VITP: An Information Transfer Protocol for Vehicular Computing [*]

Marios D. Dikaiakos[†]
Dept. of Computer Science
University of Cyprus
Nicosia, CY1678, Cyprus
mdd@ucy.ac.cy

Saif Iqbal
Dept. of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
saiqbal@cs.rutgers.edu

Tamer Nadeem[‡]
Dept. of Computer Science
University of Maryland
College Park, MD, USA
nadeem@cs.umd.edu

Liviu Iftode
Dept. of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
iftode@cs.rutgers.edu

ABSTRACT

Recent advances in wireless inter-vehicle communication systems enable the development of Vehicular Ad-Hoc Networks (VANET) and create significant opportunities for the deployment of a wide variety of vehicular applications and services. In this paper, we introduce the Vehicular Information Transfer Protocol (VITP), an application-layer communication protocol, which is designed to support the establishment of a distributed, ad-hoc service infrastructure over VANET. The VITP infrastructure can be used to provide location-based, traffic-oriented services to drivers, using information retrieved from vehicular sensors and taking advantage of on-board GPS navigation systems. In this paper, we present the key design concepts of the protocol and the infrastructure, the protocol specification, simple examples of protocol interactions that support driver inquiries, and a simulation study of VITP performance properties.

Categories and Subject Descriptors: C.2, C.2.2

General Terms: Design, Performance

Keywords: Vehicular Computing, Location-based Services, Middleware

[*]This work is supported in part by the NSF under ITR ANI-0121416 and its supplement for international collaboration CNS-0443881.

[†]Visiting the Dept. of Computer Science, Rutgers University.

[‡]Currently with Siemens Corporate Research, Princeton, NJ 08540, USA. Email: tamer.nadeem@siemens.com

1. INTRODUCTION

In the last couple of years, Inter-Vehicle Communication (IVC) has emerged as a promising field of research, where advances in Wireless and Mobile Ad-Hoc NETworks can be applied to real-life problems and lead to a great market potential [7]. Already, several major automobile manufacturers and research centers are investigating the development of IVC protocols and systems, and the use of inter-vehicle communication for the establishment of *Vehicular Ad-Hoc NETworks (VANETs)* [1, 4, 13, 17, 18].

In addition to the similarities that vehicular ad-hoc networks share with general, mobile ad-hoc networks, such as short-radio transmission range, low bandwidth, omni-directional broadcast (at most times) and low storage capacity, several new challenges arise because of the unique characteristics of the vehicular context: (i) rapid changes in link topology because of the relative fast movement of vehicles; (ii) frequent network disconnections, especially in the case of low vehicle density, where the gap between two vehicles might be several miles; (iii) data compression/aggregation required to accommodate for the limited bandwidth of the wireless medium; (iv) the feasibility of partially predicting vehicular position since vehicles normally run along pre-built roads that remain unchanged over the years; (v) the fact that energy is not a big issue since the vehicle itself can be used as a source of electric power. Last, but not least, an important challenge is the exploitation of vehicular ad-hoc networks for the provision of higher-level services to vehicles and drivers.

In this paper, we focus on the problem of providing *location-based* services to moving vehicles by taking advantage of short-range, inter-vehicle wireless communication and vehicular ad-hoc networks (VANETs). We concentrate on services that distribute on-demand information describing road conditions and available facilities in some geographic area; in particular: *traffic conditions* (e.g., congestion, traffic flow), *traffic alerts* that result from on-road emergencies (e.g., a traffic accident or a broken vehicle obstructing traffic on a road), and *roadside service directories* (e.g., location and price-lists of gas stations, location and menus of restaurants). This information can be fused with GPS navigation information, extending the functionality of state-of-the-art, on-board navigation systems.

For the deployment and provision of vehicular services, we propose the development and deployment of an ad-hoc service infrastructure on top of emerging vehicular ad-hoc networks (VANETs). We introduce the *Vehicular Information Transfer Protocol (VITP)*, an application-layer communication protocol specifying the syntax and the semantics of messages between *VITP peers*, i.e. the software components of our proposed service infrastructure. A VITP peer runs on the computing device of a vehicle, uses its IVC capabilities, and accesses the vehicle's sensors to retrieve useful information. VITP peers establish on-demand dynamic, ad-hoc groups, which collect, communicate, and combine information from the on-board sensors of different vehicles in order to resolve incoming requests. VITP is inspired by the HyperText Transfer Protocol of the World-Wide Web. However, it differs from HTTP in several fundamental aspects, such as: (i) the semantics of VITP-peer interaction vs. the HTTP request-reply exchange; (ii) the functionality and the role of VITP software components vs. the clients and servers in HTTP, and (iii) the support for push-based communication in VITP. These differences are mainly due to the ad-hoc, dynamic, and unreliable nature of vehicular ad-hoc networks, and the need to deploy VITP services even on top of embedded computers with limited resources.

The remaining of this paper is organized as follows. In Section 2, we present a motivating scenario, introduce the VITP service model and present the key design concepts of VITP. In Section 3, we give a brief overview of the VITP specification. Section 4, presents a simulation study which uses a vehicular traffic generator for investigating key performance metrics of VITP in a realistic scenario. We conclude in Section 5.

2. A SERVICE MODEL FOR VANETS

2.1 Context

To describe the infrastructure required for providing vehicular services and to present the VITP design, we introduce a simple motivating scenario taking place in the city setting of Figure 1. This Figure represents the plan of a small city-district, which is traversed by five streets; the direction of traffic is depicted with arrows placed near the street names. A snapshot of traffic conditions is superimposed on the plan. This snapshot depicts a number of vehicles (shown as grey boxes) of various sizes located on the district streets.

We assume that most vehicles are equipped with an embedded computer with a display interface, a GPS receiver, a wireless network interface for inter-vehicle communication (compliant to standards like 802.11x or DSRC) [12], and an on-board diagnostics (OBD) interface. Some vehicles may have alternative wireless network connectivity support thanks to an on-board cellular GSM/GPRS device. The OBD can be used to acquire a small set of data values from mechanical and electronic sensors mounted on the vehicle (e.g., current speed and acceleration, direction of motion, average speed during the last few minutes). We assume that all subsystems (GPS, OBD, wireless networking) are connected and provide data to the embedded computer. We

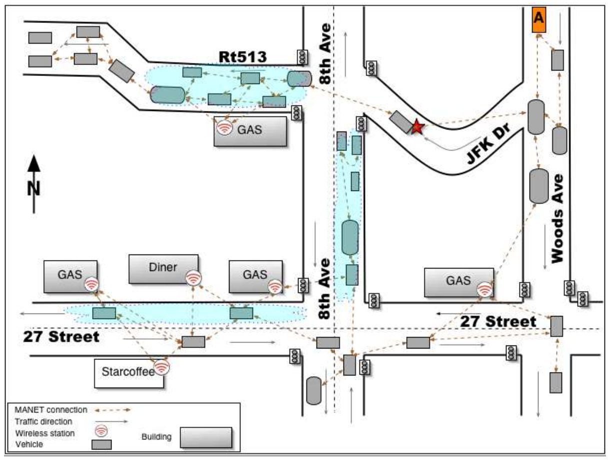

Figure 1: A vehicular service provision scenario.

also assume that a navigation software system is installed on the computer and enables the association of the vehicle's geographic position to an internal data-structure representing the road networks of a large geographic area around the vehicle. Such a data structure can be easily constructed from publicly available geographic referencing systems [3]. The feasibility of such vehicular systems has been demonstrated in a number of recent projects [5, 15]).

The vehicles establish a vehicular ad-hoc network (VANET) infrastructure through their wireless connections; Figure 1 depicts VANET connections as dashed, double-headed arrows connecting vehicles. A number of roadside service facilities (gas stations, coffee shops, restaurants) are also equipped with short-range wireless interfaces and participate in the VANET infrastructure. Vehicles and roadside stations use this infrastructure to exchange messages. Multi-hop message delivery can be supported by geographic routing protocols, which push messages toward their geographic destination [6, 8, 9, 10, 11]. In the absence of a VANET infrastructure, messages can be transported to their destination area through alternative wireless (cellular/GPRS) or wireline networks (Internet), and then passed onto moving vehicles via roadside wireless gateway stations. The details of message routing are outside the scope of this paper.

2.2 Motivation and Problem Statement

The types of services that we wish to support are illustrated with the following example: We assume that vehicle A, located at the top right of Figure 1 and moving southward on Woods Ave, is heading to a destination on the West side of the city. The driver wants to go either through Rt513 or through 27th Str. She is also interested in getting gas along the way, but is not willing to pay over 1.8 dollars per gallon for gasoline. The driver asks the on-board navigation system for the traffic conditions on alternative routes that lead to Rt513 West and 27th Str West, and for the location of drive-in coffee shops and gas stations along those routes. Notably, a possible way to Rt513 goes through JFK Dr, which is only a few meters down the road from the present location of A. Therefore, the service infrastructure should try to come up with a reply to the driver's requests, well before the driver decides whether or not to take the JFK Dr exit. The interaction between the driver and the on-board navigation system can be performed either with a voice or with a simple touch-screen interface.

The information requested by the driver of vehicle A can be *computed* out of data available on vehicles and roadside facilities located in the road segments specified by A's inquiries. For instance, the traffic-flow on the segment of Rt513 shown in Figure 1 can be derived by estimating the average speed of vehicles moving on that segment for a short period of time; a congestion in that road segment can be established from a consistently low average speed and/or a high density of vehicles on that road. Similarly, the operation of a gas station on 27th Str can be deduced from information dispatched by the gas-station's wireless access point, which specifies the type of service offered (selling gas), the business address, and gas prices.

To retrieve that information, the on-board system of vehicle A has to translate end-user inquiries into a sequence of location-sensitive queries. Each of these queries should be routed toward its designated location of interest, either via the vehicular ad-hoc network or through some other available network. Upon arrival to its destination area, the query must be picked up by the local vehicular service infrastructure. Nodes of that infrastructure (vehicles and/or roadside services) collaborate on-the-fly to compute a reply, which is dispatched back to the location where the query came from.

The goal of our work is to take advantage of VANETs established among vehicles equipped with the capabilities described in section 2.1, in order to design a vehicular service infrastructure that is capable of carrying out transactions like the ones described above. As key building blocks of this infrastructure, we introduce: (i) The **Vehicular Information Transfer Protocol**. VITP is an application-layer, stateless communication protocol, which specifies the syntax and the semantics of messages carrying location-sensitive queries and replies between the nodes of a vehicular ad-hoc network. VITP is independent of underlying VANET protocols that undertake the transmission and routing of VITP messages. (ii) The **VITP peer**, which is a lightweight software component to be deployed on the embedded computer of modern vehicles. VITP peers implement the VITP protocol and operate as clients, intermediaries, or servers in a VITP-protocol interaction. (iii) A **location encoding scheme**, which organizes and represents symbolically road segments and directions. This scheme is used by VITP for the specification of location-aware queries and for supporting underlying geographic routing protocols, which make use of on-board navigation services to transform symbolic locations into GPS coordinates [14, 5]. (iv) A number of protocol features designed to support **performance optimizations** (message caching, VITP traffic reduction), **quality assurance** for VITP results (termination conditions of VITP queries), and the **protection of privacy** of vehicle drivers.

In contrast to recent traffic monitoring systems, which are based on the continuous dissemination of traffic conditions through vehicular ad-hoc networks [14], VITP proposes the *pull-based* retrieval of traffic information, which can be triggered on-demand by *location-sensitive queries* issued from VITP-enabled vehicles. VITP supports also the *push-based* propagation of messages, as a mechanism for disseminating various alerts that carry information about emergencies or serious deviations from normal traffic conditions.

2.3 Key Design Concepts

We anticipate that service provision over vehicular ad-hoc networks will be based on an extended client-server computing model. In this model, a driver inquires information about traffic conditions or available facilities on some road segment. This inquiry is translated into query messages sent toward that road segment, via the underlying VANET. Vehicles in the destination area collaborate to establish a server, to resolve the incoming queries and to send back messages carrying the requested information. The *Vehicular Information Transfer Protocol* specifies the format and the semantics of query and reply messages exchanged between vehicular clients and servers. For the design of the VITP architecture and message specification we must take into account the following key observations:

2.3.1 Location-aware requests

In a vehicular-service provision model, service requesters are interested primarily in attributes describing traffic conditions and service facilities available to drivers in some particular geographic area. Therefore, vehicular-service queries must be *location-sensitive*, specifying explicitly the *target location* of their inquiry. Given that the motion of vehicles is constrained within the road system, we can assume that the geographic areas of interest are restricted to roads, road segments, directions of traffic, and adjacent roadside areas.

Accordingly, locations are represented in VITP as two-value tuples `[road_id,segment_id]`, where `road_id` is a unique key representing a road and `segment_id` is a number representing a segment of that road; opposite traffic directions on the same part of a road are represented as different road-segments. The `[road_id,segment_id]` representation of a vehicle's location is used in GPS-based, driver-friendly navigation support systems. It can be derived from a vehicle's GPS position `(longitude,latitude)` with a transformation calculation that uses information from public databases that store the correspondence of `(longitude,latitude)` positions to `road_id,segment_id` tuples [3]. The feasibility of such transformations has been demonstrated in recent literature [15, 5].

2.3.2 Virtual Ad-Hoc Servers (VAHS)

Because of the extensive size of road networks and the dynamic nature of traffic-related information, centralized approaches for vehicular-service information provision can be very hard and prohibitively expensive to implement. The collection, indexing, continuous update, and timely publication of traffic-related information through centralized servers requires: (i) an extensive infrastructure of sensors that continuously monitor traffic and service conditions, collecting information even in the absence of information consumers and queries; (ii) the management of a high volume of messages carrying updates to central servers on a continuous basis; (iii) a highly efficient server infrastructure that can process numerous simultaneous updates throughout large geographic areas and can provide timely and accurate replies to queries; (iv) a communication infrastructure that can carry large volumes of service queries and replies between moving vehicles and central servers, and (v) a costly effort to maintain the monitoring and communication infrastructure in good running condition. Consequently, it is questionable whether a centralized approach would scale to a wide geographic area and to a large number of vehicles.

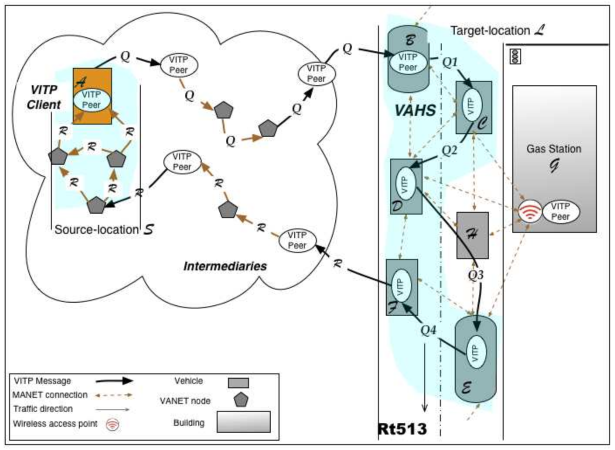

Figure 2: Clients and Servers in a VITP transaction.

To address the scalability problems of centralized approaches, VITP relies on no additional infrastructure, besides the one assumed to be available on moving vehicles and roadside facilities (see Section 2.1). Consequently, the server that computes the reply to a VITP query is essentially a dynamic collection of VITP peers, each of which: (i) Runs on a vehicle that moves inside the query's target-location area. (ii) Is willing and able to participate in the query's resolution by contributing information from its on-board diagnostics sensors or local memory. The establishment of this collection of peers is done in an ad-hoc manner, and relies on the vehicular ad-hoc network established by vehicles moving inside the target-location area. To better reflect the dynamic and ad-hoc establishment of VITP servers, we refer to the dynamic collection of VITP peers that are inside the target-location of a VITP query and take part in the query's resolution, as a *Virtual Ad-Hoc Server (VAHS).*

It is important to note that the collection of peers that constitute a VAHS, and the VITP peers that manage the VITP communication, follow a *best-effort approach* in their operation. In other words, they provide no guarantees or special features for the recovery of lost or dropped messages. It is also worth noting that a VITP peer, which has joined a Virtual Ad-Hoc Server, does not have information about other members of the group. It is also possible that a VITP peer joins a VAHS, participates in its computation, and leaves the target-location area before the completion of the query's resolution. The Virtual Ad-Hoc Server, on the other hand, does not maintain explicit knowledge of its members. The VAHS is established on-the-fly; its constituents can be derived only by the choice that VITP peers make individually about serving or simply forwarding VITP requests, and by the semantics of the VITP messages they exchange. In other words, the Virtual Ad-Hoc Server is identified with a query and its target-location area, rather than with the VITP peers that participate in it. In that perspective, the VAHS is similar to the SpatialViews system introduced in [16]. We have taken this approach in order to make the VITP protocol stateless and lightweight, and to keep the VITP state-machines as simple as possible. This is necessary in order to design simple and efficient VITP peers that can easily fit even on on-board embedded processors.

2.3.3 VITP transactions

Figure 2 depicts a typical VITP-transaction that takes place in the context of the service-provision scenario presented earlier in Figure 1. This transaction is initiated by vehicle $\mathcal{A}$, which is located in road segment $\mathcal{S}$ (Woods Ave in Fig. 1) and inquires information about the average speed of at least four vehicles inside road segment $\mathcal{L}$ (first segment of Rt513), as an estimate of traffic-flow conditions in $\mathcal{L}$. To this end, the VITP peer of $\mathcal{A}$ submits a VITP request $\mathcal{Q}$ with $\mathcal{A}$'s inquiry. We assume that the source and the target-location areas, $\mathcal{S}$ and $\mathcal{L}$, are connected through a vehicular ad-hoc network as shown in Figure 1.

The VITP transaction consists of four phases. In the initial *dispatch-query phase*, $\mathcal{Q}$ is transported through the underlying VANET toward target area $\mathcal{L}$. $\mathcal{Q}$ goes through a number of intermediary VANET nodes, which push the message toward its destination using geographic routing. It is worth noting that intermediary nodes may not be VITP-enabled (these are depicted as grey pentagons in Figure 2); these nodes simply pass the message on toward $\mathcal{L}$.

When $\mathcal{Q}$ is received by a peer $\mathcal{B}$ that is inside target-area $\mathcal{L}$ and is willing to join a Virtual Ad-Hoc Server to resolve $\mathcal{Q}$, the VITP transaction enters its second phase, the *VAHS-computation phase*. During this phase, the VITP request is routed between the VITP peers of the VAHS. These peers modify the VITP request in order to: (i) indicate that the request is part of an ongoing VAHS computation (this modification takes place only at the first peer that joins the VAHS), and (ii) piggyback partial query results to the VITP message's payload. Referring, for example, to Figure 2, when peer $\mathcal{B}$ receives the VITP request $\mathcal{Q}$, it parses the request, extracts the requested information from its on-board diagnostics system, *rewrites* the query in order to store the partial result into the query's body and to indicate that the query is now part of a VAHS computation, and passes the message on to its neighbor. The semantics of the query indicate how the underlying network protocol will treat the rewritten VITP query (unicasting or broadcasting it to neighboring peers).

A VITP request is transported between VAHS peers until some *Return Condition* is satisfied. The VAHS peer that detects the upholding of the Return Condition, creates the VITP reply and posts it toward source-region $\mathcal{S}$ through the VANET. The transportation of the VITP reply toward $\mathcal{S}$ corresponds to the third phase of the VITP transaction, the *dispatch-reply phase*. When the VITP reply reaches area $\mathcal{S}$, the VITP transaction enters its final phase, the *reply-delivery phase*. During this phase, the underlying network protocol broadcasts the VITP reply to the VANET nodes of $\mathcal{S}$, so that the reply can be received by the VITP peer of $\mathcal{A}$. The case that a vehicle has moved outside its initial road segment by the time a VITP reply reaches that road segment, can be dealt by the VAHS by specifying an extended region over which the reply should be broadcast.

2.3.4 Return Conditions

An important issue arising in the context of the VAHS-computation phase is how to define the Return Condition for a VITP request. A Return Condition determines at which point the resolution of a VITP request can be considered done. In other words, the Return Condition indicates if a VITP reply can be created and dispatched back to the originator of the request. The decision on what constitutes success in the resolution of a VITP query, however, depends on the semantics of the query itself. Therefore, the Return Condition must be defined explicitly as part of the query's specification.

For instance, referring to our example of Figure 2, suppose that vehicle $\mathcal{A}$ is looking for a gas station on road segment $\mathcal{L}$; when the corresponding VITP request reaches the VITP peer of gas station $\mathcal{G}$, the peer switches to the VAHS-computation phase, parses the incoming query, detects that the query requests information about *at least one* gas station in $\mathcal{L}$, and decides that it can fully resolve the query and that the Return Condition is satisfied. Consequently, it creates a VITP-reply message with $\mathcal{G}$'s coordinates and prices, and sends the reply to $\mathcal{S}$. In contrast, if $\mathcal{A}$ is looking for the prices of *more than one* gas stations in road segment $\mathcal{L}$, $\mathcal{G}$'s peer will start the VAHS-computation phase, re-write the incoming query, and try to pass the query on through the VANET, in search of other gas stations: clearly, the query's Return Condition is not satisfied yet. In the absence of other gas stations in $\mathcal{L}$, however, this Return Condition will never be met, the original VITP query will not be resolved, and $\mathcal{A}$'s peer will not receive any VITP reply. To address such cases, VITP supports an alternative Return Condition, which is constrained on the total time the infrastructure can spend in a VAHS-computation phase.

2.3.5 Protocol layering

The operation of the VITP protocol presumes an underlying networking infrastructure, which undertakes the transport and routing of VITP messages between peers installed in vehicles and roadside services. Typically, networking support will be provided by vehicular ad-hoc networks, although VITP messages could also be transported via other networks, such as GSM/GPRS. It is important to observe that the way VANET nodes handle VITP messages is influenced by VITP semantics. In particular, a VITP message that is part of a VITP transaction in either dispatch-query or dispatch-reply phase, must be routed *geographically* toward its target location. In contrast, a VITP reply that is part of a VITP transaction in the reply-delivery phase, should be *broadcast* inside its target-location area (and/or nearby areas), so that it reaches the VITP peer that originated the VITP transaction. Finally, the routing of a VITP message that is part of a VAHS-computation phase, depends on the semantics of the method specified in the corresponding VITP request. For example, in Figure 2, the resolution of VITP query $\mathcal{Q}$ can be satisfied with a simple tour through the VAHS peers of road segment $\mathcal{L}$. This can be achieved by unicasting $\mathcal{Q}$ inside $\mathcal{L}$. A different request method, or a different resolution method for the same request, may dictate the broadcasting of $\mathcal{Q}$ inside $\mathcal{L}$.

The interaction between VITP and the routing protocol of the underlying VANET is as follows: whenever a VITP message arrives at a VANET node, the network layer *always* makes a call to the local VITP peer (see Figure 3). This call is made even if the peer is an intermediary, i.e., it is not placed inside the target location of the message. If the node is not VITP-enabled or if its VITP peer is busy or down, the call will fail; in that case, the network layer will simply retransmit the message to a neighboring node. Otherwise, the peer will receive and parse the message. Depending on the active VITP phase and the semantics of the message, the peer may re-write the message before retransmission. The peer will also signal the VANET routing module about the routing method to be used when transmitting the outgoing message (unicast or broadcast).

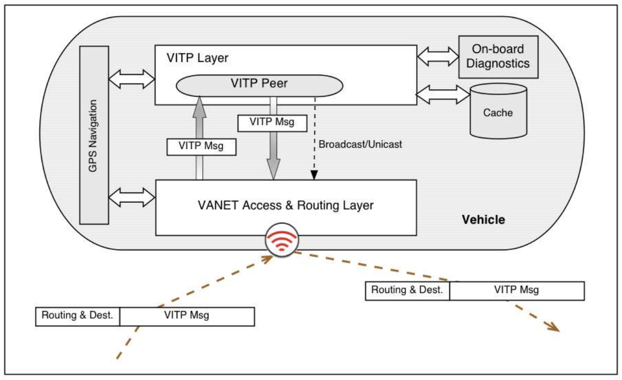

Figure 3: Protocol layering.

2.3.6 Other Issues

Caching: In the scenario of Figure 2, if the VITP peer is an *intermediary* and the incoming message is a request, the peer can search into its local cache for a matching reply. The matching test should take into account both the semantics of the VITP query (as described by the query's `uri`) and the specification of the target region. In the case of a match, the peer can send the cached reply back to the VITP client and either complete the VITP transaction or retransmit the incoming message toward its target location. This decision affects the Return Condition of the VITP request and must be based on the semantics of the incoming VITP message. Therefore, VITP provides cache-control headers that can be included in VITP messages and act as directives to VITP-peer caching decisions.

Message Identifiers: To achieve the delivery of a VITP reply to the peer that requested it and to preserve the correctness of a VAHS computation, we must ensure that: (i) A VITP peer can match incoming replies against its pending requests and can detect replies belonging to other peers. (ii) A VITP peer will not act again on the same VITP request even if it receives this request multiple times. To this end, a unique, random identifier (`msgID`) is attached to every new request. The same identifier is also attached to messages derived from the original VITP request, that is, to modified requests exchanged during a VAHS-computation phase and to the resulting VITP replies. VITP peers maintain a cache with recently received `msdID`'s. Every time a peer receives a VITP message, it compares its `msgID` against cached `msgID`'s in order to determine how to handle the incoming message. New identifiers are cached for a default period of time.

Driver privacy: VITP message identifiers can be produced by random-number generators minimizing the possibility of clashes (where two different VITP messages have the same identifier). It is worth noting that the use of a randomly produced, unique identifier for every new VITP request protects driver privacy, because the messages exchanged in the context of a VITP transaction do not carry any information identifying the driver or the vehicle that initiated the transaction.

VITP message syntax	
Line 1:	METHOD <uri> VITP/<version_number>
Line 2:	Target: [rd_id-dest,seg_id-dest]
Line 3:	From: [rd_id-src,seg_id-src] with <speed>
Line 4:	Time: <current_time>
Line 5:	Expires: <expiration_time>
Line 6:	Cache-Control: <directive>
Line 7:	TTL: <time_to_live>
Line 8:	msgID: <unique_key>
Line 9:	Content-Length: <number of bytes>
Line 10:	CRLF
Line 11:	<message body>
URI syntax	
/<type>/<tag>?[<rc_expr>,...]&<param_expr>&...	

Table 1: VITP Syntax.

Dissemination of traffic alerts: The VITP features described so far support a pull-based model of vehicular service provision extracted from Figure 1. According to this scenario, all VITP transactions are triggered by driver inquiries. However, vehicular applications can benefit equally from a push-based model of information provision. For example, in the motivating scenario described in Section 2.1 and depicted in Figure 1, the vehicle moving in JFK Dr may detect a slippery road. Information about such a dangerous condition (depicted with an asterisk in Figure 1) should be propagated to other vehicles moving into the area. To this end, the vehicle that detects this condition must generate an alert message and transmit it via the underlying VANET. To support the transmission of traffic-alert information, VITP provides a special message-type dedicated to information dissemination ("push"). A "push" message carries a special VITP method, the VITP representation of a target-location area, a description of the alert, a unique identifier, and other VITP-specific attributes (expiration time, caching directive, etc). The protocol treats VITP alerts similarly to VITP replies: an alert message is transported to its target-location via geographic routing; upon arrival to it target-location, the message is broadcast to all vehicles in that area. Alternative implementations could combine geographic routing with broadcast inside areas along the way between the source and the target location.

3. VITP SPECIFICATION

In this section, we present a brief overview of the Vehicular Information Transfer Protocol specification, focusing on the format of VITP messages. The syntax of a generic VITP message is given in Table 1. VITP provides two types of messages, distinguished by the `METHOD` entity placed at the beginning of each message (see Line 1 in Table 1). `METHOD` takes the values `GET` and `POST`: `GET` represents a VITP request that queries the attributes of some geographic area (pull model); `POST` is used for VITP replies and for messages that disseminate an attribute of some particular location toward some other geographic area (push model). The information requested by or transported through a VITP message is specified further by the `<uri>` attribute (see Table 1, Line 1). The last part of Line 1 declares the protocol used and its version number.

The syntax of the `<uri>` attribute is presented in the second part of Table 1. The `<type>` field of the `uri` specifies the classes of VITP-enabled physical-world entities involved in the resolution of a `GET` message or in the generation of a `POST` message. Currently, we anticipate two types of entities: `vehicle` and `service`. The `service` entity corresponds to roadside facilities that offer services to vehicles (gas stations etc). Other entities could be easily supported, such as `traffic_light`. The `<type>` field can also take the value `all`, which indicates that VITP peers running on *any* kind of physical entity may take part in the computation of the VITP-request. The `<tag>` field describes the actual information sought or disseminated by a VITP request. For example, a tag value `traffic` indicates that the request queries for information about road-traffic conditions expressed in terms of the average speed of vehicles in the area of interest. A tag value `alert` indicates that the request is either trying to retrieve pending alerts (if used with a `GET`) or to post a new alert (if used with a `POST`). A special type of VITP query, defined with the tag-value `index`, returns the types of queries supported by VITP peers at the query's destination area.

```
GET /vehicle/traffic?[cnt=10&tout=3000msec]&tframe=3min
GET /vehicle/traffic?[cnt=*&tout=1800msec]&tframe=0min
GET /service/gas?[cnt=4&tout=1800msec]&price<2USD
GET /vehicle/alert?[cnt=20&tout=500msec]&type=accident
POST /vehicle/alert?[cnt=*&tout=*]&type=slippery_road
```

Table 2: Type and query tag combinations.

The "`?`" character that follows the tag field in the URI syntax of Table 1 separates the request specification from its parameter list. This list is a series of `name, value` expressions separated by the "`&`" character. VITP distinguishes between two types of parameter expressions:

1. Expressions used to define the *Return Conditions* of VITP requests (`rc_expr`). These expressions are placed inside a pair of brackets immediately after the "`?`" character. There are two default return-condition parameters, `tout` and `cnt`. `tout` (time-out) specifies the maximum lifetime of a GET-request resolution (VAHS-computation phase); `cnt` (count) specifies the number of peers that should contribute to the resolution of a request. Every VITP peer that receives a request and participates in its resolution reduces the `cnt` value by one. Then, the peer checks whether the value of `cnt` equals zero or if the timeout period has been exceeded. In such a case, the Return Condition of the request is satisfied and a reply can be produced and transmitted back to the VITP client that issued the request.

2. Expressions placed after the Return Condition brackets. These, specify the values that are passed to the actual query that is to be executed on the VITP peer (`<param_expr>` field in Table 1).

Typical examples of VITP requests are given in Table 2. The first two queries seek to retrieve traffic-flow conditions; traffic-flow information is represented in terms of the average speed of cars or the density of cars in some area. The first query requests the average speed of 10 vehicles moving in the area of interest and specifies that this computation should be completed within 3000 msec. Upon arrival to a VITP peer, the query will retrieve from the OBD the speed of the vehicle, averaged over the last 3 mins; the 3min averaging is specified with the `tframe=3min` request parameter, which is specific to the `traffic` query. The second query is designed to estimate the density of vehicles in some area; the `cnt` parameter is assigned a value of infinity ("`*`") indicating that the query should go through all the reachable vehicles and retrieve their speed; the VAHS-computation phase must not take longer than 1800msec. The following three requests specify respectively: (i) a query for any gas-station facilities with a price of not more than 2 dollars per gallon; (ii) a query for any posted alerts regarding traffic accidents; (iii) the generation of an alert regarding a slippery road condition. For the "slippery road" alert, which is a POST, the sender is not interested in getting back any information. The goal is rather to disseminate the information. Therefore there is no Return Condition specified and it is up to the recipients of the alert to cache and/or keep posting it for as long as they see fit. The VITP implementation can impose a default timeout Return Condition of several hours, if the `tout` field of a query is set to "`*`."

The `Target` and `From` headers of Lines 2 and 3 in Table 1 specify the target and source-location areas of a VITP message, respectively. Locations are formatted according to a standard scheme that specifies the road and segment identifiers, as retrieved by an on-board navigation and positioning system. In the case of `GET` messages, the source-location area can be followed by an optional entity specifying the speed of the vehicle at the time that the request is created. This information can be useful in case the VAHS needs to estimate the location of the source vehicle when routing a reply back to it.

The `Time` header (Line 4 in Table 1) carries a time-stamp specifying the point in time at which the VITP message was generated by its originating peer. The `Expires` header of a VITP message (Line 5 in Table 1) specifies a point in time after which the corresponding VITP transaction has to be terminated. In the case of a `GET` request, the expiration time indicates that the originating peer wants to receive a reply before the specified expiration time. Consequently, any peer that receives a VITP message (request or reply) as part of the transaction after this time, can drop the message and not propagate it further. In the case of a `POST` request, the expiration-time header indicates when the VITP peers should stop propagating the corresponding alert.

The `Cache-control` header of Line 6 defines the caching directives that should apply to a VITP request. In the case of `GET` messages, the possible values that can be assigned to this header are: (i) "`no`," which indicates that the originator peer does not accept as reply a message generated during a previous transaction and cached in the infrastructure; (ii) "`yes`," which indicates that the originator peer is willing to accept a reply previously cached during some other transaction, and (iii) "`fwd`," which indicates that the peer wishes that the transaction goes forward but is also willing to accept cached replies. The use of caching (`Cache-control: yes`) can result to VITP peers receiving replies faster and to an overall reduction of VITP messages exchanged, at the expense of the accuracy and freshness of query results. In the case of `POST` messages, the `Cache-Control` header specifies whether the `POST` message is cacheable (value set to "`yes`") or not (value set to "`no`"). If caching is allowed, the `TTL` header can be used to determine for how long a reply can be kept in intermediary caches.

The `msgID` header is used to carry the unique numerical message identifier that is assigned to every VITP query. As mentioned in the previous section, and in order to protect driver privacy, instead of using as `msgID` the Vehicle Identification Number (VIN) which is unique for every vehicle, we generate a random `msgID` by hashing a combination of the VIN, current time, and the vehicle location. The message identifier is very important for various reasons: A copy of the `msgID` is cached by VAHS participants so as to avoid counting the same information twice during a query resolution; this is also true for the identifier of `POST` requests, so that the same alert is not registered twice in the same peer. Also, a corresponding reply is identified by the same message identifier as the original query, so that the source peer will know exactly what it is looking for when receiving replies through its VANET interface.

The remaining lines (Lines 9-10 in Table 1) are used only in VITP requests that carry intermediate results during VAHS-computation phase or in `POST` messages. The `Content-Length` header declares the size of the data carried by the request. The actual data follow after the `CRLF` character.

4. SIMULATION STUDY OF VITP

In this section, we present simulation results for VITP performance. Our goal is to investigate the feasibility of VITP and to analyze its performance in large scale vehicular networks.

4.1 Simulation Setup

To simulate VITP we used the *ns-2* simulator [2] in combination with our own traffic generator tool, which accepts as parameters the simulation time, road length in meters, number of lanes per road, average speed of the vehicles in meters/sec, average gap distance between vehicles on same lane, number of service nodes on the road, and the number of users on the road. The tool uses a simplified traffic model: (i) vehicles may enter or leave the road through evenly distributed entries and exits located along the road every 1000 meters; (ii) vehicles can change their speeds and lanes independently of other vehicles, and (iii) vehicles are evenly distributed on the road; once a vehicle leaves the road a new vehicle enters the road randomly. In the following simulation scenarios, we generated traffic for a 25km-long highway with 3 lanes. The average vehicle speed is 20m/s and the simulation time is set to 500 seconds.

For the purposes of this study, we choose as wireless medium an 802.11-compliant network with a data transmission rate of 11Mb and a transmission range of 250 meters [1]. To allow vehicles to maintain neighbor connectivity, each vehicle broadcasts a `Hello` packet every period selected randomly from the range of 0.75 to 1.25 seconds. The received signal-strength threshold used in maintaining information about neighbors is set to distances below 200 meters in order to accommodate with the fast dynamics of the network and to maintain consistency in neighborhood information. Once a vehicle enters the road, it initiates its query at a random time selected uniformly over its remaining simulation time. The vehicle re-sends the query if no answer is received within a specified timeout of 10 seconds.

As mentioned earlier, VITP messages are forwarded to their destination region with geographic routing. In doing this, we select the next vehicle in the route to be the one that is closest to the target destination. If a vehicle fails three consecutive times in transmitting the message to its next hop, it selects another neighboring vehicle. After trying

[1]In practice, the wireless transmission range is less than 250 meters. However, this transmission range could be restored with the use of external antennas.

with three different neighbor vehicles, the query message is dropped indicating a failure of the query-dispatch phase and, consequently, a failure of the VITP transaction.

In the following sections, we experiment with the `traffic` query discussed earlier. This query requests the average vehicular speed within a road segment of S meters long, in order to derive an estimate of the traffic-flow at that segment. We assume that the road segment in question is D meters away from the query sender vehicle (QS); we refer to D as the *query distance*. We use as Return Condition *cnt*, i.e., the total number of vehicles of the target road-segment that should be sampled for their speed. As mentioned earlier, a vehicle in the target segment may receive the same query message multiple times, but it participates in updating the query results only when it receives the query message for the first time.

4.2 Metrics and Results

To evaluate the performance of VITP we employ the following metrics in the evaluation:

- *Response time* is the average time of a *successful* VITP transaction. It measures the elapsed time between the time at which a query is initiated and the time at which its corresponding reply is received.

- *Dropping rate* is the percentage of unsuccessful queries, i.e., queries for which a vehicle times-out before getting a valid reply.

- *Accuracy* measures how close the estimated average speed (calculated usually by a subset of the available vehicles in the target segment) is to the actual average speed in the region of interest (calculated by considering all present vehicles in the same area).

- *Efficiency* measures the percentage of the number of exchanged query messages that were actually employed in calculating a result over the total number of query messages exchanged both in geographic routing and inside the target location. A message is considered to have participated in the result computation when it is received by a vehicle in the target segment for the first time.

4.2.1 Effects of query distance D

First, we study the effects of query distance to the metrics defined above. For this study, we set the average gap between consecutive vehicles on the same lane to 100m. We vary the query distance D from 500 to 5000 meters, with the query's target segment length S fixed to 800 meters; this translates to an average of 30 vehicles moving inside the target segment. The value of *cnt* ranges from 1 to 20 vehicles. The computation time within a vehicle is assumed to be negligible.

Figure 4 plots the response time versus the query distance D for different *cnt* values. The response time increases almost linearly with D. However, as the value of *cnt* increases and becomes comparable to the total number of vehicles in the target road-segment, a VITP request would have to cover a large percentage of the target vehicles in order to satisfy the Return Condition. Therefore, the query message would have to re-visit many vehicles before succeeding to discover unvisited vehicles. This translates to longer VAHS-computation times and, consequently, to longer response times. In our scenarios, with approximately 30 vehicles in the target road-segments, we observe that response time increases substantially for values of *cnt* greater or equal to 15 (see Figure 4).

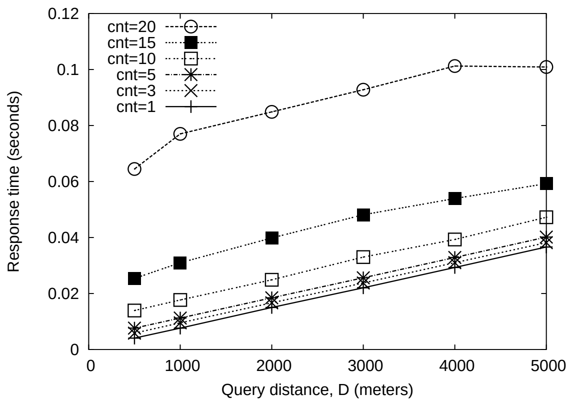

Figure 4: Response time vs. query distance (D).

Query distance (D)	Forward dropping rate (%)	Backward dropping rate(%)
500	11.84	0.47
1000	18.41	0.64
2000	36.06	1.52
3000	50.70	2.72
4000	60.69	3.65
5000	65.95	4.24

Table 3: Dropping rates vs. query distance (D).

Table 3 reports the dropping rates for different query distances. The forward-dropping and backward-dropping rates correspond to the query-dispatch and reply-delivery phases: the forward dropping rate is measured as the percentage of the failed queries due to the failure of the query-dispatch phase, over the total number of generated queries. The backward dropping rate is the percentage of the failed queries, due to failure of the reply-delivery phase, over the number of queries that successfully reach the region of interest. From Table 3 we observe that both dropping rates increase with query distance D. For very distant queries, the forward dropping rate becomes prohibitively high (i.e., 65% for D=5000m), making it harder for queries to complete successfully. Nevertheless, once a query message finds its way to its target location, it is highly possible that the reply message will be routed successfully to the QS vehicle, since the connectivity between vehicles during the VAHS-computation phase remains stable. The data reported here correspond to a cnt = 10 (as we will see later, this value results to greater efficiency and a high accuracy for our simulation scenario). Using different values of *cnt* had no effect on both dropping rates.

Figure 5 plots the accuracy of the query results versus *cnt* for different query distances D. It is interesting to note that for our simulation scenario we achieve a maximum accuracy of approximately 90% for *cnt* values greater than 5. Furthermore, that the query distance has negligible effect on the accuracy.

Figure 6 plots the query efficiency versus *cnt* for different query distances. As expected, the efficiency is higher

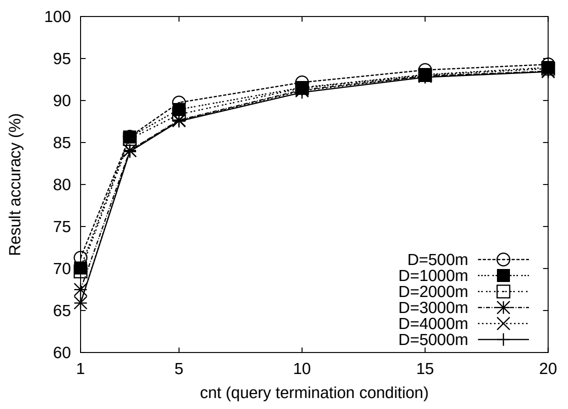

Figure 5: Result accuracy vs. ***cnt*** **for different query distances (D).**

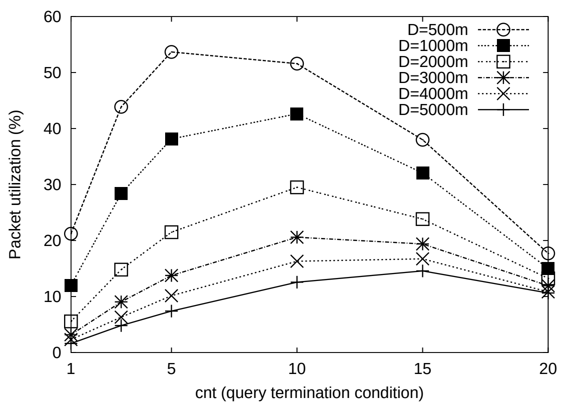

Figure 6: Query efficiency vs. ***cnt*** **for different query distances (D).**

for smaller D's because, as D decreases, the number of forwarded query messages in the query-dispatch and reply-delivery phases becomes smaller. It is interesting to observe that for each query distance examined, there is an optimal value of *cnt* for which the efficiency is maximized. When using smaller *cnt* values, the overhead of forwarding the query messages during the query-dispatch and reply-delivery phases dominates. Adopting *cnt* values greater than the optimal value dictates the visit to a large number of the vehicles in the target segment and this results to a large number of query messages that have to be forwarded to previously visited vehicles. For the parameters of our simulation study and for most query distances examined, this optimum value of *cnt* is around 10.

4.2.2 Effects of vehicle density

We study the effects that vehicle density has on VITP performance. We use a simulation scenario similar to the one used when evaluating the effects of query distance D. However, we fix D to 2000 meters and change the vehicle density by changing the gap between consecutive vehicles on the same lane from 50 to 200 meters. The response time increases with the gap (see Figure 7) but this effect becomes pronounced for larger values of *cnt*.

Examining the dropping rates, we find that the forward and backward rates increase significantly with the gap, for all values of *cnt*. For example, the forward dropping rate for $cnt = 10$ increases from 0.81% to 89.14% when the gap increases from $50m$ to $200m$; for the same increase in the gap, the backward dropping rate increases from 0.1% to 7.89%. Finally, an increase in inter-vehicle gap reduces significantly the measured efficiency (see Figure 8), especially for large *cnt* values. The reason behind this decrease is that as the vehicle density decreases when increasing the gap, it becomes more difficult to reach the required number of vehicles in the target region. This difficulty increases even more when *cnt* is large.

We have also studied the effects that the query request rate has on VITP performance. We measured the response time for the successful queries and found that changing the request rate did not have a proportional effect on response time. The request rate, however, had a effect on dropping rates: an increase in the request rate resulted to a significant increase in the forward and backward dropping rates.

5. CONCLUSIONS

In this paper, we introduced the *Vehicular Information Transfer Protocol* (VITP), an application-layer communication protocol designed to support the establishment of distributed, ad-hoc, best-effort service infrastructures over ve-

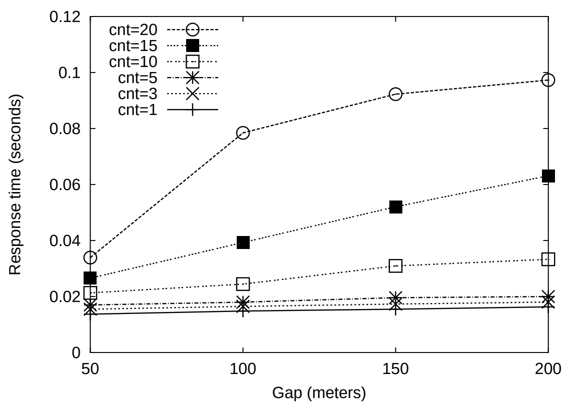

Figure 7: Response time vs. gap between consecutive vehicles for different ***cnt*** **values.**

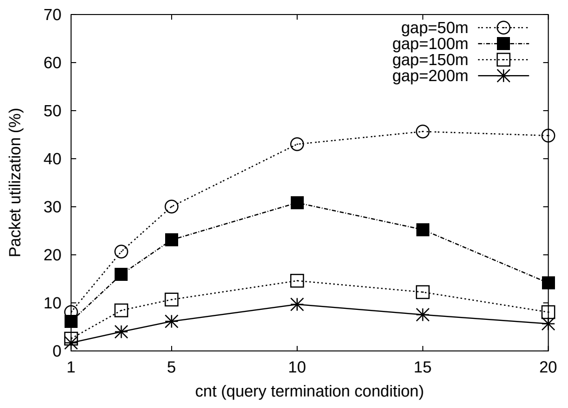

Figure 8: Query efficiency vs. ***cnt*** **for different gap values.**

hicular ad-hoc networks (VANET). We described the semantics of VITP transactions and provided the specification of VITP messages. Furthermore, we discussed the functionality of the *VITP peer*, the software component that should be installed on VAHS-enabled vehicles to support VITP interactions. We introduced the concept of the *Vehicular Ad-Hoc Server*, which is established on-demand as an ad-hoc collection of VITP peers that collaborate to resolve an incoming VITP request.

The Vehicular Information Transfer Protocol has the expressive power to define location-aware queries seeking and integrating information from vehicle sensors and roadside facilities, taking advantage of on-board GPS navigation systems. VITP is simple, stateless and lightweight; therefore, it can be easily implemented on embedded processors and resource-limited computing devices that are found on-board of modern vehicles. Initial evaluation of VITP relied upon simulations of large-scale vehicular networks. The simulation results demonstrated the viability of VITP and proved the feasibility of our approach in VANETs.

Our simulations showed also that the choice and the tuning of the Return Condition for a VITP request is very important as it determines the effort involved in a VAHS computation phase. This effort affects the accuracy of VITP results, the dropping rate of VITP transactions, and the response time of successful VITP transactions. When using the *cnt* parameter to define the Return Condition, we observed that there is a range of *cnt* values that result to optimal efficiency while producing replies of adequate accuracy. Therefore, in realistic scenarios, *cnt* has to be selected carefully according to the traffic patterns in order to achieve maximum efficiency and good accuracy.

The dropping rate in the query-dispatch phase is high and grows substantially with increasing query distances and query request rates, and with decreasing vehicle densities. The dropping rate is the parameter that determines the percentage of VITP transactions that complete successfully and, consequently, the end-to-end performance of VITP-based applications. The high dropping rates observed suggest that various optimization techniques be adopted to enhance the overall VITP performance; for example, the caching of VITP replies inside the VANET infrastructure and the use of alternative routing mechanisms in the absence of accessible routes in low-density VANETs (e.g., routing through cellular GSM/GPRS networks).

6. REFERENCES

[1] General Motors Collaborative Laboratory website. http://gm.web.cmu.edu/.

[2] The Network Simulator ns-2. http://www.isi.edu/nsnam/ns/.

[3] U.S. Census Bureau. Topologically Integrated Geographic Encoding and Referencing system. http://www.census.gov/geo/www/tiger/ (accessed May 2005).

[4] Z. Chen, H. Kung, and D. Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad-hoc Networking and Computing*, pages 247–250, Long Beach, CA, Oct 2001.

[5] C. Jensen, H. Lahrmann, S. Pakalnis, and J. Runge. The INFATI Data. Technical Report TR-79, TimeCenter, April 2005. http://www.cs.auc.dk/TimeCenter.

[6] B. Karp and H. Kung. Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000*, pages 243–254. ACM, August 2000.

[7] W. Kellerer. (Auto)Mobile Communication in a Heterogeneous and Converged World. *IEEE Personal Communications*, 8(6):41–47, December 2001.

[8] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Geographic Routing Made Practical. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI '05)*. USENIX Association, May 2005.

[9] Y. Ko and N. Vaidya. Location-aided routing in mobile ad hoc networks. In *Proc. ACM/IEEE Mobicom*, October 1998.

[10] J. Li, J. Janotti, D. D. Coutu, and R. M. D. Karger. A scalable location service for geographic ad hoc routing. In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, August 2000.

[11] C. Lochert, M. Mauve, H. Fussler, and H. Hartenstein. Geographic Routing in City Scenarios. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1), 2005.

[12] J. Luo and J.-P. Hubaux. A Survey of Inter-Vehicle Communication. Technical Report IC/2004/24, School of Computer and Communication Sciences. EPFL, Lausanne, Switzerland, 2004.

[13] R. Miller and Q. Huang. An Adaptive Peer-to-Peer Collision Warning System. In *IEEE Vehicular Technology Conference (VTC)*, Birmingham, AL, May 2002.

[14] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. TrafficView: A Scalable Traffic Monitoring System. In *Proceedings of the 2004 IEEE International Conference on Mobile Data Management*, pages 13–26, January 2004.

[15] T. Nadeem, S. Dashtinezhadd, C. Liao, and L. Iftode. TrafficView: Traffic Data Dissemination Using Car-to-Car Communication. *ACM Sigmobile Mobile Computing and Communications Review, Special Issue on Mobile Data Management*, 8(3):6–19, July 2004.

[16] Y. Ni, U. Kremer, A. Stere, and L. Iftode. Programming Ad-hoc Networks of Mobile and Resource-Constrained Devices. In *ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI)*, Chicago, Illinois, USA, June 2005. ACM.

[17] C. Schwingenschloegl and T. Kosch. Geocast Enhancements for AODV in Vehicular Networks. *ACM Mobile Computing and Communications Review*, 6(3), July 2002.

[18] E. Welsh, P. Murphy, and P. Frantz. A mobile testbed for gps-based its/ivc and ad hoc routing experimentation. In *International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Honolulu, HI, Oct 2002.