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.

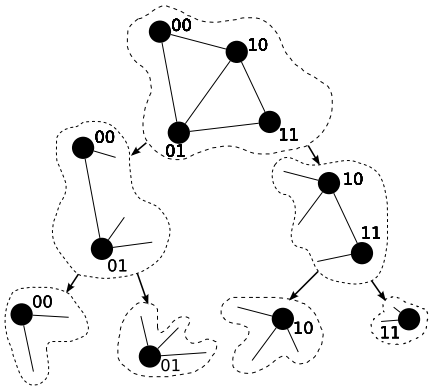


Figure 2 A street network graph with four nodes and its partitioning tree.

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.

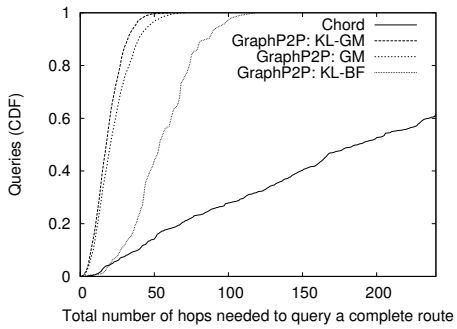
6.2 Graph-based peer-to-peer overlay

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.

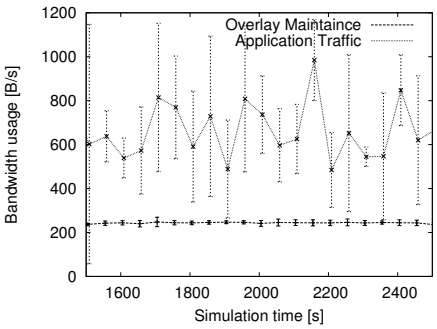
We were also interested in the bandwidth usage of our approach in order to see whether it can be realized over existing mobile access technologies. Fig. 3b shows the per-peer bandwidth required for maintaining the overlay and transporting the application data when the alternative routes approach with $n = 5$ 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.

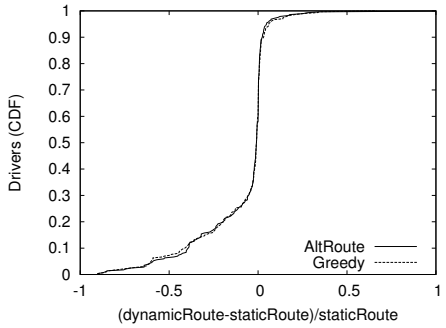


(a) Number of hops to answer a complete query.

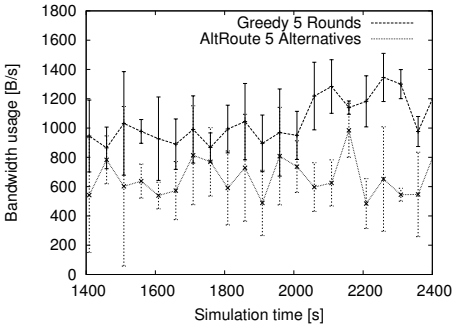


(b) Bandwidth usage (AltRoute 5 alternatives).

Figure 3 Performance of the street graph-based overlay.



(a) Relative travel time changes.

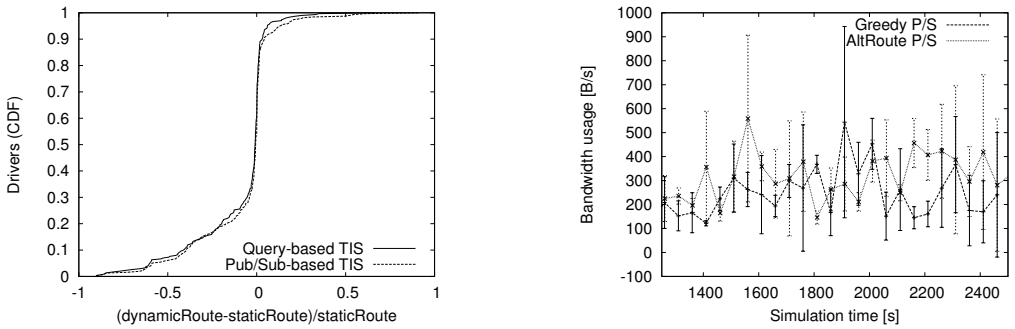


(b) Total bandwidth usage of the application.

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.