Adaptive Navigation of Vehicles in Congested Road Networks

Vasilis Verroios
University of Athens
Athens, Greece
v.verroios@di.uoa.gr

Konstantinos Kollias
University of Athens
Athens, Greece
k.kollias@di.uoa.gr

Panos K. Chrysanthis
University of Pittsburgh
Pennsylvania, USA
panos@cs.pitt.edu

Alex Delis
University of Athens
Athens, Greece
ad@di.uoa.gr

ABSTRACT

We examine the problem of routing vehicles in a road network where traffic congestion affects the time required to traverse an edge. We propose a fully distributed approach that uses only the computational resources and communication capabilities of vehicles and requires no fixed infrastructure or centralized servers. Our approach bases its operation on wireless ad-hoc communications and offers a protocol for alerting vehicles regarding traffic conditions in areas to be travelled through. Vehicles exchange estimations for the time required to reach areas of the road network and every vehicle dynamically determines the path it will follow based on estimations received from fellow travellers. Considering vehicles as selfish players in a game-theoretic framework, we relate the steady-state of our protocol with theoretical results. In this direction, our simulation focuses on confirming that the protocol adjusts rapidly to congestion variation, leading to a steady state. We also evaluate the performance of our protocol, compared to a system which uses static navigation to route vehicles, and as experimental results show, our approach achieves a better traffic distribution on the road network and provides improved average latency.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*

General Terms

Algorithms

1. INTRODUCTION

Modern vehicles are equipped with GPS-enabled Personal Digital Assistants (PDAs) that provide wireless communication capabilities of limited range and can form ad-hoc communication networks among themselves. Such Vehicular Ad Hoc Networks (VANETs) operate without any legacy client-server communication and allow a variety of applications including traffic monitoring [9]. Vehicles in such networks are able to cope with traffic congestion in urban environments, by selecting alternative paths, with smaller expected latency, towards their destination.

The traditional approach for navigating vehicles in urban areas, where numerous paths between a point of origin and a destination may exist, requires fixed infrastructure for gathering real time traffic data from vehicles and centralized servers for processing this data and suggesting paths to vehicles [1, 4, 8]. Besides the additional cost of installing and maintaining the fixed infrastructure, this approach might suffer from scalability problems depending on system-specific issues, the frequency with which real time data is produced, and the number of vehicles using the system.

The disadvantages of the centralized approach have been highlighted in a number of efforts [5, 6, 12–16] where several decentralized traffic information systems are proposed. These systems mainly deploy dissemination mechanisms that effectively dispatch global traffic information to all PDA-equipped vehicles. In this paper, we propose a distributed protocol that allows for vehicles to exploit selective information obtained from others traveling ahead in a trajectory towards a destination. Moreover, vehicles in our protocol consolidate this information to better alert and inform those following. In general, we assume that vehicles function in an ad-hoc fashion and exchange messages that allow them to create estimations for the time required to reach specific areas of a road network. Due to the limited range of its wireless communication, every PDA-equipped automobile depends on information received from other vehicles moving ahead which are yet found within range. In particular, a vehicle receives from anyone moving ahead estimations on how fast the transmitter can reach specific regions of the road network and what the transmitting vehicle experienced in its most recent movement. Once a vehicle receives such information, likely from multiple neighboring transmitters, it updates its own estimations for traveling to geographic regions and ultimately determines the next road segment to travel. The leading vehicles have shaped their own estimations recursively, following the same idea. Our approach provides for

ICPS'08, July 6–10, 2008, Sorrento, Italy.

much flexibility as automobiles can incrementally and on a demand-basis acquire information for the state of the road-network ahead. We accomplish this by utilizing condensed information regarding traffic conditions. At the same time, we avoid global and unqualified dissemination of pertinent information in contrast to existing approaches [5, 12–16].

The rest of the paper is organized as follows. Related work is discussed in Section 2. In Section 3, we present our assumptions for the application environment and the details of our protocol. In Section 4 we associate our protocol's behavior with game-theoretical results [10]. In Section 5 we evaluate our protocol in a simulation environment and present the derived results. In Section 6 we conclude the paper.

2. RELATED WORK

Most approaches for navigating vehicles through congested road networks use centralized servers or fixed infrastructure to monitor the traffic conditions. [4] presents a centralized system that predicts, based on the movement model developed in [3], the future position and speed of vehicles in each part of the road network, by using the starting time, initial speed and destination of each vehicle. Based on these data, the system computes a shortest path for each vehicle and assumes that the vehicle will actually follow this path, forming new predictions based on this assumption. [1] proposes the use of a hierarchical indexing method to represent the road network and a method for shortest-path computation. These methods in [1] can be used in centralized systems that constantly receive information about the state of the road network from applications such Traffic Message Channel (TMC) [2]. In [8], the area covered by the system is partitioned into cells and there is a server for each cell that maintains real time information about the traffic conditions in road segments. Road sensors provide this information to the server of the corresponding cell and the server disseminates this information to other servers. The vehicles communicate with the servers via a low bandwidth wireless network and are able to submit shortest path queries to the server of the cell in which they move.

The common theme in decentralized traffic information systems [5, 12–16] is that each vehicle records the time required to traverse predefined road segments and this traffic information is widely disseminated using the vehicles' communication capabilities. In contrast, in our approach, information exchanged between vehicles has a more compact form, that summarizes estimations for the traffic congestion in several road segments. [12] presents the architecture and mechanisms of a self-organizing traffic-information system that uses a "provoked" broadcast scheme for traffic information dissemination based on Inter-Vehicle Communication. In [16], a theoretical analysis of a dissemination mechanism is discussed and the effectiveness of a "zero infrastructure" traffic information system is evaluated. In [15], each participating vehicle adapts its transmission interval, according to its current traffic speed and disseminates the time latency of different road segments at different rates, according to the distance from its current position. *Grassroots* [5] also focuses on a dissemination mechanism that attempts to effectively propagate road segment traffic information in large-extent road networks. An alternative approach for decentralized traffic information systems is proposed in [6]. A vehicle that wishes to find the shortest path towards a destination, forms a query, that must be routed through other vehicles, all the way until it reaches a vehicle moving in the destination. Then, a response has to be routed back in a reverse manner to the vehicle that initiated the query.

3. MODEL AND ASSUMPTIONS

3.1 Model Description

Each vehicle is equipped with a *GPS-enabled PDA* that has *wireless communication capability* with limited range. The GPS provides vehicles with instant location and speed information. The PDA's storage maintains a directed graph, which represents the road network. The nodes of the graph are the junctions of the road network and the edges are one-direction road segments between two consecutive junctions. Figure 1 presents a part of Brooklyn's road network and the corresponding subgraph. The road network is partitioned into a set of *regions*, with every edge belonging to one region. The approach we follow in this paper, is to define geographic regions whose extent is small enough so that there is no significant benefit in intra-region routing and the destination of each vehicle is a specific such region. For example, in an urban area, each territory could be partitioned into a number of regions, proportional to the territory's congestion level and extent. The partition of the road network into regions can rely on statistical data and is fixed and stored in the PDA. Our protocol navigates a vehicle until it enters the region that contains its destination. Then, the vehicle can follow any path to its exact destination point without any significant gains or losses. For each edge e, the PDA maintains information about the length of the edge, the ending node, the outgoing edges of the ending node (termed *next outgoing edges* of e) and the region where it belongs. The location information, provided by the GPS, are sufficient to indicate the region, the edge and the exact distance that the vehicle has covered on the edge on which it moves. Throughout this paper we denote this information as *region_id*, *edge_id* and *edge_offset* respectively. Vehicles communicate by exchanging messages with other

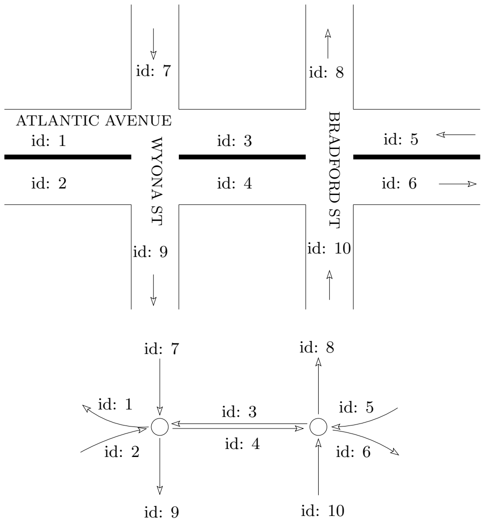

Figure 1: A part of a road network and the corresponding sub-graph.

vehicles within range, thus, radio-frequency communication is essential [2]. These messages contain estimations for the time required to reach every region in which the road network extends in and other useful information. We designate R for the *maximum transmission distance* of all vehicles.

The protocol is based on the construction of time estimations, using information provided by leading vehicles that has been recursively, yet asynchronously, constructed in the same way. Each vehicle maintains a table for each next outgoing edge of the edge it traverses. Each such table includes one time estimation per region of the road network. In addition, a basic table is used by each vehicle to summarize the best estimation for each region and the next outgoing edge, of the edge it traverses, that gives this estimation. Despite the fact that a given vehicle has a destination that lies in a specific region, it also maintains time estimations for all the other regions of the road network, which must propagate, implicitly through the protocol, to other vehicles with destinations that lie in different regions, so that they can be informed of the traffic conditions in distant areas.

All vehicles periodically broadcast their position and based on this location, vehicles that receive the broadcasted message are able to determine whether the broadcasting vehicle is a leading one and if there is any interest in sending a request to obtain information from it. Each vehicle is interested in receiving estimations from vehicles that move on the same edge, but with larger offset, and from vehicles that move on a next outgoing edge not far from the junction. Procedure *EvalInterest* (Figure 7), which is presented in Subsection 3.3, describes in detail the conditions under which a vehicle is interested in receiving estimations from another vehicle. Figure 2 indicates that vehicle 1 is interested in receiving estimations from vehicles 2, 3, that move on the same edge and have larger offsets and from vehicles 4, 5, 6, 7, 8, 9, 10 that move on next outgoing edges e_2, e_3, e_4 not far from the junction. Vehicle 1 is not interested in vehicles 11, 12, 13 that move on next outgoing edges, but far from the junction, in vehicles 14, 15, 16, 17 that move on the next incoming edge e_5 and in vehicles 18, 19 that move on e_1, but with smaller offset.

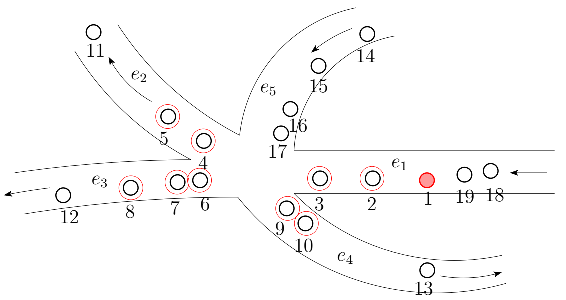

Figure 2: Encircled vehicles (2–10) are those of interest to vehicle 1.

When a leading vehicle o_l receives a request for estimations by a requesting vehicle o_r, it provides o_r with its basic table ET_l together with several data that will be used to estimate the time required by o_r to reach the current position of o_l. These data include the time t_l that was required for o_l to cover the distance between the two and the average speed s_{av_1} of the leading vehicles in range of o_l, t_l time ago. If s_{av_2} is the average speed of leading vehicles in range of o_r, then o_r estimates that it requires time $t_r = t_l \frac{s_{av_1}}{s_{av_2}}$ to reach the current position of o_l. We use the average speed of the PDA-equipped vehicles, that participate in the protocol's function, as an indication for the actual traffic congestion ahead of the vehicle o_r. In this way, the protocol remains functional even if the percentage of the PDA-equipped vehicles is low. The fraction $\frac{s_{av_1}}{s_{av_2}}$ expresses the percentage with which traffic congestion ahead of o_r has either increased or decreased by the time o_l was in the current position of o_r. If o_l were moving from a different edge, we would use the corresponding position time-wise on the edge in question. The average

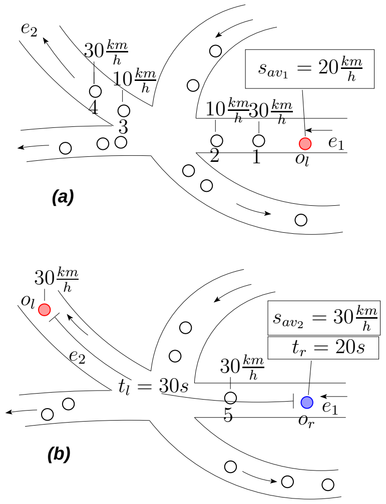

Figure 3: (a) o_l computes the average speed of the leading vehicles 1, 2, 3 and 4 found within range. (b) o_r estimates the time required to reach the new position of o_l.

speed s_{av_1} of leading vehicles 1, 2, 3, and 4 was $20\frac{km}{h}$ when o_l was in the current position of o_r as Figure 3(a) shows. In Figure 3(b), o_r computes the average speed $s_{av_2} = 30\frac{km}{h}$ of the leading vehicles o_l, 5, on e_2, e_1 and the time required to reach the position of o_l is $t_r = t_l \frac{s_{av_1}}{s_{av_2}} = 30\frac{20}{30}s = 20s$. How the average speed is computed is presented by procedures *TimeElapsed* and *GenerateEstimation* (Figure 7). t_r is added to the entries of ET_l to generate a new table of estimations. In Figure 4 the road network is partitioned in regions Reg_1, Reg_2, Reg_3 and the new table of estimations is produced by o_r by adding $t_r = 20s$ to the entries of ET_l. The entries of the derived table are used to update the entries of the tables maintained by o_r for the corresponding next outgoing edges. Details of this update are presented in Procedure *EstUpdate* (Figure 7), which is described in Subsection 3.3. The basic table is also updated by keeping the best estimation for each region.

The table of each next outgoing edge indicates the estimated time required by a vehicle to reach each possible region, if the vehicle selects the corresponding edge at the next junction. So, when a vehicle crosses a junction to a new edge, its basic table is substituted with the table of the new edge. A vehicle that moves within a specific region, maintains an estimation of value 0 for this region. Naturally, all estimations are reduced with time. Requests for information and the resulting updates are performed constantly by a vehicle while it moves on the road network. The data

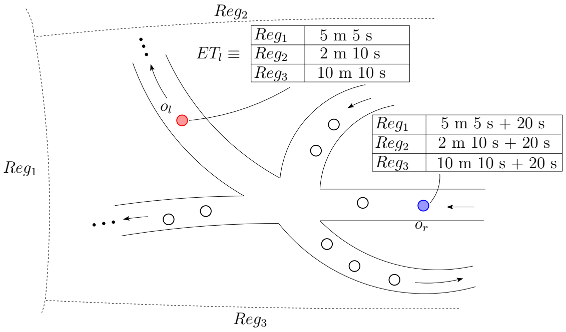

Figure 4: Deriving a table for the next outgoing edge where o_l is moving.

structures and the detailed description of the protocol are presented in the following subsections.

3.2 Data Structures

Each vehicle maintains two basic data structures. The first, termed *Est*, is a detailed representation of the tables of the next outgoing edges and the basic table discussed informally in the previous subsection. It consists of a list of tables of estimations, with one node *table[e]* for each outgoing edge e, of the edge the vehicle traverses, and one basic table of estimations, *basic_table*. Each row of a table refers to a specific region, with row i containing the estimation for the region with *region_id* i. In the *basic_table*, row i includes the minimum estimation (*est_time*) to travel to region i and the next outgoing edge that gives this minimum value (*best_edge*). For the next outgoing edge e_i, the estimations in *table[e_i]* refer to the time required to reach each region of the road network in case the vehicle selects e_i at the next junction.

The second data structure, *Distance_table*, describes the vehicle's recent movement by keeping the average speed of the leading vehicles and timing information. If we define a fixed distance quantum q, then each vehicle stores the following information for each of its last $\frac{2R}{q}$ positions (R is the assumed transmission range):

- The time elapsed (termed *time_elapsed*) since the vehicle was in that position.

- For the edge and the next outgoing edges, which correspond to that position, the average speed of the leading vehicles that were in range of the vehicle and were moving on the corresponding edge at the time. The average speed value *average_speed[0]* is kept for the edge that the vehicle was traversing and one value *average_speed[e]* is kept for each next outgoing edge e of that edge.

The $2R$ distance has been selected due to the fact that, as we will see in procedure *EvalInterest* (Figure 7), the maximum distance between a vehicle and a leading vehicle that is of interest to it, is $2R$. As shown in Figure 5, the structure *Est* of the vehicle, that moves on e_4, has a *basic_table* and 3 tables for the next outgoing edges e_1, e_2, e_3. The *Distance_table* structure has 4 *average_speed* rows, 1 for the current edge and 3 for its next outgoing edges, and 500 columns, one for each of the vehicle's $\frac{2R}{q}$ last positions ($R = 250$, $q = 1$).

Est								
basic_table			table[e_1]		table[e_2]		table[e_3]	
SegID	est_time	best_edge	SegID	est_time	SegID	est_time	SegID	est_time
1	18m 41.43s	1	1	18m 41.43s	1	19m 59.38s	1	21m 44.71s
2	22m 12.56s	2	2	29m 58.01s	2	22m 12.56s	2	56m 43.40s
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
87	06m 14.32s	1	87	06m 14.32s	87	13m 44.20s	87	08m 11.82s
88	11m 09.11s	3	88	48m 57.10s	88	31m 12.20s	88	11m 09.11s

Distance_table				
	1	2	...	500
time_elapsed	01m 52.41s	01m 51.47s	...	00m 00.45s
average_speed[0]	40 km/h	37 km/h	...	29 km/h
average_speed[1]	11 km/h	12 km/h	...	32 km/h
average_speed[2]	05 km/h	18 km/h	...	23 km/h
average_speed[3]	44 km/h	31 km/h	...	04 km/h

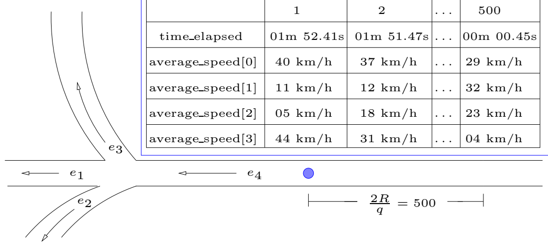

Figure 5: Vehicle data structures with $R = 250$ and $q = 1$.

For every period of length T, the *Distance_table* of each vehicle is updated. At the end of this period, if the vehicle covered an lq distance during the period, we delete the l oldest records of the *Distance_table* and we add l new records as described in the rest of the paragraph. In the *elapsed_time* field, we enter the values T, $T\frac{l-1}{l}$, $T\frac{l-2}{l}$, $\ldots$, $T\frac{1}{l}$. Each vehicle periodically broadcasts its instant speed and position. During each period, a vehicle keeps track of the speeds broadcasted by leading vehicles in range, taking its own speed into consideration, on the same or on next outgoing edges and computes the corresponding average speeds. These values are stored in the *average_speed* list of the *Distance_table*. Figure 6 presents an example of this update for $T = 0.9$ and a $3q$ distance covered during this period.

	Distance_table									
	1	2	3	4 → 1	5 → 2	...	50 → 47	48	49	50
time_elapsed				1m3s →1m3.9s	1m2.1s →1m 3s	...	0.9s →1.8s	0.9s	0.6s	0.3s
average_speed[0]				40 km/h	38 km/h	...	31 km/h	33 km/h	33 km/h	33 km/h
average_speed[e_1]				11 km/h	13 km/h	...	12 km/h	12 km/h	12 km/h	12 km/h
average_speed[e_2]				05 km/h	05 km/h	...	18 km/h	19 km/h	19 km/h	19 km/h
average_speed[e_3]				44 km/h	43 km/h	...	14 km/h	16 km/h	16 km/h	16 km/h
	obsolete data							NEW		

Figure 6: Update of *Distance_table* with $R = 100$, $q = 4$, $T = 0.9$, $l = 3$.

3.3 Protocol

Each vehicle periodically broadcasts its instant location, expressed as a pair (*edge_id*, *edge_offset*), and its instant speed. A vehicle o_2 that receives the transmission (*edge_id*$_1$, *edge_offset*$_1$, *speed*$_1$) of a vehicle o_1, decides, by calling *EvalInterest*(*edge_id*$_1$, *edge_offset*$_1$) (Figure 7), whether to request information from o_1 or not. Procedure *EvalInterest* returns *true* in case the two vehicles move on the same edge, with o_1 having greater offset, or in case o_1 moves on a next outgoing edge of o_2 and both are within distance R of the junction between them. In case *EvalInterest* returns *true*, a *SendDataReq*(*edge_id*$_2$, *edge_offset*$_2$) message is sent to o_1.

Once the request is received, o_1 sends a *DataResp* message to o_2, which includes the basic table E_1 of o_1, the time t_1 that took o_1 to cover distance equal to the distance between the two vehicles and the average speed s_1 of the leading vehicles in range of o_1 before that distance was covered. Procedure $TimeElapsed(edge\_id_2, edge\_offset_2)$ (Figure 7) describes how o_1 computes t_1 and s_1 based on its *Distance_table* structure, depending on whether o_2 is moving on the same edge as o_1 or not.

After receiving the $DataResp(E_1, t_1, s_1)$ message, o_2 calls procedure $GenerateEstimation(E_1, t_1, s_1, edge\_id_1)$ (Figure 7) and computes a temporary table E_i, which is used by procedure $EstUpdate(E_i, edge\_id_1)$ (Figure 7) to update the *Est* structure of o_2. *GenerateEstimation* predicts that time $t_1 \frac{s_1}{s_2}$ is required for o_2 to reach the position of o_1, with s_2 being the average speed of the leading vehicles in range of o_2. This time is added to the entries of E_1 in order to produce the aforementioned E_i. In case the congestion on the area between o_1 and o_2 has increased since o_1 was in position of o_2, the fraction $\frac{s_1}{s_2}$ will have a great value which indicates that o_2 requires much more time to reach the current position of o_1, than o_1 did. $EstUpdate(E_i, edge\_id_1)$ updates the estimations of the tables of the next outgoing edges by combining the current estimations with the newly formed ones of E_i. In case o_1 was on a next outgoing edge, *EstUpdate* only needs to update the table corresponding to this specific edge. In case o_1 and o_2 were on the same edge, *EstUpdate* has to update the tables of all next outgoing edges that give the best estimation for some region, according to E_i. The *basic_table* stores the best estimation for each region and the next outgoing edge that provides this estimation. *EstUpdate* uses a decay factor $a < 1$ in order to to reduce the influence of earlier estimations; a is 1 if no earlier estimations exist. The procedures presented in Figure 7 are given in a high level description and could be enhanced in order to deal with practical issues.

A vehicle upon approaching a junction, selects the next outgoing edge that provides the minimum estimation for its destination region. After crossing a junction to a new edge, the *basic_table* of the *Est* structure is replaced by the table of the edge that the vehicle followed. Since the next outgoing edges' tables don't have a *best_edge* field, once the vehicle begins traversing a new edge, it has empty *best_edge* values in the *basic_table*. The next outgoing edges' tables are empty when the vehicle passes to a new edge. As discussed, a vehicle that moves within a region has a fixed estimation of 0 for this region in its *basic_table*.

The fact that each vehicle is allowed to select any next outgoing edge leaves open the possibility that the algorithm will lead the vehicle to cross the same junction twice when moving towards a destination. Consider the case where a vehicle is informed that it should leave a major avenue to follow a smaller road due to heavy congestion on the avenue. Now consider that a traffic accident happens on the smaller road, increasing the road's traffic congestion. Thus, the algorithm might suggest that the vehicle should return to the major avenue. To avoid this possibility, since it may be undesirable to lead a driver to perform a circle under any circumstances, we could add restrictions so that, for a given destination region, each vehicle is allowed to select only from a predefined subset of the next outgoing edges of a junction. These restrictions could be based on the suggestions of a static navigation system or on the notion of "view" introduced in [1], which can be thought as the rectangular area whose diagonal is the line connecting the current position of the vehicle and a destination point. With these restrictions the protocol must be slightly adjusted so that the *basic table* does not include best estimations from next outgoing edges which are not allowed to be followed for a given region. Part of our experimental effort in Section 5 examines how the introduction of restrictions affects the protocol's performance.

When a vehicle reaches within $\frac{R}{4}$ of the next junction and has not traced any vehicles on a next outgoing edge, it forms estimations with an alternative mechanism. We apply a shortest path algorithm to determine the time required to reach a region through the specific edge. This is repeated for all regions. These estimations are used to update the tables, as if they were provided by a leading vehicle. The edge weights used by the algorithm express the expected time required to traverse each edge, and are fixed and stored in the PDA. Possibly an edge could have a different weight for each time during a given day or, alternatively, the algorithm's result could be multiplied by a factor expressing the expected congestion of the road network at the time. At the initiation phase of the protocol, this mechanism can be used to produce the very first estimations of the vehicles.

The proposed protocol functions effectively when each vehicle has leading vehicles within its maximum transmission range R. This holds, given the wireless communication capability of modern commercial devices (a few hundred meters), in congested urban environments. In the absence of traffic congestion, on the contrary, the maximum transmission range may not be enough. In that case, the shortest path mechanism that relies on stored data and is described in the previous paragraph could be sufficient.

It is interesting to see that a vehicle's computational load is not affected by the total number of vehicles in the road network or the level of traffic congestion, because a fixed frequency can be used for receiving/sending estimations from/to other vehicles. The volume of information exchanged between two vehicles and the cost for updating the estimation tables, increase linearly with the total number of regions. As a consequence, a scalability issue may arise, if the increase of the road network's extent leads to a large number of regions. We expect that the approach we proposed for defining the regions in the road network (see Subsection 3.1), produces an efficient total number of regions, for real environments. In Section 6, we mention a different approach for defining the regions in the road network, that addresses the aforementioned scalability issue.

4. STEADY STATE AND OPTIMALITY CONSIDERATIONS

We assume our road network is a directed graph G with a single source node s and a single destination node t. Constantly, new vehicles begin their trip from node s to node t. Our protocol can be thought as a distributed algorithm that routes each vehicle through a path of minimum latency, by indicating a next outgoing edge each time the vehicle approaches a junction. Actually, the algorithm suggests the same path to all vehicles until the congestion increases its latency so that it stops being the path of minimum latency. When this is detected, the algorithm routes the vehicles through the new path of minimum latency. This is repeated as long as a new path of minimum latency arises. Assume

Procedure *EvalInterest*(***edge_id***$_1$,***edge_offset***$_1$)

(∗ *my_edge_id, my_edge_offset* describe the current position of the vehicle that called the procedure. ∗)
if *my_edge_id* = *edge_id*$_1$
  then if *edge_offset*$_1$> *my_edge_offset*
      then return *true*
      else return *false*
distance_to_junction ←*my_edge_id.length*−*my_edge_offset*
if *edge_id*$_1$ ∈ *next_outgoing_edges*(*my_edge_id*)
  then if *distance_to_junction*< R **and** *edge_offset*$_1$ < R
      then return *true*
      else return *false*
return *false*

Procedure *TimeElapsed*(***edge_id***$_2$,***edge_offset***$_2$)

if *edge_id*$_2$ ≠ *myedge_id* (∗ If we move on a next outgoing edge of the requesting vehicle o_2. ∗)
  then *distance* ←*myedge_offset*+(*edge_id*$_2$*.length*−*edge_offset*$_2$) (∗ The distance between o_1 and o_2 ∗)
    $j \leftarrow \frac{2R-distance}{q}$
    t_1 ←*Distance_table[j].time_elapsed*
    av_1 ←*Distance_table[j].average_speed[0]*
    av_2 ←*Distance_table[j].average_speed[myedge_id]* (∗ *myedge_id* was a next outgoing edge when we were at position j ∗)
    $s_1 \leftarrow \frac{av_1+av_2}{2}$
  else (∗ If we move on the same edge with the requesting vehicle o_2. ∗)
    distance ←*myedge_offset*−*edge_offset*$_2$
    $j \leftarrow \frac{2R-distance}{q}$
    t_1 ←*Distance_table[j].time_elapsed*
    s_1 ←*Distance_table[j].average_speed[0]*
return (t_1, s_1)

Procedure *GenerateEstimation*(E_1, t_1, s_1, ***edge_id***$_1$)

$j \leftarrow \frac{2R}{q}$ (∗ Current position ∗)
if *edge_id*$_1$ ≠ *myedge_id* (∗ If o_1 moves on a next outgoing edge ∗)
  then av_1 ←*Distance_table[j].average_speed[0]*
    av_2 ←*Distance_table[j].average_speed[edge_id*$_1$*]* (∗ *edge_id*$_1$ is a next outgoing edge ∗)
    $s_2 \leftarrow \frac{av_1+av_2}{2}$
  else (∗ If we move on the same edge with o_1. ∗)
    s_2 ←*Distance_table[j].average_speed[0]*
for all regions k
    E_i*[k].est_time* ←E_1*[k].est_time*+$t_1 \frac{s_1}{s_2}$
    E_i*[k].best_edge* ←E_1*[k].best_edge*
return E_i

Procedure *EstUpdate*(E_i, ***edge_id***$_1$)

if *edge_id*$_1$ ≠ *myedge_id* (∗ If o_1 moves on a next outgoing edge ∗)
  then for all regions k
      (∗ Update the table of *edge_id*$_1$ for all regions k ∗)
      Est.table[edge_id$_1$*][k].est_time* ←$(1-a)\cdot$ *Est.table[edge_id*$_1$*][k].est_time* + $a\cdot$ E_i*[k].est_time*
  else (∗ If we move on the same edge with o_1. ∗)
    (∗ The next outgoing edge E_i*[k].best_edge* gives the best estimation for region k. Update this edge's table only for k. ∗)
    for all regions k
      Est.table[E_i*[k].best_edge][k].est_time* ←$(1-a)\cdot$ *Est.table[*E_i*[k].best_edge][k].est_time* + $a\cdot$ E_i*[k].est_time*
  for all regions k
    (∗ For each region k, the basic table will have the minimum estimation given by a next outgoing edge ∗)
    Est.basic_table[k].est_time ←min{next outgoing edge e: *Est.table[e][k].est_time*}
    e_m ←the edge that gives the minimum above
    Est.basic_table[k].best_edge ←e_m

Figure 7: Protocol procedures.

that the algorithm's time estimations adapt to congestion variation, such that there is a negligible delay between the time that the path of minimum latency switches and the time when the algorithm suggests the new path. Based on this assumption, our algorithm avoids a situation where several paths periodically take over being the path of minimum latency and converges to a state where all paths followed by vehicles are estimated to have the same latency. Assuming that our estimations are precise, every path that is followed by a fraction of the vehicles will have the same latency. Note that there may be paths, not followed by any vehicle, that have greater latency than the ones followed by the vehicles. Part of our experimental studies in the next section, focuses on determining the degree to which the aforementioned assumptions hold and the convergence to a steady state is achieved.

Each vehicle applies our algorithm in order to select a path of minimum latency towards the destination node t and does not care about the average latency of all vehicles. Intuitively, it seems that in the aforementioned steady state, vehicles are distributed in a uniform manner among the available paths from source to destination, such that all paths have the same total latency. It would be interesting to examine the difference between the minimum possible average latency (perhaps achieved by treating several vehicles in an unfair manner) and the average latency in our algorithm's steady state. In this direction, the algorithm's steady state can be associated with the game-theoretical [10] results presented in [11]. The *selfish routing* model of Roughgarden and Tardos consists of a directed graph and an infinite number of players that wish to route a negligible amount of flow from the source node to the destination node, trying to minimize the total latency of the path each one selects. The latency on each edge is a linear function of the flow passing through it and the strategy of a player is a path from source to destination. The authors assume that the outcome will be a *Nash equilibrium*. A Nash equilibrium is an outcome such that no player benefits from switching strategy unilaterally. In this setting, all Nash equilibria are such that all paths from a source node to a destination node, that carry positive flow, have the same latency. All other paths have greater or equal latency, even for flow equal to 0. This situation is similar to the steady state of our algorithm if we consider the road network as the directed graph and the vehicles as the selfish players. In [11], the authors evaluate the efficiency of the Nash equilibrium state using the notion of the *price of anarchy* [7], which is the worst case ratio of a defined social cost in a Nash equilibrium to the optimal social cost. The social cost used in [11] is the average latency of all players and the main result is that the price of anarchy for the selfish routing setting is $\frac{4}{3}$. The result also holds for the general case of multiple source-destination pairs.

The assumed ideal behavior of our algorithm and the convergence to a steady state indicate that our algorithm provides players with sufficient information, that allows them to choose their paths as if they had complete knowledge of the road network's state. As implied by the theoretical result of [11], the selfish path selection by each player leads to an outcome that, in the worst case, is not far from the optimal, regarding the average latency of all vehicles.

5. EXPERIMENTAL EVALUATION

Our experimental section consists of two categories of experiments. In the first subsection, we examine how close the protocol's behavior comes to the ideal behavior described in Section 4, in a network where all vehicles have a common source and a common destination. In the second subsection we evaluate our protocol's performance by comparing it with a static navigation system, using several metrics. In both subsections we have used the following simplifications in our simulation. We assume that message transmission times are negligible, that wireless communication is not affected by any kind of obstacles and that the maximum transmission range is $150m$.

5.1 Convergence to a Steady State

In this subsection we examine the behavior of our protocol and whether it converges to a steady state. The network used in this subsection's experiments consists of 30 edges, 19 nodes and 11 paths from the source node s to the destination node t. The corresponding graph is pictured in Figure 8. The time required to traverse an edge of this graph, is defined by a linear function of the number of vehicles that traverse this edge, at a given time. The parameters of the linear function, depend on the length and the width of the corresponding edge. Each vehicle begins its movement from a random position on the starting edge e_s, which is common

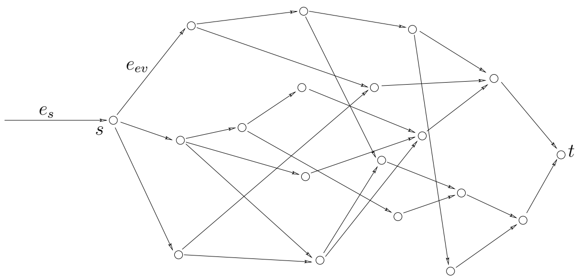

Figure 8: Graph used in experiments of Subsection 5.1.

for all vehicles, and wishes to reach the destination t, which is also common for all vehicles, by following one of the possible paths. The time interval between the beginning of two consecutive vehicles' movement follows the uniform distribution. Our simulation uses a fixed number of vehicles and, for each vehicle that reaches the destination t, a new one immediately starts its movement from a random position on the starting edge e_s. The experiment is completed once the total number of arrivals reaches a predefined limit. In order to establish whether the system converges to a state where all possible paths have approximately the same latency, we periodically measure the latency of each path and for the measured values we compute the relative standard deviation, RSD ($=100\% \frac{\text{standard deviation}}{\text{mean value}}$). In the experiment, whose results appear in Figure 9 the road network is constantly traversed by $5,000$ vehicles and measurements of path latencies are taken for every unit of time. The RSD is high for the 20 first time units, where it reaches above 50%. Then, RSD begins to descend, drops below 10% and stabilizes between 1% and 6%. We observe that the steady state of our protocol does not achieve the ideal 0% for the RSD, which would imply the exact same latency for every path. This is due to the fact that the elapsed time between the moment that the path of minimum latency is changed and the moment that the protocol begins suggesting the new path

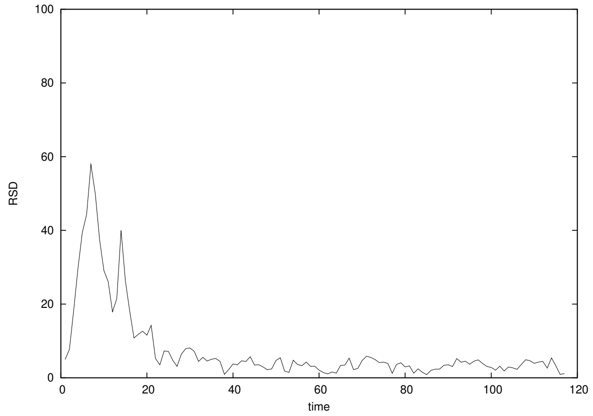

Figure 9: Experiment with single source and destination and no event triggered.

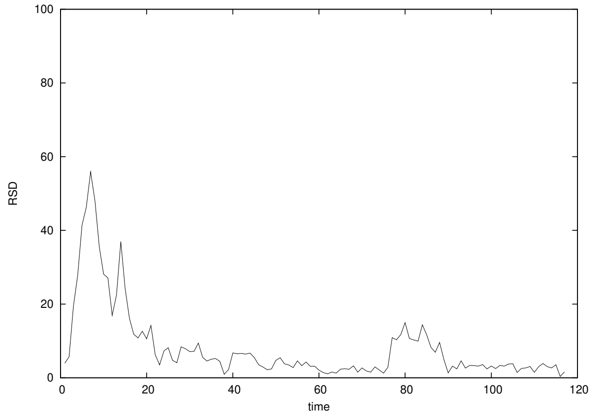

Figure 11: Experiment with single source and destination and event triggered at time 75.

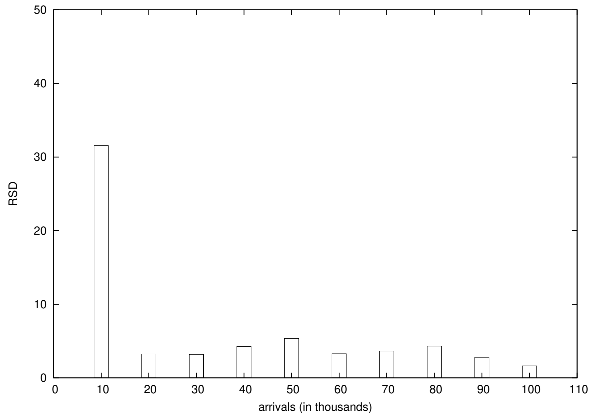

Figure 10: Experiment with single source and destination and no event triggered.

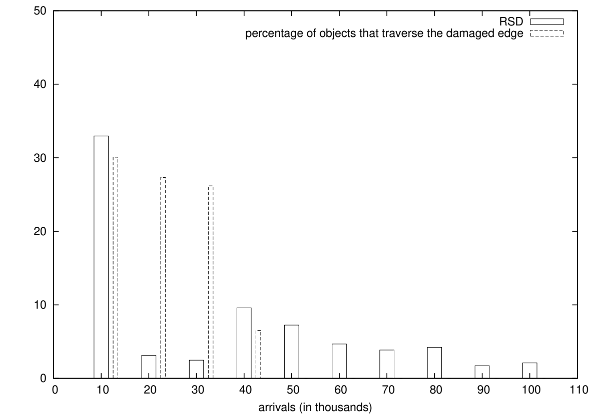

Figure 12: Experiment with single source and destination and event triggered during the fourth group of arrivals.

is not negligible. In Figure 10, for $5,000$ vehicles, we have partitioned the $100,000$ total arrivals in groups of $10,000$ consecutive arrivals. For each arrival, we measure the time that was needed for the vehicle's movement from source to destination. For the values that correspond to a group of $10,000$ arrivals we compute the RSD, in order to examine if vehicles tend to have the same source-destination latency and thus evaluate the estimations' precision. For the first group of $10,000$ arrivals, the RSD is high and approaches 30%. For the next and the following groups, as the steady state is approached, the RSD drops to 3%-6%.

The experiment, whose results appear in Figures 11 and 12 is similar to the experiment mentioned above, with the difference that we have triggered a virtual event at time 75, causing high congestion on one of the edges, e_{ev} (see Figure 8), thus deactivating a subset of the possible paths. We performed this experiment in order to examine the consequences that a traffic accident would have on our protocol's behavior. Figure 11 corresponds to Figure 9 and Figure 12 corresponds to Figure 10. RSD values of Figure 11 from time 75 and later are produced only from the paths that do not include e_{ev}. As Figure 11 indicates, up to time 75, our protocol had achieved a steady state, with the RSD not exceeding 6%. The event causes an increase to the values of the RSD, which exceeds 10% until the protocol leads the system to a steady state once again, after time 90. In Figure 12, the additional boxes for each group of arrivals, present the percentage of vehicles that followed a path which includes e_{ev}. As previously, the RSD for the first $10,000$ arrivals is close to 30% and for the next two groups of arrivals, it drops below 5%. Before the event, 25%-30% of the vehicles follow paths that include e_{ev}, which is normal since these paths are approximately $\frac{1}{4}$ of the possible paths. The event happens during the fourth group of arrivals and causes the increase of the RSD at approximately 10% and decrease in the percentage of vehicles passing through e_{ev} at 6%. After the sixth group of arrivals, the RSD drops to values not exceeding 5%, while from the fifth group of arrivals and later, no vehicle passes through e_{ev}.

5.2 Performance Evaluation

In this subsection we compare the performance of our protocol to a system of static navigation, whose suggestions rely on already stored statistical data about traffic conditions in the road network. We also examine how our protocol is affected by the addition of restrictions to the set of next outgoing edges of a junction, for each destination region. These restrictions aim to prevent the possibility that a vehicle might cross the same junction twice while moving towards its destination (see Subsection 3.3). For this purpose, the experiments also evaluate another version of our protocol, that uses restrictions.

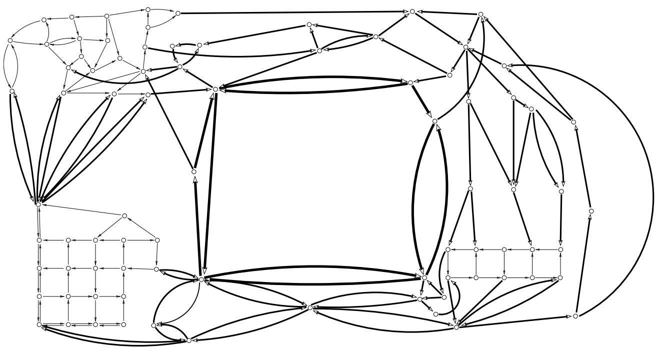

Figure 13: Graph used in experiments of Subsection 5.2.

In our experiments, we used a road network with 171 edges. The corresponding graph is pictured in Figure 13. The edges of this graph derive from three different road categories, major avenues (extra bold edges in Figure 13), secondary avenues (bold edges), and common streets (thin edges). The time required to traverse an edge and the speed of the vehicles that traverse this edge, depends on the number of vehicles located in this edge at a given time. Different functions are used for each road category. Each vehicle, using the uniform distribution, randomly selects a starting point and a destination point, which define a starting region and a destination region. Once the vehicle reaches its destination point, it randomly selects a new destination, thus, the total number of vehicles traversing the network remains constant during the experiment. The total number of vehicles expresses the road network congestion. The experiment is completed once a predefined number of arrivals is reached. The metrics used in these experiments include the average value, the maximum value and the RSD of the latency experienced by the vehicles from their source to their destination. These metrics are measured for each pair of source region - destination region and then a mean value is computed for all source-destination pairs.

Figure 14 depicts the average latency, when the road network is constantly traversed by $1,000$, $2,000$, $4,000$ or $8,000$ vehicles. As the congestion of the road network increases, we observe that the benefit of applying our protocol instead of static navigation increases significantly. The average latency for $8,000$ vehicles is almost triple when vehicles use the static navigation system. The performance of our protocol's version with restrictions is slightly worse, but close to the one without restrictions for this metric. Thus, we conclude that the addition of restrictions attaches only a very limited cost in order to prevent the possibility that the protocol leads a vehicle to cross the same junction twice. The values computed for the version with restrictions are also worse for the metric of the RSD for $2,000$, $4,000$ and $8,000$ vehicles, as depicted in Figure 15. It seems that by restricting some of

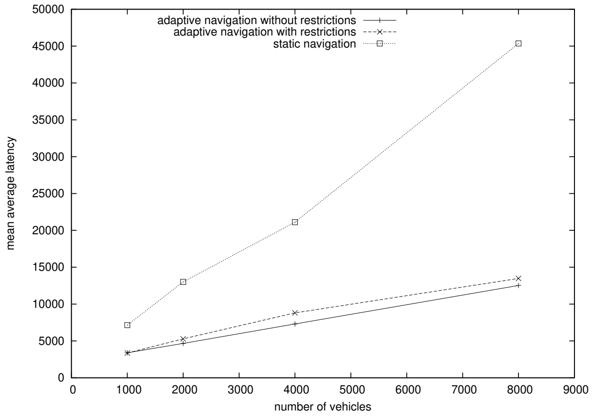

Figure 14: Mean average latency.

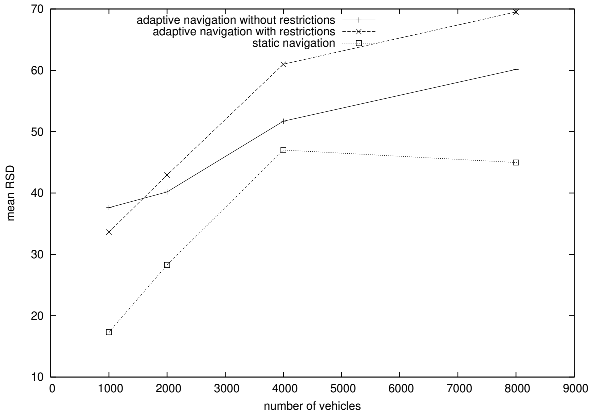

Figure 15: Mean relative standard deviation.

the vehicles from choosing any next outgoing edge of a junction, towards a destination, the deviation of the latency experienced by the vehicles increases for high congestion. For $1,000$ vehicles the RSD is lower when we apply restrictions, which could be explained by the fact that there is a small fraction of vehicles that are lead by the protocol to cross the same junction twice when no restrictions are applied. The static navigation system achieves better RSD values for any level of congestion. This is expected since all paths from one region to another are identical, for their largest part, as indicated by the statistical data of the static navigation system. Figure 16 presents the performance of the 3 navigation systems relative to the maximum latency metric. Similarly to the RSD metric, the version of our protocol that uses restrictions performs better than the one without restrictions for $1,000$ vehicles and slightly worse for $2,000$, $4,000$ and $8,000$ vehicles. Our protocol performs better than the static navigation system, for the maximum latency metric, with the difference being greater as congestion increases.

A direct observation is that the performance difference between our protocol and static navigation becomes greater as congestion increases, for the mean average latency and maximum average latency metrics. This can be explained by the fact that when congestion is increased, the distance

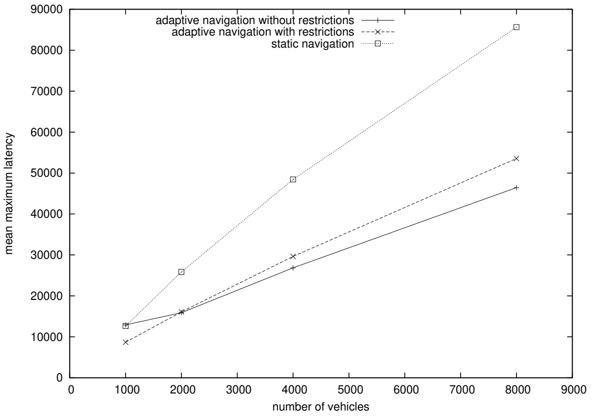

Figure 16: Mean maximum latency.

between vehicles is reduced and information can be propagated effectively throughout the road network. Moreover, the vehicles' computational load is not affected by the total number of vehicles.

6. CONCLUSION AND FUTURE WORK

In this paper we presented a distributed protocol that facilitates the dynamic navigation of vehicles in congested urban road networks. Our approach requires no centralized servers or fixed infrastructure as vehicles become aware of traffic conditions in distant parts of the road network by receiving information by fellow travellers geographically located ahead of them in the direction of movement. To this effect, leading vehicles provide estimations for the time required to reach specific areas of the road network. Experimental results show that our approach adjusts rapidly to congestion variation and achieves a better traffic distribution on the road network, compared to a static navigation technique, yielding improved average and maximum latency.

In our approach, we partition road networks into small regions, such that the routing within a region has no significant benefit. This one-level fragmentation may lead to large number of regions and furnish overheads as far as the transport and processing of protocol requisite data is concerned. We plan to investigate the effectiveness of adopting regions organized in a hierarchy to address potential scalability issues as far as wide geographic areas is concerned. In this hierarchy, a vehicle with a destination region of level j must first be navigated successively to the regions of level 1, 2, $\ldots$, j-1 that all include its destination point. Although our protocol can be effectively applied to congested road networks, it may become an under-performer in extreme situations where portions of the road network may experience very low traffic density and at the same time at least one congestion point may have been formed. In this context, we plan to extend our protocol with a specialized dissemination mechanism that harnesses information from oncoming vehicles as well.

7. REFERENCES

[1] Y. Bai, Y. Guo, X. Meng, T. Wan, and K. Zeitouni. Efficient Dynamic Traffic Navigation with Hierarchical Aggregation Tree. *8th Asia Pacific Web Conference*, Harbin, China, January 2006.

[2] H. L. Bertoni. *Radio Propagation for Modern Wireless Systems*. Prentice Hall, Upper Saddle River, NJ, 2004.

[3] H. D. Chon, D. Agrawal, and A. E. Abbadi. Query Processing for Moving Objects with Space-time Grid Storage Model. *Int. Conf. on Mobile Data Management*, Singapore, January 2002.

[4] H. D. Chon, D. Agrawal, and A. E. Abbadi. FATES: Finding A Time dEpendent Shortest path. *Int. Conf. on Mobile Data Management*, Melbourne, Australia, January 2003.

[5] S. Goel, T. Imielinski, K. Ozbay, and B. Nath. Grassroots - a Scalable and Robust Information Architecture. Technical Report DCS-TR-523, Department of Computer Science, Rutgers University, 2003.

[6] E. S. Kim, S. Y. Hwang, and K. J. Li. Arrival Time Dependent Shortest Path by On-Road Routing in Mobile Ad-Hoc Network. *4th Int. Workshop on Web and Wireless Geographical Information Systems*, Goyang, Korea, November 2004.

[7] E. Koutsoupias and C. H. Papadimitriou. Worst-case Equilibria. *16th Annual Symposium on Theoretical Aspects of Computer Science*, Trier, Germany, March 1999.

[8] K. Y. Lam, E. Chan, T. W. Kuo, S. W. Ng, and D. Hung. RETINA: A REal-time TraffIc NAvigation System. *ACM Int. Conf. on the Management of Data*, Santa Barbara, CA, June 2001.

[9] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. TrafficView: A Scalable Traffic Monitoring System. *Int. Conf. on Mobile Data Management*, Berkeley, CA, January 2004.

[10] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, Boston, MA, 1994.

[11] T. Roughgarden and E. Tardos. How Bad is Selfish Routing? *Journal of the ACM*, 49(2):236–259, 2002.

[12] L. Wischhof, A. Ebner, and H. Rohling. Information Dissemination in Self-organizing Intervehicle Networks. *IEEE Transactions on Intelligent Transportation Systems*, 6(1):90–101, 2005.

[13] L. Wischof, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. Adaptive Broadcast for Travel and Traffic Information Distribution Based on Inter-Vehicle Communication. *IEEE Intelligent Vehicles Symposium*, Columbus, Ohio, June 2003.

[14] L. Wischof, A. Ebner, H. Rohling, M. Lott, and R. Halfmann. Sotis - a self-organizing traffic information system. *57th IEEE Semiannual Vehicular Technology Conference*, Jeju, South Korea, April 2003.

[15] H. Xu and M. Barth. An Adaptive Dissemination Mechanism for Inter-Vehicle Communication-Based Decentralized Traffic Information Systems. *IEEE Intelligent Transportation Systems Conference*, Toronto, Canada, September 2006.

[16] A. Ziliaskopoulos and J. Zhang. A Zero Public Infrastructure Vehicle Based Traffic Information System. *Transportation Research Board's 2003 Annual Meeting*, Washington, D.C., January 2003.