10.1007/s11227-010-0501-4.

6 Y. Zeng, S. Zhang, S. Guo, and X. Li. Secure hop-count based localization in wireless sensor networks. In *CIS '07: Proceedings of the 2007 International Conference on Computational Intelligence and Security*, pages 907–911, Washington, DC, USA, 2007. IEEE Computer Society.

Efficient Distributed Intrusion Detection applying Multi Step Signatures

Michael Vogel and Sebastian Schmerl

Brandenburg University of Technology
Cottbus, Germany
{mv|sbs}@informatik.tu-cottbus.de

Abstract

Intrusion Detection Systems (IDS) offer valuable measures to cope with today's attacks on computers and networks. But the increasing performance of networks and end systems and the growing complexity of IT systems lead to rapidly growing volumes of observation data and large signature bases. Therefore, IDS are forced to drop observations in high load situations offering chances to attackers to act undetectable. We introduce an efficient dynamically adaptable, distributed approach for a multi-step signature based IDS. Finally, we discuss initial performance evaluations of a prototype implementation and motivate future work scopes.

Keywords and phrases Computer Security, Distributed Intrusion Detection, Attack Signatures

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.188

1 Motivation

Intrusion Detection Systems (IDS) have been proven as important instruments for the protection of computer systems and networks. IDSs consist of sensors and analysis units. Sensors are either network-based or host-based. *Host sensors* monitor the activities of applications and the operating system on observed hosts (e.g. system calls). *Network Sensors* monitor network traffic at mirror ports of routers and switches. The sensors capture security relevant activities from these observations and continuously output a stream of *audit events* which is passed to an *analysis unit*. IDSs apply either pattern anomaly or misuse detection. Misuse detection searches for traces of security violations in captured observations (audit data), using known attack patterns – the signatures.

A challenge that all intrusion detection systems are facing today is the increasing performance of both networks and end systems. This leads to a rapid growth of audit data volumes to be analyzed. On the other hand, the growing complexity of the IT-systems causes novel vulnerabilities and offers new possibilities for running attacks so that the number of signatures to be analyzed increases as well. Already today intrusion detection systems are forced to drop audit data in high load situations. Thus, countermeasures become impossible. This provides the attackers also the possibility to apply a "be patient"-attack-strategy which portions their malicious activities over several days to exploit the fact that overloaded IDS discard detection results regularly, due to limited memory resources.

To cope with this situation several approaches have been proposed, e.g. the detection of intrusions based on an analysis of more compact, less detailed network log data [3]. But all these approaches aim at optimizing the non-distributed, single threaded signature analysis. The Gnort approach in [7] utilizes the massive parallel computing capabilities of expensive, high end graphic processors (GPUs), but it only doubles the analysis throughput compared to the sequential Snort tool. In contrast to today's highly loaded IDS, extensive spare computing resources are available on cheap desktop machines in every network. Therefore,

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 188–193

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a distributed signature analysis is a possible way to overcome this issue. So far, today's network-based IDS only apply primitive means to parallelize packet analysis by load balancing mechanisms [1]. There are also almost no approaches to parallelize host-based audit data analyses [2].

2 Efficient Distributed Intrusion Detection

Today, most IDS used in practice use a centralistic approach and apply misuse detection (e.g. Snort [6]). The implementation and configuration of misuse detection systems are simpler and allow for a significantly higher detection accuracy compared to anomaly detection. Typically, single step signatures are applied to detect attacks by analyzing a single audit event (e.g. network packet). In contrast to single step signatures, multi-step signatures allow for a fine-grained modeling of attacks. They specify more detailed characteristic attack traces, their dependencies, and the chronological order of the steps. Many semantic aspects of multi-step attacks cannot be modeled by single-step signatures or only insufficiently, which results in an increased false alarm rate (false positives). Therefore, in the following we focus on misuse detection based on the analysis of multi-step signatures. Our approach bases on an existing non-distributed IDS, which applies multi-step signatures defined in the Event Description Language (EDL) [5]. The state/transition approach EDL uses a Petri net like description principle.

2.1 EDL Signatures

EDL descriptions consist of places and transitions connected by directed edges. *Places* represent relevant system states of an attack. Hence, they characterize the attack progress. *Transitions* describe state changes triggered by audit events from the audit data stream recorded during an attack. An example of an EDL signature with four places (P_1 – P_4) and three transitions (T_1 – T_3) is depicted in Fig. 1. Ongoing attacks are represented by tokens

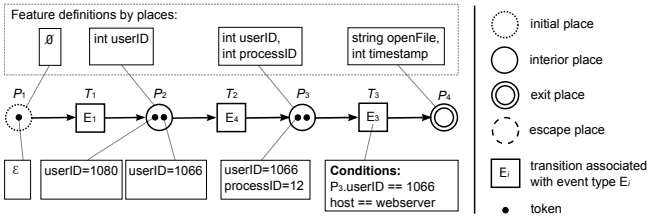


Figure 1 EDL Signature Example

on places. Tokens can be labeled with values like colored Petri nets. A place defines zero or more features which specify the properties of tokens located on this place. EDL distinguishes four place types: initial, interior, escape and exit. *Initial places* are signature's starting places which are marked with an initial token at start up. The *exit place* represents the completion of an attack instance. Thus, a token reaches this place implies that an attack has been detected. *Escape places* indicate that events occurred in the audit data stream which make the completion of an attack instance impossible. Therefore, tokens reaching places of this type are discarded. All other places are *interior places*.

A transition that is triggered by an event of a certain type can also contain a set of conditions. As shown in Fig. 1, these conditions can specify constraints over certain features of the triggering event (e.g., `host=webserver`) and token values (e.g., `P3.userID=1066`). The

evaluation of these transition conditions requires CPU time depending on the complexity of the conditions and the frequency of the evaluation, which is determined by the number of occurring events in the audit data and the number of tokens on input places of a transition.

2.2 Distributed Analysis

In order to cope with overload situations, which often forces today's IDS to drop audit data, the required analysis effort can be distributed among a set of cooperating analysis units on different hosts or multiple CPU cores. Primitive load balancing mechanisms, assigning audit events to some analysis units cannot be used for multi-step signature-based IDS, because an attack consists of a chain of distinct attack traces (audit events). Instead, the signature base of the IDS can be split up into a number of distinct subsets which are assigned to different cooperating analysis units. But this requires to transmit each audit event many times (from the sensor to each host that runs an analysis unit). Therefore, a more sophisticated signature distribution, which minimizes the need to duplicate captured audit events is desirable. This can be achieved by identifying fine grained minimal parts in each multi-step signature that can be independently assigned to different analysis units. This allows to optimize the required communication effort for distributing and duplicating captured audit data for each distributed analysis unit. By pooling minimal signature parts which analyze the same type of audit events and assigning them to the same analysis unit, events of this type only have to be sent to a subset of the analysis units. As an example, we consider an EDL signature that

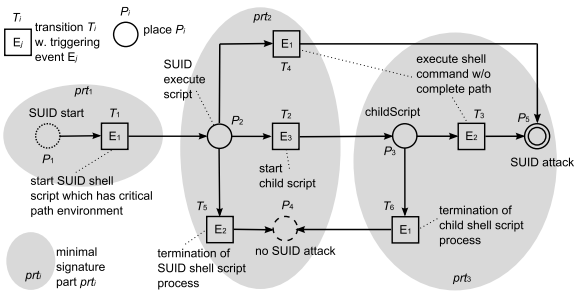


Figure 2 Minimal parts of SUID example signature

describes the SUID (set user ID) script attack on a former version of the Solaris OS, which is depicted in Fig. 2. Without explaining the attack in detail, the attacker tries to gain administrative privileges by exploiting a vulnerability of the extended file access rights.

The gray shaded spheres in Fig. 2 represent the three minimal parts, this signature can be partitioned into. These parts cannot be partitioned further, because the transitions have to evaluate the tokens on their input places. Therefore, a transition and its input places have to be assigned to the same signature part. Now, if signature parts are assigned to different analysis units, a token forward mechanism is required. In Fig. 2, the execution of transition T_2 from signature part prt_2 requires to place a token on the output place P_3 which belongs to different signature part prt_3. If prt_2 and prt_3 are assigned to different analysis hosts the token has to be forwarded via a communication channel between both analysis hosts.

The basic concept of a prototype implementation of our approach is depicted in Figure 3. A *sensor* logs audit events and classifies them into different event types (e.g. network protocol, port, specific system calls). The signature base is splitted up into minimal signature parts (prt*ij*) and each part is assigned to one *analysis unit* (AU), which allows for many different assignments of the signature base to avaliable analysis units. A configurable *filter*, knowing

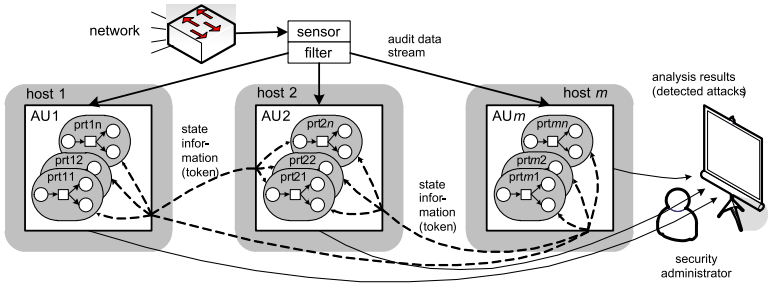


Figure 3 Distributed IDS Overview

the current signature partitioning, discards non-relevant audit data and transmits only those event types to each analysis unit, which are analyzed by them to limit the communication effort. The distributed analysis units examine incoming audit events by evaluating the transition conditions of assigned signature parts. If a transition is activated and a token has to be placed on an output place, which was assigned to a remote analysis unit, then this token is transmitted to the responsible analysis unit. The results of the distributed analysis are aggregated and evaluated by a *security administrator*.

3 Initial Evaluations

The distribution strategy described in the previous section has been implemented and evaluated using a distributed Intrusion Detection System, which bases on EDL multi-step signatures [4]. We modified and extended our existing Intrusion Detection tool SAM (signature analysis module) according to Figure 3 to convert it into a distributed IDS. SAM sensors forward audit events to a configurable number of analysis units over network connections (sockets). Each SAM analysis unit possesses a configurable set of EDL signatures. This implies that a configurable set of minimal signature parts is assigned to each analysis unit. The analysis units exchange state information (tokens) between each other also over network connections. We choose three example signatures and calculated all possible partitions (assignments) of the contained signature parts to three analysis units. We evaluated each of these 965 partitions on three parallel running SAM analysis units. These partitions include very efficient distributions as well as completely inappropriate, inefficient ones, causing badly balanced loads of the analysis units. Because of limited space, only two of the used signature examples are explained in the following.

The first example describes the SUID (set user ID) script attack that already has been introduced in the previous section. Another used signature example, consisting of 8 places and 15 transitions, which is not depicted here due to lack of space, describes the link shell attack which exploits a vulnerability of a former version of the Solaris OS. The attack exploits a specific shell function as well as the SUID (set user ID) mechanism. If a link refers to a shell script and the scripts filename starts with a hyphen "-" then an attacker can get an interactive shell having the access rights of the script owner (e.g. root) by executing the link.

In order to evaluate the analysis efficiency of our distributed system we first used a generic set of audit data. The data set was created by capturing system calls of a host while the described attacks were executed. Afterwards, all logged system calls, which do not belong to the executed attacks have been discarded manually. Therefore, the audit data set only contains relevant attack traces of the applied signatures. Concerning the required analysis effort, this represents the worst case scenario, demanding for the maximum computation

effort for analysis. Additionally, the captured attack traces have been duplicated to create a sufficiently large audit data file of 6,000 events (system calls). The experiments were conducted on four separate machines (Intel Xeon, 2.66 GHz, 512 KB L2 cache, 2 GB RAM) which are connected by switched Fast Ethernet links. One machine executes the sensor; the others each run an analysis unit. At first we applied the generic audit data set and evaluated, that even assigning the three example signatures to different analysis units, without splitting them into parts, leads to a runtime improvement. We measured the run time separately for each analysis unit. Then, we evaluated if further runtime improvements can be achieved, by fine grained assignment of signature parts to different AUs and which signature partition turned out to be efficient. Therefore, we evaluated all 965 possible different distributions of our signature examples to three analysis units. The Table 1 contains runtime evaluations for some selected signature distributions. The sensor runtime is related to the slowest analysis unit, as the sensor terminates after transmitting the last audit event (to the slowest AU). The first row (id 0) represents the non-distributed case. Thus, only one analysis unit (client)

distribution	**Sensor**	**Client 1**		**Client 2**		**Client 3**	
id.	real [s]	real [s]	user [s]	real [s]	user [s]	real [s]	user [s]
0	47.953	47.625	46.89				
96	29.625	37.718	28.672	29.328	9.828	32.390	7.844
302	110.641	110.656	108.469	116.343	33.891	113.718	19.313
626	19.719	19.750	17.948	25.437	18.078	23.453	17.969

Table 1 Selected runtimes for different signature distributions

is used, which gets all signatures assigned. This is the benchmark for any optimizations by signature distribution. The second row (id 96) represents the obvious distribution, which assigns each of the three example signatures completely to a different analysis unit (clients). Thus, the signatures are not split up into parts and no state information (tokens) have to be exchanged between different analysis units. The runtime evaluation shows a relevant improvement for the distributed case. The real runtime of the sensor, which captures and sends the audit data decreases by roughly 60 %. That means the distributed analysis run completes 60 % faster than the non-distributed run. But the really consumed CPU time (user time) of the three clients indicate that the analysis distribution is not well balanced. Client 1 (28.672 sec) consumes significantly more CPU time than the other clients (9.828 sec, 7.844 sec). Further, the last row shows the signature distribution (id 626) requiring the least runtime to complete, which is significantly more efficient than the distribution (id 96). Here, parts of a signature are fine-grained assigned to the different AUs and the sensor requires only half of the runtime compared to the non-distributed case. Further the required CPU times (user time) of the clients indicate a well balanced signature distribution among three clients. Finally, the third row (id 302) shows runtime results of the expected worst suitable analysis distribution requiring the maximum runtime (110 sec). Thus, by choosing a bad signature distribution, where signatures are split up poorly, the distributed analysis can take substantially longer compared to the non-distributed case (id 0). The evaluation of all possible 965 different partitions of the signature base shows that there are many efficient signature partitions offering short run times as well as some completely unsuitable partitions.

An efficient distributed IDS applying EDL signatures has to choose one of the partitions, requiring a low overall runtime. Therefore, it is necessary to introduce, resp. create a metrics which maps features of the audit data characteristics (e.g. number of occurred

specific audit events), monitored by the sensor, as well as the statistics maintained by the analysis units (detailed logs of spend communication and computation effort) for the currently used signature distribution to a metrics value M. This metrics then should be applied to continuously predict suitable assignments of signature parts to available analysis units. The distributed IDS then can use the metrics predictions to dynamically adapt the analysis configuration to the ever changing characteristics of the captured audit data, as well as changing available analysis resources, by reassigning signature parts to other analysis units. But even without a predicting metrics our approach is not useless. A suitable, resp. efficient initial partition can always be achieved by simply equally assigning whole signatures (without splitting them) to the analysis units in a round-robin manner (e.g. id 96 in Table 1).

4 Summary and Future Work

We presented an efficient approach for distributed IDS, which aims at balancing the computation load of complex signature-based IDS across multiple analysis units. The approach is not limited to EDL signatures and thus can be adapted to any multi-step signature based IDS. A prototype implementation was used to initially examine the achievable performance improvements by distributing the audit data analysis to a number of parallel running analysis units. We applied a set of three example signatures, split up into minimal signature parts. Then, we evaluated the analysis performance for the non-distributed case (as a benchmark) and all possible, different assignments of signature parts to three distinct analysis units. Evaluation shows many suitable signature partitions, which run significantly faster, but also worse ones, requiring much more run time, compared to the non-distributed baseline. Further work to create a metrics which predicts expectable computation and communication effort for different signature partitions based on audit data characteristics is in progress. The metrics should enable a distributed IDS to dynamically reconfigure its analysis distribution in order to adapt to changing analysis effort and available resources.

References

1 Michele Colajanni and Mirco Marchetti. A parallel architecture for stateful intrusion detection in high traffic networks. In *Proc. of the IEEE/IST Workshop on "Monitoring, attack detection and mitigation" (MonAM 2006)*, Tuebingen, Germany, September 2006.

2 Christopher Krügel, Thomas Toth, and Clemens Kerer. Decentralized event correlation for intrusion detection. In *ICISC*, volume 2288 of *Lecture Notes in Computer Science*, pages 114–131. Springer, 2001.

3 John McHugh. Sets, bags, and rock and roll: Analyzing large data sets of network data. In *ESORICS*, volume 3193 of *Lecture Notes in Computer Science*, pages 407–422. Springer, 2004.

4 Michael Meier. A model for the semantics of attack signatures in misuse detection systems. In *ISC*, volume 3225 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 2004.

5 Michael Meier, Sebastian Schmerl, and Hartmut König. Improving the efficiency of misuse detection. In *DIMVA*, volume 3548 of *Lecture Notes in Computer Science*, pages 188–205. Springer, 2005.

6 Martin Roesch. Snort: Lightweight intrusion detection for networks. In *LISA*, pages 229–238. USENIX, 1999.

7 Giorgos Vasiliadis, Spyros Antonatos, Michalis Polychronakis, Evangelos P. Markatos, and Sotiris Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In *RAID*, volume 5230 of *Lecture Notes in Computer Science*, pages 116–134. Springer, 2008.

Node Degree based Improved Hop Count Weighted Centroid Localization Algorithm

Rico Radeke[1] and Stefan Türk[2]

1 **Technische Universität Dresden, Chair for Telecommunications**
01069 Dresden, Mommsenstrasse 13, Germany
rico.radeke@tu-dresden.de

2 **Technische Universität Dresden, Chair for Telecommunications**
01069 Dresden, Mommsenstrasse 13, Germany
tuerk@ifn.et.tu-dresden.de

Abstract

Hop-count based weighted centroid localization is a simple and straightforward localization algorithm, which uses anchors with known positions and the hop count to these anchors to estimate the real position of nodes. Especially in sensor networks, where energy restrictions prevent more complex algorithms, this fast and simple algorithm can be used. Unfortunately the localization error of the algorithm can hinder the practical usage.

In this paper we will improve the weighted centroid algorithm for hop count based localization by adding the node degree on the paths to the referenced anchors into the weights. After an analysis to obtain theoretically optimal coefficients we will show by means of simulation that for longer hop counts to the anchors and areas with different node degrees the proposed ND-WCL algorithm outperforms the known hop count based weighted centroid localization algorithm.

Keywords and phrases Localization, Weighted Centroid

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.194

1 Introduction

Fast and adaptive algorithms for distributing messages are needed in ad hoc networks, as these may be mobile, fast changing or even partially disrupted[6]. Geographic Routing may be one of the enabling technologies to ensure communication in these networks [2][4]. Unfortunately Geographic Routing is based on the knowledge of real location information. GPS equipment is expensive as well as energy consuming and relies on line of sight to the GPS satellites. The Ad Hoc Positioning System [5] extended GPS on a hop by hop behavior for usage in networks where only a fraction of nodes have the capability to detect their real location. A novel approach was made in [7] to construct virtual coordinates and use these as base for geographic routing. The proposed algorithm managed to construct virtual coordinates without any knowledge of the real coordinates of the network nodes. But still all nodes need to have a logical position assigned to work with geographic routing.

Our paper will improve the known general centroid algorithm (CL) [3] as well as the hop count based weighted centroid algorithm (WCL) [1]. The aim is to improve the performance of the localization in terms of localization error without using decentralized or global information or introducing additional communication messages for gathering data. The proposed novel algorithm ND-WCL will use the average node degree on the shortest paths to the anchors in the network, which is an easily obtainable information.

The remainder of this paper is organized as follows: Section 2 analyses the theoretic connections between node degree, distance between node pairs and n-hop-neighborhoods in

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 194–199

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

ad-hoc networks with uniformly distributed nodes. Supported by simulations we obtain in section 3 approximations for the average distance between n-hop-neighbors, which are used in section 4 to define a novel node degree based weighted centroid algorithm for localization. Simulation in section 5 show the performance of the novel algorithm in relation to already known centroid algorithms.

2 Analysis of n-hop-Neighborhoods

For the analytic investigation we assume a static two dimensional plain scenario with N random uniformly distributed nodes which leads to a constant average node degree. Each node uses the same wireless communication module with a fixed unidirectional communication range. Nodes within the communication range R may communicate with each other and are denoted as 1-hop-neighbors. For the analytic investigation we place one node as central node (Node Zero) in the middle of the observance area.

Node density ND in the scenario is strongly linked to the node degree $deg(N)$, which is the number of neighbors.

$$ND = \frac{deg(N)}{\pi \cdot R^2} \tag{1}$$

Two nodes are denoted as n-hop-neighbors, if the shortest communication path, by means of shortest hop count, is equal to n.

In the following we observe the probability $P_n(d)$ of a random node pair with distance d to each other to be n-hop-neighbors. We also observe the mean distance $\bar{d}_n$ between n-hop-neighbors. As all nodes are independently and randomly placed with a uniform distribution the mean value of the distance of all possible n-hop-neighbors and the expected distance of randomly chosen n-hop-neighbors is the same. The communication range R will be set to 1 (distance unit) as simple scaling, resulting in a unit-disc-graph.

The probability $P_1(d)$ that two nodes with distance d to each other are 1-hop-neighbors is

$$P_1(d) = \begin{cases} 0 & \text{for } d \leq 0 \\ 1 & \text{for } 0 < d \leq 1 \\ 0 & \text{for } d > 1 \end{cases} \tag{2}$$

as two nodes are 1-hop-neighbors if and only if they have a positive distance less equal to the communication range.

The average distance $\bar{d}_1$ between 1-hop-neighbors can be computed and simplified as

$$\bar{d}_1 = \frac{1}{A} \int_A P_1(d) ddxdy = \frac{2}{3} \tag{3}$$

with A being the unit circle as all nodes in the unit circle around Node Zero are a 1-hop-neighbor of it. For 1-hop-neighbors $\bar{d}_1$ is independent of the node degree.

Two nodes with a distance d equal less to 1 are already 1-hop-neighbors. If the distance d is greater than 2 no common neighbor can be found for the node pair. Therefore only node pairs with a distance $1 < d \leq 2$ are potential 2-hop-neighbors. The probability $P_2(d)$ is equal to the probability to find a third node to establish a 2-hop-neighborhood. This third node must be placed in the intersection area A of the communication range circles of the first two nodes. This area A can be computed for communication range $r = 1$ as

$$A = 2 \arccos(\frac{d}{2}) - \frac{d}{2}\sqrt{4 - d^2} \tag{4}$$

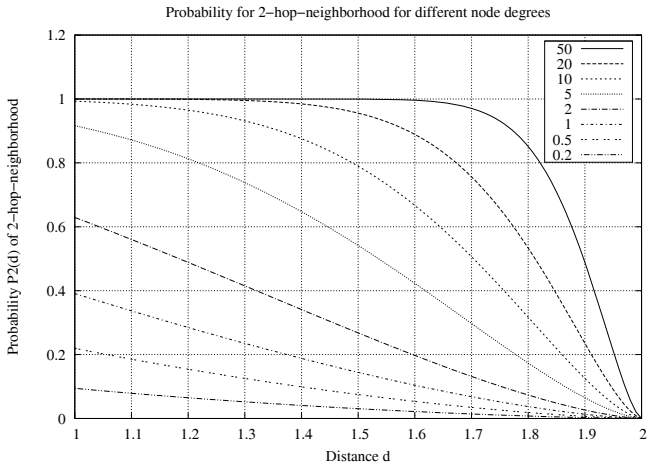


Figure 1 2-hop-neighbors

The probability $p_2(d)$ that one random but specific neighbor node is placed in the intersection of the two communication areas is

$$p_2(d) = \frac{A}{\pi r^2} \text{ for } 1 < d \leq 2 \tag{5}$$

and $P_2(d)$ can be computed as

$$P_2(d) = 1 - (1 - \frac{2\arccos(\frac{d}{2}) - \frac{d}{2}\sqrt{4-d^2}}{\pi})^{deg(N)} \tag{6}$$

The probability for two nodes with distance d to be 2-hop-neighbors is shown in Fig. 1 for different average node degrees. Note that even in scenarios with an average node degree less than 1 a small probability exists to find a third node for establishing a 2-hop-neighborhood. As expected a high node degree offers more possibilities to find a third node and an increased probability to find this third node even for node distances d close to the maximum of 2.

The average distance of 2-hop-neighbors can be computed using polar coordinates again

$$\bar{d}_2 = \frac{1}{A}\int_A P_2(d)ddxdy = \frac{2}{3}\int_{r=1}^{2}(1 - (1 - \frac{2\arccos(\frac{r}{2}) - \frac{r}{2}\sqrt{4-r^2}}{\pi})^{deg(N)} r^2 dr \tag{7}$$

which is unfortunately not independent of the node degree.

For node degrees close to infinity we may assume that all nodes within an annulus with a large radius $R = n$ and a smaller radius $r = n - 1$ are n-hop-neighbors if they are not placed on the inner circle. For example all nodes within the centered annulus $A_{1,2}$ with $r = 1$ and $R = 2$ are 2-hop-neighbors of the central node at $(0, 0)$. Therefore the probability $\widehat{P}_n(d)$ of 2 nodes with distance d to be n-hop-neighbors for high node degrees is

$$\widehat{P}_n(d) = \begin{cases} 0 & \text{for } d \leq n-1 \\ 1 & \text{for } n-1 < d \leq n \\ 0 & \text{for } d > n \end{cases} \tag{8}$$

The average node distance $\bar{d}_n$ for an almost infinite high node degree can be computed as

$$\bar{d}_n = \frac{2}{3}\frac{n^3 - (n-1)^3}{n^2 - (n-1)^2} \tag{9}$$

which equals $2/3$ for $n = 1$ as expected and converges to $n - 1/2$ for $n \to \infty$.

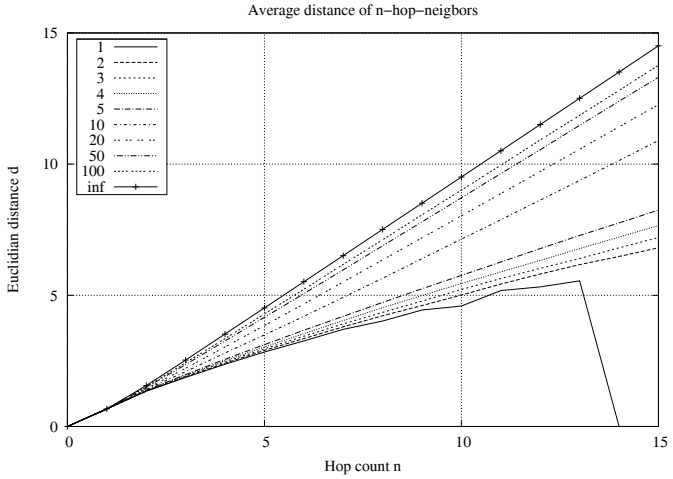


Figure 2 n-hop-neighbors

3 Simulation of n-hop-Neighborhoods

For higher n-hop-neighborhoods we used simulations to achieve information about the average distance between n-hop-neighbors. We simulated up to one million random scenarios for different node degrees. Therefore nodes were randomly placed with a uniform distribution around the Node Zero. The observance area was larger than the maximum observed hop count to prevent border effects. The average distance of n-hop-neighbors is shown in Figure 2 as well as the previously theoretically found boundary values for infinite node degrees. Note the insufficient simulation results for a node degree of 1, as with an average of just one neighbor per node, long paths between nodes are only to obtain by very long or rare event simulations.

To achieve one simple formula for the average distance of n-hop-neighbors we made linear approximations. The achieved linear coefficients can be also linear approximated for node degrees from 1 to 10 with an additional cut off at the theoretical values of node degrees towards infinity. This cut off is necessary as the linear approximation exceeds the theoretical bounds for higher node degrees. This approximation is used to achieve the average distance of n-hop-neighbors

$$\bar{d}_n(deg(n)) = (0.0391 \cdot deg(N) + 0.3338) * n + -0.1108 \cdot deg(N) + 0.9917 \tag{10}$$

for $n \geq 2$. For $n = 1$ we assume a node degree independent $\bar{d}_1 = \frac{2}{3}$.

4 Node Degree based Weighted Centroid Localization Algorithm

As base for our improvement we assume that a node N_i in a network, which wants to localize itself, has only chances to send messages to its direct neighbors. By flooding a localization request with a hop counter through the network all anchors are reached. These send back their position as well as the hop count from N_i to the anchor. After receiving this messages from the anchors, N_i may use different localization algorithms.

In our improved algorithm ND-WCL the average node degree $\bar{deg}(N)$ on the shortest path to an anchor is also send to node N_i. This information is easily obtainable, as all nodes know through neighbor detection and listening to hello messages it's own node degree, which is the number of direct neighbors. No additional message is needed here.

As first reference algorithms we used the centroid algorithm (CL) [3], taking the mean of the positions of the n anchors to estimate the own position. The second algorithm observed is the weighted centroid localization algorithm (WCL) [1], taking the weighted mean of the positions of the n anchors to estimate the own position. The weight for each anchor is computed as the reciprocal of the hop count to the anchor.

Our algorithm uses the reciprocal of the formerly computed average distances to the anchors based on hop count and node degree as weights w_i.

$$w_i = \frac{1}{\bar{d}_{HopCount_i}(\bar{deg}(n))} \tag{11}$$

$$Pos_{ND-WCL} = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \cdot Pos_{Anchor}(i) \tag{12}$$

This algorithm is easily implementable and does not need any further messages. It can be computed locally on the node to localize as only one additional step with a linear equation must be computed. The computation complexity is close to the normal hop count based weighted centroid algorithm.

5 Simulation of Localization

To compare the performance of the new algorithm with the two existing ones we used extensive simulations in MATLAB. Up to 1000 nodes were placed in a two dimensional square of 50x50m with 4 or 9 anchors. Simulation details are shown in the left part of Table 1 with the number of nodes, communication range and number of placed anchors. To have paths to anchor nodes with rather different node degrees, we shifted half of the nodes from one side of the simulation area to the other in the even numbered scenarios, producing unbalanced scenarios with a balancing of 3 to 1. As the anchors are placed in the corners of the simulation area and at least three anchors are taken for the centroid algorithm, at least one path to an anchor has a quite different node degree than the other two paths.

The main evaluation metric is the localization error LEr [1] as distance between the exact real position and the estimated position of a node.

We computed and compared for all non anchors in a scenario the localization error for centroid, weighted centroid and node degree based improved weighted centroid algorithm. As expected the weighted centroid algorithm outperformed the simple centroid algorithm tremendously. Therefore we only compare the two weighted centroid algorithms here.

The localization error was averaged for all non anchors in each randomly generated scenario. The minimum, maximum and mean improvement in percent over 1000 scenarios using the node degree approach is shown in Table 1. Negative values show that some randomly generated scenarios had a better average performance for the normal weighted centroid algorithm. Nevertheless in all scenarios the mean localization error is reduced using node degree based improved weighted centroid. It performs better with longer routes in scenarios V to X with shorter communication range and more nodes. It also performs better with less anchors to choose, resulting also in longer and more different long paths to anchors. The most increase in performance was obtained with the unbalanced scenarios leading to paths with rather different node degrees.

The overall best performance of the node degree based improved weighted centroid localization algorithm was obtained in scenario X with the longest paths to the four anchors and an unbalanced scenario.

Table 1 Simulation Scenarios and Localization Error Improvement

Scenario	No. of Nodes N	Range R	No. of Anchor Nodes	Balancing	*LEr* Improvement in %		
					Min	Max	Mean
I	100	10m	4	none	-3.92	6.39	1.38
II	100	10m	4	3:1	-8.37	15.04	4.45
III	100	10m	9	none	-2.43	3.16	0.25
IV	100	10m	9	3:1	-5,15	7,45	1,06
V	400	5m	4	none	-1,49	5,20	2,55
VI	400	5m	4	3:1	-0,11	8,78	3,05
VII	400	5m	9	none	0,19	5,11	2,76
VIII	400	5m	9	3:1	1,51	8,57	4,07
IX	1000	3m	4	none	0.97	3.92	2.91
X	1000	3m	4	3:1	5,43	14,87	8,64

6 Conclusions

In this paper we improved the weighted centroid algorithm for hop count based localization. With the small functional addition of counting the node degree on the paths to the referenced anchors we could improve the performance of hop count based localization. Simulations showed that for longer hop counts to the anchors and areas with different node degrees the proposed algorithm outperforms the known hop count based weighted localization algorithm.

References

1 J. Blumenthal, R. Grossmann, F. Golatowski, and D. Timmermann. Weighted centroid localization in Zigbee-based sensor networks. In *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*, pages 1–6. IEEE, 2008.

2 P. Bose, P. Morin, I. Stojmenović, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

3 N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE personal communications*, 7(5):28–34, 2000.

4 B. Karp and H.T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

5 D. Nicolescu and B. Nath. Ad-hoc positioning systems (APS). In *Proceedings of IEEE GlobeCom*, volume 1, pages 25–29, 2001.

6 R. Radeke, D. Marandin, F.J. Claudios, P. Todorova, and S. Tomic. On reconfiguration in case of node mobility in clustered wireless sensor networks. *Wireless Communications, IEEE*, 15(6):47–53, 2009.

7 A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 96–108. ACM, 2003.

Industry Papers

Automated generic integration of flight logbook data into aircraft maintenance systems*

Oliver Hunte[1], Carsten Kleiner[1], Uwe Koch[1], Arne Koschel[1], Björn Koschel[2], and Stefan Nitz[1]

1 **Fachhochschule Hannover, Fakultät IV (Wirtschaft und Informatik)**
Ricklinger Stadtweg 120, 30459 Hannover, Germany
carsten.kleiner@fh-hannover.de, arne.koschel@fh-hannover.de

2 **edatasystems GmbH, Am Bugapark 60, 45899 Gelsenkirchen, Germany**
bjoern.koschel@edatasystems.de

The automated transfer of flight logbook information from aircrafts into aircraft maintenance systems leads to reduced ground and maintenance time and is thus desirable from an economical point of view. Until recently, flight logbooks have not been managed electronically in aircrafts or at least the data transfer from aircraft to ground maintenance system has been executed manually. Latest aircraft types such as the Airbus A380 or the Boeing 787 do support an electronic logbook and thus make an automated transfer possible. A generic flight logbook transfer system must deal with different data formats on the input side – due to different aircraft makes and models – as well as different, distributed aircraft maintenance systems for different airlines as aircraft operators. This article contributes the concept and top level distributed system architecture of such a generic system for automated flight log data transfer. It has been developed within a joint industry and applied research project. The architecture has already been successfully evaluated in a prototypical implementation.

1998 ACM Subject Classification C.2.4 Distributed Systems, H.2.8 Database Applications, J.2 Physical Sciences and Engineering

Keywords and phrases system integration, data mapping, XML, aerospace engineering, generic interface, configurable mapping

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.201

1 Introduction and Problem Description

Ground and maintenance time is costly for aircraft carriers, thus they try to minimize them for economical reasons. Today's still mostly manual transfer of flight logbook data from an aircraft into the operator's maintenance systems should be automated to get closer to achieving this goal. This will eventually reduce information transfer time and is also likely to be less error prone. Thus in total it should result in reduced maintenance time and cost.

A generic automated flight log data transfer system needs to support the differently structured flight log information from different aircraft types and manufacturers on one side. On the other side different aircraft maintenance systems used by different operators have to be supported. Technically, all these systems are very likely distributed, even though typically in a common network within a single organization. Moreover, fault tolerance and

* In cooperation with Lufthansa Technik AG. The project is part of the Aviation Cluster Hamburg Metropolitan Region's Leading-Edge Cluster Strategy and is sponsored by the Federal Ministry of Education and Research.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 201–204

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(transactional) persistence to prevent data loss are required. Performance demands however, are not very critical due to a limited amount of log data per flight in practical applications.

To support those requirements, a *generic* transfer system for flight logbooks into maintenance systems needs to be designed and implemented. In a joint industry and research cooperation (*Verbundprojekt*) the eLog system has been designed and prototypically implemented by the University of Applied Sciences Hannover in cooperation with Lufthansa Technik AG and edatasystems GmbH. Technically this system supports different – currently XML-based, but with heterogeneous XML schemata – versions of different aircraft flight log systems on the *input* side. On the *output* side different airline systems are supported which may have different data models. As an example a relational DBMS with tables is used, that map (almost) 1:1 to the data structures of the *output* system. The *output* system is Lufthansa's core aircraft maintenance system.

The resulting eLog system offers a generic distributed system architecture for the integration and mapping of the mentioned different XML-based flight log input data formats to different output aircraft maintenance systems. The mapping itself is flexibly configurable as well. Initial tests of the prototypical implementation already validated the practical usefulness of the concept. Only very few logbook data transfer systems exist. Those that do exist are flight operator internal and/or aircraft type and maintenance system specific.

Although concepts and approaches for application/data integration in general of course do exist ([4, 7, 6, 5, 2]) their application to flight logbook data is a novelty. To the best of our knowledge a *generic* flight logbook data transfer system has not been implemented before and is thus the key contribution of our work.

The remainder of this article gives a high level eLog system overview and discusses some technical aspects in detail. Due to space constraints a discussion of related work is omitted. The paper ends with a conclusion and outlook.

2 eLog: System Overview

The designed generic flight log data transfer system is based on ideas frequently found in extract transform load (ETL) processes in data warehouse systems [5]. Note though that the implementation itself is not based on data warehouses but rather uses a transactional DBMS-based approach [3]. Next we explain the data flow within the eLog system followed by some selected eLog design decisions.

2.1 Data flow within eLog

The data flow within eLog follows several steps to map XML input data to arbitrary output aircraft maintenance systems on the conceptual level. Figure 1 shows the high level flow of input data from the flight logbook system until it is procured to the maintenance system.

An input reader component – where different implementations for different source data formats may exist – uses in step 1 polling to check whether new XML files have been delivered by aircrafts as electronic logbooks. Polling is implemented by frequently checking a predefined directory on a dedicated server for newly added files. The data is validated against the XML schema (step 2) of the particular aircraft's electronic logbook format, e.g., against different XML schema versions like those that are defined in [1]. If the data is formally valid, it is transferred into a buffer database (step 3), where it is stored in its original format in an *XML-type* attribute. Else an error handling takes place. As long as aircrafts provide source data in XML format a single database schema is sufficient for the buffer database.

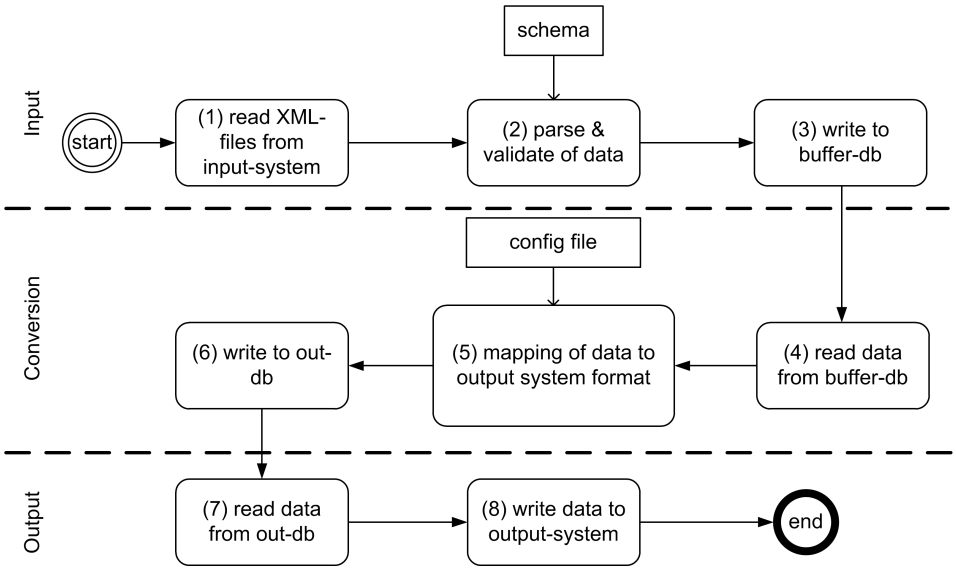


Figure 1 Generic eLog data flow

Again using polling, a mapping component checks for newly arrived source data (step 4) in the buffer database. It utilizes a flexibly configurable sequence of conversion functions to map the input data to a specific output target system (step 5). The output data is stored in a database (Out DB) again (step 6). It closely resembles the data structure of the relevant airline operator's maintenance system.

Consequently for each maintenance system there will be an individual schema in the output database. Options in the mapping configuration file include checks for already present dependent source data, flexible mapping to one or more new entities in the Out DB for every input entity as well as features to update already existing entities in the Out DB.

Eventually another upload component transfers data from the Out DB (step 7) into the airline maintenance system (step 8) whenever a meaningful entity of the maintenance system has been assembled in the Out DB from one or more input entities. Amongst other options, the airline maintenance system used here provides a Web services interface for access.

2.2 Selected eLog design decisions

To give an idea how the overall eLog architecture and design was established, we explain selected eLog design decisions:

- Error checking and handling is added to all components of the system. For example, schema errors in input documents are logged in the underlying database and an appropriate message is sent to the eLog system operator.
- Throughout the conversion steps DB transactions are utilized to ensure data consistency. In combination with operating system based fault tolerance (automated process restarts), a high degree of fault tolerance of the overall system is thus achieved.
- Technically the system is designed according to the ETL paradigm and it is implemented with different Java processes which together provide the tasks from figure 1. They are combined with a relational DBMS, that also allows for XML data storage (Oracle).

3 Conclusion and Outlook

3.1 Conclusion

So far the concept and prototypical implementation have proven to be very promising. The implementation already covers a single exemplary input and also output system. It includes entities that require almost all possible mapping options between input and output systems. The mapping configuration is defined in an XML file which is of a proprietary structure. The implementation employs several XPATH expressions to navigate in input files in order to select the information to be mapped. The output DB resembles the simple relational structure of most airline maintenance systems.

Both mapping configuration as well as input and output data formats are sufficiently generic in order for the system to be easily adjusted to specific data formats. The overall modular design of the system by decoupling input and output system leads to a highly scalable overall architecture.

3.2 Outlook

By now only a brief evaluation of the first eLog prototype took place. This evaluation mostly included general tests of the eLog system architecture (based on the prototype), some XML database performance tests based on realistic data volume assumptions, tests with different generated flight log data sets, and some error handling / restart experiments. All conducted tests with the final eLog prototype showed very promising results for further development.

Future work includes evaluation of eLog in a production environment. Since only selected representative entity types from [1] are currently with eLog processable, the remaining ones have to be added. Moreover, additional different aircraft log file source systems and target airline maintenance systems shall be connected to eLog.

The former will provide additional proof for the scalability and reliability aspects. The latter will lead to the definition of a universal XML schema for the mapping configuration in order to overcome the currently proprietary XML format used.

In summary the generic integration of flight logbooks that has been implemented in this project shows the potential for reduced transfer times of maintenance information coupled with increased correctness when compared to the current manual process. This will eventually lead to reduced maintenance times for aircrafts and thus increase profitability of the airline.

References

1 Air Transport Association of America, Inc. (ATA). *Industry Standard XML Schema for the Exchange of Electronic Logbook Data*, 1.2 edition, May 2008.

2 Jürgen Dunkel, Andreas Eberhart, Stefan Fischer, Carsten Kleiner, and Arne Koschel. *Systemarchitekturen für Verteilte Anwendungen*. Hanser, München, 2008.

3 Ramez Elmasri and Shamkant Navathe. *Fundamentals of Database Systems (5th Edition)*. Addison Wesley, U.S.A, 2006.

4 Gregor Hohpe and Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Longman, Boston, MA, USA, 2003.

5 W. H. Inmon. *Building the Data Warehouse*. Wiley, U.S.A, 2005.

6 Dirk Krafzig, Karl Banke, and Dirk Slama. *Enterprise SOA: Service Oriented Architecture Best Practices*. Prentice Hall, Upper Saddle River, NJ, 2005.

7 David S. Linthicum. *Enterprise Application Integration*. Addison-Wesley Professional, 1999.

Alister 2.0 - Programmable Logic Controllers in Railway Interlocking Systems for Regional Lines of the DB Netze AG

Reiner Saykowski, Elferik Schultz, and Joachim Bleidiessel

Funkwerk Information Technologies GmbH
Edisonstraße 3, 24145 Kiel, Germany
{reiner.saykowski, elferik.schultz, joachim.bleidiessel}@funkwerk-it.com
http://www.funkwerk-it.com

Abstract

Railway interlockings are dominated by highly proprietary systems. We present the development project Alister 2.0 – an interlocking system based on industry-proven standard components: Safety PLCs in distributed nodes communicate over safe network protocols. This enables a highly productive and highly maintainable fail-safe interlocking system for centralised traffic control.

Keywords and phrases electronic interlocking, PLC, regional lines, ESTW-R

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.205

1 Introduction

About 1000 railway interlocking systems in Germany reach the end of their life-cycle within the next years, most of them controlling regional lines. Furthermore, the DB Netze AG as operator faces increasing cost pressure and wants to raise its competitiveness. Hence, it is the defined aim to perform the modernisation of these interlocking systems with cost-effective, standardised and modular components, while still meeting the requirements and the railway standard CENELEC EN 50126[1], EN 50128[2] and EN 50129[3]. Among other things these standards impose a development process and safety requirements for the hard- and software. Consequently the Functional Specification ESTW-R (Electronic Interlocking for Regional Lines) was published, which encouraged a strongly modularised structure that consists of industry-proven common off-the-shelf products (COTS). This implicates a new system architecture and at the same time offers the chance of opening the separated market to new competitors. In summer 2006 Funkwerk Information Technologies GmbH was commissioned to equip the regional line connecting the cities of Kiel and Flensburg with a new interlocking system. With this innovative approach, the development project entered the phase of safety approval at the end of 2009. This was the first time COTS products were deployed in a interlocking system using a distributed architecture.

2 System Concept and Architecture

Although the regional line between Kiel and Flensburg is more than 50km long, it is monitored and controlled by just one control centre. This overall architecture is depicted in Figure 1 and will be explained now. The distributed computing nodes at the stations are called local ESTW-R. They are redundantly connected to the control centre via standard ethernet connection over optical fibre. The signalman operates the line from the control centre with one technical procedure-protected workstation running under Linux. Supported by train describer and train routing functionality, the signalman sends commands to set the

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 205–207

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

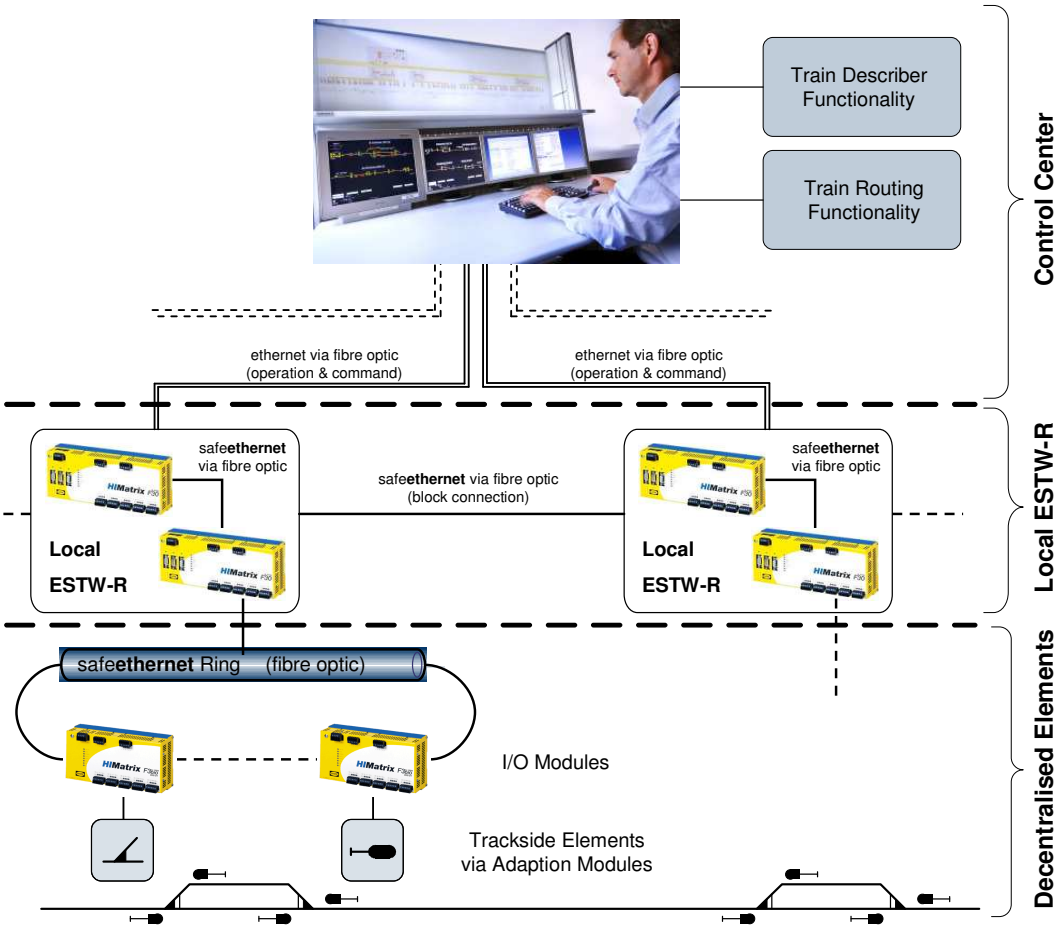


Figure 1 Network architecture of the development project Alister 2.0

train routes correctly. In the local ESTW-R computation is done by industry-proven safety Programmable Logic Controllers (PLC). PLCs follow a model of successive cycles, where each cycle consists of three sequential phases: read input, compute, write output. The PLCs communicate with so called decentralised I/O-modules via TCP/IP protocol over *safeethernet*. Safeethernet is a certified safety protocol, based on standard Ethernet technology IEEE 802.3. It is developed by the company of HIMA[1]. The I/O-modules drive the adaption modules, which in the end control the trackside elements, e.g. points, signals or level crossings.

2.1 Local ESTW-R — Interlocking logic with safety PLCs

A single local ESTW-R is composed of two safety PLCs, whereby every PLC is internally based on a 2oo2-architecture ("2-out-of-2"). Therefore, if one failure is detected by one of the PLCs, the PLCs will switch to a safe state – this is an integral requirement for safe railway operations. The PLCs are programmed in Function Block Diagram language (FBD), corresponding to the IEC 61131-3 Standard. IEC 61131[4] describes the standard use of PLCs, including general information, equipment requirements, communication and

[1] safeethernet is a registered trademark of HIMA Paul Hildebrand GmbH + Co KG, http://www.hima.de

standardised programming languages. Furthermore, to meet the requirements of the railway CENELEC standard only a subset of the FBD language is used, e.g. to avoid infinite loops. To set up train routes from one local ESTW-R to the next local ESTW-R, both communicate over safeethernet. Thus, data corruption, loss of data and data sequencing problems in communication between the nodes are detected. The decentralised I/O-Modules, which get their commands over safeethernet from the PLCs, are connected in a ring-structure, due to availability-considerations. The time to switch the direction of access in the ring-structure after a problem occurred is less than the minimum cycle-time of a PLC. Therefore, a switch in the flow of information will not interrupt the system. The I/O-modules drive the adaption modules. In turn these are not limited to actuate typically trackside elements. Because of their configurable and open communication interface, they can also actuate arbitrary digital in- and outputs, like axle counters.

2.2 Technically procedure-protected workstation

The complete route between Kiel and Flensburg is controlled by the technically procedure-protected workstation. At each point in time it presents an up-to-date process image on the workstation of the signalman. The integrity of the process image is secured by a patented image signal comparing system. The incoming image signal is received via two independent channels and compared twice a second. Single failures in communication or single components will be detected by the system and lead to a safe state. In this state only standard operations are allowed.

3 Outlook

With its maximum number of 10 distributed interlocking nodes already today the ESTW-R Alister 2.0 is based on industry-proven, highly maintainable COTS components and industry protocols. Hence, the railway interlocking market has been opened for innovation and sustainable components which are already proven and tested in automation and process industry. A new trend has started that will lead to further cost reductions and great efficiency enhancements.

References

1 European Committee for Electrotechnical Standardization (CENELEC). *EN 50126 – Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*. 1999

2 European Committee for Electrotechnical Standardization (CENELEC). *EN 50128 – Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems*. 2001

3 European Committee for Electrotechnical Standardization (CENELEC). *EN 50129 – Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling*. 2003

4 International Electrotechnical Commission (IEC). *IEC 61131 – Programmable controllers*. 2000-2007

Browser as a Service (BaaS): Security and Performance Enhancements for the Rich Web

Nils Gruschka[1] **and Luigi Lo Iacono**[2]

1 **NEC Laboratories Europe**
Heidelberg, Germany
nils.gruschka@neclab.eu

2 **European University of Applied Sciences (EUFH)**
Brühl, Germany
l.lo_iacono@eufh.de

Abstract

This paper introduces an architectural approach to access the Web via a virtual Web browser executed within a secure Cloud environment.

Keywords and phrases Web, Security, Performance, Cloud, SaaS

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.208

1 Introduction

The Web has become an indispensable prerequisite of everyday live and the Web browser is the most used application on a variety of distinct devices. The content delivered by the Web has changed drastically from static pages to media-rich and interactive Web applications offering nearly the same functionality as native applications, a trend which is further pushed by the Cloud and more specifically the Cloud's SaaS layer. In the light of this development, security and performance of Web browsing has become a crucial issue.

From a security perspective, Web browsers superseded other attack targets like email [1] and are more and more the gate for installing malware on victims computer [2]. No matter if malicious or reputable Web sites, both may contain manipulated active content. In the most dangerous form—the so-called drive-by downloads—the malware is downloaded and installed to the visiting client without knowledge and notice of the user.

Another problem of Web browsing is the "memory" of the browser. Browser cookies and histories can lead to privacy breaches and especially becomes a problem if recognition can be performed between different Web sites. A well known attack of this type is called *history stealing* allowing a Web site to access the user's history using JavaScript and CSS [3]. In an advanced form this attack even allows to identify a user—i.e. learning his name, address etc.—by analyzing the sites a user has visited [5]. And even if a user disables or cleans his cookies and browser history, recent published techniques are still able to store re-identification data [4].

Further, performance issues arise from the increased complexity of Web applications running inside the browser. Nearly all functionalities of native applications can nowadays be found in the browser part of Web applications. These programs—most commonly written in JavaScript—require high memory and processing resources. This makes JavaScript engines and specifically their efficiency the most promoted feature for new browsers or browser versions. In addition to JavaScript also CSS rendering or DOM access are resource consuming and can become an execution bottleneck [6].

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 208–210

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

This paper contributes and discusses an architectural approach towards faster and more secure Web browsing taking into account contemporary attack vectors and adhering to the demands of the SaaS era.

2 Browser as a Service

The basic underlying idea of the proposed architecture is to provide access to a remote Browser as a Service (BaaS, see Figure 1). The remote browser is executed in a secure cloud environment for a very limited lifetime. The interaction with the remote browser takes place with a locally installed browser. In contrast to the current "standard" scenario, the locally installed browser does not execute any active content, besides the one required to capture user inputs and events as well as to display the graphical output the remote browser has rendered and sent.

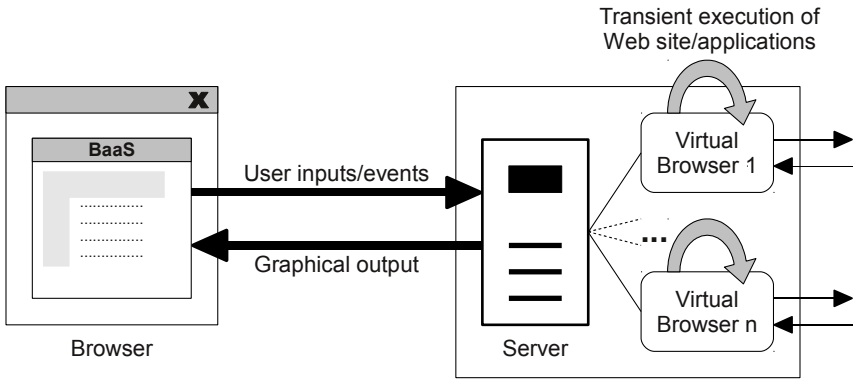


Figure 1 BaaS Architecture

The server-side runs a virtual machine containing a Web browser instance denoted as *Virtual Browser*. Such an instance runs an up-to-date Web browser engine which includes all state-of-the-art extensions required to consume services and applications in the Web. The lifetime of the instance is transient. It does not save any state and is only available during the actual session of the user. Henceforth, any malicious code eventually installed to the virtual browser while surfing the Web with this particular instance will be destroyed as well.

The client-side still requires a Web browser to access and use a virtual browser. The virtual browser appears within the local browser's output area as a Web application (browser in a browser). When the user interacts with the virtual browser within the local browser, the inputs and events are captured by the local browser and then send to the virtual browser. It then triggers the related actions and renders the associated output accordingly. This includes the execution of active content which is performed exclusively within an isolated environment on the remote server. Thus, a possible infection will harm the proxying virtual browser instance, but not the acting end-user client, as the virtual browser does only transmit the rendered output to the client.

Additionally, such an approach has positive effects on the performance of executing the client-side portion of Web applications. Since this part is increasing steadily—especially in SaaS-based applications—the demands in terms of local computing resources are increasing in the same way. This is mainly caused by the trend to offer Web applications with a similar user-experience like in standalone programs. The proposed BaaS approach off-loads the execution of complex and large JavaScript programs to the virtual browser, which enables

the usage of resource-demanding Web applications also for restricted devices such as thin clients, netbooks and smart phones.

3 Conclusion and Open Issues

Performance of complex Web applications, privacy breaches and malware vulnerabilities are major problems of today's Web usage. The presented idea for a virtual browser running "in the cloud" has the potential to provide a solution for these issues. However, a number of issues and questions have to be solved for realizing BaaS. The main problem is the protocol for accessing the virtual browser. Standard remote desktop protocol (like RDP or RFB) might be sufficient for office applications or online shopping but probably not for video applications. In any case, parameters like responsiveness, rendering quality and network traffic must be studied intensively.

References

1 David Barroso. Botnets – the silent threat. *ENISA Position Paper*, 2007.

2 Marc Fossi, Dean Turner, Eric Johnson, Trevor Mack, Téo Adams, Joseph Blackbird, Stephen Entwisle, Brent Graveland, David McKinney, Joanne Mulcahy, and Candid Wueest. Symantec global internet security threat report: Trends for 2009. XV, April 2010.

3 Artur Janc and Lukasz Olejnik. Feasibility and real-world implications of web browser history detection. In *W2SP 2010: Web 2.0 Security & Privacy*, 2010.

4 Samy Kamkar. evercookie – never forget. `http://samy.pl/evercookie/`, 2010.

5 Gilbert Wondracek, Thorsten Holz, Engin Kirda, and Christopher Kruegel. A practical attack to de-anonymize social network users. In *IEEE Symposium on Security and Privacy*, 2010.

6 Kaimin Zhang, Lu Wang, Aimin Pan, and Bin Benjamin Zhu. Smart caching for web browsers. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 491–500, New York, NY, USA, 2010. ACM.

Model Driven Development of Distributed Business Applications

Wolfgang Goerigk

b+m Informatik AG
Rotenhofer Weg 20, D-24109 Melsdorf, Germany
wolfgang.goerigk@bmiag.de

Abstract

The present paper presents a model driven generative approach to the design and implementation of destributed business applications, which consequently and systematically implements many years of MDSD experience for the software engineering of large application development projects in an industrial context.

Keywords and phrases MDSD, Software Architecture, Modelling, Code Generation

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.211

1 Introduction

MDSD (Model-Driven Software Development) is a generic term covering methods and techniques used to automatically generate executable software or other software related artefacts from formal models [4].

The MDSD platform b+m gear Java (gear) is the result of many years of experience with enterprise application development using various MDSD tool chains. gear supports the classical 3 tier architectural layers, *i.e.* entity, service and front end, and also a workflow partition similar to the BPMN (Business Process Modeling Notation). gear focusses on business software and in particular on client/server applications, and it is designed to support several non functional and technical requirements like *e.g.* safe distributed transaction handling or business driven historization. b+m gear Java is based on the Eclipse Modeling Project [2] and the well known generator framework openArchitectureware [3], which has originally been initiated and founded by the b+m Informatik AG. Since 2003 it is an open source framework and it is now part of the Eclipse Modeling Project.

2 Model Driven Development Using b+m gear Java

A comprehensive description of MDSD can be found in [4]. In the talk we will discuss the model driven software engineering approch and its maturity and state of the art in more detail. In the paper we want to focus on modeling and some technical advantages of the architecture centric generative software development process. Figure 1 depicts the so far supported DSL partitions and underlying platforms of gear:

Entity is used to model the persistence layer. OR mappers (like *e.g.* Hibernate) are used to generate database schemata and access code. The entity partition declaratively supports historization (cf. below).

Service is used to model business services and entity access. The service partition guarantees safe distributed transactions and uses the Java Transaction API (cf. below).

Frontend is used to model screen flows, which are coupled synchronously or asynchronously. From screen flow models, JavaServer Faces (JSF) and Spring WebFlows are generated.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 211–213

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Workflow is used for business service orchestration. It also organizes persistent tasks and their user and rôle assignments. For Workflow, the underlying jBPM platform is supplemented by task management, distribution strategies and enhanced by concepts like delegation, escalation, resubmission, task forward.

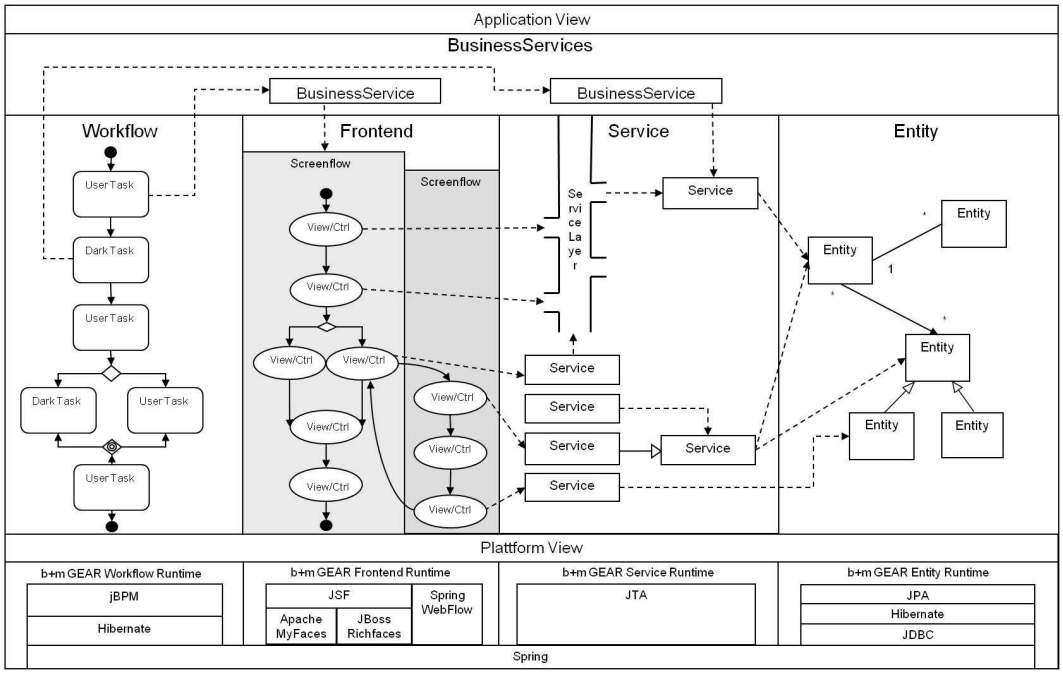


Figure 1 DSL partition map and underlying platforms in gear

Workflow and its formal and orthogonal modeling and generative support, as well as of business rules and underlying rule engines, become increasingly important. gear supports workflow. The present development for b+m gear Java aims at the BPMN 2 standard [1], and the integration of a rule engine is planned in the future as well. Modeling is supported by grafical editors within the Eclipse IDE; the syntax resembles UML (Unified Modelking Language). However, for practical and pragmatical reasons we prefer a more light weight approach using specifically tailored domain specific languages.

We want to illustrate technical advantages of the MDSD approch using some examples: **Historization** is often used in order to guarantee, that software systems *e.g.* in the finance or insurance domain are audit-proof – often required by law. Every transaction has to be plotted, which requires additional history tables for entities, and complicated write access has to be weaved horizontally into the entire code. In gear, historization is a simple annotation of entity definitions, and weaved generatively. An equally prominent example is **transaction safety**, necessary for distributed client/server applications with persistent data. In gear, the service partition uses the Java Transaction API. In both cases, sophisticated and concise architectural enhancements for DSL and generator support the mastery of potentially complex horizontally weaved code aspects. Quality and implementation efficiency are improved.

References

1 Business Process Model and Notation (BPMN), Version 2.0. http://www.omg.org/spec/

BPMN/2.0, 2010.

2 Eclipse Modeling Project. http://www.eclipse.org/modeling/, 2010.

3 openarchitectureware.org. http://www.openarchitectureware.org/, 2010.

4 T. Stahl, M. Völter, S. Efftinge, and A. Haase. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. dpunkt.verlag, 2 edition, 2007.

Summaries of Prize-Winning Theses

A Mobility Model for the Realistic Simulation of Social Context

Daniel Fischer

SAP AG, Germany
daniel.fischer@sap.com

Abstract

Simulation is a fundamental means for evaluating mobile applications based on ad-hoc networks. In recent years, the new breed of social mobility models (SMMs) has risen. Contrary to most classical mobility models, SMMs model the *social aspects of human mobility*, i.e. which users meet, when and how often. Such information is indispensable for the simulation of a wide range of socially-aware communication protocols mostly based on delay-tolerant networks, including opportunistic ad-hoc routing and data dissemination systems. Each SMM needs a model of the relations between a set of relevant people (called social network model – SNM) in order to simulate their mobility. Existing SMMs lack flexibility since each of them is implicitly restricted to a specific, simplifying SNM.

We present GeSoMo (General Social Mobility Model), a new SMM that separates the core mobility model from the structural description of the social network underlying the simulation. This simple and elegant design principle gives GeSoMo generalizing power: Arbitrary existing and future SNMs can be used without changing GeSoMo itself. Our evaluation results show that GeSoMo produces simulations that are coherent with a broad range of empirical data describing real-world human social behavior and mobility.

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.215

1 Introduction

In recent years, powerful mobile devices with short-range wireless network interfaces have become widely used that leverage the idea of mobile ad-hoc networks. Connectivity between mobile nodes is created sporadically and highly depends on the movement of the users carrying those devices. This human movement is strongly influenced by social relations [13, 16]: We do not meet other people on a purely random basis; instead, we meet some people more frequently and regularly than others, e.g. friends, colleagues, family. We call the network of social relationships of a larger group of people a *social network*.

Since the social network directly influences the mobility of people (and therefore those peoples' devices), many protocols and distributed algorithms, mostly in the area of opportunistic networks, exploit the structural properties of this underlying social network [2, 5, 15, 16]. To evaluate such approaches, several social mobility models (SMMs) have been proposed recently [3, 8, 11, 17]. The general task of a SMM is to simulate the encounters between mobile users such that they meet according to their social relationships. A SMM is always (possibly implicitly) based on a *social network model* (SNM) defining the structure of the relations between mobile users. The existing SMMs, however, have drawbacks. Each of them is tightly integrated with a specific SNM. Thus, by selecting an SMM, the SNM is fixed and cannot be changed. This is disadvantageous since there are a number of competing SNMs, and the selection of the right SNM is scenario-dependent and can be subject to debate. Furthermore, the used SNMs are quite simplistic which can have significant consequences on the evaluation results of applications simulated on top of it. SNMs with more realistic properties (e.g. [19]) are not used by existing SMMs. Also note that social network research is still a young discipline such that the SNMs are not yet established and improved models will be

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 215–220

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

proposed in the future. Finally, none of the existing SMMs models group mobility, i.e. the fact that humans tend to move in groups because of their social relations.

This paper provides an overview of our work in [6]. We introduce *GeSoMo* (General Social Mobility Model) [7], a new SMM that generalizes a number of existing models by separating the *social mobility model* and the *social network model* on which the SMM is based. GeSoMo receives a social network as input and creates a mobility pattern simulating a series of encounters between the users in this network. Thus, arbitrarily structured social networks can be used without changing the simulation model. Additionally, a number of parameters may be used for tuning GeSoMo to different scenarios. We show that GeSoMo produces realistic mobility patterns by comparing its characteristics to real measurements of human mobility.

2 Spatio-temporal Characteristics of Human Mobility

Apart from the fact that human mobility is influenced by the underlying social network, the investigation of real human mobility traces has shown several omnipresent spatio-temporal characteristics that a realistic SMM should capture:

- Inter-contact times, i.e. the time intervals between two successive contacts (connectivity) of two nodes, are of particular interest since they strongly influence the speed of relaying information in the social network and, thus, in any communication network resulting from it. Independent of the investigated trace, the **inter-contact distribution** is characterized by a **dichotomy**: There is a power-law up to an inter-contact time of about half a day, thereafter followed by an exponential decay [4, 14].
- The probability that individuals return to previously visited locations after a certain time t is characterized by distinct peaks at multiples of a day [9]. This **temporal regularity** confirms the intuition behind human habitual behavior.
- **Spatial Regularity**: Individuals have a strong preference for a small number of locations while visiting all other locations only with low probability [9].
- Frequently, humans do not move independently of each other from location to location, but rather move as a group. **Group mobility** is a realistic human characteristic that is already exploited by existing mobile applications [18], e.g. by identifying groups and using their combined resources.

3 The General Social Mobility Model

In this section, we first introduce our idea of how the simulated mobility should conform to the underlying social network. Subsequently, we give a brief overview of the concepts to realize a mobility model which both exhibits a high conformance and realistic spatio-temporal characteristics.

3.1 Conformance

The general idea behind GeSoMo is show in Figure 1. It takes an arbitrary social network as input. A social network is essentially a graph where nodes represent people and weighted edges represent social relationships. The weight is an indicator of the strength of the social relation. Two people having a social relation are called *social acquaintances*. Social networks can be generated by a number of social network models (SNMs), e.g. [10, 19, 20] describing different typical structural properties (e.g. scale-free node degree, high clustering, ...). GeSoMo then translates the static social network into realistic spatio-temporal mobility characteristics described by a mobility trace. This trace can be used as input to a simulation tool such as ns2 [1]. Such simulation tools determine at which time two nodes have connectivity resulting in a connectivity graph. At the moment, we

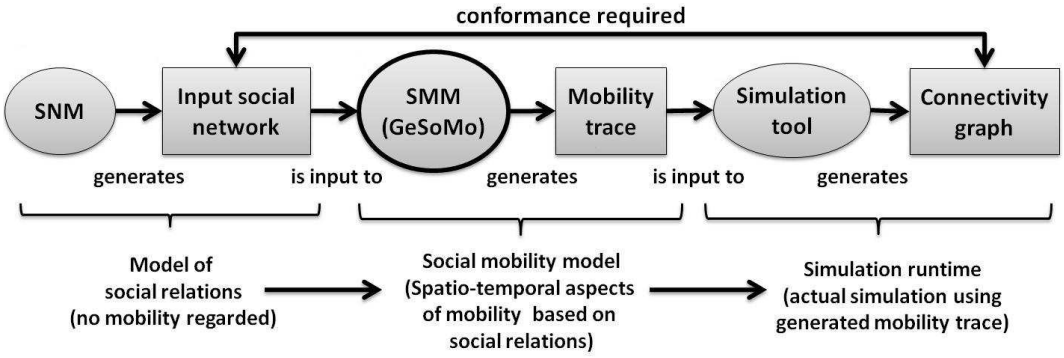


Figure 1 Dependencies between social network and simulated mobility

know that important structural properties of the input social network are mapped to the connectivity graph [13, 16] and the frequency and duration of contacts between social acquaintances is typically an order of magnitude larger than unrelated node pairs [16]. However, so far, social studies do not show how exactly the strength of social relations relate to the number of contacts (i.e. times of connectivity) between nodes. Hence, we define **conformance** as follows: *The number of contacts between nodes should be proportional to the strength of the relationship*. Note, however, that this interpretation can be adjusted in an arbitrary way, without changing GeSoMo, as soon as corresponding future results in social studies are known (for details, see [6]).

3.2 Outline of the General Concepts

For each node in the input social network, GeSoMo simulates a *mobile node* (or short: node) assuming that the corresponding user carries a single mobile device. Social interactions take place at so-called *anchors* which are an abstract representation of real locations (e.g. a bus stop or a coffee shop). Nodes move between anchors, and if a node reaches an anchor, it stays there for a specific duration, before it starts to move towards the next anchor. The movement between anchors can be performed by several nodes together in order to simulate **group mobility**. GeSoMo allows to configure the frequency of such group movements.

The probability that a node chooses a certain anchor as its next desination is proportional to its *anchor attraction* which consists of three additive components: First, anchors themselves exert a time-varying attraction on nodes: Nodes are attracted to a subset of anchors leading to a **spatial regularity** and this attraction is periodic which creates a **temporal regularity**. Second, nodes are also attracted by other nodes currently traveling to or residing at certain anchors. This node-based attraction depends on the strengths of the social relations in order to yield frequent meetings between social acquaintances. Third, in cases where nodes do not have a social relation with each other, it is necessary that nodes are *repulsed* (i.e. negatively attracted) by each other. Both node-based types of attractions (positive and negative) are necessary in order to yield a **high conformance**.

Influences such as different travel-times between anchors may lead to less or more meetings between some pairs of social acquaintances than required by the conformance criterion. To achieve strong conformance independent of such disturbances, we propose a feedback control mechanism: If too many meetings between two nodes have been produced, their social attraction is decreased. Conversely, if too few meetings have been produced, their social attraction is increased. Hence, after the adjustment, less (or more) meetings between them will occur, such that GeSoMo yields a **robust conformance** despite physical effects such as heterogeneous speed/anchor distributions, obstacles, or streets.

SNM	Caveman	Toivonen	Holme-Kim
Percentage Matched	99.91%	99.89%	99.90%
95% Conf. Int.	0.15%	0.11%	0.11%

Table 1 Results of the conformance metric (standard parameters)

4 Evaluation Results

In this section, we show that GeSoMo conforms to arbitrary structural properties of the input social network with high accuracy and still produces realistic spatio-temporal properties. We show the latter by evaluating several known omnipresent characteristics of human behavior (cf. Section 2) and comparing them to results of real measurements. We used three different SNMs (Caveman model [20], Holme-Kim model [10], Toivonen model [19]), for generating input social networks to show that the results detailed in this section are independent of the concrete SNM (and hence, their structural properties). The detailed simulation setup can be found in [6].

4.1 Conformance

We proposed the following metric to measure the conformance to the input social network: Assume a node u has a social relation with node v and node w, and the strength of the relation with v is signficantly stronger than the relation with w. Then, based on the conformance criterion, we expect that u has more contacts with v than with w after a simulation runtime of sufficient length. The percentage of node pairs for which this was true is shown in Table 1 for different input SNMs.

The results clearly show that the conformance is very high independent of the SNM used to generate input social networks. Furthermore, we have performed simulations with a broad range of parameters (see [6, 7]) and report that the conformance exceeds 99% in all cases with each of the three used SNMs. This shows that GeSoMo is characterized by a high conformance independent of the concrete parameters of our model. Among others, this good result is mainly due to the proposed concept of robust conformance.

4.2 Inter-contact Characteristics

In the following, we compare the inter-contact distribution of GeSoMo to the Infocom-trace [12]. This mobility trace is one of the few available direct-contact traces that has a high number of representative contacts. At the same time, its power-law/exponential decay dichotomy is representative for many other traces. Figure 2 shows the complementary aggregated inter-contact CDF of GeSoMo for three different SNMs in comparison to the Infocom-trace. Independent of the used SNM, we observe a distinct power-law (straight line in a log-log plot) followed by an exponential degradation after about half a day ($\approx 40000s$) in all cases. These results correspond to the omnipresent dichotomy found in real human mobility traces [4, 14]. GeSoMo exhibits this characteristic for a broad range of parameters [6, 7] .

4.3 Spatial and Temporal Regularity

In order to investigate the characteristics of spatial regularity, Gonzales et al. have analyzed the reappearance of humans at specific locations in the trajectories of 100000 mobile phone users [9]. For each location, they ranked the number of visits where rank L represents the L-most visited location. They report that the probability of finding a user at a location of rank L is well approximated by $P(L) \sim L^{-1}$ independent of the total number of locations a user has visited. We have performed a similar measurement for GeSoMo in order to conduct a comparison. Figure 3 shows the average

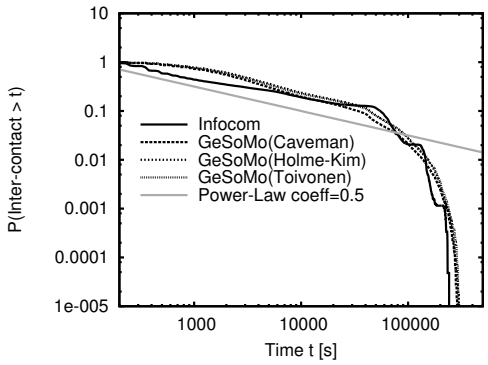


Figure 2 Inter-contact distribution in comparison to the Infocom-trace

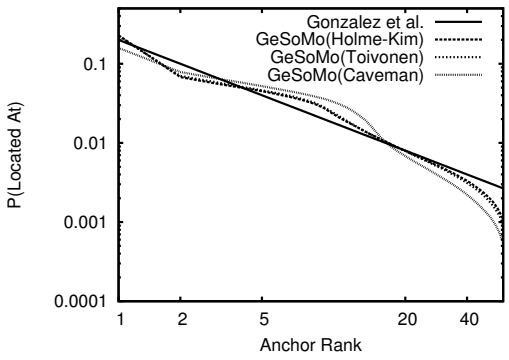


Figure 3 Spatial characteristics compared to an approximation of real-world mobility

visiting probability over all nodes for each anchor rank in comparison to the approximation of Gonzales et al. The good fit shows that nodes in GeSoMo have a preference for a small subset of anchors which they visit with high probability, very similar to real-world mobility behavior.

The evaluation GeSoMo's temporal regularty shows the tendency of users to periodically return to certain locations with high probability (see [6] for details). Very similar characteristics were also reported for existing real traces of human mobility.

5 Conclusions and Outlook

We have presented GeSoMo, a new social mobility model that can be used to evaluate applications for people-based delay tolerant and opportunistic networks. GeSoMo generalizes a number of existing social mobility models by leveraging the elegant and simple concept of separating the simulation model from the structural description of the simulated social network. GeSoMo builds on the principle of *conformance*: an arbitrary input social network is simulated such that the connectivity graph produced among the simulated mobile devices conforms to this input network as closely as possible. This enables researchers to use any existing social network model that is appropriate to their scenario and also the (more realistic) ones that will be discovered in the future without changing the mobility model. This is a major step forward compared to the existing social mobility models which build on specific simplifying social network models and, thus, limit the flexibility of a simulation severely.

In our evaluation which is based on a broad range of empirical observations and data of real user behavior, we have shown that GeSoMo creates simulations that behave realistically with respect to 1. inter-contact distribution, 2. spatial characteristics, and 3. temporal characteristics of human behavior. Furthermore, we have shown that the conformance is very high for all used SNMs. Hence, it is the first SMM which shows all these omnipresent characteristics of human mobility. It is also the first SMM that incorporates group mobility. All these results are independent of the model parameters. We conclude that GeSoMo is a major step forward in the field of social mobility models due to its flexibility and generalizing power over the used SNM.

References

1 The network simulator - ns-2. http://www.isi.edu/nsnam/ns/.

2 Jó Ágila Bitsch Link, Nicolai Viol, André Goliath, and Klaus Wehrle. SimBetAge: Utilizing Temporal Changes in Social Networks for Delay/Disconnection Tolerant Networking. In *MobiQuitous 2009*, 2009.

3 Chiara Boldrini, Marco Conti, and Andrea Passarella. Users mobility models for opportunistic networks: the role of physical locations. In *IEEE Wireless Rural and Emergency Communications*, 2007.

4 Augustin Chaintreau, Pan Hui, Christophe Diot, Richard Gass, and James Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6(6):606–620, 2007.

5 Elizabeth M. Daly and Mads Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 32–40, 2007.

6 Daniel Fischer. A mobility model for the realistic simulation of social context. Master's thesis, University of Stuttgart, 2009.

7 Daniel Fischer, Klaus Herrmann, and Kurt Rothermel. GeSoMo - A General Social Mobility Model for Delay Tolerant Networks. In *Proceedings of the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS '10)*, pages 1–9. IEEE, 2010.

8 Sabrina Gaito, Giuliano Grossi, and Federico Pedersini. A two-level social mobility model for trace generation. In *MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, pages 457–458, 2008.

9 M. C. Gonzalez, C. A. Hidalgo, and A. L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.

10 Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Phys. Rev. E*, 65(2):026107–026112, 2002.

11 W. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy. Modeling Time-variant User Mobility in Wireless Mobile Networks. In *Proceedings of IEEE INFOCOM*, 2007.

12 P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 244–251, 2005.

13 Pan Hui. People are the network: experimental design and evaluation of social-based forwarding algorithms. Technical report, UCAM-CL-TR-713, University of Cambridge, 2007.

14 Thomas Karagiannis, Jean-Yves Le Boudec, and Milan Vojnović. Power law and exponential decay of inter contact times between mobile devices. In *Proceedings of the 13th ACM conference on Mobile computing and networking*, pages 183–194, 2007.

15 A.G. Miklas, K.K. Gollu, K.K.W. Chan, S. Saroiu, K.P. Gummadi, and E. De Lara. Exploiting social interactions in mobile systems. *Lecture Notes in Computer Science*, 4717:409–428, 2007.

16 A. Mtibaa, A. Chaintreau, J. LeBrun, E. Oliver, A.K. Pietilainen, and C. Diot. Are you moved by your social network application? In *Proceedings of the first workshop on Online social networks*, pages 67–72, 2008.

17 Mirco Musolesi and Cecilia Mascolo. A Community based Mobility Model for Ad Hoc Network Research. In *Proceedings of the 2nd ACM/SIGMOBILE International Workshop on Multi-hop Ad Hoc Networks: from theory to reality*, 2006.

18 M. Thomas, A. Gupta, and S. Keshav. Group based routing in disconnected ad hoc networks. *Lecture Notes in Computer Science*, 4297:399–410, 2006.

19 R. Toivonen, J.P. Onnela, J. Saramäki, J. Hyvönen, and K. Kaski. A model for social networks. *Physica A: Statistical Mechanics and its Applications*, 371(2):851–860, 2006.

20 D.J. Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton University Press, 1999.

Anonymous Communication in the Digital World*

Andriy Panchenko

Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg
`http://www.securityandtrust.lu`
`andriy.panchenko@uni.lu` `http://lorre.uni.lu/~andriy/`

Abstract

Privacy on the Internet is becoming a concern as an already significant and ever growing part of our daily activities is carried out online. While cryptography can be used to protect the integrity and confidentiality of contents of communication, everyone along the route on which a packet is traveling can still observe the addresses of the respective communication parties. This often is enough to uniquely identify persons participating in a communication. Anonymous communication is used to hide relationships between the communicating parties. These relationships as well as patterns of communication can often be as revealing as their content. Hence, anonymity is a key technology needed to retain privacy in communications.

This paper provides a very brief overview of my doctoral dissertation *"Anonymous Communication in the Age of the Internet"* [2] and then concisely focuses on one randomly selected aspect, namely, the attack on the anonymization concept called *Crowds*.

Keywords and phrases Security, privacy, anonymity, anonymous communication, confidentiality

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.221

1 Introduction: Brief Thesis Overview

The goal of the thesis *"Anonymous Communication in the Age of the Internet"* [2] is to enhance the state-of-the-art in the field of anonymous communication and to contribute to a broader understanding of the topic and its primitives within the community of researchers as well as to create solid fundamentals for future designs of the systems empowering users with tools for strengthening their privacy protection on the Internet.

We first propose a practical attacker model for risk analysis in anonymous communication. We justify the applicability of the model by an analysis of the strength of anonymizing techniques compared to each other as well as some widely known attacks on them. We then present the design and evaluation of a novel lightweight anonymization protocol based purely on open standards. It significantly outperforms other existing approaches for anonymization while only slightly sacrificing the level of provided protection. We next propose and analyze two innovative approaches for scalable distribution of information about anonymization networks. They have security properties similar to a centralized directory, but scale gracefully and do not require trust in any third party. We use analytical models and simulations to validate our approaches. We also consider performance issues of anonymization networks. To this end we develop and evaluate path selection metrics for performance-improved onion routing. The results show that applying our methods, users can obtain a significant increase in performance without harming their anonymity. Alternatively, users can get a dramatic performance boost with little sacrifice in anonymity. We provide a practical approach to empirically analyze the strength of anonymity different methods of path selection provide

* This work was conducted at the RWTH Aachen University, Germany.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 221–226

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

in comparison to each other. Finally, we investigate several attacks against anonymous communication systems. Most notably, we present a traffic analysis attack against encrypted HTTP streams sent through different anonymizers with surprising results showing the effectiveness and ease of website identification in encrypted channels transferred through the commonly used anonymizers. Moreover, we show under which conditions and how innocent-looking application layer data can be used to speed up traditional attacks that are targeted at the network layer identification of a user's communication partners. We also propose and analyze different countermeasures hampering these attacks.

2 Crowds

Crowds [5] is a peer-to-peer system for anonymous web browsing. It is based on a simple randomized routing protocol, in which all participants forward messages on behalf of other users as well as their own. Crowds provides security by means of increased path length and was meant to be a trade-off between performance and security. The main idea of Crowds is to hide each user's communications by routing them randomly within a group of similar users (*"blending into a crowd"*). When a user requests a web page, he sends the request to another (randomly chosen) crowd member (called a *jondo*). Upon receiving such a request, this jondo (and, if any, all consecutive) decides whether to forward the message to the final destination or to another randomly chosen jondo by making a *biased coin toss*. More formally, the message is forwarded to its final destination with probability $1 - p_f$, or to some other random participant with probability p_f. Communication between jondos is encrypted, however each of them sees the content of passing messages, including the address of the final destination. The final request to the server is usually sent in plain.

3 Predecessor Attack on Crowds

Due to the fact that the message initiator always forwards messages to a randomly chosen node, but all consecutive nodes only with a certain probability, there exist an information leakage in the system: the one who forwards a message to a colluding jondo is more suspicious to be the message initiator than any other honest jondo. In the following we will show how to make use of this information leakage in order to deanonymize the users.

Let $n + c$ be the size of the crowd, c is the number of colluding jondos and p_f is the probability of forwarding in the system as defined in [5]. $n + c$ and p_f are system parameters known to everyone, while c is known only to the adversary.

Let p_h denote the probability that a colluding jondo receives a message directly from the path initiator[1]. Let p_l denote the probability for one honest jondo, which is not the initiator, to forward the message to a colluding jondo (note that $p_h + (n-1) \cdot p_l = 1$). The interested reader is referred to [2, 3] for the details how to calculate the probabilities p_h and p_l.

Because of the information leakage mentioned above, $p_h > p_l$ for all admissible parameter values [2]. This fact can be used by an attacker to make precise statements about the users' peers. We assume that the attacker participates in the crowd with c jondos ($c \geq 1$) and, without loss of generality, only stores the passing communication to a single external entity, namely Bob. In this section we will show how to estimate the amount of communication that each honest user initiated with Bob. The estimation can be performed with an arbitrary precision [2].

[1] h stands for "high", l for "low"

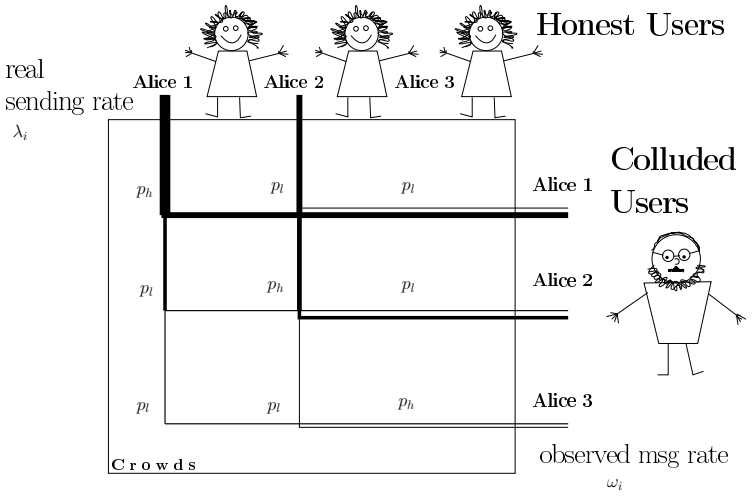


Figure 1 Information flow for the attacker: matrix M

From the adversary's point of view, there are n jondos that are sending messages to Bob, each with its own average rate $\lambda_i \geq 0$ per time interval. We model these as Poisson processes $A_i = \mathcal{P}_{\lambda_i}$ for $i \in 1 \ldots n$. This kind of arrival distribution is in our opinion a fair trade-off between the analytical complexity and realism. Herewith we want to provide a first approximation which can be further refined by modeling arrivals in a more sophisticated manner. The colluding entities observe on average the following number of messages to Bob per time interval from the i-th system member:

$$E[\text{msg to Bob from } i] = p_h \cdot \lambda_i + \frac{1 - p_h}{n - 1} \cdot \sum_{j=1, j \neq i}^{n} \lambda_j \tag{1}$$

With the help of the following matrix M, we can model the number of messages O_i arriving at the attacker from jondo i to Bob:

$$M = \begin{pmatrix} p_h & p_l & \cdots & p_l \\ p_l & p_h & \cdots & p_l \\ \vdots & \vdots & & \vdots \\ p_l & p_l & \cdots & p_h \end{pmatrix} \tag{2}$$

$$O_i = \sum_{j=1}^{n} \mathcal{P}_{m_{i,j}\lambda_j} \tag{3}$$

The observations can be seen as a vector of messages (each element is the number of messages received from the corresponding jondo by colluding members and addressed to Bob), which is a product of the vector of actually sent messages with the matrix M (see Figure 1). The thickness of the line corresponds to the sending rate of the corresponding user (the thicker it is, the higher is the sending rate). Note that the missing line from one of the users on the right-hand side means that he does not send own messages, but rather only forwards some on behalf of the others (which corresponds to the thin line pointing to colluding users). Since O_i is a sum of the Poisson processes, it is also a Poisson process with

the following properties:

$$O_i = \mathcal{P}_{\omega_i} \quad \text{for} \quad \omega_i = \sum_{j=1}^{n} m_{i,j}\lambda_j \tag{4}$$

$$E[O_i] = V[O_i] = \omega_i \tag{5}$$

Because the cardinality of ω_i depends on the value of the λ_i, from the observations of O_i it is possible to draw conclusions about the λ_i (for $t \to \infty$):

$$(\omega_1, \ldots, \omega_n) = M(\lambda_1, \ldots, \lambda_n) \tag{6}$$

$$\iff \quad (\lambda_1, \ldots, \lambda_n) = M^{-1}(\omega_1, \ldots, \omega_n) \tag{7}$$

Indeed, the matrix M is invertible to M^{-1} because Crowds is unable to provide perfect security ($p_h \neq p_l$), thus:

$$M^{-1} = \begin{pmatrix} \tilde{p_h} & \tilde{p_l} & \cdots & \tilde{p_l} \\ \tilde{p_l} & \tilde{p_h} & \cdots & \tilde{p_l} \\ \vdots & \vdots & & \vdots \\ \tilde{p_l} & \tilde{p_l} & \cdots & \tilde{p_h} \end{pmatrix} \tag{8}$$

$$\tilde{p_h} = \frac{n + p_h - 2}{n p_h - 1} \tag{9}$$

$$\tilde{p_l} = \frac{p_h - 1}{n p_h - 1} \tag{10}$$

Under the common values for p_f, n, and c ($0 < p_f < 1,\ n \gg c$) the following inequalities hold: $\tilde{p_h} > 0$ and $\tilde{p_l} < 0$.

Having observed ω_i, it is thus possible to calculate an estimation for the λ_i, namely $\tilde{\lambda_i}$, as follows:

$$\tilde{\lambda_i} = \sum_{j=1}^{n} {m^{-1}}_{i,j}\omega_j \tag{11}$$

Using Chernoff style bound for the probability that a Poisson process deviates from its mean, it is even possible to calculate the number of observations needed in order to estimate sending rates λ_i with any desired precision. The interested reader is referred to [2] for further details.

4 Predecessor Attack Speed-Up: Cross-Layer Attack

The classical predecessor attack presented above aims to identify a user's peer partners at the network layer only. We will show in this section that finding a user's peer partners by a predecessor attack can be sped up by building an extensive user profile on the application layer in parallel, i.e., identifying values of different communication attributes.

While a user typically communicates with many arbitrary peers, his application layer profile (set of accepted languages, browser version, etc.) remains usually the same. The required number of observations to confirm (identify) with arbitrary precision user A_i's peer partner B proportionally depends on the message rate from A_i to B observed by colluding users [2]. The same is valid for the application layer profile. Therefore, an attacker will discover the application layer profile of his victim much faster than the communication profile. If the attacker is using a statistical attack on the network layer, he can then use information from the application layer to bias the input for the network layer attack, e.g., by filtering out improbable combinations or in general, messages that do not fit to the victim's profile.

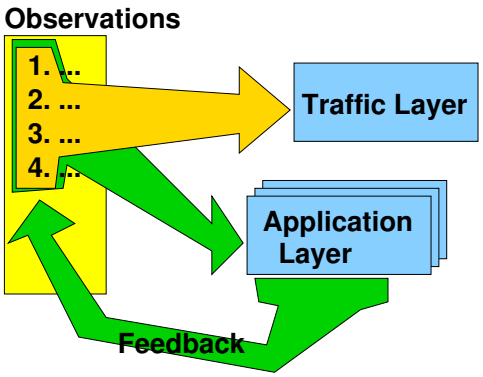


Figure 2 Cross-layer information flow

Figure 2 illustrates how the attack works: at first a user's application layer profile is built applying the classical predecessor attack – but rather on application layer data. This information is further used in order to refine the classical attack on the network layer: the feedback is given to improve the observations which are used as an input to the attack on the traffic layer [2, 4].

After the profiles of users are built, it is possible to use the same data in order to identify the user's peer partners. Actually, even during the process of building the application layer profile, the calculations for the network layer can already be adapted on the fly. We propose two attacks using application layer data that work as follows:

combined attack: the classical predecessor attack (as described in the previous section) is applied only for messages matching the profile;

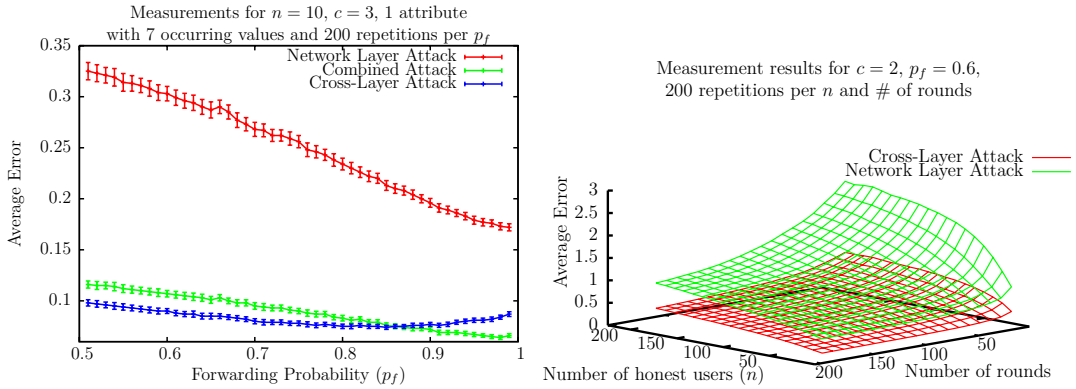
cross-layer attack: same as above, but additionally p_h^{new} and p_l^{new} are used instead of p_h and p_l. p_h^{new} is the probability that A_i is the originator of the message, given that the message is received from A_i by an attacker and that it has a corresponding application layer attribute. Similarly, $p_l^{new} = \frac{1-p_h^{new}}{n-1}$.

In order to evaluate the effectiveness of the new attacks we have performed simulations. The simulations are done as follows: every honest user has exactly two random communication peers. These two peers do not change during the whole simulation run. In each round every honest user sends between 0 and 3 messages to each of his communication peers. On the application layer, we used a single attribute with 7 possible values of the attribute in the communication profile (e.g., accepted languages in an HTTP request).

Figure 3 shows the simulation results including 95% confidence intervals in the 2D plot. Both of the introduced attacks are significantly more precise than the original network layer attack. We found out that the advantage of our attacks is higher for networks with a lower fraction of colluding members. For the considered scenario the accuracy is improved by a factor of 3. An interesting finding is that up to the forwarding probability of 0.86 the cross-layer attack is more precise than the combined one. However, for $p_f \geq 0.86$ the combined attack is more accurate. These results can be observed in Figure 3a.

5 Discussion and Conclusion

We showed how the information leakage in Crowds can be used in order to deanonymize its users. We also showed that enriching network layer information with innocent-looking



(a) Different values of forwarding probability **(b)** Growing no. of observation rounds

Figure 3 Cross-layer attack: simulation results

application layer data (e.g., browser strings in HTTP requests) can be used to significantly increase accuracy and speed up traditional attacks targeted at the network layer only. It is naïve to think that an attacker would not make use of all the information which is available. Hence, information from both network and application layer has to be considered in future research in order to provide usable and realistic metrics for anonymity.

The attack speed-up can be circumvented if the filtering system on the application layer would substitute the identifying information from the browser with the known statistical distribution of the browser data like, e.g., [1]. To be even more unidentifiable, one could observe the communication of the others as an attacker does, and calculate this statistic on his own. This is due to the reason that distribution of identifying information among the users in anonymizing networks may be different from those on the Internet in general.

The network layer attack itself, however, cannot be circumvented completely. Crowds leaks routing information under any admissible parameter values [2]. The number of observations needed to determine the sending rate of the users in the crowd precisely enough can be relatively small [2]. This rises a reasonable doubt on the possibility of using the system for strong anonymity in an open environment.

For more information about this and other topics around anonymous communication, the interested reader is referred to [2].

References

1 Browser statistics. http://www.w3schools.com/browsers/browsers_stats.asp, January 2011.

2 Andriy Panchenko. *Anonymous Communication in the Age of the Internet*. PhD thesis, Department of Computer Science, RWTH Aachen University, 2010. Wissenschaftsverlag Mainz, Aachen.

3 Andriy Panchenko and Lexi Pimenidis. Crowds revisited: Practically effective predecessor attack. In *Proceedings of the 12th Nordic Workshop on Secure IT-Systems (NordSec 2007)*, Reykjavik, Iceland, October 2007.

4 Andriy Panchenko and Lexi Pimenidis. Cross-Layer Attack on Anonymizing Networks. In *Proceedings of the 15th International Conference on Telecommunications (ICT 2008)*, St. Petersburg, Russia, June 2008. IEEE Xplore.

5 Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, pages 66 – 92, April 1998.

Optimized DTN-Routing for Urban Public Transport Systems

Tobias Pögel

Institute of Operating Systems and Computer Networks
Technische Universität Braunschweig, Germany
poegel@ibr.cs.tu-bs.de

Abstract

Communication is crucial to the coordination and efficient operation of public transport systems. However, deployment of infrastructure based communication systems is expensive, esp. due to long-term operational costs. Delay tolerant vehicular networks are a promising alternative since only very few infrastructure elements are required. This paper presents a DTN routing algorithm for urban public transport systems which exploits the knowledge about system characteristics and node mobility to improve the routing performance. Various DTN routing schemes are compared with the presented algorithm. Moreover, the impact of disturbances on the routing performance is examined.

Keywords and phrases DTN, Routing, Public Transport, RUTS, The ONE

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.227

1 Introduction

Communication systems have a large impact on the efficiency and user-friendliness of public transport. Vehicle dispatching, traffic management, arrival prediction, dynamic passenger information and dynamic rerouting are just a few examples for processes that require communication. The increasing integration of information technology into the management and operations of public transport leads to a growing demand of wireless data transfer. Cellular data services provide sufficient range but relatively low data rates. Furthermore, the deployment and operation of a dedicated cellular infrastructure require relatively high investments. For the operator of public transport, permanent cellular access costs might be inhibiting. An alternative is direct vehicle-to-vehicle communication based on IEEE 802.11a/b/g/n/p. These WLAN technologies are capable of significantly higher data rates but also have a lower range, so it is unlikely that each vehicle is constantly in the radio range of other vehicles. This problem could be solved by roadside infrastructure, which would again require high investments. A delay tolerant network (DTN) is a more economical approach since for many applications a (slightly) delayed delivery of data can be accepted.

In a DTN, end-to-end connections are not required. Instead, the protocol data units, referred to as bundles, may be stored until a connection with the next hop is available. Then, the bundle is forwarded and stored again, and so on, until it finally reaches its destination. Choosing the 'right' sequence of next hops to minimize delay and maximize reliability is the routing algorithm's task. The proposed DTN routing algorithm takes advantage of the characteristics of public transport systems to improve the efficiency. In the next section the characteristics of contacts in public transport systems are analyzed. Based on this, the design of a specialized routing algorithm is described. In the evaluation, the routing algorithm is compared with various common DTN routings.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 227–232

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Analysis and Characteristics

This paper focus on urban public transport systems with vehicles operating on surface infrastructure. Buses and trolleys running on roads or rails are the most common vehicles in such transport systems. These vehicles move on lines with stops. The arrival and departure time at each stop is defined by a timetable, which also defines the frequency of service on a given route. Usually, the frequency of service changes several times a day, according to a static plan based on the average number of passengers at a given time. For example, the frequency of service may be five minutes during rush-hour and 30 minutes during the late evenings. It should be noted that timetables are hard to maintain since unpredictable disturbances may happen such as traffic congestion, vehicle failures, traffic accidents or blocked roads.

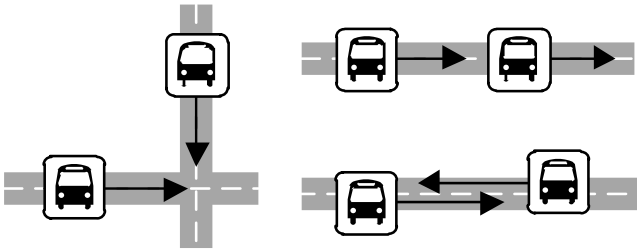


Figure 1 Left: intersection, Right: overlapping segments

A possible contact occurs whenever two vehicles are in each other's radio range. Vehicles move on lines and follow a timetable with potential contacts being roughly predictable. However, there are always minor variations, e.g. caused by traffic lights or changing numbers of passengers getting on and off at stops. In certain situations, these minor variations make the contact prediction very difficult. For example, a variation of just a few seconds can prevent a potential contact between vehicles of two different lines crossing at an intersection. For this reason, it is important to assess the probability and the duration of a possible contact. Currently, the routing differentiates between two situations which are shown in figure 1. The first situation is the crossing of two lines in an intersection in which the driving direction is insignificant. The contact probability and duration for this case is typically low. The other situation is where contacts overlaps in segments of the lines. In this case, lines overlap on a section of the map where vehicles can drive next to each other or one after another. The length and the direction are relevant for the contact probability and duration. Finally, combinations of these contact types are possible.

3 DTN-Routing in Urban Public Transport Systems

The goals in the design of Routing in Urban Public Transport Systems (RUTS) are resource-friendliness as well as timely and reliable delivery as far as possible in a public transport vehicular DTN. As mentioned before, there are some specific characteristics, e.g. timetables and network maps, which can be exploited for routing. If this context information is available on each DTN node, an efficient routing is possible. First, the possible contacts between the vehicles are determined and stored in a routing-graph. Based on this routing-graph, different ways from the sender to the receiver can be identified. Each way represents a possible routing path for a bundle. Due to external disturbances and local conditions, contacts have different performances and probabilities. Therefore, every contact involved in the routing paths has

to be evaluated. This result is a rating for every possible routing path. Now, the bundle can be passed on to the most suitable path. Multiple copies of a bundles can be used to increase the delivery probability and to reduce latency. After the routing path for a bundle has been determined, it is stored in the bundle. Now, the routing path can be followed unless there is a discrepancy which occurs when a planned contact fails. Only in this case a new path has to be calculated.

3.1 RUTS Routing

At first, the number of bundle copies has to be determined which can increase the delivery speed and rate. As a result of unpredicted disturbances, it could happen that not every bundle can be routed on its calculated path. Therefore, n is defined as the number of bundle copies to be issued plus the original bundle. n is usually small and ≥ 1. For each new bundle, the transmitting node sets the initial value of n into an additional field in the bundle to save storage and transmission capacity for 'real' copies. As long as $n > 1$, the bundle may be split by a DTN node to follow a previously calculated path for each bundle copy. This means that a bundle is transmitted with an adjusted value of n to another node, whereby multiple transmissions can be saved. Now, both nodes have the same bundle and the sum of available copies of both parts corresponds to the value before. Over the whole bundle lifetime, it must be ensured that the number of copies in the DTN network is always $\leq n$. In addition to the splitting, it is still possible to forward bundles to follow the calculated paths.

In order to send the bundles and their copies on different routing paths, the different possibilities have to be detected. For this, k routing paths must be identified where typically $k > n$ to get a higher choice because each routing path can have a different quality. To find the necessary paths, RUTS uses the available timetables and the network map from the urban public transport system. This context information can be used to calculate possible contacts between vehicles. Each possible contact is stored in a special routing-graph. In the first step, the search starts from the sender DTN node. All possible contacts are inserted into the routing-graph. Now, the search will be resumed for all nodes contained in the routing-graph. The search is iteratively continued and is stopped if at least k potential paths are available. This routing-graph is explained in more detail in [1] and [2].

Now, the k fastest paths from the transmitter to the destination must be found in the routing-graph. As already mentioned, the paths vary in quality such as different contact probability and duration between the vehicles. Therefore, the paths are evaluated and the n best ways are selected for forwarding. In addition, a score value for each of the k-paths is calculated. It reflects the estimated time required for the transport from the sender to the receiver, the number of required hops and the line characteristics for each DTN node. All of the data can be obtained from the routing-graph and the available context information. The score for each k-path is determined as follows:

$$Score_k = \text{Time Difference} + \frac{\sum(\text{Contact-Rating})}{\text{Number of Bundle Transmissions}} \tag{1}$$

The time difference indicates the maximum transmission time of all paths minus the transmission time of the current examined path. For the rating of the contacts, the length of the overlapping or rather a fixed value for intersections is used. Each parameter can be parameterized to control its impact on the score. In addition, the rating models for the possible DTN node transmissions are interchangeable. After each of the k paths has been rated, the bundles are sorted according to the scores and distributed on the most suitable paths.

4 Evaluation

The RUTS routing was implemented for 'The ONE' [3], a DTN simulator featuring several previously proposed routing protocols which are used for performance comparison. For the simulation, realistic mobility traces with the real lines and timetables from the urban public transport system in Braunschweig are used. The studied scenario has been simulated for 5 hours, has 54 stops and 13 lines which are defined by a series of stops. For each line, one or several vehicles are assigned so that 28 vehicles are simultaneously on the lines. For the data exchange between nodes, the simulator generates a bundle each 5 seconds. Source and destination are random but are equal in each run. The RUTS uses a local copy of the timetables and the network map on each vehicle. In addition to the original bundle, two additional copies are used ($n = 3$). In the routing-graph five possible paths ($k = 5$) are determined.

4.1 Simulation Results

For the evaluation, various performance criteria are used to compare the different routing protocols. First, the number of successfully delivered bundles is considered as well as the bundles that never reached their destination because all bundle copies were discarded. Moreover, bundles may exist that have not yet been delivered because the simulation was terminated.

The next criterion is the average memory consumption at the nodes over the entire simulation. This value has an impact on the number of discarded bundles and the number of transmitted bundles. Furthermore, the average time to transport a bundle from the source to the destination is investigated. The number of delivered bundles has a direct influence on this value. Hence, these values must be examined jointly.

In figure 2, the successfully delivered bundles are illustrated as well as the lost bundles and the bundles remaining in transit at the end of the simulation. In the comparison of all routing protocols, RUTS can deliver the most bundles and also causes the lowest bundle loss. Figure 3 shows that RUTS occupies less storage capacity than nearly all other routing protocols. Only FirstContact uses slightly less memory in the simulation because it uses no additional bundle copies. However this causes a low delivery rate.

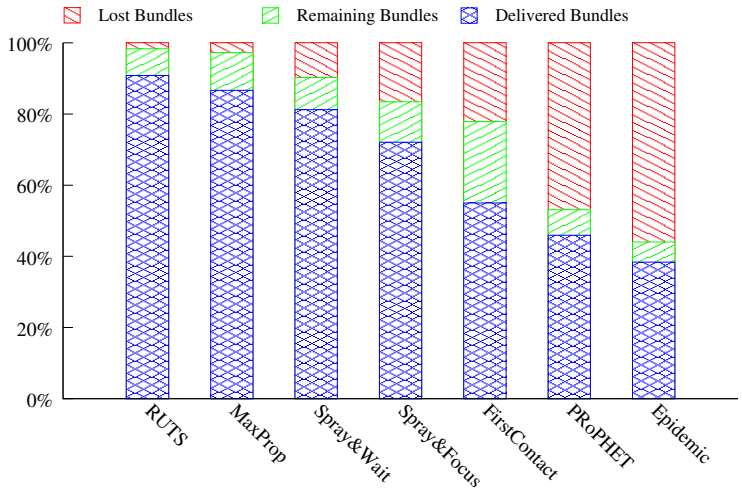


Figure 2 Lost, remaining and delivered bundles

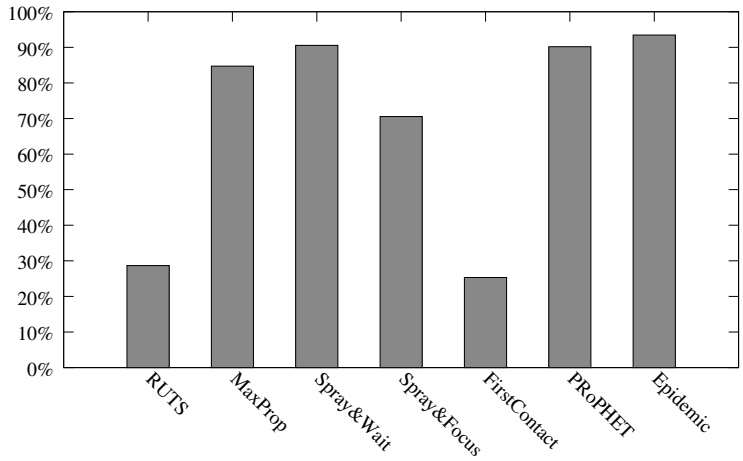


Figure 3 Average memory usage

Figure 4 compares the transmission times of successfully delivered bundles. Due to the random selection of the source/destination pairs, it is possible that bundles are generated on nodes which are in contact with the destination. These bundles can be delivered immediately, so that the minimum transmission time is very low. Average and maximum transmission times can only be calculated for bundles which are delivered before the simulation ends. Therefore, transmission time can only be interpreted in relation to the amount of delivered bundles because a higher number of delivered bundles increases the probability that bundles with high transmission times are included. In this context, RUTS achieves a low average transmission time, although it has the best delivery ratio.

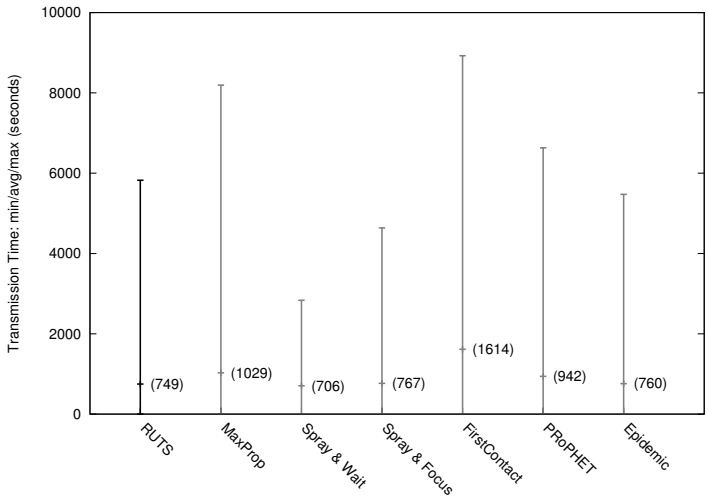


Figure 4 Transmission time (min/avg/max)

To investigate the impact of disturbances in the public transport system, several variations in the scenario were generated. 20% of the vehicles are chosen at random and are assigned a random delay of 1-5 minutes. Thereafter, the result is calculated as the average of several simulations. Figure 5 shows the differences between the previous results and the average results with random delays. Here, the differences are less than 1%.

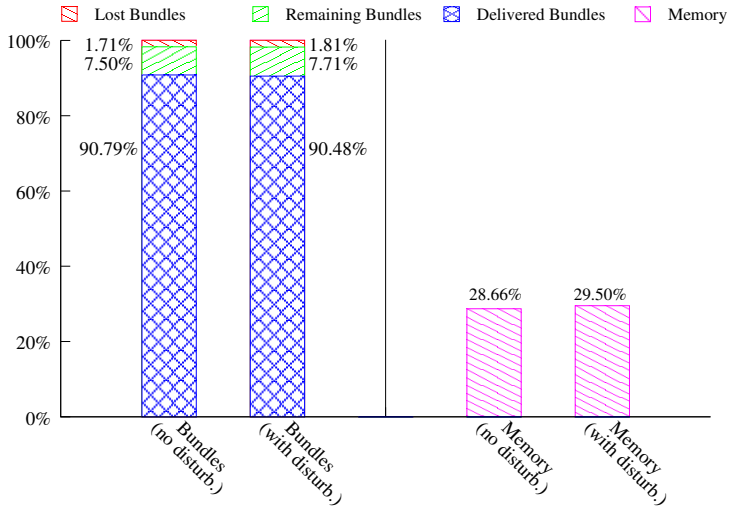


Figure 5 Impact of disturbances

The performance of RUTS is very promising taking into account all performance criteria. In all investigated areas, RUTS is at least similar to the best performing routing protocol. In most cases it archives the best results. Moreover, through the usage of some bundle duplicates, which are routed via another path, it is robust against disturbances.

5 Conclusion and Future Work

In this paper, the RUTS routing was presented which exploits the characteristics of urban public transport systems. The results show a low resource utilization, low latencies and high delivery rates. This is also the case if delays or breakdowns occur. In the future, major scenarios should be studied and simulated, e.g. the urban public transport systems in Chicago [4]. In addition, the scoring functions are further optimized.

Acknowledgements This work was part of a thesis [1]. I would like to thank Michael Doering, Sven Lahde and Lars Wolf for their constructive comments and assistance during this time.

References

1 Tobias Pögel. Entwurf und Analyse eines DTN-Routingverfahrens für ÖPNV-Netze. Master's thesis, Technische Universität Braunschweig, Germany, 2009.

2 Michael Doering, Tobias Pögel, and Lars Wolf. DTN routing in urban public transport systems. In *Proceedings of the 5th ACM workshop on Challenged networks*, CHANTS '10, pages 55–62, New York, NY, USA, 2010. ACM.

3 Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

4 Michael Doering, Tobias Pögel, Wolf-Bastian Pöttner, and Lars Wolf. A new mobility trace for realistic large-scale simulation of bus-based dtns. In *Proceedings of the 5th ACM workshop on Challenged networks*, CHANTS '10, pages 71–74, New York, NY, USA, 2010. ACM.

A Novel Algorithm for Distributed Dynamic Interference Coordination in Cellular Networks

Marc C. Necker

Institute of Communication Networks and Computer Engineering, University of Stuttgart, Pfaffenwaldring 47, Germany, *

Abstract

Wireless systems based on Orthogonal Frequency Division Multiple Access (OFDMA) multiplex different users in time and frequency. One of the main problems in OFDMA-systems is the inter-cell interference. A promising approach to solve this problem is interference coordination (IFCO). In this paper, we present a novel distributed IFCO scheme, where a central coordinator communicates coordination information in regular time intervals. This information is the basis for a local inner optimization in every basestation. The proposed scheme achieves an increase of more than 100% with respect to the cell edge throughput, and a gain of about 30% in the aggregate spectral efficiency compared to a reuse 3 system.

Keywords and phrases 3GPP LTE, 802.16e, OFDMA, WiMAX, beamforming, graph coloring, interference coordination, scheduling

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.233

1 Introduction

IFCO has been an active topic especially in the 3GPP standardization body. It is a promising approach to solve the problem of inter-cellular interference in a reuse 1 scenario. Most activities have focused on local schemes operating on local state information in every basestation. These schemes are often based on power regulation [4] or Fractional Frequency Reuse (FFR) [3]. A number of FFR-based schemes was compared in [14] and [13]. Another local scheme was proposed by Xiao et al. in [15]. Kiani et al. propose a distributed scheme based on local measurements [7]. The authors measure the increase in the overall network capacity, but do not consider fairness issues, such as the throughput at the cell edge. In [8], Li et al. propose a distributed scheme by formulating a local and a global optimization problem, thus being able to include global state information in the coordination process. They consider one strongest interferer and do not consider fairness issues. However, fairness is crucial since it is easy to sacrifice cell edge throughput in favor of overall network throughput [12].

In this paper, we present a novel distributed interference coordination scheme. We explicitly consider the performance at the cell edge compared to the aggregate throughput as a fairness metric. A central coordinator solves an outer optimization problem based on global information collected from the basestations, and the basestations solve a local inner optimization problem based on local state information. The communication with the central coordinator can be in intervals in the order of seconds. The performance is significantly increased compared to Coordinated FFR [10] and pushed further towards the (theoretical) performance of a globally coordinated system. In particular, the presented scheme outperforms a frequency reuse 3 sytem by more than 100% with respect to the cell edge throughput while increasing the aggregate throughput performance by more than 30%.

* Work was done at the University of Stuttgart. The author thanks Prof. Kühn and Prof. Tran-Gia for their great support and many invaluable discussions. He is now with Daimler AG, marc.necker@daimler.com.

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 233–238

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2 Overview of Cellular 802.16e System Model

We consider a cellular 802.16e-system [6], focusing on the Adaptive Modulation and Coding (AMC) zone in the downlink subframe. We consider the AMC 2x3 mode, which defines subchannels of 16 data subcarriers by 3 OFDM-symbols. Our scenario consists of a hexagonal cell layout comprising 19 basestations at a distance of $d_{\mathrm{BS}} = 1400$ m with 120° cell sectors. The scenario is simulated with wrap-around, making all cells equal with no distinct center cell. All cells were assumed to be synchronized on a frame level. Every basestation has 3 transceivers, each serving one cell sector. The transceivers are equipped with linear array beamforming antennas with 4 elements and gain patterns according to [9].

3 Graph-Based Interference Coordination

In [9], we introduced a scheme for global interference coordination based on an interference graph. In this graph, the vertices represent the mobile terminals, and the edges represent critical interference relations between them. It is constructed by evaluating the interference that a transmission to one mobile terminal causes to any other terminal. For each terminal, we first calculate the total interference that the mobile receives from other basestations within a radius of d_{ic}. We then block the largest interferers from using the same set of resources by establishing a relation in the interference graph. This is done such that a desired minimum SIR D_S is achieved. Note that if $d_{ic} = 0$, only interference from other sectors of the same basestation is considered, whereas if $d_{ic} \geq 1$ inter-cellular interference from other basestations is considered as well. For more details, please refer to [12].

Resources need to be assigned to the mobile terminals such that no two mobile terminals are assigned the same resources if they are connected in the graph. This is equal to coloring the graph if the resources correspond to colors [12]. Resources may be mapped to colors $c_{k,i} \in \mathbf{C}$ as shown in Fig. 1. Every AMC zone is subdivided into a certain number N_p of resource partitions. Several AMC zones in subsequent MAC frames form one *virtual frame* such that the total number of resource partitions in the virtual frame corresponds to the number of required colors during the coloring process (precisely: to the next larger multiple of N_p).

4 Distributed Interference Coordination

In this section, we present a novel scheme for distributed IFCO. The scheme uses a central coordinator which is responsible for the coordination of neighboring basestations based on a global interference graph with $d_{ic} \geq 0$. At the same time, every basestation creates a local interference graph with $d_{ic} = 0$ to coordinate the transmissions in its three sectors. The scheme takes a formal approach by formulating an *inner* optimization problem, which needs to be solved by every basestation. This inner optimization problem is subject to constraints delivered by the *outer* optimization problem, which is solved in the central coordinator.

4.1 Outer Optimization Problem

The outer optimization problem is solved by the central coordinator based on an interference graph with $d_{ic} \geq 1$. This graph creation may be based on measurements obtained from the mobile terminals and collected by the central coordinator which is out of the scope of this paper. The goal of the outer optimization problem is to find a set of colors $\mathbf{C}_i \subseteq \mathbf{C}$ (i.e., a set of resource partitions) for every mobile terminal m_i such that there is no conflict between any combination of colors in the sets. This problem is known as *fractional graph coloring*. An

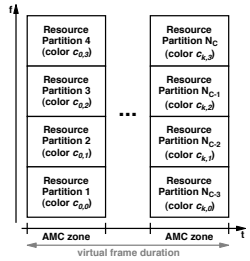


Figure 1 Resource partitioning with $N_p = 4$

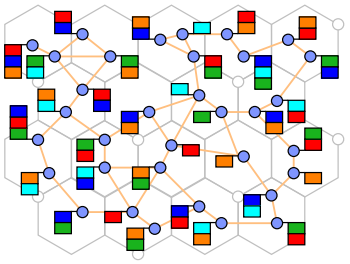


Figure 2 Graph coloring of outer optimization problem.

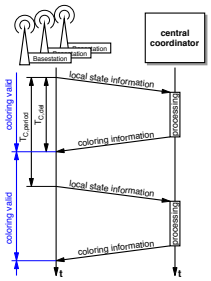


Figure 3 Timing diagram

example for a possible coloring is shown in Fig. 2. Fractional graph coloring is NP-hard. We take the following approach to solve the outer optimization problem. We first color the graph by means of the sub-optimal heuristic Dsatur [5]. Next, we traverse all mobile terminals in a random order and assign a second color to every mobile terminal where possible. We repeat this step until no more extra colors can be assigned to any mobile terminal.

4.2 Inner Optimization Problem

The goal of the inner optimization problem is to assign every mobile terminal to one or more resource partitions $c_{k,l}$ of the respective cell sector. This means that every mobile terminal is assigned a set of colors $\mathbf{R}_i \subseteq \mathbf{C}_i$ on which it is served. That is, $\mathbf{R}_i$ must be chosen from the color set $\mathbf{C}_i$ assigned to mobile terminal m_i by the coordinator. To formulate the optimization problem we introduce for every mobile m_i the matrix $(x_{i,k,l})$, which describes the resource allocation for a particular cell sector:

$$x_{i,k,l} = \begin{cases} 1 & \text{if mobile } m_i \text{ is served in resource } c_{k,l} \\ 0 & \text{if mobile } m_i \text{ is not served in resource } c_{k,l} \end{cases} . \tag{1}$$

This means that $\mathbf{R}_i$ can be defined as

$$\mathbf{R}_i = \{c_{k,l} \mid x_{i,k,l} = 1\} \quad . \tag{2}$$

We further define a utility u_i for every mobile terminal m_i. u_i is a real number and denotes the utility if the mobile terminal is scheduled in a MAC frame.

The objective function of the inner optimization problem for basestation b then is

$$\max \left(\sum_{m_i \in M_b} \sum_k \sum_l u_i x_{i,k,l} \right) \quad , \tag{3}$$

where M_b contains all mobiles m_i which are served by any of the three transceivers of basestation b. Note that eq. (3) maximizes the utility sum for one virtual frame. Hence, the resource allocation problem has to be solved at the beginning of every virtual frame.

The inner optimization problem is subject to a number of constraints.

1. Every mobile m_i has to be served using one color from $\mathbf{C}_i$ assigned by the coordinator:
$$\forall \{x_{i,k,l} \mid x_{i,k,l} = 1\} : \; c_{k,l} \in \mathbf{C}_i \quad . \tag{4}$$
2. Every mobile terminal has to be served at least once in every virtual frame:
$$\forall i : \sum_k \sum_l x_{i,k,l} \geq 1 \quad . \tag{5}$$
3. Every mobile terminal must not be served more than once per MAC frame:
$$\forall i \forall k : \sum_l x_{i,k,l} \leq 1 \quad . \tag{6}$$

4. The constraints of the local interference graph have to be met:

$$\forall \{(i,j) \mid e_{ij} = 1\} : \mathbf{R}_i \cap \mathbf{R}_j = \varnothing \tag{7}$$

The optimization problem formulated by equations (3)–(7) is a binary integer linear program (BILP), which is NP-hard. BILPs can be treated by standard optimization packages for integer linear programs (ILPs), but represent a particularly difficult class of ILPs.

4.3 System Architecture

The system architecture comprises a central coordinator which creates the interference graph and solves the outer optimization problem. All basestations communicate the necessary data to the central coordinator, as shown in Fig. 3. The set of colors is then communicated from the central coordinator to the basestations, which periodically solve the inner optimization problem. Communication with the central coordinator takes place with an update period $t_{C,up}$. The delay $t_{C,delay}$ contains all signalling delays, processing delays and synchronization delays.

5 Solution of Inner Optimization problem with Genetic Algorithms

The inner optimization problem can be solved by means of genetic algorithms. Genetic algorithms are based on generations of solutions. Every generation contains a number $|P|$ of possible solutions to the optimization problem, which are called *genomes*. Every genome is evaluated and assigned a *fitness* value. This fitness value is usually a real number and indicates how "good" the solution is. For details about the applied genetic algorithm see [11].

The genetic representation of a solution is problem-specific and often not obvious. For our problem, we choose a list representation as detailed in [11]. The list contains references to the mobile terminals along with the colors $\mathbf{C}_i$ that were assigned during the outer optimization. The order of the mobile terminals in the list determines the assignment of resources to each mobile terminal. To assign resources, a placement algorithm traverses the list and assigns the first possible and free resource partition to the mobile terminals. The resource partitions must not yet be occupied, and the assignment must not be in conflict with the inner interference graph.

The placement algorithm takes care that constraints (4), (6), and (7) are fulfilled. Constraint (5) is taken into account during the subsequent evaluation of the genome by counting the number n_u of mobile terminals that have not been assigned resources. The second factor during the evaluation is the number n_o of occupied resource partitions (i.e., the resource utilization). Finally, we set $u_i = 1$ for all mobiles m_i. Hence, the number of scheduled mobile terminals will be maximized, i.e., it will be attempted to schedule a mobile terminal in every resource partition. The overall fitness of a genome is then calculated as $Fitness = n_o - n_u$.

6 Performance Evaluation

6.1 802.16e scenario and simulation model

We consider a system with a bandwidth of 10 MHz and a MAC-frame-length of 5 ms. The AMC zone was set to 9 OFDM-symbols with $48 \cdot 3$ subchannels. AMC was applied from QPSK 1/2 to 64QAM 3/4. This results in a max. raw data rate of 6.2 Mbps in the AMC zone.

The system model was implemented as a frame-level simulator using the event-driven simulation library IKR SimLib [1] with a detailed physical layer model as described in [11]. Throughput measurements were done on the IP-layer in downlink direction with greedy traffic sources, capturing all effects of SINR- and BLER-variations, retransmissions by ARQ and HARQ with chase combining, and MAC overhead.

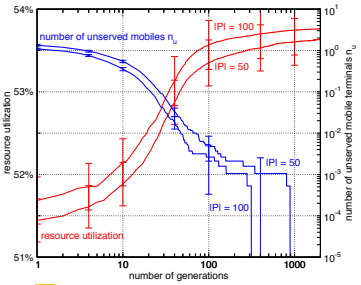


Figure 4 Monte-Carlo Runs: Convergence of algorithm

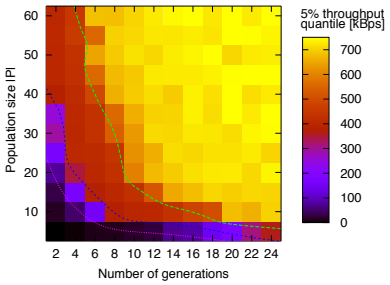


Figure 5 Monte-Carlo Runs: Tradeoff between complexity and performance

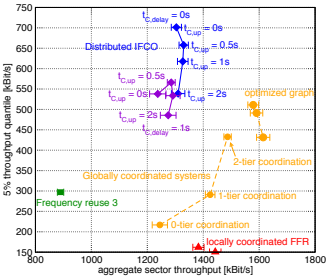


Figure 6 Mobile scenario: Compariosn with other IFCO-schemes.

We consider two mobility scenarios. In the *static scenario*, Monte-Carlo-like simulations are performed, where $N = 9$ terminals are randomly replaced for every drop. The drops have a duration of 4 s with full event-driven simulation. In the *mobile scenario*, each cell sector contains $N = 9$ fully mobile terminals moving at a velocity of 30 km/h, which are restricted to their respective cell sector (see [9]). The Monte-Carlo runs were used to explore the parameter space, since they are much faster to perform, whereas a fully time-continuous simulation was performed in the mobile scenario to achieve final performance values.

The considered throughput performance metrics are the aggregate system throughput, which is proportional to the overall spectral efficiency, and the 5% quantile of the individual throughputs of all terminals, which correlates with the throughput of terminals close to the cell edge [2]. Hence, the 5% quantile is a very good fairness indicator.

6.2 Convergence and Complexity of genetic algorithm

The convergence behavior is shown in Fig. 4. Plotted is the overall resource utilization and the average number of unserved terminals after a certain number of generations N_{gen}. A higher resource utilization leads to a larger aggregate throughput, while the number of unserved terminals affects the fairness. In particular, terminals in unfavorable positions at the cell edge will most likely be unserved for small N_{gen}, thus decreasing the cell edge performance. From Fig. 4 we can see that as few as $N_{gen} = 10$ generations bring the number of unserved terminals below one. For $N_{gen} = 100$, the algorithm already comes close to its optimum performance. The graph also shows that the aggregate throughput, which is mainly determined by the resource utilization, depends much less on the number of generations than the cell edge throughput. This also holds for the population size $|P|$.

The computational complexity is mainly proportional to $N_{gen} \cdot |P|$. Figure 5 plots the 5% throughput quantile depending on N_{gen} and $|P|$. The chart shows that the best performance for a particular computational effort can be achieved for $|P| \approx 2N_{gen}$. Figure 5 further shows that it requires only a small computational effort to achieve near optimal results.

6.3 Comparison with existing IFCO schemes

Figure 6 compares the performance of the proposed distributed coordination scheme with an uncoordinated frequency reuse 3 system (also with beamforming antennas), and a system with FFR, which is locally coordinated based on local state information in every basestation (see [13]). Furthermore, the chart contains reference curves with global coordination according to [9] and [10]. All results in Fig. 6 were obtained in the mobile scenario with only $N_{gen} = 20$.

The performance of the proposed distributed IFCO scheme is plotted for different values of $t_{\mathrm{C,up}}$ and $t_{\mathrm{C,delay}}$. The results show a big performance increase compared to the reference frequency reuse 3 system, and the cell edge performance compared to a locally coordinated

FFR system is greatly increased. Even for update periods and delays in the order of seconds the IFCO schemes achieves a large performance gain. Note also that in a nomadic scenario, which is a realistic use case for 802.16e networks, the performance gain will be mostly independent of $t_{\mathrm{C,up}}$ and $t_{\mathrm{C,delay}}$, hence the performance gains will even be larger compared to the mobile scenario when $t_{\mathrm{C,up}}$ and $t_{\mathrm{C,delay}}$ are in the order of seconds.

7 Conclusion

We efficiently solved the problem of IFCO by separating the initial global optimization problem into two separate problems, namely the inner and the outer optimization problem. We presented efficient approaches to solve these problems based on graph coloring heuristics and genetic algorithms. The complexity is well manageable, since the genetic algorithm converges after very few generations, allowing for efficient hardware-based real-time implementations. We evaluated the performance in a fully mobile scenario. The proposed scheme outperforms a reference reuse 3 system by more than 30% with respect to the aggregate spectral efficiency, and by more than 100% with respect to the cell edge throughput.

References

1 *IKR Simulation Library.* [Online] http://www.ikr.uni-stuttgart.de/Content/IKRSimLib/.

2 3GPP TS25.814. *Physical layer aspects for evolved Universal Terrestrial Radio Access (UTRA) (Rel. 7).* 3rd Gen. Partnership Project, June 2006.

3 3GPP TSG RAN WG1#42 R1-050841. Further analysis of soft frequency reuse scheme. Technical report, 3rd Gen. Partnership Project, 2005.

4 3GPP TSG RAN WG1#47bis R1-070040. DL power allocation for dynamic interference avoidance in E-UTRA. Technical report, 3GPP, Sorrento, Italy, January 2007.

5 D. Brélaz. New methods to color the vertices of a graph. *Comm.ACM*, 22(4):251–256, 1979.

6 IEEE 802.16e. *IEEE Std. for Local and metropolitan area netw., Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Phys. and Medium Access Control Layers for Combined Fixed and Mobile Op. in Licensed Bands*, Feb. 2006.

7 Saad G. Kiani, Geir E. Oien, and David Gesbert. Maximizing multicell capacity using distributed power allocation and scheduling. In *Proc. IEEE WCNC 2007*, Kowloon, China.

8 Guoqing Li and Hui Liu. Downlink dynamic resource allocation for multi-cell OFDMA system. In *Proc. IEEE VTC 2003-Fall*, Orlando, FL, USA.

9 M. C. Necker. Towards frequency reuse 1 cellular FDM/TDM systems. In *Proc. ACM/IEEE MSWiM 2006*, pages 338–346, Torremolinos, Spain, Oct. 2006.

10 M. C. Necker. Coordinated fractional frequency reuse. In *Proc. ACM/IEEE MSWiM 2007*, Chania, Crete Island, October 2007.

11 M. C. Necker. A graph-based scheme for distributed interference coordination in cellular OFDMA networks. In *Proc. 67th IEEE VTC2008 - Spring*, Singapore, May 2008.

12 Marc C. Necker. Integrated scheduling and interference coordination in cellular OFDMA networks. In *Proc. Broadnets*, Raleigh, NC, Sep. 2007.

13 Marc C. Necker. Local interference coordination in cellular 802.16e networks. In *Proc. IEEE VTC 2007-Fall*, Baltimore, MA, Oct. 2007.

14 Arne Simonsson. Frequency reuse and intercell interference co-ordination in E-UTRA. In *Proc. IEEE VTC 2007-Spring*, pages 3091–3095, Dublin, Ireland, April 2007.

15 Weimin Xiao, R. Ratasuk, A. Ghosh, R. Love, Yakun Sun, and R. Nory. Uplink power control, interference coordination and resource allocation for 3GPP E-UTRA. In *in Proc. IEEE VTC-2006*, pages 1–5, 2006.

Does Proactive Secret Sharing Perform in Peer-to-Peer Systems?

Nicolas C. Liebau[1], Andreas U. Mauthe[2], Vasilios Darlagiannis[3], and Ralf Steinmetz[1]

1 **Multimedia Communications Lab, Technische Universität Darmstadt**
2 **Computing Department, Lancaster University**
3 **Informatics and Telematics Institute, Centre for Research and Technology Hellas**

Abstract

Trustworthy applications in fully decentralized systems require a trust anchor. This paper describes how such an anchor can be implemented efficiently in p2p systems. The basic concept is to use threshold cryptography in order to sign messages by a quorum of peers. The focus is put on advanced mechanisms to secure the shares of the secret key over time, using proactive secret sharing. This mechanism was researched in context of the token-based accounting scheme.

Keywords and phrases peer-to-peer, proactive secret sharing

Digital Object Identifier 10.4230/OASIcs.KiVS.2011.239

1 Introduction

Trustworthy applications require an trust anchor, a trustworthy foundation that security mechanisms can employ. Typically in IT systems a trust anchor is a trusted entity with its associated public and private key. This key pair is used as focal point with which the security of a mechanism can be checked.

In p2p systems a single trusted entity does not exist by definition. Thus, if a p2p application requires trust, an alternative is to use a group of peers as the trust anchor. Such a so called quorum delivers trust to the application by probability - the probability that in the quorum fraudulent behavior will not prevail. E.g., in a quorum requiring unanimous judgment a quorum size of 17 guarantees trust with a probability of 99.999%, if there are up to 50% fraudulent peers in the system [7]. Therefore, threshold cryptography has been researched as a way to implement such unanimous quorum judgment. This would build a decentralized trust anchor.

A successful implementation of threshold cryptography in a p2p system consists of several elements. Threshold cryptograhy splits a secret key s into n shares and requires only t shares to create a signature of a message. This is called a (t, n)-threshold scheme.

The focus of this paper is on mechanisms that secure the key shares over time. Peers get compromised over time and their key shares get know to fraudulent peers. If a fraudulent peer collects the knowledge of t shares it can forge system signatures. Proactive Secret Sharing (PSS) was invented to deal with this issue. It introduces time periods and invalidates all key shares at the end of a period by updating them and recovering shares for peers that have been corrupted.

Within the token-based accounting scheme (TbAS) [7, 8] PSS has been employed to ensure the system's long term trustworthiness. However, updating a complete p2p system seems prohibitive expensive. Among other things, this is the reason for introducing so called

17th GI/ITG Conference on Communication in Distributed Systems (KiVS'11).
Editors: Norbert Luttenberger, Hagen Peters; pp. 239–244

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

trusted peers in TbAS. This is a subset of peers that have been selected as share owner due to their trustworthiness using a reputation mechanism.

This paper proves by example of the TbAS that PSS is not prohibitive expensive for applying it to p2p systems in order to build a decentralized trust anchor.

2 Related Work

Threshold cryptography has been suggested to be used in p2p systems by several authors. In [14, 15, 13] the costs of applying threshold cryptography in p2p systems is evaluated; however, focus is on the signature process. The costs of maintaining a distributed secret over time is not shown. In [2] the application of PSS to MANET is discussed but not evaluated. In [11] a DRM system based on PSS is presented, however a thorough evaluation of the created traffic by the PSS mechanism is missing. An evaluation of PSS for up to 100 peers is given presented in [10], assuming that all peers are updated.

3 Cryptographic Background

In order to allow building a trust anchor threshold cryptography (see e.g. [3, 6, 17, 4]) offers the required mechanisms;

3.1 Threshold Cryptography

When selecting a threshold cryptography scheme attention must be paid to the secret sharing mechanism. In additive secret [6] a secret s is split into n shares, and share s_n is computed from the other $n-1$ parts. When a secret key is created and shared among peers, all shares have to be created at the same time. Additional shares cannot be created later. Therefore, additive secret sharing is not applicable to p2p system, where membership is dynamic. In [16] Shamir presented polynomial secret sharing, that does not have this drawback. Here, the shares are calculated by using a polynomial $f(x)$ of degree $t-1$, where the secret key is $s = f(0)$. Accordingly, only threshold schemes based on polynomial secret sharing are considered for building a distributed trust anchor.

There are several challenges to overcome in order to apply threshold cryptography to p2p systems (see [8]); The URSA scheme[9] and BLS scheme [1] are build on polynomial secret sharing and fulfill all remaining requirements.

3.2 Proactive Secret Sharing

This paper focuses Proactive Secret Sharing (PSS) [5] applied in p2p systems. PSS schemes were introduced to protect long-lived shares of cryptographic keys. The principle of PSS is to introduce time periods. In each time period the shares of the shareholders are updated by adding new polynomials $g(x)$ with $g(0) = 0$. That is, all key shares change with an update period, however the secret key remains unchanged. Further, peers with corrupted shares can be recovered; that is a new share for the new sharing polynomial is calculated for them is a distributed fashion by a group of k updated peers. Using recovery, also new peers in the system can get assigned a new share. For details about the protocols used in the TbAS see [8].

Both considered schemes, Threshold BLS and URSA, require the same message flow for a share update or a share recovery. When the shares are distributed, each peer requires an individual share. Accordingly, updating large systems seems to be expensive traffic-wise.

Table 1 Experiments for System-key Maintenance

$L = 99,99\%$				$L = 99,999\%$			
#	T	t	β	#	T	t	β
1.1	100	13	26	2.1	100	15	30
1.2	500	14	28	2.2	500	17	34
1.3	1000	14	28	2.3	1000	17	34
1.4	2000	14	28	2.4	2000	17	34

Legend: T: Number of trusted peers, t: quorum size, β: Size of update group

In [8] different update strategies are evaluated. In [9] a scheme is suggested where only a specific ratio of peers is updated and the remaining peers recover their share. This has the advantage that the knowledge of all trusted peers' IDs is not required (see [8]). However, an evaluation for large p2p systems was not performed. In context of TbAS this Limited Update and Self-Initialization mechanism was evaluated with the target of building highly trustable p2p mechanisms.

4 Simulation of Key Management Traffic

In order to simulate key management, two parts of proactive secret sharing, namely the update phase and the recovery phase were implemented in PeerfactSim.KOM [12]. The objective of this simulation was to assess the traffic created when all trusted peers receive an updated key share of the system-wide private key.

4.1 Experiments

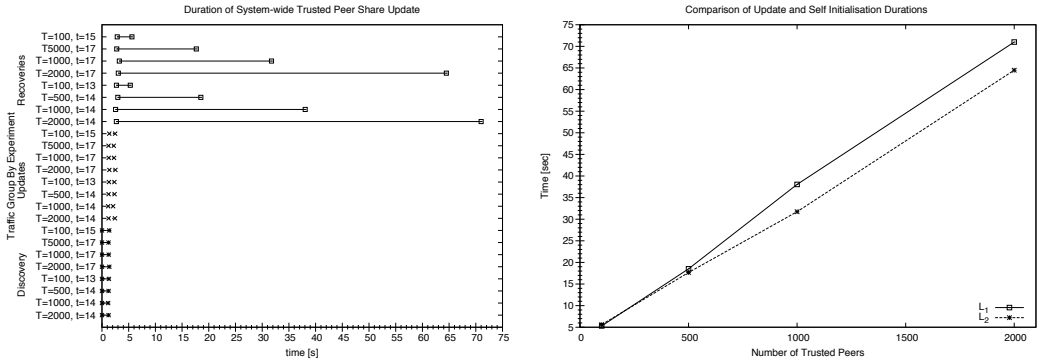
When the proximity group B is found the update shares have to be distributed among the peers in this group. This traffic is fully deterministic, because only direct communication between the group members is applied. Therefore, the general concept of the experiment is to perform the update using only one update polynomial by one peer. This allows an exact analysis of the created traffic that can be easily extrapolated to the use of more update polynomials.

the proximity group B's size β must have at least the size t in order to enable recoveries by that group. Within the simulation, the update group size β is set to $\beta = 2t$ in order to achieve redundancy.

In order to evaluate the scalability of the update and self initialization scheme, different numbers of peers between $T = 100$ and $T = 2000$[1] to be updated and recovered as well as different Trust Levels L (determined by the threshold t) will be simulated[2]. Table 1 summarizes the executed experiments. In all experiments we use a key length of 1024 bit.

[1] Within TbAS scheme threshold cryptography is executed by a subgroup of peers that are called trusted peers. Even for very large p2p systems a number of $T = 2000$ trusted peers is sufficient. See [7, 8].

[2] The computation of required threshold depending on the desired system's Trust Level L, number of peer T, and ratio of good peer p_g was presented e.g .in [7, 8]. For the experiments a ratio of good peer $p_g = 0.5$ is assumed.



(a) Durations by partial mechanism **(b)** Comparison of durations by system size

Figure 1 Durations for update and self-initialization

4.2 Results

The update and self initialization mechanism is a very deterministic process. The simulation does not require any probability distributions. Accordingly, the confidence intervals are very small and cannot be seen in the figures.

4.2.0.1 Duration of Update and Self initialization Process

First, the time required for completely updating the trusted peer system is looked at. Figure 1 a shows the processes' duration.[3] The discovery of the initial update group requires between 1.16 seconds and 1.36 seconds. The difference is due to message transmission delays. The update phase required a bit less time and needed between 0.98 seconds and 1.18 seconds. the reason is that lookup messages might require several hops on the way towards the peer, where update requires direct communication. The recovery phase required 2.58 seconds for systems with $T = 100$ trusted peers and a quorum size of $t = 13$ and 68.3 seconds for a system size of $T = 2000$ and a quorum size of $t = 14$ trusted peers. The average time for a quorum size of $t = 17$ trusted peers is here a bit lower, which indicates that the quorum size has no influence on the duration of the recovery process. The reason is that the number of communication steps required within a recovery of one trusted peer is constant.

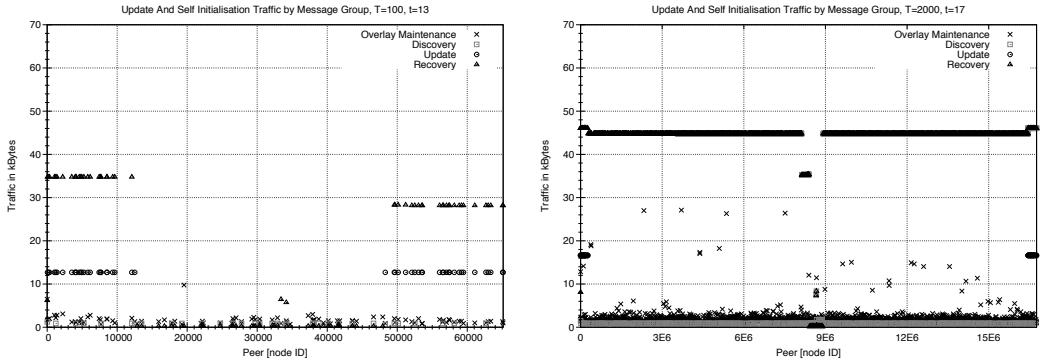
The difference in the duration stems from the different sizes of the beginning start-up group. With a higher Trust Level, the update group in the beginning is larger; updating peers happens in parallel. Therefore, the update duration does not increase with the update group size. However, with a larger update group size less recoveries are required. Therefore, for $L_2 = 99.999\%$ the update duration is shorter.

4.2.0.2 Traffic Generated by Update and Self initialization

The traffic is observed on a per-peer basis. Figure 2 shows the traffic generated for the smallest and largest simulation setzup. The x-axis represents the peers' ID.[4]

It can be observed that the update group is symmetrically located around the peer ID 0. Especially in the graphs for large system sizes with $T = 2000$, the recovery traffic

[3] Note that we did not simulate computation time. The durations all stem from message transfer.
[4] For $T = 2000$ the ID space was increased in order to enable complete message logging. Therefore, peer IDs are distributed from 0 to 2^{24}.



(a) Traffic by message type, T=100, t=13 **(b)** Traffic by message type, T=2000, t=17

Figure 2 Update and self initialization traffic by message type

is almost equal for all peers. Only at the middle of the graph is the traffic distinctively lower, because here the two recovery fronts meet. Recovery traffic is the major source for traffic, although it is below 60 kBytes in total per peer. It can be seen that recovery traffic and overlay maintenance traffic have the lowest proportions of traffic load. Overall, a very even distribution of traffic load over the trusted peers is achieved by the update and self initialization mechanism.

When observing the main source for traffic increase, it is obvious that the quorum size has an influence on it, however the system size influence seems minimal. It can be seen that there are very few peers with distinctively higher loads. The maximum load for the complete update and self initialization process is 240.13 kBytes, which happened in all experiments with $t = 17$. Assuming for trusted peers a DSL-connection with 128 kBit upload, a trusted peer would need 15.37 seconds to send this traffic. Accordingly, even the very few trusted peers with the maximum load do not get overloaded by the update and self initialization process. The statistics show that there are only very few peers with a traffic load above 100 kBytes.

5 Conclusion

This paper focused on a specific issue when applying threshold cryptography to p2p systems. The key shares have to be protected over time. Proactive secret sharing (PSS) is designed for that purpose. PSS introduces time periods and provides the ability to update key shares and recover corrupted and lost key shares at the end of a time period. Also new key shares for new peers can be created. However, these mechanisms seem to be expensive and unattractive for large p2p system. In this paper we proved the opposite. Updating a system of 2000 peers requires approximately and 71 seconds and the average load per peer is 49.10 kBytes.

Accordingly, threshold cryptography based p2p systems can be secured with PSS. Update phases can be performed once a day. Peers not participating in an update will be recovered when they join again. This basic finding can be employed to build decentralized trust anchors for new trustworthy p2p mechanisms, like the token-based accounting scheme [7, 8].

References

1 Alexandra Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *PKC '03: Proceedings of the 6th*

International Workshop on Theory and Practice in Public Key Cryptography, volume 2567 of *LNCS*, pages 31–46, London, UK, 2003. Springer-Verlag.

2 Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. Cryptographic techniques for mobile ad-hoc networks. *Computer Networks*, 51:4938–4950, 2007.

3 Y. Frankel, P. Gemmell, P. D. MacKenzie, and Moti Yung. Optimal-Resilience Proactive Public-Key Cryptosystems. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 384, Washington, DC, USA, 1997. IEEE Computer Society.

4 Amir Herzberg, Markus Jakobsson, Stanislaw Jarecki, and Hugo Krawczyk. Proactive Public Key and Signature Systems. In *Proceedings of ACM Conference on Computer and Communications Security*, pages 100–110, 1997.

5 Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '95)*, volume 963 of *LNCS*, pages 339–352, London, UK, 1995. Springer-Verlag.

6 Stanislaw Jarecki and Nitesh Saxena. Further Simplifications in Proactive RSA Signatures. In *Proceedings of Theory of Cryptography Conference'05*, pages 510–528, 2005.

7 Nicolas Liebau, Vasilios Darlagiannis, Oliver Heckmann, and Andreas Mauthe. *Peer-to-Peer Systems and Applications*, volume 3485 of *LNCS*, chapter Accounting in Peer-to-Peer Systems, pages 547 – 566. Springer-Verlag, 2005.

8 Nicolas C. Liebau. *Trusted Accounting in Peer-to-Peer Environments - A Novel Token-based Accounting Scheme for Autonomous Distributed Systems*. PhD thesis, Technische Universität Darmstadt, Germany, 2008.

9 Haiyun Luo, Jiejun Kong, Petros Zerfos, Songwu Lu, and Lixia Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks. *IEEE/ACM Transactions on Networking*, 12(6):1049–1063, 2004.

10 Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang. Self-Securing Ad Hoc Wireless Networks. In *ISCC '02: Proceedings of the 7th International Symposium on Computers and Communications (ISCC'02)*, page 567, Washington, DC, USA, 2002. IEEE Computer Society.

11 Ling Ma, Shouxun Liu, and Yongbin Wang. A DRM model based on proactive secret sharing scheme for p2p networks. In *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pages 859 – 862, 2010.

12 Multimedia Communications Lab – Technische Universität Darmstadt. PeerfactSim.KOM. http://peerfact.kom.e-technik.tu-darmstadt.de/, 2007.

13 Nitesh Saxena. *Decentralized Security Services*. PhD thesis, University of California, Irvine, 2006.

14 Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *SASN '03: Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 104–113, New York, NY, USA, 2003. ACM Press.

15 Nitesh Saxena, Gene Tsudik, and Jeong Hyun Yi. Threshold Cryptography in P2P and MANETs: The Case of Access Control. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(12):3632–3649, 2007.

16 A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.

17 Victor Shoup. Practical Threshold Signatures. In *Proceedings of Eurocrypt 2000*, 2000.