Kenneth Birman, Flaviu Cristian, Friedemann Mattern, André Schiper (editors):

Unifying Theory and Practice in Distributed Systems

Dagstuhl-Seminar-Report 96;
05.09.-09.09.94 (9436)

ISSN 0940-1121

Copyright © 1994 by IBFI GmbH, Schloss Dagstuhl, D-66687 Wadern, Germany
Tel.: +49-6871 - 2458
Fax: +49-6871 - 5942

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Verantwortlich für das Programm ist das Wissenschaftliche Direktorium:
Prof. Dr. Thomas Beth.,
Prof. Dr.-Ing. José Encarnação,
Prof. Dr. Hans Hagen,
Dr. Michael Laska,
Prof. Dr. Thomas Lengauer,
Prof. Dr. Wolfgang Thomas,
Prof. Dr. Reinhard Wilhelm (wissenschaftlicher Direktor)

Gesellschafter: Universität des Saarlandes,
Universität Kaiserslautern,
Universität Karlsruhe,
Gesellschaft für Informatik e.V., Bonn

Träger: Die Bundesländer Saarland und Rheinland-Pfalz

Bezugsadresse: Geschäftsstelle Schloss Dagstuhl
Universität des Saarlandes
Postfach 15 11 50
D-66041 Saarbrücken, Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397
e-mail: office@dag.uni-sb.de

Contents

Report on the Dagstuhl Seminar

Unifying Theory and Practice in Distributed Systems

September 5th - September 9th, 1994

Kenneth Birman, Flaviu Cristian,
Friedemann Mattern, André Schiper

During the past 20 years, a substantial theoretical and practical base has evolved in the area of distributed computing. However, this work has been done by largely disjoint sets of researchers, with the result that much theory is inapplicable to real-world systems, and many of the real-world systems that have been built suffer from weaknesses that could be overcome using the existing theoretical methodology. It was therefore the intention of the seminar "Unifying Theory and Practice in Distributed Systems" to bring together a diverse group of pragmatically inclined theoreticians and theoretically inclined practitioners with the goal of sharing insight, educating one another, and laying the groundwork for the next generation of distributed systems research and development.

The Dagstuhl seminar was organized by Kenneth Birman (Cornell University), Flaviu Cristian (University of California at San Diego), Friedemann Mattern (University of Saarland at Saarbrücken), and André Schiper (Ecole Polytechnique de Lausanne). It brought together 38 participants – established experts from academia and industry as well as young scientists – from nine countries. 30 talks (some with on-line and video demonstrations) were given during the week, often followed by lively and sometimes controversial discussions. The emphasis of the talks was on the following general themes:

- important paradigms and influential concepts
- fundamental algorithms and principles
- fault-tolerance
- real-time
- system structures, basic services, toolkits
- large scale issues, application domains, case studies

In two evening discussion sessions the implications of the Fischer-Lynch-Paterson Theorem (impossibility of distributed consensus with one faulty proces-

sor) for practical system design and the various notions of "real time" were discussed. In total, the seminar was considered to be successful and very interesting, and the publication of proceedings is now considered.

The special nature and atmosphere of Schloss Dagstuhl offered ample opportunities for personal discussions, the computer science library was also used extensively by some participants. The fine food, the good wine, and the perfect organization by the office and the local staff of Schloss Dagstuhl was much appreciated by all participants. If there is anything to blame, then it is the weather—we hope that the next time we will have some sunny days!

Architectural Issues in the StormCast System

Dag Johansen and Gunnar Hartvigsen
Department of Computer Science, University of Tromsø, Norway
Email: johansen@cs.cornell.edu

We briefly present the architectural approach to reasoning about large scale distributed applications used in StormCast. A high-level architecture and well-defined abstractions are exploited to master the complexity that application developers are confronted with in a particular application domain. This has shown to be useful in promoting both reusability and rapid prototyping.

The Rampart Toolkit for Building High-Integrity Services

Michael K. Reiter
AT&T Bell Laboratories, Holmdel, New Jersey, USA
Email: reiter@research.att.com

Rampart is a toolkit of protocols to facilitate the development of *high-integrity* services, i.e., distributed services that retain their availability and correctness despite the malicious penetration of some component servers by an attacker. At the core of Rampart are new protocols that solve several basic problems in distributed computing, including asynchronous group membership, reliable multicast (Byzantine agreement), and atomic multicast. Using these protocols, Rampart supports the development of high-integrity services via the technique of *state machine replication*, and also extends this technique with a new approach to server output voting. To our knowledge, Rampart is the first system to demonstrate reliable and atomic multicast in asynchronous systems subject to malicious process failures.

In this talk we give an overview of Rampart, focusing primarily on its protocol architecture. We also discuss its performance in our prototype implementation, application services that we are currently developing, and other ongoing work.

Efficient Information Dissemination in Multi Machine Systems

Danny Dolev
Danny Dolev, Institute of Computer Science, Hebrew University, Givat Ram, Jerusalem 91904, ISRAEL
Email: dolev@cs.huji.ac.il

Transis is a high availability distributed services system being developed at the Hebrew University. In the talk we cover its specific services and applications using it. Transis is the first distributed system to handle partition as integral part of the transport layer. It offers a reliable multicast transport layer that employs hardware's broadcast features. It provides applications with causal order, and completes order of messages. On top of these services it offers "safe messages" that is a higher knowledge level service about message delivery. Using safe messages, a variety of replications algorithms can be developed without the need for end-to-end acknowledgements at the application layer.

Some Practical Experience from Delta-4 on Implementing Distributed Fault-Tolerance

David Powell
LAAS-CNRS, Toulouse, France
Email: dpowell@laas.fr

Software-implemented approaches to fault-tolerance are very resilient to change since changes in hardware technology do not require extensive re-design of specialized hardware. The presentation argues the case for implementing fault-tolerance in a distributed fashion and reports the approach adopted in the Delta-4 project. Fault-tolerance is achieved by replicating capsules (the run-time representations of application objects) on distributed nodes interconnected by a local area network. Capsule groups can be configured to tolerate either stopping failures or arbitrary failures. Multipoint protocols are used for coordinating capsule groups and for error processing and fault treatment. The presentation concludes with a critical analysis of the project's results regarding in particular: design assumptions about failure modes; active replication schemes; and performance issues.

Issues on Reusable Spacecraft Control Center Software

Vu Tien Khang
CAP SESA, 8 rue Paul Mesple, 31100 Toulouse, France
Email: vutien@capsogeti.fr

Spacecraft Control Centers can be considered at first view as instances of the general facilities that perform real-time remote supervision of an automated system. The system itself, the spacecraft, is not overly complex when compared with ground systems such as nuclear power plants. However, because of the hostile space environment, spacecrafts present specific issues on accessibility, on availability, on remote diagnosis tools or on known restart points.

On the other hand, Spacecraft Control Centers rely on a variety of tracking and telecommands stations that are scattered around the world, each presenting a different set of features, a different data interface, and operating with different constraints and work habits.

Because of all these constraints, Spacecraft Control Center software have, until the present, been implemented specifically for a spacecraft inside a specifically negotiated environment. Cost factors come in second, after the constraints of the space segment of the global system are solved. As a result, mostly proprietary and specific techniques are used.

Recently, some factors put a new accent on the approach used for SCC software. They are: rising software costs, rising transmitted data volume, rising launch rates as more countries can afford space systems (telecommunication, earth observation, localization). In the same time, scientific users also demand more flexibility to access to on-board experiments. Facing these demands, a possible way to go is to take profit of the trends in the consumer market: RISC and Open Systems for more computing power and more compatibility, high speed switched public networks (Data Highway) for more bandwidth at lower costs, *distributed tools for scalable systems*, multimedia facilities (sound and images) for a better resource management.

Research and development in the Cap Sesa Space Skill Center is concerned with moving beyond the current situation, as summarized above. A new technology base is being developed that will address all aspects of this general problem. We expose the current situation as summarized above, then describe the issues specific to the SCC software field that faces R&D team in Cap Sesa Space Skill Center, such as: upward compatibility, availability, security (on public networks), *interoperability* and, finally, the most difficult issue, the human factor.

Asynchronous Systems with Failure Detectors A Practical Model for Fault-Tolerant Distributed Computing

Sam Toueg
Department of Computer Science, Upson Hall, Cornell University, Ithaca, NY 14853
Email: sam@cs.cornell.edu
Vassos Hadzilacos
Computer Systems Research Institute, University of Toronto, 6 King's College Road, Toronto, Ontario M5S 1A1
Email: vassos@cs.toronto.edu

In 1983 Fischer, Lynch and Paterson proved that Consensus, which is a fundamental problem in fault-tolerant distributed computing, cannot be solved in asynchronous systems, even in the presence of at most one crash failure. To circumvent this impossibility result, we introduce the concept of *failure detectors*. We define a hierarchy of failure detectors including several unreliable ones – i.e. failure detectors that falsely suspect correct processes as crashed. We show how to solve Consensus using such failure detectors in asynchronous systems. Finally we identify a particular failure detector in this hirarchy, called the eventually weak failure detector $W_\diamond$, and show that it is the *weakest* failure detector that can be used to solve Consensus. Our theoretical framework and results have useful implications about practical fault tolerant systems, some of which we discuss.

Towards a Synthesis of the Synchronous and Asynchronous Distributed Computing Models

Kenneth P. Birman
Dept. of Computer Science, Cornell University
Email: ken@cs.cornell.edu

It has become common to treat distributed systems as having either purely synchronous or purely asynchronous execution models. In particular, most work on protocols for real-time computing and for tolerance on severe faults has been undertaken in a synchronous model, where communication delays can be bounded and clocks are assumed to have bounded skew and drift. The asynchronous model, on the other hand, underlies most work on distributed consistency, temporal logics and distributed knowledge, and is the framework in which the "virtual synchrony" model used in my work on Isis and Horus was based.

Yet, real-world distributed systems are neither synchronous nor asynchronous, but rather exhibit elements of both approaches. Current distributed systems are asynchronous when viewed on a temporal scale at which the resolution of the execution model approaches the latency for sending messages on the network. Yet these same systems are synchronous, to increasingly high precision, when the load on the system is well below its maximum and when the temporal scale is sufficiently coarse. In this presentation, I will suggest that because the developers of real distributed systems are normally interested in solving problems at a range of temporal resolutions and with properties that may depend upon circumstances, we need to start to conceive of distributed systems in terms that directly link the temporal scale to the properties it offers.

In effect, we need to think of protocol suites as having properties and corresponding probability distributions, parameterized by the time scale on which we wish to examine the system and perhaps the "strength" of assumptions made about the environment. In this view, a protocol would not guarantee virtual synchrony or guarantee temporal properties, but rather would achieve each of a set of properties spanning consistency and temporal behaviors of the system, and corresponding to a parameterized probability distribution.

The development of tools and technology that are realistic about the environment in which "real" distributed applications are developed and operated will not be an easy undertaking. The assumptions underlying the synchronous and asynchronous models are ultimately simplifying assumptions, and if by relaxing them a huge new form of complexity is introduced, it is unlikely that useful tools would result. However, if methods for controlling complexity can be identified, we may be able to achieve powerful new tools for engineering reliable distributed systems – tools in which reliability and real-time properties are attacked from multiple perspectives, to arrive at correct solutions with very high probability. Such an outcome would represent a major step forward for our field of research, and is a challenge worthy of considerable effort.

Transaction Model vs Virtual Synchrony Model: Bridging the Gap

Rachid Guerraoui and Andre Schiper
Laboratoire de Systemes d'Exploitation, Departement d'Informatique, EPFL CH1015 Lausanne, Switzerland
Email: rachid@lsesun3.epfl.ch, schiper@lse.epfl.ch

Two important models for building fault-tolerant applications have been independently proposed in the literature, the *transaction* model (developed within

the context of database applications) and the *virtual synchrony* model (proposed initially by the Isis system). An interesting question is then the following: are the basic mechanisms needed to implement both models exactly the same? We answer this question by defining the Dynamic-Terminating-Multicast problem and showing that it can be seen as a generic problem that allows to implement both the transaction and the virtual synchrony model. It should thus be possible to build a system offering, in an integrated way, both the transaction model and the virtual synchrony model. Such a system could offer powerful primitives that are currently cumbersome or impossible to express in the existing systems.

Sequential Consistency in Distributed Systems: Theory and Implementation

Michel Raynal
IRISA, Campus de Beaulieu, 35042 RENNES cedex - France.
Email:Michel.Raynal@irisa.fr

Masaaki Mizuno
James Z. Zhou
Dept. of Computing and Info. Sciences, Kansas State University, Manhattan, KS 66506.

Recently, distributed shared memory (DSM) systems have received much attention because such an abstraction simplifies programming. It has been shown that many practical applications using DSMs require competing operations. We have aimed at unifying theory and implementations of protocols for sequential consistency, which provides competing operations. The results are useful not only to clarify previously proposed implementations but also to develop new efficient implementations for consistency criteria which provide competing operations, such as sequential consistency, weak ordering (with sequential consistency for competing accesses), and release consistency (with sequential consistency for competing accesses). By adopting concepts from concurrency control, we have developed a theory for sequential consistency, called *sequentializability theory.* This paper first presents the sequentializability theory, and then demonstrates the correctness of existing protocols using the theory. Finally, the paper presents new protocols which require significantly less communication than previously proposed protocols in systems which do not provide hardware atomic broadcasting facilities.

A Case Study in Distributed Algorithm Design using Mixed Specifications

Beverly Sanders
ETH Zürich, Institut fuer Informatik, ETH Zentrum, CH 8092 Zuerich, Schweiz
Email: sanders@inf.ethz.ch

Formal proofs can significantly increase ones confidence in the correctness of distributed algorithms and encourage a helpful way of thinking about a problem which often yields some general algorithms. The structure of the proof can suggest a nice way of presenting algorithms after they have been designed. In the talk we use mixed specification to derive a previously published (incorrect) algorithm of mutual exclusion in distributed systems.

Correctness Proofs of Distributed Algorithms

Wolfgang Reisig
Institut für Informatik, Humboldt-Universität zu Berlin, 10099 Berlin, Germany
Email: reisig@informatik.hu-berlin.de

For *small* distributed algorithms, e.g. algorithms for mutual exclusion, wafe- and echo algorithms or round based algorithms, it will be shown that

- such algorithms are frequently simpler to handle than usually done, and particularly that

- formal correctness proofs are feasible at moderate expenditure.

This is achieved by a (Petrinet based) modeling technique that sticks to synchronization issues, avoiding variables and assignment statements. Atomic actions are shaped "If $\alpha_1, \ldots, \alpha_n$ are arrived, then is sent $\beta_1, \ldots, \beta_m$". Furthermore, a proof technique is suggested, fully exploiting the operational model. For state properties ("it is always true that ..."), powerful invariants can be constructed. Arguments on the static structure of algorithms allow for example to prove mutual exclusion for Peterson's algorithm, or the termination of all child-nodes in an echo-algorithm. For progress properties ("eventually it is true that ..."), a temporal logic based technique is suggested: Basic progress properties can be picked up from the static structure of the algorithm directly. More involved properties are derived by help of rules.

Various examples show the adequacy and generality of the suggested technique.

Some Key Issues in the Design of Distributed Garbage Collectors

Marc Shapiro
INRIA Rocquencourt and Cornell University
Email: mjs@cs.cornell.edu

David Plainfossé, Paulo Ferreira, Laurent Amsaleg
INRIA Rocquencourt

The design of garbage collectors combines both theoretical aspects (safety and liveness) and practical ones (such as efficiency, inobtrusiveness, ease of implementation, fault tolerance, etc.). Although distributed garbage collection is an instance of a consistency problem, practical designs often use weaker, "conservative" safety conditions, and/or weaker, "incomplete" liveness conditions. We report on our experience designing a number of distributed garbage collection algorithms in different settings, and explore the various design dimensions. The cost of each design alternative depends on the scale of the distributed system.

Speedup Anomalies in Distributed Computations

Reinhard Schwarz
University of Kaiserslautern, Germany
Email: schwarz@informatik.uni-kl.de

One of the motivations for distributed and parallel computing is the potential for speeding up the execution of an algorithm. Typically, it is assumed that a "perfect" realization and execution of a distributed program should yield a speedup which is linear in the number of processors being used. In practice, however, such performance gains can hardly ever be obtained. The discrepancy between the expected linear and the observed speedup is generally considered as a measure for the quality of the runtime environment, as well as of the quality of the parallelized algorithm. However, it can be shown that the underlying hypothesis that linear speedup is achievable is generally not justified, even if zero-delay

communication, zero-delay synchronization, and optimal load-balancing among the processors is assumed, and even if the parallelization of the algorithm does not cause any additional overhead. We present an analytical study which proves that tightly-coupled distributed computations – when executed on realistic operating system platforms – are subject to reduced availability of the runtime system. Three different models analyzing three different classes of application profiles are studied, and precise upper bounds on the achievable speedup are derived which reflect the dramatic speedup loss for certain types of parallel programs. The models give an intuitive explanation for the analytical results that are obtained.

Timing Failures and Timeliness Proofs in the Case of Real-Time Distributed Systems

Gerard LeLann
INRIA, B.P. 105, F 78153 Rocquencourt Cedex, France
Email: Gerard.Le_Lann@inria.fr

Most often, real-time centralized or distributed systems are designed considering a synchronous model of computation, i.e. lower and upper bounds are assumed to exist for communication and computation delays, and the values of such bounds is advance knowledge. Furthermore, upper bounds as well as knowledge of their values are also postulated for densities of failures.

The first part of this presentation is concerned with timing failures. It can be easily observed that, so far, timing failures have not received as much attention as crash or omission or byzantine failures. A timing failure is a (run-time) violation of the value of some postulated bound on communication or computation delays. Recently, a number of papers on "hard" real-time distributed systems have suggested that it is possible to perform on-line detection of timing failures, so as to transform them into omission failures, which we know how to tolerate or compensate in synchronous systems. In the case of communication delays, which have postulated bounds denoted *min* and *max*, a particular class of on-line schemes has been considered in these papers, which will be referred to as one-way timeliness checks. One-way timeliness checks are based on the simple idea that synchronized clocks can be used to measure interprocessor communication delays. Some precision e being maintained among clocks, these one-way timeliness checks simply consist in verifying whether the following condition holds for every incoming message: $min - e <$ measured delay $< max + e$. Contrary to claims made in published papers, such tests provide no "deterministic" (i.e. guaranteed) detection of timing failures. The solutions presented either are tautologies or are

incorrect or are blocking (i.e., no messages can be accepted). This is demonstrated in the talk. Conditions under which such tests do not reject messages indefinitely are also given.

The second part of this talk is concerned with timeliness proofs. A timeliness proof is a demonstration that, for any given task, response times of a computing system are comprised between two bounds, which match the earliest and latest deadlines specified for that task. Timeliness proofs are mandatory in the case of "hard" or "critical" real-time systems. The establishment of timeliness proofs is also notoriously difficult in the case of distributed systems, for the reason that advance knowledge of future arrivals of events (external and internal to a system) and associated timeliness constraints is very limited. This is the reason why, most often, such proofs are not given, even in papers that are concerned with "hard" real-time systems. When given, such proofs usually are based on very restrictive assumptions, e.g., postulating advance knowledge of the future (typically, periodic event arrivals), which is not acceptable in the case of "critical" systems, as such assumptions can be easily violated at run-time. Techniques that have proved useful in establishing timeliness proofs for the general case are those based on adversary arguments. In this talk, we present the basic ideas behind such proofs and examine the different types of adversaries that can be contemplated. An example of such proofs is given for the case of a multiaccess broadcast network which implements a deterministic variant of the basic Ethernet protocol.

Paradigms for Fault-Tolerant Services: from Practice to Theory

Keith Marzullo
University of California at San Diego, Department of Computer Science, 9500 Gilman Drive, La Jolla, CA 92093-0114, USA
Email: marzullo@cs.ucsd.edu

Message logging protocols are an integral part of a technique for implementing processes that can recover from crash failures. All message logging protocols require that, when recovery is complete, there be no *orphan processes*, which are surviving processes whose states are inconsistent with the recovered state of a crashed process. Orphans are either avoided through careful logging or are eliminated through a somewhat complex recovery protocol.

We give a precise specification of the consistency property "no orphan processes" From this specification, we describe how different existing classes of message logging protocols (namely *optimistic*, *pessimistic*, and a class that we call *causal*)

implement this property. We then propose a set of metrics to evaluate the performance of message logging protocols, and characterize the protocols that are *optimal* with respect to these metrics. Finally, starting from a simple protocol that relies on causal delivery order, we show how to derive optimal causal protocols that tolerate f overlapping failures and recoveries for a parameter $f : 1 \leq f \leq n$.

Lessons Learned from Building and Using Arjuna Distributed Programming System

Santosh K. Shrivastava
Department of Computing Science, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, England
Email: santosh.shrivastava@newcastle.ac.uk

Arjuna is an object-oriented programming system implemented entirely in C++, that provides a set of tools for the construction of fault-tolerant distributed applications. Arjuna exploits features found in most object-oriented languages (such as inheritance) and only requires a limited set of system capabilities commonly found in conventional operating systems. Arjuna provides the programmer with classes that implement atomic transactions, object level recovery, concurrency control and persistence. These facilities can be overridden by the programmer as the needs of the application dictate. Distribution of an Arjuna application is handled using stub generation techniques that operate on the original C++ class headers normally used by the standard compiler. The system is portable, modular and flexible. Arjuna has been used regularly by us for teaching distributed computing to undergraduate and graduate students. In addition, it has been used successfully for building a variety of distributed applications. This has given us useful insights into the strengths and weaknesses of Arjuna. This talk presents the overall design details of Arjuna and takes a retrospective look at the system based on the application building experience of users. Ideas on restructuring the system to overcome the limitations of the present design are presented.

Causally Consistent Observations of Distributed Systems

Friedemann Mattern
University of Saarbrücken, Germany
Email: mattern@cs.uni-sb.de

Observing an asynchronous distributed system is non-trivial not only from a technical point of view (instrumentation, intrusiveness), but also because of inherent conceptual problems: Since event notification messages sent to an observer are subject to unknown delays, it is generally not possible to observe all processes at the same instant in (global) time. This has serious consequences when "detecting" global predicates (such as deadlock or garbage) of distributed computations. Fortunately, there exist several means to guarantee that the observer gets at least a causally consistent view (i.e., a linearly ordered sequence of events with respect to the causality relation), namely using timestamps based on real time, on Lamport time, or on vector time.

Furthermore, we show that if two or more causally consistent observers observe a single computation, they may not agree on the value of a global predicate which, for example, makes the notion of global (or "distributed") breakpoints rather doubtful. We explain this phenomenon and shortly mention the concept of observer independent predicates (i.e., "objective facts").

A closely related problem is causal order message delivery. Here, each process within the system must get a causally consistent view of all messages addressed to it. By generalizing the vector timestamp realization of the previous problem in a canonical way, causal order message delivery can be implemented by timestamps based on "matrix clocks" (i.e., vectors of vectors). We show, however, that there exists a more space-efficient implementation based on input-output buffer processes with FIFO message queues that communicate in a handshake-way.

Detection of Global Predicates in Distributed Systems

Bernadette Charron–Bost
Laboratoire d'Informatique, LIX,
Ecole Polytechnique,
91128 Palaiseau Cedex, France
Email: charron@lix.polytechnique.fr

The major problem for detecting a global predicate in a distributed system is due to the fact that a distributed computation can be observed in many different (correct) manners and, according to the observation to which one refers,

one will claim that a given predicate is satisfied or not. Recently, Cooper and Marzullo addressed this issue by distinguishing the predicates which "possibly" hold, namely the predicates which hold in some observation, from those which "definitely" hold, i.e., which hold in all observations. In this way, for a given predicate ϕ, they define two new predicates referred to as "Possibly ϕ" and "Definitely ϕ".

The definition of the predicates Possibly ϕ and Definitely ϕ, where ϕ is a global predicate, leads to the definition of two predicate transformers *Possibly* and *Definitely*. We show that *Possibly* plays the same role with respect to *time* as the predicate transformers K_i in Knowledge Theory play with respect to *space*. Pursuing this analogy, we prove that local predicates are exactly the fixed–points of the K_i's while the stable predicates are the fixed–points of *Possibly*.

In terms of the predicate transformers *Possibly* and *Definitely*, we define a new class of predicates that we call *observer–independent* predicates and for which the detection of Possibly ϕ and Definitely ϕ is quite easy. We study this new class of predicates and we give some non–trivial examples of observer–independent predicates.

Cheaper Matrix Clocks

Frederic Ruget
Chorus Systemes, 6 Avenue Gustave Eiffel, 78182 Saint Quentin en Yvelines Cedex, France
Email: ruget@chorus.fr

Matrix clocks have nice properties that can be used in the context of distributed database protocols and fault tolerant protocols. Unfortunately, they are costly to implement, requiring storage and communication overhead of size $O(n^2)$ for a system of n sites. They are often considered a non feasible approach when the number of sites is large.

In this paper, we firstly describe an efficient incremental algorithm to compute the matrix clock, which achieves storage and communication overhead of size $O(n)$ when the sites of the computation are "well synchronized". Secondly, we introduce the *k-matrix clock*: an approximation to the genuine matrix clock that can be computed with a storage and communication overhead of size $O(kn)$. k-matrix clocks can be useful to implement fault-tolerant protocols for systems with crash failure semantics such that the maximum number of simultaneous faults is bounded by $k-1$.

These cheaper matrices will be useful within the CDB project. CDB is a debugger for distributed applications running on top of the CHORUS microkernel, with an execution replay facility. During record and replay phase, CDB maintains a distributed data base of all IPC objects of the application, that enables it to translate old identifiers to new identifiers (among other things). The implementation of the distributed database currently relies on the use of matrix clocks. We plan to optimize it by using these cheaper matrices.

Probing and Fault Injection of Distributed Systems

Farnam Jahanian
University of Michigan
Email: farnam@eecs.umich.edu

We present a technique for probing and fault injection of distributed protocols. The proposed technique, called "script-driven probing and fault injection", can be used for studying the behavior of distributed systems and for detecting design and implementation errors of fault-tolerant protocols.

We view a distributed protocol as an abstraction through which a collection of participants communicate by exchanging a set of messages. Each layer in a protocol stack, from the device layer to the application-level protocol, provides an abstract communication service to higher layers. In the proposed approach, a fault injection layer is inserted between any two layers in a protocol stack to filter and to manipulate the messages that are exchanged between the two layers. The fault injection layer supports the execution of deterministic or randomly-generated test scripts to probe the participants and to inject faults into the system under various failure models including daemon/link crash, send/receive omissions, and timing failures. In particular, by intercepting messages between two layers in a protocol stack, the fault injection layer can delay, drop, reorder, duplicate, and modify messages. Furthermore, by selective reordering of messages and spontaneous transmission of new messages, we were able to orchestrate a distributed computation into a particular path without instrumenting the protocol implementation.

To demonstrate the capabilities of this technique, we performed several experiments that studied the behavior of two protocols: the Transmission Control Protocol (TCP) and a Group Membership Protocol (GMP). These experiments identified three types of information about these protocol implementations: design decisions made by the developers, violations of protocol specifications, and design/implementation errors. The talk also discusses some of the ongoing theoretical and experimental work.

On the Role of Clock-Less Protocols in Real-Time Systems

Paulo Verissimo
Technical University of Lisboa, IST-INESC, Lisboa, Portugal
Email: paulov@inesc.pt

In a former paper, we have informally pointed out that clock-less and clock-driven protocols are both able to serve distributed time-critical applications, providing time boundedness and temporal order. The objective of this talk is to formalize a set of application correctness conditions equally valid for clock-less and clock-driven protocols. This will confirm our point about suitability of clock-less protocols. Then, we derive the correctness limits in real settings, to assess the applicability of clock-less protocols to real time-critical applications. We show that in an adverse environment, only clock-driven protocols are able to meet those criteria, and with limitations. In well-behaved environments, clock-less protocols may be as able as clock-driven ones. These results open the door to exploring new forms of communication in time-critical systems, such as supporting mixed event- and time-triggered operation. We expect that the our results will give insight to that problem.

RMP – A High Performance, Totally Ordered Multicast Protocol

Brian Whetten
Department of Computer Science, University of California at Berkeley, Berkeley, CA 94720, USA
Email: whetten@tenet.ICSI.Berkeley.EDU

This paper presents the Reliable Multicast Protocol (RMP). RMP provides a totally ordered, reliable, atomic multicast service on top of an unreliable multicast datagram service such as IP Multicasting. RMP is fully and symmetrically distributed so that no site bears an undue portion of the communication load. RMP provides a wide range of guarantees, from unreliable delivery to totally ordered delivery, to K-resilient, majority resilient, and totally resilient atomic delivery. These QoS guarantees are selectable on a per packet basis. RMP provides many communication options, including virtual synchrony, a publisher/subscriber model of message delivery, a client/server model of delivery, an implicit naming service, mutually exclusive handlers for messages, and mutually exclusive locks.

It has commonly been held that a large performance penalty must be paid in order to implement total ordering–RMP discounts this. On SparcStation10's on a 1250 KB/sec Ethernet, RMP provides totally ordered packet delivery to one destination at 842 KB/sec throughput and with 3.1 ms packet latency. The performance stays roughly constant independent of the number of destinations. For two or more destinations on a LAN, RMP provides higher throughput than any protocol that does not use multicast or broadcast.

Selling Heterogeneous RPC to the Masses

Richard D. Schlichting
Department of Computer Science, The University of Arizona, Tucson, Arizona 85721, USA
Email: rick@cs.arizona.edu

Heterogeneous Remote Procedure Call (RPC) allows computational components executing on different architectures or written in different programming languages to communicate. This talk describes our experience collaborating with computational scientists on using Schooner, an interconnection system that provides heterogeneous RPC facilities, to construct realistic scientific applications that span the Internet. Several of these applications are associated with the Numerical Propulsion System Simulation (NPSS) project, a NASA project designed to expand the use of simulation in the development of next generation jet engine technology. Among the conclusions are that heterogeneous RPC is a feasible tool for such applications, especially if care is taken to make its semantics resemble those of normal procedure calls as closely as possible.

Towards Open Service Environments

Kurt Geihs
Department of Computer Science, University of Frankfurt, P.O.Box 11 19 32, D-60054 Frankfurt, Germany
Email: geihs@informatik.uni-frankfurt.de

Rapid advances in computer and communications technologies have lead to a continuously progressing distribution of information processing. In the future

we will see very large distributed systems in which many independent service providers will provide a variety of different services to a large population of service consumers. Customers, looking for some service, will be able to choose from a number of similar service offers that differ e.g. in pricing and quality of service. We call such a distributed system an open service environment (OSE).

The focus of our contribution is on service types and service mediation. Characteristic for an open service environment will be the wide spectrum of service types and the inherently dynamic nature of the service configurations. Service providers will come and go, new service types will be introduced, and other service types may no longer be supported. In current systems, service types are defined by a unique name and a syntactical specification of the interface, written in some Interface Definition Language (IDL). The IDL-specification basically describes how the service can be used, but not what it is going to do.

We propose a semantic extension to the purely syntax-based IDL specification. This extension is based on declarative semantics and supports the service type matching without the need for an a priori agreement on the exact syntax of a particular service interface. The issue of type matching brings up two questions related to conformance: Do two type specifications conform to a common supertype and does a service implementation conform to its specification? We will show how these questions can be addressed based on established logic programming and conformance testing methodology, respectively.

System Structuring: A Convergence of Theory and Practice?

Jeff Kramer and Jeff Magee
Department of Computing, Imperial College of Science Technology and Medicine, 180 Queen's Gate, London SW7 2BZ, UK
Email: jk@doc.imperial.ac.uk

Our research work concerned with the provision of sound and practical means for the construction of distributed systems has resulted in the development of configuration languages as a means of describing and managing system structure. The most recent of these languages Darwin is an attempt to provide a general structuring tool of use in building systems from diverse components and diverse component interaction mechanisms. Darwin is in essence a binding language which can be used to define hierarchic compositions of interconnected components. Distribution is dealt with orthogonally to system structuring. The language allows the specification of both static structures and dynamic structures which evolve during execution. The central abstractions managed by Darwin are

components and services. Bindings are formed in a uniform manner by manipulating references to services.

Research work on the provision of a theory of concurrency and interaction has produced process algebras such as CSP, CCS and, most recently, the p-Calculus. The last of these is particularly interesting in its attempt to provide "...a direct description of systems which change their configuration" (Milner). System structure is described as a parallel composition of processes which share particular channels for interaction. Processes can be dynamically replicated as necessary. Interaction is defined by synchronous communication along channels, although asynchronous models have also been proposed for the calculus. The key idea is the ability to handle naming (or references) of all entities in a uniform manner, including the ability to pass channel names to other processes to form bindings. This provides the basis for the binding required for both static and dynamic configuration.

The correspondence between the treatment of names in the p-Calculus and the management of service references in Darwin has lead us to work on modelling Darwin programs in the calculus. In the talk, we describe the approaches in more detail and indicate the similarities and differences and their implications.

COCOON – Support for Information Sharing in CSCW Based on Causally and Totally Ordered Group Communication

Markus Kolland
Corporate Research and Development, Siemens Munich, Germany
Email: makol@km21.zfe.siemens.de

The term CSCW describes IT support for the collaboration of people, which work together as teams in geographically dispersed settings. The main objective of each CSCW system is to provide an efficient means of sharing information within work groups in the context of supporting a specific collaborative task. In order to really improve the efficiency of team work in these scenarios, CSCW applications must closely resemble the interaction patterns within human-human collaboration, bridging the boundaries of time and place while hiding the complexity of the underlying distributed environment (geographically dispersed and mobile hosts, heterogeneous hardware, software and communication, failures and concurrency).

Addressing the above objective in a CSCW application, however, turns out to be extremely difficult. The major source of complexity lies in implementing

a uniform information access, manipulation and consistency model for shared information under the given interactivity and distribution constraints. Current distributed system technology does not provide adequate support in this area. Most of the existing commercial platforms for open, distributed software systems like OSF/DCE, ORBIX or ANSAware don't address the required consistency requirements. Others like ISIS or TUXEDO provide a set of useful mechanisms for the interaction of (operating-) system oriented software components which is however largely different from the needs of human-human collaboration. The same observation holds with respect to state-of-the-art database technologies.

These observations led us to design COCOON, a CSCW support layer based on the causally and totally ordered group communication paradigm. COCOON uses existing distributed systems technology (ISIS/News) but with specific focus on the issues arising from information sharing mechanisms in CSCW. COCOON is an object-oriented application framework, which provides collaborative applications with the abstraction of a shared information space. In this context COCOON offers a communication model, an information model, a session model, and several CSCW specific consistency models to facilitate the implementation of CSCW systems. COCOON has been successfully implemented and validated within a real CSCW design project.

New Applications for Group Computing

Robbert van Renesse, Ken Birman,
Thorsten von Eicken and Keith Marzullo
Cornell University
Email: rvr@cs.cornell.edu

Group computing (GC) encompasses two important paradigms: membership and multicast communication. So far, these paradigms are best understood in two domains: fault-tolerant applications (replicated services), and data dissemination applications (e.g. stock trading). Yet GC also has potential in many other areas of distributed applications. We will discuss some new application domains, what approaches one may take when trying to apply GC here, and what implications this has for the GC paradigms themselves. In particular, we will discuss how our own new GC system, Horus, addresses these issues. The application domains on which we focus are parallel computing, multimedia systems, and real-time control systems.

The Horus system is software development effort that started in 1991 at Cornell University. An outgrowth of our prior work on the Isis Toolkit Horus advances

over Isis in a number of ways. The protocols and algorithms used are cheaper, cleaner, and more modular than in Isis, and they scale better. The architecture uses a layering technique that allows the user (or presentation layer) to construct special-purpose communication systems which exhibit subsets of the full functionality available in the Horus kernel. Moreover, the user interface has been implemented in a way that permits high degrees of parallelism both in Horus itself, and in the application program. This design has resulted in a uniquely flexible system that considerably outperforms Isis for many operations.

We will not discuss the group computing model in any detail. Briefly, the approach provides a way to form groups of processes dynamically, using primitives by which a process can join a group or leave a group. Failed processes are automatically removed from a group. In conjunction with these basic operations, reliable multicast facilities are provided by which processes can communicate with or within a group, offering varied ordering and stability properties. Naming services, security mechanisms, and other system infrastructure are provided to extend this basic model into a comprehensive distributed programming environment, somewhat like the tools available in RPC-oriented distributed computing environments.

We view parallel computing environments as an important potential application area for group computing. GC has been exploited for parallel computing by efforts such as ORCA, a programming language that supports distributed computations. Although we believe that parallel languages may open parallel computing to a large community, the majority of existing parallel codes operate over programming libraries, such as PVM. Recently, the parallel computing community developed a new standard for parallel computing support. This new system, called MPI, is based on the assumption that a static number of processors are assigned to each parallel execution. The processors communicate with each other through messages and barrier synchronization routines that scatter and gather interim results. GC may be applied here to manage the group of processors, and implement the barrier synchronization routines. The failure detection and recovery mechanisms of GC may be applied to add fault-tolerance, transparently to the application. This would be particularly useful for long-running parallel computations.

It is important in this application domain to provide minimal communication overhead. We are in the process of studying how the very low overhead *Active Messages* paradigm can be consolidated with GC. In the strategy we are pursuing, Horus is combined with a microkernel to form a minimal operating system for use on dedicated processing nodes.

Although GC has not been applied to multi-media applications previously, this seems also an obvious and promising idea. In multi-media applications, groups of two or more participants cooperate by communicating in a variety of ways at the same time. Beside the obvious audio and video, this may also include shared whiteboards, shared windows which can be used for, e.g., cooperative debugging,

and other types of cooperative applications.

GC can be applied here to do connection management and help with synchronization. Even with failing sites and channels, the standard GC membership protocols provide easy access at every site to the group and the status of its members. The ordered communication routines can be used to maintain consistent shared state between the members to reserve communication bandwidth, and synchronize the channels. It is important for such applications that the GC system support at least synchronized clocks and prioritized communication. In addition, the GC system has to be able to reserve resources for particular connections.

Perhaps more demanding than multi-media, real-time control systems require hard guarantees on latencies of communication, and detection and recovery of failures. Besides demands on GC protocols and use of resources, this requires the GC implementation to be embedded easily into a variety of situations. Such a GC system needs to be portable and customizable with different sets of protocols. An example of a GC system with real-time properties is the HAS system, and later used as a basis for parts of an air traffic control system being constructed at IBM for the United States FAA. We have started our own project to add real-time properties to Horus.

Object Framework for Operating System Serverization

Jishnu Mukerji
Novell Inc. USG, 190 River Road, Summit NJ 07901, USA,
Email: jis@summit.novell.com

We describe our experience in the development and evolution of a framework for providing object invocation and management service in a microkernel. In the ESPRIT III funded Ouverture project we have developed an object framework to aid in the serverization of the UNIX SVR4 ES/MP system onto the Chorus microkernel. This framework consists of a set of classes and associated derivation rules that allow easy incorporation of variations to the basic design. Implementers of individual servers can specify the interfaces they support using C++ extended with OMG-IDL-like constructs for specifying interfaces. This framework allows selection and use of an appropriate implementation of the invocation mechanism, based on the relative location of the invoker and the invokee as well as configuration features desired by the server (e.g. migration persistence etc.). Much work remains to be done in understanding what is the best way of precisely defining the innovation points and specifying the allowed extensions to the framework. Using such specifications to automatically check whether a new extension is consistent with the framework is an open area of inquiry.

This work draws heavily from the Chorus COOL project and has been jointly developed by Christian Jacquemot, Peter Jensen and Frederic Herrmann from Chorus Systemes and Philippe Gautron and Jishnu Mukerji of Novell USG.

Exposing Theory to Practice in Storage Systems

John Wilkes
Hewlett Packard Labs 1U13, P.O. Box 10490, Palo Alto, CA 94303-0969, USA
Email: wilkes@hpl.hp.com

Although it might seem at first sight that research in storage systems – by which I mean here disks and file systems – is grounded firmly on the "practical" side of this boundary, the intent of this short talk is to suggest that things may not be quite so clear.

Consider the humble disk drive, acting as a "slave" peripheral to a host system. Most system designers treat that disk as a simple, passive object that pretty much does what it is told, when it is told to do it. In fact, the disk drive is a complicated system in its own right: one that exploits caching, request reordering, and asynchronous operations to improve performance; and one that has a number of "housekeeping" tasks (like thermal recalibration) that can easily interfere with the "normal" operation of the device. Thus, disk drive behavior needs to be considered in systems that are exploiting causality (request ordering), or trying to achieve fault-tolerance (caching dirty data in volatile RAM), or real-time performance (asynchronous operations, request ordering). To make matters worse, the algorithms and policies used by the disk are rarely specified (although it is usually possible to disable them – but only at a considerable performance cost).

Disks are often embedded into larger ensembles – "disk arrays", which themselves have large amounts of firmware, much of whose behavior is poorly specified.

And finally, in the larger view of storage systems, multiple hosts (the clients of these disks and disk arrays) will themselves be caching data at the file and/or block level, with all the usual cache consistency problems that entails. In such systems, the complexity of the physical and logical interconnections (e.g., multi-ported disks connected to different I/O adapters on separate hosts), means that something more than ad-hoc techniques to address failures are required.

In short, the area of storage systems is probably ready to take advantage of the same kind of theoretical understanding that has improved our ability to build working distributed systems. Indeed, it is a suitable field for fruitful research across exactly the boundary under discussion at this workshop.

Dagstuhl-Seminar 9436:

List of Participants

Jean-Pierre **Banatre**
Université de Rennes
IRISA
Campus de Beaulieu
Avenue du Général Leclerc
F-35042 Rennes Cedex
France
jpbanatre@irisa.fr
tel.: +33-99.84.72.20

Kenneth P. **Birman**
Cornell University
Department of Computer Science
4106 Upson Hall
Ithaca NY 14853-7510
USA
birman@cs.cornell.edu

Bernadette **Charron-Bost**
Ecole Polytechnique
Laboratoire d'Informatique
F-91128 Palaiseau Cedex
France
charron@lix.polytechnique.fr
tel.: +33-1-69.33.34.84

Danny **Dolev**
The Hebrew University of Jerusalem
Institute of Computer Science
Givat Ram
91904 Jerusalem
Israel
dolev@cs.huji.ac.il
tel.: +972-2-584-116

Stefan **Fünfrocken**
TH Darmstadt
FB 20 Informatik
Frankfurter Str. 69 a
63293 Darmstadt
Germany

Kurt **Geihs**
Universität Frankfurt a.M.
Fachbereich 20 Informatik
Postfach 11 19 32
D-60054 Frankfurt
Germany
geihs@informatik.uni-frankfurt.de
tel.: +49-69-798-81 96

Rachid **Guerraoui**
Ecole Polytechnique de Lausanne
Departement d'Informatique
CH-1015 Lausanne
Switzerland
guerraoui@lse.epfl.ch
tel.: +41-21-693-52 72

Vassos **Hadzilacos**
University of Toronto
Computer Systems Research Institute
6 King's College of Road
Toronto Ontario M5S 1A1
Canada
vassos@cs.toronto.edu
tel.: +1-416-978-60 28

Farnam **Jahanian**
The University of Michigan
EECS Department
PO Box 1485
Ann Arbor MI 48109-2122
USA
farnam@eecs.umich.edu
tel.: +1-313-936-29 74

Dag **Johansen**
University of Tromso
Dept. of Computer Science
N-9037 Tromso
Norway
dag@cs.uit.no

Markus **Kolland**
SIEMENS ZFE BT SE 32
Zentralabt. Forschung und Entwicklung
Otto-Hahn-Ring 6
81739 München
Germany
makol@km21.zfe.siemens.de
tel.: +49-89-636-4 13 43

Sacha **Krakowiak**
Bull-IMAG
ZI de Mayencin
2 rue Vignate
F-38610 Gieres
France
krakowiak@imag.fr
tel.: +33-76-634834

Jeff **Kramer**
Imperial College of Science
Department of Computer Science
180 Queen's Gate
London SW7 2BZ
Great Britain
jk@doc.ic.ac.uk
tel.: +44-71-589-51 11

Gérard **Le Lann**
INRIA
Domaine de Voluceau
Rocquencourt
BP 105
F-78153 Le Chesnay Cedex
France
gerard.le-lann@inria.fr
tel.: +33-1-39.63.53.64

Jeff **Magee**
Imperial College of Science
Department of Computer Science
180 Queen's Gate
London SW7 2BZ
Great Britain
jnm@doc.ic.ac.uk
tel.: +44-71-594-82-69

Keith **Marzullo**
University of California at San Diego
Department of Computer Science
and Engineering
9500 Gilman Drive
La Jolla CA 92093-0114
USA
marzullo@cs.ucsd.edu
tel.: +1-619-534-37 29

Friedemann **Mattern**
TH Darmstadt
FB 20 Informatik
Frankfurter Str. 69 a
63293 Darmstadt
Germany
mattern@iti.informatik.th-darmstadt.de
tel.: +49-6151-16-37 09

Jishnu **Mukerji**
NOVELL USG
Room A- 121
190 River RD
Summit NJ 07901
USA
jis@summit.novell.com
tel.: +1-908-522-50 24

David **Powell**
LAAS-CNRS
7 Avenue du Colonel Roche
F-31077 Toulouse CEDEX
France
dpowell@lnas.fr
tel.: +33-61-33-62-87

Michel **Raynal**
Université de Rennes
IRISA
Campus de Beaulieu
Avenue du Général Leclerc
F-35042 Rennes Cedex
France
raynal@irisa.fr
tel.: +33.99.84.71.88

Wolfgang **Reisig**
Humboldt-Universität
Institut für Informatik
Unter den Linden 6
10099 Berlin
Germany
reisig@informatik.hu-berlin.de
tel.: +49-30-20 18 12 20

Mike **Reiter**
AT&T Bell Laboratories
Room 4F-637
Crawfords Corner Road
Holmdel NJ 07733
USA
reiter@research.att.com
tel.: +1-908-949-19 57

Robbert **van Renesse**
Cornell University
Department of Computer Science
4118 Upson Hall
Ithaca NY 14853-7510
USA
rvr@cs.cornell.edu
tel.: +1-607-255-10 21

Guy **Rilba**
Services Techniques de la
Navigation Aerienne
STNA/ATC
95 Rue Rochefort
F-91000 Evry
France
rilba@stnatls.stna7.stna.dgac.fr
tel.: +33-62.14.50.92

Frédéric **Ruget**
Chorus Systems SA
6 Avenue Gustave Eiffel
F-78182 St. Quentin en Yvelines
France
ruget@chorus.fr
tel.: +33-1-30-64-82-41

Beverly **Sanders**
ETH Zürich
Institut für Computersysteme
ETH-Zentrum
CH-8092 Zürich
Switzerland
sanders@inf.ethz.ch
tel.: 41-1-632-2830

André **Schiper**
Ecole Polytechnique de Lausanne
Department d'Informatique
CH 1015 Lausanne
Switzerland
schiper@lse.epfl.ch
tel.: +41 216934248

Rick **Schlichting**
University of Arizona
Department of Computer Science
Tucson AZ 85721
USA
rick@cs.arizona.edu

Reinhard **Schwarz**
Universität Kaiserslautern
FB Informatik
Postfach 3049
67653 Kaiserslautern
Germany
schwarz@informatik.uni-kl.de
tel.: +49-631-205-32 97

Marc **Shapiro**
INRIA
Domaine de Voluceau
Rocquencourt
BP 105
F-78153 Le Chesnay Cedex
France
shapiro@sor.inria.fr
tel.: +33-1-39.63.53.25

Santosh **Shrivastava**
University of Newcastle upon Tyne
Computing Laboratory
Newcastle upon Tyne NE1 7RU
Great Britain
Santosh.Shrivastava@newcastle.ac.uk
tel.: +44-91-222-80 38

Sam **Toueg**
Cornell University
Department of Computer Science
4106 Upson Hall
Ithaca NY 14853-7510
USA
sam@cs.cornell.edu
tel.: +1-607-255-91 97

Paulo **Verissimo**
INESC
Apartado 10 105
Rua Alves Redol 9/ 6
P-1017 Lisboa Codex
Portugal
paulov@inesc.pt
tel.: +351-1-3100-2 81

Werner **Vogels**
Cornell University
Department of Computer Science
4141 Upson Hall
Ithaca NY 14853-7510
USA
vogels@cs.cornell.edu
tel.: +1-607-255-9205

Khang **Vu Tien**
CAP SESA
8 Rue Paul Mesple
F-31036 Toulouse Cedex
France
vutien@capsogeti.fr
tel.: +33-61 31 53 30

Brian **Whetten**
University of California at Berkeley
Computer Science Division
589 Evans Hall
Berkeley CA 94720
USA
whetten@cs.berkeley.edu
tel.: +1-510-664-28 03

John **Wilkes**
Hewlett Packard Labs
P.O. Box 10490
Palo Alto CA 94303-0969
USA
wilkes@hpl.hp.com
tel.: +1 415 875 3568

Zuletzt erschienene und geplante Titel:

F. Meyer a.d. Heide, H.J. Prömel, E. Upfal (editors):
Expander Graphs, Random Graphs and Their Application in Computer Science, Dagstuhl-Seminar-Report; 87; 11.04.-15.04.94 (9415)

J. van Leeuwen, K. Mehlhorn, T. Reps (editors):
Incremental Computation and Dynamic Algorithms, Dagstuhl-Seminar-Report; 88; 02.05.-06.05.94 (9418)

R. Giegerich, J. Hughes (editors):
Functional Programming in the Real World, Dagstuhl-Seminar-Report; 89; 16.05.-20.05.94 (9420)

H. Hagen, H. Müller, G.M. Nielson (editors):
Scientific Visualization , Dagstuhl-Seminar-Report; 90; 23.05.-27.05.94 (9421)

T. Dietterich, W. Maass, H.U. Simon, M. Warmuth (editors):
Theory and Praxis of Machine Learning, Dagstuhl-Seminar-Report; 91; 27.06.-01.07.94 (9426)

J. Encarnação, J. Foley, R.G. Herrtwich (editors):
Fundamentals and Perspectives of Multimedia Systems, Dagstuhl-Seminar-Report; 92; 04.07.-08.07.94 (9427)

W. Hoeppner, H. Horacek, J. Moore (editors):
Prinzipien der Generierung natürlicher Sprache, Dagstuhl-Seminar-Report; 93; 25.07.-29.07.94 (9430)

A. Lesgold, F. Schmalhofer (editors):
Expert- and Tutoring-Systems as Media for Embodying and Sharing Knowledge, Dagstuhl-Seminar-Report; 94; 01.08.-05.08.94 (9431)

H.-D. Ehrich, G. Engels, J. Paredaens, P. Wegner (editors):
Fundamentals of Object-Oriented Languages, Systems, and Methods, Dagstuhl-Seminar-Report; 95; 22.08.-26.08.94 (9434)

K. Birman, F. Cristian, F. Mattern, A. Schiper (editors):
Unifying Theory and Practice in Distributed Systems, Dagstuhl-Seminar-Report; 96; 05.09.-09.09.94 (9436)

R. Keil-Slawik, I. Wagner (editors):
Interdisciplinary Foundations of Systems Design and Evaluation, Dagstuhl-Seminar-Report; 97; 19.09.-23.09.94 (9438)

M. Broy, L. Lamport (editors):
Specification and Refinement of Reactive Systems - A Case Study, Dagstuhl-Seminar-Report; 98; 26.09.-30.09.94 (9439)

M. Jarke, P. Loucopoulos, J. Mylopoulos, A. Sutcliffe (editors):
System Requirements: Analysis, Management, and Exploitation, Dagstuhl-Seminar-Report; 99; 04.10.-07.10.94 (9440)

J. Buchmann, H. Niederreiter, A.M. Odlyzko, H.G. Zimmer (editors):
Algorithms and Number Theory, Dagstuhl-Seminar-Report; 100; 10.10.-14.10.94 (9441)

S. Heinrich, J. Traub, H. Wozniakowski (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 101; 17.10.-21.10.94 (9442)

H. Bunke, T. Kanade, H. Noltemeier (editors):
Environment Modelling and Motion Planning for Autonomous Robots, Dagstuhl-Seminar-Report; 102; 24.10.-28.10.94 (9443)

W. Maass, Ch. v.d. Malsburg, E. Sontag, I. Wegener (editors):
Neural Computing, Dagstuhl-Seminar-Report; 103; 07.11.-11.11.94 (9445)