VAN: *V*ehicle-*A*ssisted Shortest-Time Path *N*avigation

Wenping Chen[+] Sencun Zhu[*] Deying Li[+]

[+] School of Information
Key Laboratory of Data Engineering and Knowledge Engineering, MOE
Renmin University of China, Beijing
{chenwenping, deyingli}@ruc.edu.cn

[*]Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA
szhu@cse.psu.edu

***Abstract*—Traffic congestion is a very serious problem in large cities. With the number of vehicles increasing rapidly, especially in cities whose economy is booming, the situation is getting even worse. In this paper, by leveraging the techniques of Vehicular Ad hoc Networks (VANETs) we present a dynamic navigation protocol called *VAN* for individual vehicles to find the shortest-time paths toward their given destinations. Specifically, a vehicle initiates a number of queries, which are routed by VANETs along different paths toward its destination. During query forwarding, the real-time road traffic information in each road segment is aggregated from multiple participating vehicles and returned to the source after the query reaches the destination. This information enables the source to calculate the shortest-time path. We also propose two forwarding optimization methods to reduce communication costs and an error handling mechanism to deal with abnormal circumstances. To evaluate its performance, we use the real traffic data of Beijing, including 2,308 road segments at two different times. Our simulation results demonstrate that our protocol, on average, could save around 30% driving time, compared to traveling along the shortest distance paths.**

Keywords- vehicular networks; navigation; shortest-time path; traffic jams

I. INTRODUCTION

People living in large cities almost suffer from traffic jams every day. The Texas Transportation Institute estimates that the 85 largest metropolitan areas experienced 3.7 billion vehicle-hours of delay, resulting in 2.3 billion gallons in wasted fuel and a congestion cost of $63 billion in a year [17]. Traffic jam is especially serious in metropolitan cities in developing countries, such as China. In Beijing, there are more than 4,000,000 vehicles and yet the number is increasing with the rapidity of more than 1,000 each day. Even with the enforcement of transportation restriction, traffic congestion still happens on many road segments, and the average speed on arteries of urban area is only 21.7km/h at peak hours [16]. It takes working people average 70-minute to commute, which is double of what it would be if no congestions occur. To mitigate this problem, it is important to use the transportation resources more efficiently. Intelligent Transportation System (ITS) provides us a way to realize it.

Centralized ITS with infrastructure is one of the possible solutions. These centralized infrastructures rely on a large number of sensors or other devices located on roadsides or traffic lights to collect traffic data and report to a central server. Then, the real-time traffic information is broadcasted to vehicles via infrastructure equipments [1,2,3]. While such kinds of systems are useful, they are not widely deployed in large cities yet. Besides the initial investment, the infrastructure, once deployed, is difficult to change and costly to upgrade.

Meanwhile, currently vehicles equipped with navigation devices are becoming more and more popular. It is not only because such devices enhance people's driving experience and safety, but also because they are becoming affordable to many people. Nowadays, the price of a typical navigation device, which has a GPS receiver and WiFi capability, is as cheap as 100 dollars. However, traditional GPS navigation devices only consider the inherent *static* characteristics of roads, such as length and speed limit, as the parameters in determining the shortest-distance path for users. Thus, they cannot meet people's *dynamic* demand. In reality, sometimes people are more concerned with driving time than driving distance. However, in the urban area of a big city, especially at peak hours, shortest time path is often different from shortest distance path because of traffic congestions. Now the question is: *how can one find the shortest-time route at the real time?*

To answer the above question, we resort to a new type of network systems, namely Vehicular Ad hoc Networks (VANETs). VANETs are been regarded as an emerging and promising field by both industry and academia. Indeed, some effort has been made on dynamic navigation with VANETs [4-6], where vehicles exchange traffic information by broadcasting its own traffic information to neighbors. To reduce the communication cost, data filtering techniques [4,5] were

This paper was supported by the National NSF of China, National High-Tech Project (863) of China (No. 2008AA01Z120) and Renmim University of China (No. 10XNJ03)

proposed to remove uninterested traffic data, but unfortunately they also decrease the probability for vehicles to acquire the needed data.

In this paper, by leveraging the VANET technology, we propose a real-time dynamic navigation protocol to find the shortest-time path. First, a vehicle initiates a number of queries, one for each candidate path of its choice. A query is forwarded along the candidate path by a number of forwarders that are selected according to certain rules. The forwarders at each road segment aggregate the real-time traffic information at their current segment. After reaching its destination, a query reply message carrying the en-route traffic information is returned to the query initiator. Since the traffic status changes dynamically, a vehicle may run our protocol once a while (e.g. every 10 mins) for more accurate predication. To reduce the communication cost, we employ two optimization methods: caching the past query results and suppressing redundant queries. Furthermore, mechanisms are also presented to handle abnormal communication failure. Finally, we use real traffic data of Beijing from BTMB (Beijing Traffic Management Bureau) to evaluate the performance of our protocol. The dataset cover all the arteries on about 25km*25km urban area of Beijing regarding two typical times: peak hour and normal daytime. With the help of Beijing digital map, we setup the simulation environment involved 2,308 road segments and more than 160,000 vehicles. Our evaluation results demonstrate that our protocol is feasible for a metropolitan-size city, and in many cases it can provide drivers alternative routes with reduced traveling time (on average by 30%), especially when the driving distance is relatively large (e.g., above 5km).

The rest of the paper is organized as follows. Related work is discussed in Section II. In Section III, assumptions, motivation and system overview are described. In Section IV, we present the details of our navigation protocol. Performance evaluation is presented in Section V. In Section VI, some issues and future work are discussed. Finally, we conclude the paper in Section VII.

II. Related Work

There are two kinds of approaches, infrastructure-based and zero-infrastructure-based, for dynamic traffic information dissemination. Deploying a fixed infrastructure [1,2,3], such as wireless sensors [2] or RFID readers and wireless APs [10], on roadsides or traffic lights, is a relatively steady approach to collect and transmit traffic data. However, applying such models on large cities usually costs much on installation and maintenance. Furthermore, it might suffer from the scalability problem when updating the infrastructure. PeerTIS [25] is a traffic information system that creates a DHT-based P2P overlay for efficient traffic information sharing. It assumes that IP-based communication is available between cars.

Based on VANETs, recent research [4-7, 18] has introduced zero-infrastructure systems where information is exchanged only between the vehicles themselves. Data delivery between vehicles can be classified into two models: push and pull [12]. In the push mode, vehicles broadcast traffic information to other vehicles without being requested, whereas in the pull mode transmissions are only between specified requesters and repliers. Most systems adopt push mode to disseminate traffic data, where vehicles periodically broadcast traffic data to others. Based on the traffic data received from neighbors, Traffic View [6] displays the positions of the surrounding vehicles that might be beyond the driver's view. Its objective is safe driving, not path selection. To reduce communication costs, data filtering or demand-based acquiring strategies have been proposed. Vehicles only receive/broadcast information from/to the vehicles moving in the opposite direction [4], since such vehicles may carry useful information with higher probability. In [7,18], several methods are proposed to save communication resources caused by periodical broadcasting. The principle is to dynamically adjust the broadcast period according to vehicle density. In [5], traffic data transmission depends on the location relationship between vehicles. Each vehicle periodically broadcasts its location. When a vehicle hears the location of a neighbor, it requests traffic data from the neighbor only if the neighbor is on the way to its destination. In such push mode, vehicles frequently exchange data that often comprise a great deal of useless information. Data filtering helps reduce data traffic, but consequently a vehicle may not be able to acquire all data it needs, especially when the destination is relatively far away. Unlike these existing works, our protocol performs in the pull mode, where vehicles actively request the needed traffic data for navigation by sending queries regarding the possible paths.

As for traffic data transmission, a number of data dissemination or routing algorithms for vehicular networks were proposed. MDDV [15] adopts a trajectory based forwarding strategy, VADD [8] addresses the issue of frequent disconnection in VANET, SADV [9] sets up static nodes to assistant data delivery in low-density area, CAR[20] and MURU [21] measure or predict the connectivity of paths to minimize the probability of link breakage. Their objectives are to find a wireless communication path with the smallest delay from source to destination to delivery data, and such paths often have the highest vehicle density. Differently, our objective is to help drivers to find the shortest-time driving path. Thus, we must collect the traffic data of multiple possible routes from source to destination. The possible travel routes are also the routes for traffic data

(a) Peak hour (b) Normal daytime

Figure 1. Traffic condition

dissemination and the shortest-time path often has the lowest vehicle density.

III. SYESTEM MODEL

In this section, we first describe our system assumptions, followed by our motivation and protocol overview.

A. Assumption

We assume that each vehicle in a VANET is equipped with a special device, which has some computation and wireless communication capability and as well as a GPS receiver. Thus, a vehicle can know its location and velocity through its GPS device, and communicate with other vehicles by wireless channels. Furthermore, each vehicle stores a digital map on the local device. Indeed, with the popularity of smart phones, GPS and PDAs, many vehicles have already been equipped with such devices.

B. Traffic Study and Motivation

To motivate our research, we first investigate the road traffic conditions in a big city. Fig.1 shows the traffic condition of a normal day, Tuesday, March 10, 2009, in Beijing, based on real traffic data acquired from BTMB (Beijing Traffic Management Bureau). The shown area is within the 5th Ring Expressway, which is a 25km*25km core area of Beijing metropolitan. Fig.1(a) shows the traffic condition at *peak hour* (6:33pm), when commuting traffic dominates all the roads. We observe that about 30% of roads (in black) are jammed (the speeds of vehicles on these roads are below 10 km/h); about 30% of roads (in dark gray) are in low speed states (10~25 km/h); only 40% of roads (in light gray) exhibit smooth traffic (25-80km/h), where those roads are mostly distributed outside the city core. Fig.1(b) shows the traffic condition at *normal daytime* (1:05pm). The situation is better but still in serious conditions. We can observe that jammed roads account for about 15% of all the roads.

For an individual car driver, probably the more concerned problem is how to get to his/her destination at the earliest time. Traditional GPS navigation systems can compute the shortest distance paths, but shortest-time paths often differ from the shortest distance paths when there are traffic jams. Our traffic data reveals that, in a metropolitan city like Beijing, other than path length, the traffic condition, especially traffic jam, is a key factor affecting travel time.

We note that while traffic jam itself is the problem of concern, on the other hand it provides an ideal platform to adopt VANET technologies. The high vehicle density, especially in the time of traffic jams, provides a good opportunity for vehicles to be connected into a big citywide network through wireless channels. Thus, we are motivated to leverage VANET techniques to realize a dynamic navigation system to find shortest-time paths.

Suppose a vehicle wants to reach its destination as soon as possible. Based on its digital map, it identifies several possible paths (called *candidate* paths), where each path is composed of multiple road segments. If the source vehicle knows the *current* vehicle moving speeds at all these segments, it can easily estimate the driving time along each path and then follows the one giving the shortest time. It may search for the shortest-time path again after some time interval (e.g., 10 minutes) to adjust to road condition changes. The essential research question is:

How to acquire the knowledge of the current vehicle velocities at road segments of interest?

There are several possible answers to the above question. A proactive way is to request every vehicle to exchange its velocity table with its neighbors or flood its own velocity information to the entire network. Without considering packet losses due to unreliable wireless channel and node mobility, ideally the proactive way allows every vehicle to maintain a global velocity table, based on which it may locally determine the shortest-time path to anywhere in the city. Unfortunately, such proactive solutions are not scalable. Not only a node has to maintain a huge dynamic table, the wireless channels will be easily congested by the tremendous number of control messages in such routing protocols. On the other hand, general reactive approaches similar to DSR [23] or AODV [24] are not efficient either because they involve network-wide flooding to first establish a path to a destination before data forwarding.

In our application, with GPS devices, vehicle nodes know their geographical locations already, so it is the most appropriate to employ a geographic-based data dissemination protocol in which the geographic routing path of a message is specified in the message. Now, through geographic routing, if the source vehicle can send a query message to learn the moving speeds of vehicles in all the segments on the path, our problem is solved.

Though the problem sounds simple, an efficient design is not obvious. A challenging problem is to minimize the communication overhead introduced by a

huge number of vehicles. For the map shown in Fig.1, if we set the distances between vehicles as 10m, 20m, and 25m~500m for the three types of roads (i.e., jammed, low-speed, and smooth), respectively, there will be more than 160,000 vehicles in the total 2,308 road segments at the same time on peak hour. Without a careful design, extraordinarily high communication cost and channel collisions will paralyze the underlying VANET. Another research challenge comes from network disconnection. As we shown in Fig.1(a), even in a peak hour some road segments outside the city core have smooth traffic flows. As a result, the connectivity of VANET is not reliable there. Now, how can we deliver the query messages through these segments, or what actions to take if a query message cannot be forwarded because of either network disconnection or unreliable communication channels? Last but not the least, how to estimate the moving speeds of vehicles in a segment?

In summary, the main design goals of our dynamic navigation system are as follows:

- Efficiency: avoid unnecessary communications
- Robustness: deal with abnormal circumstances, such as wireless network disconnection
- Validity: provide valid real-time traffic information

C. Solution Overview

With respect to the above three design goals, we employ the following strategies. First, to reduce communication overhead, for every query, only a small number of vehicles will be selected to forward the query message and its reply message (Section IV.B). To further improve the system efficiency, we will provide query optimization exploiting the fact that the candidate paths of different querying vehicles may be overlapped. Specifically, we will propose two optimization strategies to reduce the communication cost: caching query results and suppressing redundant queries (Section IV.C).

Second, to handle network disconnection, we will borrow the idea of store-carry-forward [22]. After a number of retransmissions, if a vehicle cannot find the next-hop forwarding node toward the destination, it will carry the query message for a while (Section IV.D). Note that our application has certain timeliness requirement, that is, the source vehicle should know the query result within a time limit (say, a few minutes) to make a prompt decision on which path to take. When it does not receive a query reply message in time, it could be because either the query or the reply message was carried for an overly long time. These two cases have different indications and will be treated differently (Section IV.D).

Third, to acquire the moving speeds of vehicles in a road segment, we will sample a number of moving vehicles in the segment and aggregate their average speed. In general, more vehicles are sampled, the higher accuracy is resulted; however, the higher communication overhead will be incurred. So a tradeoff between performance overhead and sampling accuracy is needed, our protocol provides an easy way to enable such tradeoffs (Section IV.B).

IV. PROTOCOL DESCRIPTIONS

In this section, we present our dynamic navigation protocol in detail. We start by describing the overall process of our protocol in Section IV.A, while leaving the detail of our message forwarding algorithm *DDFS* (Distance Dependent Forwarder Selection) to Section IV.B. Optimization and abnormal situation handling are discussed in the end.

A. Routing &Traffic Data Aggregation

In our protocol, a source vehicle acquires the traffic information by sending a query message toward the vehicles on each candidate path, where each query message has its path information embedded. Specifically, a query message contains the following fields:

< type, origin_seg, dest_seg, sender_location, path_info, expiration_time >

- *type* : type of the message: query
- *origin_seg, dest_seg*: identifier of the source and destination road segment, respectively
- *sender_location*: geographical location of the current sender, either the original source or a forwarder
- *path_info <seg_id, aggregate, count>*: a sequential list storing information about all the segments in the candidate path. *seg_id* is the unique road segment identifier obtained from the local digital map*; count* records the number of hops the message has been forwarded in this segment and its initial value is zero; *aggregate* stores the aggregated traffic information. For our navigation application, it contains the average moving speed in the segment and its initial value is also zero.
- *expiration_time:* by which the query process is aborted. That is, the ongoing message, either a query request or a reply, should be discarded after this time. It is calculated as the sum of *request_time* (query initiating time) and *max_rtt_time* (the maximum round-trip time acceptable to the query initiator). *max_rtt_time* is application dependent. A user may set it based on his need, but should also consider the length of the path from source to destination and the traffic condition. In general, the larger the distance is, the longer time the query travels to the destination. If the path is fully connected, the travel time is determined by the multi-hop wireless communication only. In this case, it is up to a few seconds for forwarding to a very far destination. However, when the path is partially disconnected due to insufficient vehicle density, vehicles will carry the query message and forward it later. In this case, the actual round trip time of a query message could be very long when it is eventually delivered. In practice, by that time the aggregated en-route traffic information could have

become outdated to the user. With these factors in mind, we consider something between 30 and 300 seconds could be a good choice for *max_rtt_time*.

When a vehicle on the candidate path receives a query message, it will first compare the current time with *expiration_time*. If the query has expired, it will discard the message. Otherwise, if it is elected to forward the query by our DDFS algorithm (to be introduced later), it identifies the entry *<seg_id, aggregate, count>* in *path_info*, based on the id of the road segment it is currently located in. Assume the current *aggregate* is v_{avg}, the current velocity of the vehicle is v_{own}, and *count* is c, the new *aggregate* will be updated as v'_{avg} according to (1) and then *count* is increased by one.

$$v'_{avg} = (v_{avg} \cdot c + v_{own}) / (c + 1) \quad (1)$$

It also changes *sender_location* to its own location. After the update, the vehicle caches the query message and forwards the query based on our DDFS algorithm.

When the query arrives at its destination, the vehicles in the destination segment will also check the *expiration_time* field to decide whether to reply. If not expired yet, any vehicle in the segment can reply and the first response suppresses the others. A reply contains the following fields:

<type, origin_seg, dest_seg, sender_location, path_info, expiration_time >

- *type:* the type of message: reply
- *origin_seg, dest_seg:* ids of the source and destination segment, respectively. It reverses the corresponding field in the query message
- The meanings of *sender_location, expiration_time, and path_info* are the same as in a query message.

When a node receives a reply message, it will check the *expiration_time* field. If the query has expired, the reply will be discarded. Otherwise, the node will cache the *path_info*. If this node is selected as the next forwarder based on our DDFS algorithm, it changes *sender_location* to its own location and rebroadcast the reply message. Otherwise, if this node is located in *dest_seg*, it will not forward the reply message. Finally, if it is the originator of the query, it will process the message to calculate the travel time on this path if followed.

Specifically, suppose a candidate path includes k segments, the average vehicle speed at segment i is v_i, and the length of the segment is l_i, which can be acquired from the local digital map. Thus, the potential travel time travel along the candidate path can be calculated by (2). After it receives reply messages from multiple candidate paths, the path with the minimal travel time will be chosen as the driving path.

$$t_{travel} = \sum_{i=1}^{k} l_i / v_i \quad (2)$$

From our experimental results, we observe that query delay is within one minute in most cases and a querying vehicle is often still on the original road segment. Indeed, even if the vehicle has left the original segment, it will run on one of the candidate paths, so some corresponding traffic data could still be heard by the vehicle. We also note that during message forwarding, it may happen that the data forwarding path becomes broken. Section IV.D describes how to deal with such circumstances.

B. Distance Dependent Forwarder Selection (DDFS)

We need a rule to select message forwarders to avoid broadcast storm. In our protocol, vehicles do not exchange real-time location information with neighbors because the high mobility and high density of vehicles will lead to prohibitively high communication overhead. However, without such knowledge, it is not easy for a sender to determine an appropriate next-hop forwarder. So we propose the following DDFS algorithm to address this problem.

Specifically, when a vehicle receives a message, it will perform the following operations:

1. *Check its qualification for forwarding*. First, a forwarding candidate must be on the forwarding path. To know whether it is the case, the vehicle searches the *path_info* field with its current segment id. Second, its position should also be between the previous sender and the final destination of the message. The location of the previous sender is obtained from the *sender_location* field in the message and the final destination is *dest_seg*. This second condition ensures that the next forwarder will bring the message closer to the destination than the previous one did. Third, if this is a query message, only the vehicles whose driving direction is consistent with the forwarding path are considered as candidate forwarders. This is because the query initiator is only interested in the velocities of the vehicles moving in the same direction, not that on the opposite direction. If this is a reply message, a forwarder could be a vehicle heading at either direction, as long as the first two conditions hold. This is because a forwarder for reply messages does not participate in velocity sampling.

2. If a vehicle finds that it is a forwarding candidate, it will delay for a certain time interval before forwarding the message. The forwarding delay, t_{wait}, depends on its distance from the previous sender, decided by (3).

$$t_{wait} = \left| d_{cs} - D_{ref} \right| \cdot t_{ref}, \quad (3)$$

- D_{ref}: the Euclidean distance between an ideal forwarder and the sender, rounded to an integer. It is a system parameter.

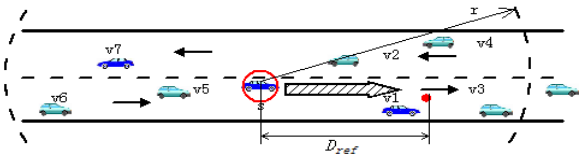

Figure 2. Forwarder selection

- d_{cs}: the Euclidean distance between the candidate and the sender, rounded to an integer
- t_{ref}: the granularity of forwarding delay, e.g.,0.05ms

From the formula, we can observe that when $d_{cs} = D_{ref}$, $t_{wait} = 0$. Thus, the vehicle at the ideal distance D_{ref} will immediately reply, which suppresses the others from forwarding. The closer a vehicle is to the ideal distance, the sooner it will respond. Clearly, the value of D_{ref} is critical. Generally, a larger D_{ref} will result in less number of hops, thus saving communication cost and reducing query delay. However, for a query message, it is desired that a good number of vehicles may contribute their velocity information so that the aggregate of velocity in a segment is more accurate. This implies that D_{ref} should not be too big. As such, for a query message a tradeoff between accuracy and performance is needed for the value of D_{ref}. When forwarding a reply message, we simply set the D_{ref} as the wireless radio range.

3. If before t_{wait} times out, the vehicle hears another candidate forwards the message, it will not forward the message; otherwise, the vehicle forwards the message. In either case, it caches a copy of the message.

4. If the vehicle hears additional forwarding of the message, it will silently drop the redundant one(s).

In Fig.2, *s* is a message sender, and *r* represents the radius of the wireless transmission range. The shadow arrow indicates the direction of the forwarding path. The red dot marks the location of an ideal forwarder for query. We can observe that only *v1, v2, v3* and *v4* are between the sender and the destination. When forwarding a query message, *v1* and *v3* are qualified as forwarding candidates. *v1*, which is closest to ideal location, will forward the message first. Other candidates will not forward the message any longer. When forwarding a reply message, *v1, v2, v3* and *v4* are all qualified as forwarding candidates. *v3*, which is nearest to wireless transmission range, will forward the message and suppress other candidates.

C. Optimization

Exploiting the fact that the candidate paths of different vehicles probably overlap, we propose two optimization techniques to reduce the communication cost. As illustrated in Fig.3, a candidate path of *v1* (solid lines) and that of *v2* (dotted lines) have the common segments *s2*, *s3* and *s4*. Our basic idea for performance optimization is to cache recent query results for future uses and suppress redundant requests.

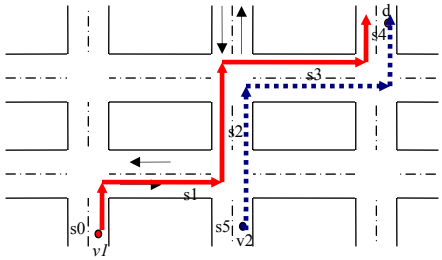

Figure 3. Forwarding optimization

Consider our first scenario. Suppose *v1* has sent a query *q1* to gather the road traffic information on its candidate path and received the reply *r1*. After a short time interval, *v2* sends a query *q2* to the same destination *d*. If the vehicles on *q1*'s path have cached *r1*, which contains the traffic information of *s2*, *s3* and *s4*, when *q2* reaches the segment *s2*, *r1* can be used to construct an immediate reply. Thus, *q2* need not be forwarded any farther, avoiding the communication cost for message forwarding.

Specifically, in our system, each vehicle maintains a table *global_info*<*seg_id, traffic_info, expiration_time*> storing the recent traffic information. Every time it receives a *reply*, it updates the table with new traffic information. New traffic information includes new segments that the table does not contain or old segments with later *expiration_time* than that in the table. The expired entries are removed from the table. Now, when a vehicle receives a *query* and finds that it is a forwarding candidate, it searches the *global_info* table first to decide whether to relay the request. If the *global_info* table contains the traffic information for all the remaining segments in the candidate path, the vehicle will generate a reply immediately. Otherwise, it will forward the query as in the normal mode.

Next let us look at the second scenario. In Fig.3, *q1* has been forwarded on *s2*, but the reply has not come back yet. Now *q2* reaches *s2*. Since *q1* covers the remaining segments of *q2*, which are *s2*, s3 and *s4*, it is unnecessary to continue forwarding *q2* to *d* as long as their expiration times are close or if *q1* is expiring after *q2*. To enable this optimization, each vehicle maintains a *history_req* table to store all received query requests that have not been acknowledged with a reply yet. Two circumstances can trigger the deletion of a request *q* from *history_req*: receiving reply of *q* or *q* has expired. When a vehicle receives a new request q_c, if there exits a request q_h in the *history_req* table which covers the remaining segments of q_c, q_c will be suppressed. Here request q_c is called dependent of q_h. A table *hold_req* is used to record such dependent among query requests. Thus, when a reply comes back, the dependent queries can also be answered and then deleted from the *hold_req* table.

D. Handle abnormal transmission

Wireless communication might have packet losses, and some areas may not have enough high vehicle density to keep network connected. Thus, queries may get lost. The question is how to infer the cause and handle the problem. Our protocol adopts the following error handling mechanisms.

As a message forwarder, if it does not hear any neighbor rebroadcasting the message after a time threshold t_{fwd}, it is likely that the message delivery is failed. Here t_{fwd} is the maximal one-hop round-trip time, whose typical value is within 20ms in our protocol. In this case, the forwarder will try to retransmit it for up to n_r times (in this study we set $n_r=1$) unless a successful delivery happens earlier. If still not succeed, the vehicle will adopt the store-carry-forward strategy. That is, it will store the message in its cache, carry the message during its movement, and forward the message when a new neighbor is detected. The message is discarded from the cache if it has expired.

Our protocol has a special treatment for query messages which were not forwarded. When a forwarder changes from the retransmission mode into the store-carry-forward mode, it will immediately generate a *partial* reply message, which includes all the traffic information collected so far. This message is sent back to the querying user, allowing the user to infer the status of path connectivity. Specifically, when the originator receives a reply which contains the traffic information only from a part of the candidate path, it can conclude that there is wireless disconnection on the candidate path. While such unpredictable factors as harsh environment or the imperfection of wireless technologies may also lead to path disruption, in this study we mainly attribute it to low network density. Accordingly, the originator will simply assume that the vehicles are moving at the speed limit at the segment where the connection was lost. For the remaining segments, unless it receives further (partial) reply, it has no information on their traffic statuses. In this case, they can be estimated based on some historical data or empirical data.

Finally, if a query originator does not receive any reply, including any partial reply, in the lifetime of the query, it is mostly likely because its *reply* message has been lost during its forwarding. The originator may resend a query at its new location.

V. PERFORMANCE EVALUATION

In this section, we first introduce our simulation environment. Then, we evaluate the performance of our protocol.

A. Simulation Environment Setup

Our simulation area covers a 25km*25km zone, which is the central region of Beijing. First, we abstract the artery segments on the digital Beijing map into a graph, where edges represent the road segments and vertexes represent the junctions of roads. Thus, this area contains 2,308 edges and 1,649 vertices. Then, we collected the real traffic data from BTMB at two times. These data sets represent the typical traffic characteristics at normal daytime and peak hour**.** Fig.1 illustrates the traffic condition at these two time points.

According to the traffic status at peak hours, we estimate that there are more than 160,000 vehicles running at the same time. We notice that although some work [26,27] have studied the simulation of VANET-based transportation systems by integrating network communication, vehicle mobility, and drivers' behavior, they are not directly applicable to such a large-scale complex urban environment considered in this paper. As such, we adopt a divide-conquer simulation method. We divide the simulation into two levels of hierarchies: intra-segment and inter-segment. First, we use an event-driven simulator called GLOMOSIM [19] to simulate our routing protocol in a segment with various traffic conditions. This is to study the intra-segment vehicular communication characteristics. Then, we develop a tool to simulate our protocol in the whole road network. According to the traffic data from BTMB, we can acquire the traffic status of each road segment. Based on the intra-segment simulation statistics, each segment is assigned a communication model based on its traffic status. We measure the overall query delay and drop ratio, which reflect the performance of our protocol. Moreover, we show the benefit of shortest-time path navigation.

B. Wireless Communication Characteristics in a Segment

We first study the delay for a message to travel through a road segment via wireless communication. With such knowledge, later on we can evaluate our protocol in a large-scale environment. Our experiments simulate the transmission along a road segment of one-kilometer long with two lanes. We set the average distance between vehicles on the same lane as 10m, 20m, 40m, 100m, which represents the typical traffic condition at jam, light jam, normal in urban and in outer areas. Thus, if we use the number of vehicles per kilometer to represent vehicle density, the corresponding vehicle density is 200, 100, 50 and 20, respectively. In our protocol, the size of a message is usually no more than 512 bytes, so we set the packet size as 512 bytes.

TABLE I. PARAMETERS USED IN OUR GLOMOSIM SIMULATION

Parameter	Value
Simulation area	1km*10m
Vehicle density (nodes/km)	200, 100, 50, 20
CBR (packets/second)	10,50,100
Data packet size	512Bytes
Radio range	203m
MAC protocol	802.11
D_{ref} (m)	200,100

We measure the transmission delay with various CBR rates and vehicle density levels. As shown in Fig.4, the maximum transmission delay is less than 0.065s, which happened at the highest vehicle density and CBR rate, whereas the minimum delay is around 0.025s. So, if no wireless disconnection on a candidate path of tens of kilometer long, the query delay should be in a few seconds. We can also observe that when changing the ideal forwarder distance D_{ref} from 200m to 100m, the transmission delay increases by 36% on average. Hence, the transmission delay is acceptable when forwarding a query at a smaller step to sample more vehicle speeds in a segment.

Fig.5 shows the transmission delay in each of the artery segments in Beijing at two different times, where black line mean that wireless communication in this segment is fully connected and the delay is within one second and blue line (light color) means wireless connection is broken in this segment and the delay is usually more than 10 seconds. We can see that at the peak hour most segments have good wireless connection.

Through simulations, we also study the packet delivery ratio and observe that the minimum packet deliver ratio in a segment is 98%, which happens when the CBR is 100 packets/second and vehicle density is 200, whereas in the other cases the delivery ratio is 100%. Therefore, our protocol can work well when less than 100 packets /second are forwarded in a segment.

C. *Query Delay*

Query delay includes the time for forwarding a query and the time for forwarding its reply message along one or multiple segments. Based on our previous intra-segment simulation results, we can easily estimate the total transmission delay of a query according to the segments it needs to traverse. With the traffic data at a peak hour and a daytime, we first estimate the number of vehicles on each segment. For each vehicle in the network, we randomly choose a destination for it. Fig.6 is the CDF (Cumulative Distribution Function) of average query delay at the peak hour for the distance of 5±2km, 10±2km, 15±2km, 20±2km, respectively. We can observe that query delay increases with the query distance. When the distance is 5±2km, 100% of query delays are within 150s, but when the distance is 20±2km, 95% of query delays are within 300s and 5% of them are even larger. There are two reasons for such large query

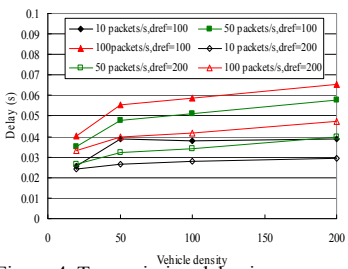

Figure 4. Transmission delay in a segment

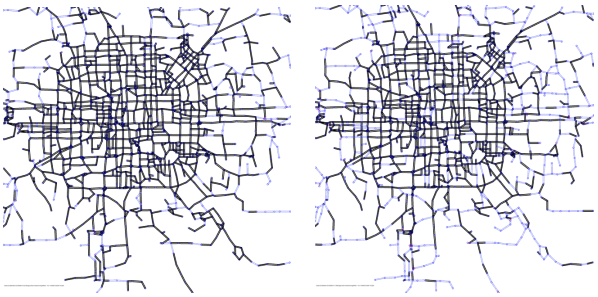
(a) Peak hour (b) Normal daytime

Figure 5. Transmission delay of segments

delay. First, the forwarding vehicles enter the store-carry-forward mode. Second, some queries/replies are lost during transmission. Nevertheless, overall, 87% queries have replies coming back from their destinations in 60s at the peak hour.

D. *Query Drop Ratio*

We also measure the packet (query) drop ratio of our protocol at different times. In our protocol, queries are dropped in two cases: wireless communication is unreliable or queries are expired. Here we set the lifetime of queries as 300s. As shown in Fig.7, when the transmission distance is as short as 5±2km, almost no queries are dropped at the peak hour, and only 1% queries are dropped at the daytime. Packet drop ratio increases with transmission distances because it is likely that a forwarding path includes more low-density segments. However, the drop ratio is always acceptable at a peak hour, and it is only 6% even at the case of driving 20±2km. In a normal daytime, the average packet drop ratio for queries traveling 15±2km is about 16%, so our protocol is still helpful in most cases. Indeed, for a city as big as Beijing, when we randomly choose a destination for each vehicle, 88% of vehicles travel no more than 15km. Finally, we note that for vehicles whose queries were dropped during long-distance transmissions, they may resend their queries after they are closer to their destinations, say, within the 10km or 5km range. From the figure we can observe very low drop ratio in such a query range.

E. *Impact of Cache*

To evaluate the effectiveness of our cache based optimizations, we measure the active query rate (AQR) of a segment. AQR is the number of queries that are actively forwarded along a segment in one second. Recall that because of our caching mechanisms, some queries need not to be forwarded further. Clearly, a smaller AQR is preferable, because it implies lower communication overhead and less channel contention.

From the BTMB data, we observe that traffic condition normally does not change rapidly, so we set the lifetime of traffic information in cache as 10 minutes, and each vehicle sends five queries in 10 minutes. Our simulations show that with cache, the average AQR is

reduced by up to 87% at the peak hour, and by 76% at the normal daytime, compared with that without cache. In more detail, Fig.8 shows the CDF of AQR comparison between with query caching and without query caching. We can see that at the peak hour, with cache 95% AQRs are below 40 packets/second in a segment; whereas without cache only 45% AQRs are below 40 packets/second. Referring to our results in Section V.D, we can see that at this level of packet rates, the packet delivery ratio is close to 100%. The figure also shows that without cache, AQRs at the peak hour are higher than that at the normal daytime. This is because more queries are issued at the peak hour. On the other hand, with cache, AQRs at the peak hour are slightly lower than that at the normal daytime. This is because the cache hit rate is higher at the peak hour.

Next we compare query delays between with cache and without cache. We find that caching can greatly decrease query delay. As illustrated in Fig.9, at the peak hour, when the transmission distance is 10±2km, with cache 86% queries can be answered in 2 seconds. On average, with cache query delay is reduced by 61% compared with that without cache.

F. Efficiency of Navigation Along Shortest Time Paths

To show the benefit of shortest-time path navigation, we compare it with shortest-distance path navigation at peak hour. For each vehicle, we randomly choose a destination for it. In Fig.10, the x-axis is the time saving ratio of shortest-time path over shortest-distance path, y-axis is its CDF, and the four curves correspond to the portion of vehicles whose travel distances are 5±2km, 10±2km, 15±2km and 20±2km, respectively, at peak hours. The point *P* means that for vehicles driving 5±2km, 40% of them have up to 19% time savings, or in other words, 60% of them have over 19% time savings, by taking the shortest-time path instead of shortest-distance path. Overall, in this study shortest-time path navigation saves 30% time on average. It is especially beneficial to those driving at a longer distance, because the time saving is more obvious when a vehicle's travel distance is longer. There are less or no time savings for short driving distances; for example, as indicated by point *Y*, 15% vehicles driving 5±2km have 0 time saving ratio. This means that these two paths are the same. The reason could be that at peak hours the potential time saving due to less jamming in a detour route is no more than the extra time needed to travel a longer distance.

Vehicles following the shortest-time path often need to drive at a longer distance in detour routes. Fig.11 shows that the driving distance is increased by 37% on average at peak hour, and the increase is more obvious for long-distance travels. In practice, a user has to make a choice between saving travel time and saving travel path length.

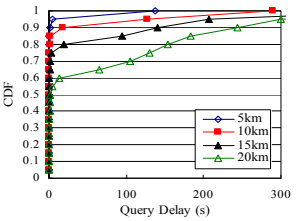

Figure 6. Query delay

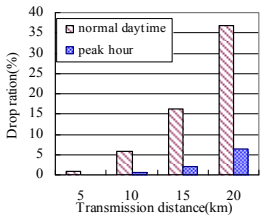

Figure 7. Query drop ratio

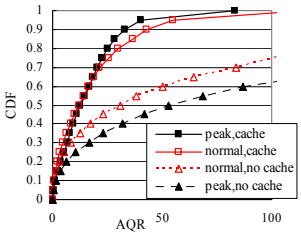

Figure 8. AQR Comparison between with cache and without cache

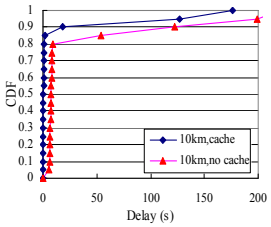

Figure 9. Query delay comparison between with cache and without cache

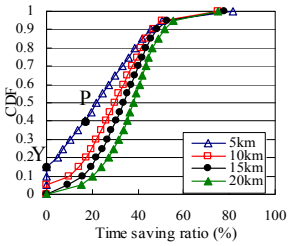

Figure 10. Efficiency of shortest-time path navigation

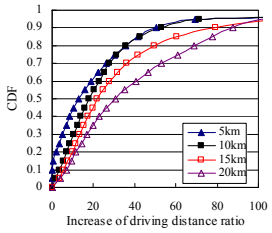

Figure 11. Increase of driving distance ratio

VI. DISCUSSIONS AND FUTURE WORK

With no surprise, there is a price to pay for the benefit of saved driving time. Vehicles following the shortest-time path often need to drive at a longer distance in detour routes. We are not proposing that every vehicle should drive at the shortest-time path, but provide a technique for those users who care more about traveling time than other factors. We believe that in practice, the introduction of our protocol will not cause all the vehicles to change their original routes. This is because (1) some users care more about driving distances because of the extra gas costs for detouring; (2) based on our experiments, we already observed in some cases shortest-time path and shortest-distance path are identical; (3) in a metropolitan city, a great number of vehicles (e.g., city buses) always take fixed routes; (4) some vehicles do not run our protocol. This makes it possible for users who take detours to benefit. In our future work, we will study the traffic pattern changing problem in more detail with modeling and simulation at different scenarios. On the other hand, sometimes people may be simply interested in estimating the remaining travel time to reach the destination along the current driving path. Our protocol can serve this purpose as well, as a byproduct.

Note that our protocol, being simple, basically provides the best effort service. There is no guarantee that it will achieve the optimal performance in all circumstances. For example, the positioning errors of GPS, although small relative to the distance between vehicles, may affect the selection of candidate forwarding nodes. Also, in the real world, road traffic pattern is very complex and the road network is hard to model, so in our initial study, we have not considered the impact of traffic lights when determining the shortest-time path. Different personal driving habits may introduce improper samples to impact the accuracy of v_{avg}. All these issues deserve further examinations in our future work.

Another problem is the incremental deployment problem; that is, what is the performance of our protocol when only a fraction of vehicles in a city participate in the VANET. Our expectation is that it will work well in most cases unless the participating vehicle density is extremely low. Indeed, in our simulation we have already evaluated the impact of vehicle density. But a detailed study on this issue is necessary. Other issues like further optimization, integration with existing data dissemination protocols, security and privacy are all interesting to us.

VII. CONCLUSION

In this paper, we present a VANET-based dynamic navigation protocol for individual drivers to find the shortest-time paths. By actively sending queries along multiple possible paths, a vehicle can gather the real-time vehicular traffic information from each road segment of these paths. Based on the resulted information, drivers can decide the best driving path. Our simulation results show that our protocol is efficient and useful, and it is feasible for deployment in a metropolitan-scale city.

REFERENCES

[1] D. Pfoser,S. Brakatsoulas, P. Brosch, M. Umlauft, N. Tryfona, G. Tsironis, "Dynamic Travel Time Provision for Road Networks", in ACM SIGGIS, 2008.

[2] K. Kwong, R. Kavaler, R. Rajagopal, P. Varaiya, "Arterial travel time estimation based on vehicle re-identification using wireless magnetic sensors", in Transportation Research Part C: Emerging Technologies, May 2009.

[3] Z. F. Syed, P. Aggarwal, X. Niu, N. El-Sheimy, "Civilian Vehicle Navigation: Required Alignment of the Inertial Sensors for Acceptable Navigation Accuracies", IEEE Transactions on Vehicular Technology, November 2008.

[4] A. K. Ziliaskopoulos and J. Zhang, "A Zero Public. Infrastructure Vehicle Based Traffic Information System". Transportation Research Board Meeting (TRB), 2003.

[5] V. Verroios, K. Kollias, P. K. Chrysanthis, "Adaptive navigation of vehicles in congested road networks", in Proceedings of the 5th InternationalConference on Pervasive services(ICPS), 2008.

[6] T. Nadeem, S. Dashtinezhad, C. Liao, L. Iftode, "TrafficView: Traffic Data Dissemination using Car-to-Car Communication", in ACM SIGMobile, 2004.

[7] H. Xu, M. Barth, "An adaptive dissemination mechanism for inter-vehicle communication-based decentralized traffic information systems", in IEEE Intelligent Transportation Systems Conference, 2006.

[8] J. Zhao, G. Cao, "VADD: Vehicle-Assisted Data Delivery in Vehic- ular Ad Hoc Networks," in IEEE INFOCOM, April 2006.

[9] Y. Ding, C. Wang and L. Xiao, "A Static-Node Assisted Adaptive Routing Protocol in Vehicular Networks", in 4th ACM International Workshop on Vehicular Ad Hoc Networks (VANET), 2007.

[10] H. Zhu, Y. Zhu, M. Li, L. M. Ni, "HERO: Online Real-Time Vehicle Tracking in Shanghai", in IEEE INFOCOM, April 2008.

[11] H.Huang, P. Luo, M. Li, D. Li, X. Li, W. Shu, M. Wu, "Performance Evaluation of SUVnet With Real-Time Traffic Data", in IEEE Transactions on Vehicular Technology, 2007.

[12] B.S. Sharif, B.S. Blythe, S.M. Almajnooni, C.C. Tsimenidis, "Inter-vehicle mobile ad hoc network for road transport system", in Intelligent Transport Systems, March 2007.

[13] K. Collins, G.M. Muntean, "A Vehicle Route Management Solution Enabled by Wireless Vehicular Networks", in Mobile Networking for Vehicular Environments Workshop of INFOCOM, 2008.

[14] K. Kim, S. Hwang, K. Li, "Arrival Time Dependent Shortest Path by On-Road Routing in Mobile Ad-hoc network", in W2GIS 2004.

[15] H. Wu, R. Fujimoto, R. Guensler, M. Hunter, "MDDV: A Mobility-Centric Data Dissemination Algorithm for Vehicular Networks", in ACM International Workshop on Vehicular Ad Hoc Networks (VANET) 2004.

[16] http://www.chinanews.com.cn/auto/auto-jtxx/news/2009/06-22/1743221.shtml .

[17] http://www.fhwa.dot.gov/congestion/.

[18] L. Wischhof, A. Ebner, H. Rohling, M. Lott, R. Halfmann, "Adaptive Broadcast for Travel and Traffic Information Distribution Based on Inter-Vehicle Communication", in Proceedings of IEEE Intelligent Vehicles Symposium, 2003.

[19] http://pcl.cs.ucla.edu/projects/glomosim/.

[20] V. Naumov, T. R. Gross, "Connectivity-Aware Routing(CAR) in Vehicular Ad Hoc Networks",in IEEE INFOCOM, 2007.

[21] Z. Mo, H. Zhu, K. Makki, N. Pissinou, "MURU: A Multi-Hop Routing protocol for Urban Vehicular Ad Hoc Networks," in MobiQuitous, 2006.

[22] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets", in SIGCOMM, 2003.

[23] D. B. Johnson, D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks", in Mobile Computing, 1996.

[24] C. E. Perkins, E. M. Royer. "Ad hoc On-Demand Distance Vector Routing." in Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999.

[25] J. Rybicki, B. Scheuermann, M. Koegel, M. Mauve, "PeerTIS - A Peer-to-Peer Traffic Information System", in VANET, 2009

[26] R. Jayakrishnan, H.S. Mahmassani, "Dynamic simulation-assignment methodology to evaluate in-vehicle information strategies in urban traffic networks", Proceedings of Simulation Conference, 1990.

[27] Y. Yang, R. BagrodiaMobile, "Evaluation of VANET-based Advanced Intelligent Transportation Systems", VANET, 2009