HERO: Online Real-Time Vehicle Tracking

Hongzi Zhu, *Student Member*, *IEEE*, Minglu Li, *Senior Member*, *IEEE*,
Yanmin Zhu, *Member*, *IEEE*, and Lionel M. Ni, *Fellow*, *IEEE*

Abstract—Intelligent transportation systems have become increasingly important for the public transportation in Shanghai. In response, ShanghaiGrid (SG) project aims to provide abundant intelligent transportation services to improve the traffic condition. A challenging service in SG is to accurately locate the positions of moving vehicles in real time. In this paper, we present an innovative scheme, Hierarchical Exponential Region Organization (HERO), to tackle this problem. In SG, the location information of individual vehicles is actively logged in local nodes which are distributed throughout the city. For each vehicle, HERO dynamically maintains an advantageous hierarchy on the overlay network of local nodes to conservatively update the location information only in nearby nodes. By bounding the maximum number of hops the query is routed, HERO guarantees to meet the real-time constraint associated with each vehicle. A small-scale prototype system implementation and extensive simulations based on the real road network and trace data of vehicle movements from Shanghai demonstrate the efficacy of HERO.

Index Terms—Distributed applications, real-time system, RFID system, peer-to-peer network, vehicle tracking.

1 Introduction

Intelligent transportation systems (ITSs) [1], [2], [3] have been evolving rapidly in the past two decades, leveraging advanced computing and communication technologies. ITSs help coordinate traffic condition, improve safety, reduce environmental impact, and make efficient use of available resources. Shanghai, the largest metropolis in China, covers an area of 5,800 km^2 and has a large population of 18.7 million. The economy of Shanghai is soaring today, and the growing traffic has become a serious challenge. In response to the challenge and the needs of the public, the Shanghai government has established the ShanghaiGrid (SG) project since 2005, with the ambitious goal of building a metropolitan-scale traffic information system. This project will construct the basic infrastructure, composed of a great number of traffic information collectors and information processing nodes connected through the Internet, for building diverse applications to facilitate the public's transportation. According to the blue paper of the project, wireless access points (APs) and RFID readers will be deployed throughout Shanghai. Exploiting the pervasive deployment of these devices, location and status information of vehicles can be actively captured and logged in a large number of distributed local nodes. The goals of the project are twofold. First, it tries to make the available transportation infrastructure to be used more efficiently. Second, it aims to provide the public with a wide spectrum of ITS applications [13], [14], ranging from real-time traffic information, trip planning and optimal route selection, to congestion avoidance and bus arrival prediction.

Among the others, *online real-time vehicle tracking* is a fundamental service in SG, which refers to tracking the current position of a certain vehicle in real time. A wide spectrum of compelling applications can be implemented on top of this basic service. For example, authorized users will be able to track individual vehicles that they are concerned with, such as their own or friends' cars, public buses, and taxies. In particular, there exist several critical types of vehicles in the city, which need to be located urgently, such as stolen cars, speeding cars, ambulances, and police cars. Besides these application scenarios, it is also an indispensable building block underpinning many other high-level applications. For example, in the bus arrival prediction application, the tracking service is used to locate the nearest feasible bus.

However, real-time vehicle tracking in the metropolitan-scale system is very challenging because of several rigorous requirements. First, users (or high-level applications) often pose a real-time requirement on tracking a certain vehicle. That is, any query for the vehicle must be answered within a certain bounded time. Otherwise, the returned answer may become invalid or useless. For example, a query tries to locate the current location of a stolen car. If the query fails to be answered within a short time, the car could actually be far away from the returned location because it may be moving at a high speed. Second, the system should be scalable to support hundreds of thousands of vehicles. In addition, SG aims to serve millions of users every day. The huge number of simultaneous queries is a serious issue. In addition, as the Shanghai City is continuously expanding, the system is required to be highly extensible to such expansion. Third, the system should be robust to node failures. In such a large-scale distributed system consisting of thousands of local nodes, system maintenance is not a trivial issue.

- *H. Zhu and M. Li are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Room A-303, SEIEE Building, 800 Dong Chuan Road, Min Hang, Shanghai 200240, P.R. China. E-mail: hongzi@cs.sjtu.edu.cn, mlli@sjtu.edu.cn.*
- *Y. Zhu is with the Distributed Software Engineering Center, Department of Computing, Imperial College London, Huxley Building, 180 Queen's Gate, SW7 2RH London, UK. E-mail: yzhu@doc.ic.ac.uk.*
- *L.M. Ni is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: ni@cse.ust.hk.*

Manuscript received 7 Mar. 2008; revised 19 July 2008; accepted 28 July 2008; published online 1 Aug. 2008.
Recommended for acceptance by T. Abdelzaher.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2008-03-0091.
Digital Object Identifier no. 10.1109/TPDS.2008.147.

1045-9219/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society

To realize this service, a centralized scheme is straightforward, where location information of all vehicles is sent back to a centralized database and constantly maintained. A user, who wants to track a vehicle, can send a query to the central server. The server then processes the query and returns the location information of that vehicle to the user. However, it is infeasible for the metropolitan-scale system due to the huge amount of vehicle data streams. For example, there are 22,413 crossroads in Shanghai. Even in 2001, the average number of vehicles running across a crossroad per minute in daytime was up to 33 [4]. This produces, in total, about 12,000 events per second. Such a large volume of location updating data can easily overwhelm the centralized server. Therefore, it necessitates efficient designs of distributed solutions. As an alternative scheme, captured vehicle information can be stored locally at distributed nodes. As a result, there is little updating data as in the centralized scheme. Nevertheless, by this means, there is no hint about the inquired vehicle for a query. To track the vehicle, an intuitive scheme is to flood the query across the network which can always locate the desired vehicle. However, flooding search incurs a large amount of network traffic and, hence, is subject to poor scalability. To reduce query traffic, there is another search scheme based on random walks, which introduces modest network traffic. But, this scheme is limited by the problem of unbounded response latency of the query. As a result, there is no existing successful solution, to the best of our knowledge, to tracking vehicles in real time in a large-scale distributed system.

In this paper, we propose a novel scheme, Hierarchical Exponential Region Organization (HERO), which satisfies the unique requirements of real-time vehicle tracking in a metropolitan-scale distributed system. In essence, HERO connects local nodes into an overlay network matching the underlying road network. A hierarchical structure over the overlay network is constructed and dynamically maintained while the vehicle is moving along. Exploiting the inherent spatiotemporal locality of vehicle movements, this hierarchy enables the system to conservatively update location information of a moving vehicle only in nearby nodes. The distinctive features of HERO are twofold. First, it guarantees that any query, which can be injected anywhere in the city, can meet the real-time constraint associated with each vehicle, by bounding the maximum number of hops that the query is routed. Second, it significantly reduces the communication overhead of both location updating and query routing, and therefore is truly scalable to support hundreds of thousands of vehicles and millions of system users. Moreover, HERO is a fully distributed lightweight protocol extensible to the increasing scale of the system. In addition, it is robust to node failures and able to tolerate inaccurate location readings.

The original contributions that we have made in this paper are highlighted as follows:

- We propose a novel protocol HERO for real-time vehicle tracking in a metropolitan-scale ITS. HERO employs a distributed technique to store the large volume of vehicle information. It guarantees to meet the real-time constraint, in terms of the number of hops that a message has to traverse, associated with a vehicle for answering any query about the vehicle, and is truly scalable with respect to the number of vehicles, the number of queries, and the system scale.
- We conduct in-depth theoretical analysis, identify the tradeoff relationship between the communication overhead and the query response time, and draw an optimal configuration of system parameters of HERO, minimizing network traffic under real-time constraints.
- We have built a small-scale prototype system to track experimental vehicles in the campus which covers an area of 322 acres. This experiment system validates our design of HERO and demonstrates its practical implementation.
- We evaluate the performance of the HERO approach through precise trace-driven simulations. We base our simulations on the real road network and trace data of vehicle movements in Shanghai, and compare the performance of HERO with other alternative schemes.

The rest of this paper is structured as follows: Section 2 compares HERO with related work. In Section 3, we introduce the infrastructure that will be deployed in the SG project. Section 4 elaborates the design of HERO and presents theoretical analysis for the optimal configuration of the protocol parameters. Several design issues that may be encountered in practice are discussed in Section 5. Section 6 describes our prototype implementation of the vehicle tracking system realizing the HERO protocol. In Section 7, we introduce the trace-driven methodology that we use to evaluate the performance of HERO and present simulation results. Finally, we present concluding remarks and outline the directions for future work in Section 8.

2 Related Work

The Globe system [15] has constructed a static worldwide search tree for mapping object identifiers to the locations of moving objects. It is not flexible to expand or adjust the structure and may have the bottleneck problem near the root of the directory tree structure. In [16], the authors have introduced a distributed approach for load balance but they have not taken the number of system users into consideration. In contrast, HERO needs no dedicated directory servers and achieves good scalability and flexibility.

In database community, indexing techniques have been proposed for tracking moving objects but they are based on the assumption of the existence of centralized databases [17], [18], [19], [20]. Despite the large number of existing methods, there is no applicable one for update-intensive applications, where it is infeasible to continuously update the index and process queries at the same time [21]. HERO does not need any centralize routing information is distributed to every node in the system.

In structured peer-to-peer (P2P) networks, various DHT schemes have been proposed to map objects to peers in a decentralized way, thus enabling the system to satisfy queries efficiently [22], [23], [24], [25]. However, DHTs may cause large computation and traffic overhead for a large number of rapid updates of moving objects. In unstructured

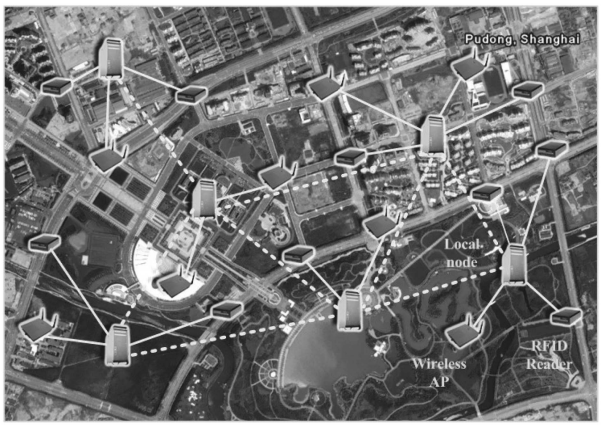

Fig. 1. The infrastructure of SG; a small part of the Pudong District of Shanghai is shown.

P2P networks, the most typical query methods are based on flooding [12]. Using flooding is not scalable. Several randomized approaches, such as random walks [26], [27] and randomized gossip-based methods [28], [29] have been introduced to distribute and locate objects. Random walks are resilient to node failures but need sufficiently long walks before finding the results in a stable network. Random gossip-based methods can retrieve global information with high probability after approximately logarithmic rounds but introduce large traffic. Furthermore, none of these schemes provides real-time guarantees for queries. HERO introduces minimal updating cost to guarantee the real-time constraints desired by the applications.

3 System Description

As RFID technology continuously evolves, it has been widely used in tracking various mobile objects such as vehicles [5], [6]. The US government also enacts the TREAD Act [7] which demands RFID tags to be planted in every new tire before September 2007. The SG project exploits the promising RFID and local-area wireless communication technologies. The infrastructure of SG, which is still underway, is illustrated in Fig. 1. RFID readers and wireless APs will be deployed throughout the urban area of Shanghai, typically installed at crossroads. A local node is responsible for collecting data from several close RFID readers and wireless APs within its own domain, and accepts queries from nearby users or applications. A local node is basically a server which connects to a dedicated underlying network for communication.

In SG, the vehicles' information is gathered both actively and passively. In the initial prototype of SG, a vehicle is captured passively using active RFID technology. An active RFID tag emits its ID at a fixed interval and has an effective communication range of about 2 to 80 m. The battery can sustain the operation of an active RFID tag for about 6 years [10]. A moving vehicle attached with an active RFID tag can be captured if the emitted signal reaches some reader. Besides active RFIDs, a vehicle can actively communicate with wireless APs as it passes by them. A Cisco Aironet 1240AG AP working under IEEE 802.11g has an effective outdoor communication range of about 280 m at the transmission rate of 2 Mbps [8]. The vehicle can actively push important vehicle status information, such as vacancy status, to local nodes.

Precisely speaking, we aim at providing real-time guarantee of tracking a vehicle by bounding the maximum number of hops that a query could traverse in the system. Since the provision of such a real-time service depends on the underlying network for communication, a dedicated network such as an ATM can be used which provides a reliable and predictable data transmission between any two endpoints. With the bounded maximum number of transmission, such a system for the purpose of tracking vehicles can guarantee rigid real-time requirements. In the initial prototype of SG, we connect local nodes to the wide-area ATM network provided by Shanghai Telecom [9] through a dedicated connection or a cheap ADSL connection.

As another initial pilot effort in Shanghai, certain vehicles (around 6,850 taxies and 3,620 buses) are equipped with Global Positioning System (GPS) receivers, which can provide coarse-grained location information. A vehicle actively reports its location information back to a centralized database through a wireless cell-phone data channel (i.e., GPRS). Several crucial reasons prohibit this initial effort from being extended for vehicle tracking in Shanghai. First, with crowded high buildings squeezed along most of the narrow streets in the city, it is very difficult for the GPS system to work accurately without any other assistant devices. It is often the case that the reported GPS position of a vehicle can be more than 100 m deviated from its actual location. To make things worse, a large number of major roads are covered by viaducts that prevent satellites from seeing the vehicles running under them. Second, the intervals of location information reports can be notably long. Due to the GPRS communication cost for transmitting the GPS location information back to the data center, drivers prefer to choose relatively large intervals. The typical value would be from 1 to 3 minutes. Third, the expense of a GPS receiver as well as data communication cost is quite high, which limits the wide deployment of this technology. However, the trace data of vehicle movements in the urban area of Shanghai obtained from this prototype using GPS technology is very valuable for study of traffic conditions. We evaluate HERO using the real trace data.

4 Design of HERO

In this section, we first give an overview of the HERO protocol, introducing its basic rational. Next, we delve into the conservative location updating based on the assistance of a dynamically maintained hierarchy. Finally, we discuss the optimal configuration of the protocol parameters of HERO.

4.1 Overview

To meet the rigid requirements in vehicle tracking in real time, we need to solve two critical issues. First, the system should limit the maximum query response time to guarantee the real-time constrains from applications. Second, the system should minimize network traffic to support a large number of vehicles and queries as well as the continuous extension of the network.

However, there is an intrinsic tradeoff between network traffic and query response time in vehicle tracking. As

mentioned earlier, by aggressively updating location information of a vehicle to all the other nodes, the system provides minimal query response time whereas introducing high updating network traffic overhead. In contrast, the system suffers from long query response time if the system does not perform any location updating. In general, more rigid real-time requirement on tracking a vehicle implies higher network traffic overhead.

HERO elegantly manages to solve the two critical issues in an integrated way. The core idea of HERO is to dynamically update location information of a moving vehicle to all the nodes in the system *in a controlled way*. Generally, the nodes closer to the vehicle are updated more frequently than those further from it and, therefore, have more accurate information about the current location of the vehicle. By this means, HERO effectively exploits the inherent spatiotemporal locality of vehicle movements in an urban setting, and consequently reduces location updating cost. Upon receiving a query, the node unlikely has the exact information. However, it knows some other node which has more accurate information about the vehicle. Thus, it forwards the query to that node. Following an elaborately organized routing path, the query can eventually reach the destination node, which keeps the most updated information of the vehicle. The typical latency between two nodes can be easily measured. Thus, by bounding the maximum number of hops that the query is routed, HERO can also meet the real-time constraint for the vehicle.

The key to the design of HERO is how to realize the controlled location updating while bounding the maximum number of hops a query is routed. To accomplish this, HERO integrates four effective components:

Overlay construction. To exploit the locality of vehicles' movements, HERO organizes local nodes into an overlay network that matches the real underlying road network in Shanghai (as depicted in Fig. 1, dashed lines present the overlay connections of local nodes). There is a connection between two geographically adjacent local nodes in the overlay network if there is a road between the two corresponding regions. This overlay is easy to build and maintain, with each node having to know its neighbors. Additional overlay connection may also be added for two nodes that are geographically close to each other even if they are not connected by a real road. Such connections enhance the reliability of the overlay network when a local node has only one road connecting itself to other local nodes.

Hierarchy organization. For every vehicle, HERO divides local nodes into different regions which constitute the hierarchy on the overlay network. The regions are organized in the following way, as illustrated in Fig. 2. The first region ($\mathcal{R}_1$) has the smallest size and covers the vehicle. For the example, in Fig. 2, $\mathcal{R}_1$ covers node e, which is closest to the vehicle and has the latest information about it. The second region ($\mathcal{R}_2$) has a larger size and covers $\mathcal{R}_1$. More generally, a region ($\mathcal{R}_i$) has a larger size than the immediate inner region ($\mathcal{R}_{i-1}$) and covers it.

Restricted location updating. When the vehicle is moving within $\mathcal{R}_1$, the location updating involves only the small set of nodes in $\mathcal{R}_1$. When the vehicle is moving out of $\mathcal{R}_1$, the location updating is extended to more regions. In this case, part of the hierarchy needs to be reorganized. This reorganization aims to restrict location updating in $\mathcal{R}_1$ as much as possible, thereby minimizing network traffic cost for location updating.

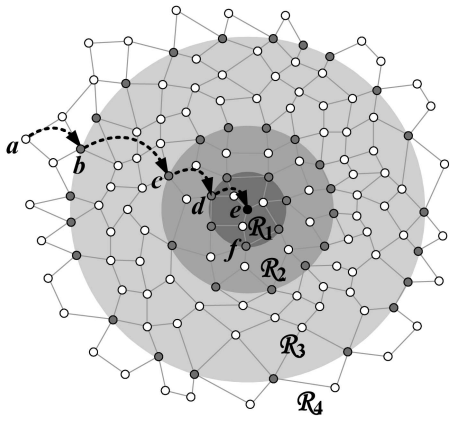

Fig. 2. Illustration of hierarchical regions and query routing.

Query routing. With the hierarchy and restricted location updating, a region always has more up-to-date location information of the vehicle than its outer regions. In HERO, each node has a pointer pointing to a boundary node of its immediate inner region. A query can be injected from any node in the system. For example, in Fig. 2, node a receives a query. Node a will forward the query to b. The query will further be forwarded by nodes c and d, and eventually arrives at e. Node e will return the location information directly back to node a. To restrict the maximum number of hops that the query is routed, we limit the number of regions that the hierarchy for the vehicle contains.

In the following sections, we first describe the process of hierarchy initialization when a new vehicle is joining the system. Next, we describe the detailed mechanism for restricted location updating while the vehicle is moving based on the established hierarchy. Finally, the optimal configuration of design parameters is discussed.

4.2 Hierarchy Initialization

The first node that captures a new vehicle triggers an initialization procedure to establish the hierarchy for the vehicle. As the vehicle may move toward any direction, a region is initially designed as a disk in the overlay network. Note that, the deployment of local nodes is not necessary to be uniform in the city. They can be more densely deployed where more refined tracking accuracy is required. We will discuss more on this in Section 5. In the rest part of this paper, without explicit specification, distance is measured in terms of hops in the overlay network. Each region $\mathcal{R}_i$ has a radius r_i (in hops). A node, which has a distance d from the first node, belongs to region $\mathcal{R}_k$ if this region is the smallest one that covers the node. The radius r_k of $\mathcal{R}_k$ is

$$r_k = \min_{i=1}^{h}\{r_i,\ r_i \geq d\}, \tag{1}$$

where h is the maximum number of regions in the system. If d equals to certain r_i, $1 \leq i \leq h$, the node is on the boundary of r_i. Moreover, for query routing, every node maintains a pointer that points to a node which is on the boundary of the immediate inner region. Let `next-insider` denote this pointer.

TABLE 1
Data Structures Used in the Algorithm

Local node	Init. packet	Update packet

`boundary`
`next-insider`
	`router`
	`journey`
		`router`
		`journey`
		`scale`

To establish `next-insider` pointers in the nodes, the first node initiates an initialization packet which contains a `router` field for setting up these pointers and a `journey` field for maintaining the distance that the packet has traversed. The first node initializes `router` and `journey` to its own IP address and one, respectively. Then, the first node floods the initialization packet throughout the network. Upon receiving the packet, a node first sets its `next-insider` to `router` contained in the packet. Then, it checks `journey` in the packet. If `journey` equals to the radius of certain region r_i, the node marks itself as a boundary node of region $\mathcal{R}_i$. It also modifies `router` in the packet to its own IP. Otherwise, it leaves that field unchanged. Next, it increases `journey` in the packet by one and rebroadcasts the packet to its neighbors. In addition, duplicated initialization packets with larger `journey` are silently dropped. After the initialization procedure terminates, the regions are centered at the first node and the hierarchy is established (as illustrated in Fig. 2). Note that the structure of the hierarchy is distributed in all local nodes (the data structure for a node is shown in Table 1). Therefore, the storage overhead for tracking the vehicle at a local node is very small.

4.3 Restricted Location Updating

When a vehicle is moving in the city, its information is captured by the local nodes that it passes by. When a node captures the vehicle (we call this node *chaser*), it performs location updating, and maintains the hierarchy for the vehicle if necessary. There are three cases. For presentation clarity, we define a node as a *boundary node* of $\mathcal{R}_i$ if it is a most outer node within $\mathcal{R}_i$. The nodes in $\mathcal{R}_i$ except boundary nodes are *interior nodes* of $\mathcal{R}_i$.

Case 1: The chaser is an interior node within $\mathcal{R}_1$. In this case, the hierarchy for the vehicle remains unchanged. The chaser floods the location information of the vehicle to all the other nodes in $\mathcal{R}_1$.

Case 2: The chaser is a boundary node of $\mathcal{R}_1$. In this case, it is possible that the vehicle will move out of $\mathcal{R}_1$ shortly. For example, in Fig. 3, node a is the current chaser which is a boundary node of $\mathcal{R}_1$ (the dashed circle). When the vehicle moves along the depicted direction, $\mathcal{R}_1$ will not cover the vehicle any more. Two consequences follow. First, a future query cannot be routed to the chaser properly because the information on the boundary nodes of $\mathcal{R}_1$ is out-of-date. Second, to enable the proper routing of a future query, the chaser has to flood the location information of the vehicle to $\mathcal{R}_2$ every time, which will incur larger network traffic overhead. Therefore, HERO needs to reorganize $\mathcal{R}_1$.

To this end, the chaser initiates an *update packet* in which its `router` and `journey` is initialized to its own IP address and one, respectively, as in an initialization packet. The update packet includes an additional `scale` filed that is used to indicate the area that the update packet should be propagated to. In this case, the chaser floods the packet within $\mathcal{R}_2$ by letting the boundary nodes of $\mathcal{R}_2$ stop the flooding. On the one hand, the new $\mathcal{R}_1$ is rebuilt from the current chaser within $\mathcal{R}_2$. At the same time, location information is also updated in the new $\mathcal{R}_1$. On the other hand, it updates nodes in $\mathcal{R}_2$ about the current position of the new $\mathcal{R}_1$.

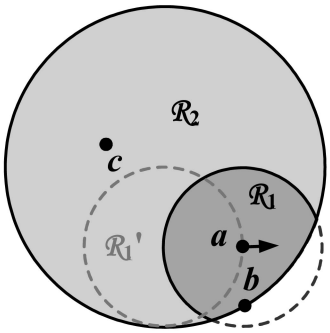

Fig. 3. Reconstruction of $\mathcal{R}_1$, node a is the chaser and is a boundary node of the first region $(\mathcal{R}_1')$.

There is a special situation during the reconstruction of $\mathcal{R}_1$, where the new $\mathcal{R}_1$ is truncated by the boundary of $\mathcal{R}_2$. This happens when the chaser is close to the boundary of $\mathcal{R}_2$ (e.g., node a in Fig. 3). In this situation, a boundary node of $\mathcal{R}_2$ receives an update packet whose `journey` is less than or equals to r_1 (e.g., node b in Fig. 3). As a result, this node sets itself as a boundary node of both $\mathcal{R}_1$ and $\mathcal{R}_2$. We call such a node a common boundary node of $\mathcal{R}_1$ and $\mathcal{R}_2$. In this case, $\mathcal{R}_1$ is no longer a disk because it is restricted in $\mathcal{R}_2$. But, this does not affect the operation of our protocol.

Case 3: The chaser is a common boundary node of several regions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_j$ $(j > 1)$. This is actually a more general situation of Case 2. This situation results from constant reconstructions of regions as the vehicle is moving. In this case, it is possible for the vehicle to move out of all the regions from $\mathcal{R}_1$ to $\mathcal{R}_j$. The system needs to reorganize regions from $\mathcal{R}_1$ to $\mathcal{R}_j$. For example, in Fig. 4, the situation occurs if node b is the current chaser, where b is also a common boundary node of $\mathcal{R}_1$ and $\mathcal{R}_2$ (the dashed circles).

To rebuild regions from $\mathcal{R}_1$ to $\mathcal{R}_j$, the chaser floods an update packet within $\mathcal{R}_{j+1}$. As a result, all regions from $\mathcal{R}_1$ to $\mathcal{R}_j$ are reconstructed within $\mathcal{R}_{j+1}$. In addition, the

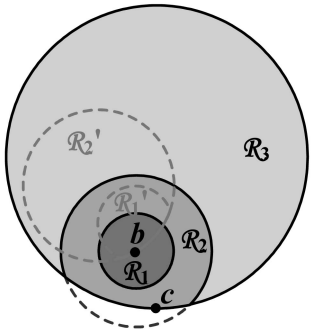

Fig. 4. Reconstruction of $\mathcal{R}_1$ and $\mathcal{R}_2$, node b is the chaser and also is a common boundary node of the first and second region ($\mathcal{R}_1'$ and $\mathcal{R}_2'$).

location information of the vehicle within $\mathcal{R}_{j+1}$ is updated. Similar to Case 2, there is also a special situation during the reconstruction of regions from $\mathcal{R}_1$ to $\mathcal{R}_j$, where several regions, say from $\mathcal{R}_i$ to $\mathcal{R}_j$, might be truncated by some boundary nodes of $\mathcal{R}_{j+1}$. Such a boundary node of $\mathcal{R}_{j+1}$ sets itself as the common boundary node of regions $\mathcal{R}_i, \mathcal{R}_{i+1}, \ldots, \mathcal{R}_{j+1}$, $(1 \le i \le j)$. For example, in Fig. 4, node c is a resulting common boundary node of $\mathcal{R}_2$ and $\mathcal{R}_3$.

Note that the hierarchy needs to be established only once at the time when the vehicle is first introduced in the system. Afterward, it is dynamically maintained in a fully decentralized manner. Therefore, the storage overhead for tracking the vehicle at each local node is small. HERO automatically reorganizes the hierarchy to control the flooding for location updating to happen mostly in the first few smallest regions. Using flooding for the controlled location updating and hierarchy maintenance is robust and effective when the flooding scale is small [30]. In addition, duplicated useless packets during the flooding are silently dropped which also mitigates the network traffic for location updating. The efficacy of HERO design can be examined more intensively by our prototype system implementation and extensive simulations.

4.4 Protocol Analysis and Parameter Optimization

By far, a key question remaining unestablished is the configuration of the radii r_i $(1 \le i \le h)$ in (1). To conveniently control the maximum number of regions in the hierarchy and to restrain the location updating in small regions close to the vehicle, HERO organizes the hierarchical regions with exponentially increasing sizes.

More specifically, we introduce two protocol parameters: first radius r and amplification factor k. The radius of the first region is r (i.e., $r_1 = r$), and the radius of $\mathcal{R}_i$ is $k^{i-1}r$ (if k is an integer). Fig. 2 shows an example with r and k both equal to 2. More generally, k can take any real number greater than one. Since the radius of a region must be an integer in hops, we take the ceiling of $k^{i-1}r$ as the radius of $\mathcal{R}_i$ and further make sure that a region is larger than its immediate inner region. Then, the radius of $\mathcal{R}_i$ is defined as

$$r_1 = r, \qquad r_i = \begin{cases} \lceil r \cdot k^{i-1} \rceil, & if\ r_{i-1} < \lceil r \cdot k^{i-1} \rceil, \\ r_{i-1} + 2, & otherwise. \end{cases} \tag{2}$$

We are interested in the maximum number of hops that a query is routed, and we have the following theorem.

Theorem 1. *Given a network with the network diameter (i.e., the maximum hop distance between any pair of nodes) D hops, it takes at most $\lceil \log_k(D/r) \rceil$ hops for a query to be answered.*

Proof. The worst case of a query, where it traverses the maximum number of hops, occurs when the hierarchy is constructed from one end of the network diameter and the query is injected at the other end of the diameter. In this case, according to the definition of the exponential hierarchy, the maximum number of regions contained in the network is $\lceil \log_k(D/r) \rceil$. Since nodes in $\mathcal{R}_1$ always have the latest location information, a query only needs to be routed to a boundary node of $\mathcal{R}_1$. Thereby, a query takes at most $\lceil \log_k(D/r) \rceil - 1$ hops to reach that boundary node. It takes the boundary node one more forwarding hop to finally return the result back to the node that initiates the query. This concludes the proof. □

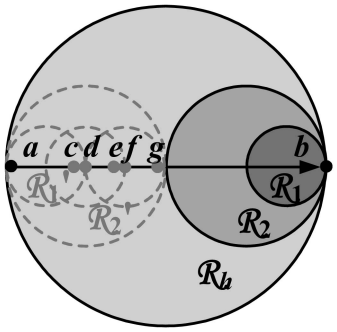

Fig. 5. Worst case of location updating, when the vehicle traverses the whole network from nodes a to b.

We study the location updating overhead caused by the movements of a vehicle. Since the patterns of the vehicles' movements could be very different, we analyze the updating overhead in the worst case where a vehicle moves straight. In this case, the movement continuously breaks the maximum number of regions, and therefore arouses the most significant updating overhead. We have the following theorem.

Theorem 2. *Suppose that the topology of a network is a disk, the maximum network traffic overhead of location updating for a vehicle moving a distance of D is $\eta(D) = c(kD^2 + 2r(r-k-1)D - 6r^2)$, where D is the network diameter and c is a constant coefficient.*

Proof. Fig. 5 depicts the worst case of location updating among all possible movements with a distance of D, where all constructed regions in the network need to be reconstructed during the movement from node a to node b. For analysis simplicity, we assume that k is an integer. With uniform deployment of local nodes, the network traffic for flooding in $\mathcal{R}_i$ (denoted as S_i) can be approximately evaluated by the area of $\mathcal{R}_i$. Let ζ_i denote the updating overhead incurred as a vehicle moves from the boundary of $\mathcal{R}_{i-1}$ to the node immediately next to the boundary of $\mathcal{R}_i$, $(i \ge 2)$. For example, in Fig. 6, the updating overhead introduced when the vehicle moves from node a to node b is denoted as ζ_1, and that from node c to node d is denoted as ζ_2. We have

$$\begin{cases} \zeta_1 = (r-1)S_1 = c_0\pi(r-1)r^2, \\ \zeta_i = (k-1) \cdot \left(S_i + \sum_{j=1}^{i-1} \zeta_j \right), \end{cases} \tag{3}$$

where c_0 is a constant coefficient. Let ω_m denote the updating overhead incurred as the vehicle traverses the diameter of $\mathcal{R}_m$ from node a, as shown in Fig. 5.

To formulate ω_m, we need to go through the whole process of the restricted location updating when the vehicle moves from one end of $\mathcal{R}_m$ to the other. Obviously, we can recursively express ω_m in terms of ω_{m-1} and ζ_{i-1}:

$$\omega_m = \omega_{m-1} + 2(k-1)(S_m + \zeta_{m-1}). \tag{4}$$

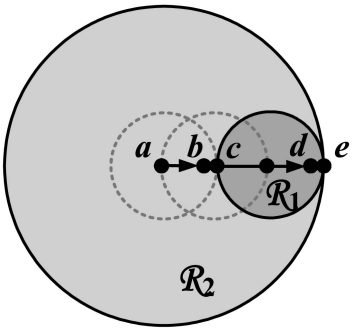

Fig. 6. Example of continuous reconstruction of $\mathcal{R}_1$ during the movement from nodes a to d.

For example, in Fig. 5, where k is 2, ω_2 (i.e., the updating overhead incurred when the vehicle moves from node a to node g) consists of ω_1 (from node a to node c), S_2 (from node c to node d), ζ_1 (from node d to node e), S_2 (from node e to node f), and ζ_1 (from node f to node g).

Thus, with (3) and (4), ω_m can be formulated as follows:

$$\begin{aligned}\omega_m &= (2r-1)\cdot S_1 + 2(k-1)\left(\sum_{i=2}^{m} S_i + \sum_{i=1}^{m-1}\sum_{j=1}^{i}\zeta_j\right) \\ &= c_0\pi\left(2k\cdot r_m^2 + 2r(r-k-1)r_m - 3r^2\right).\end{aligned} \tag{5}$$

Denote $\eta(D)$ as the total updating traffic caused while the vehicle traverses the network, and then $\eta(D) = \omega_h$. Let $c = c_0\pi/2$. This concludes the proof. □

We aim to meet the real-time constraint of a vehicle and meanwhile minimize network traffic overhead. The typical latency between a pair of local nodes connected using ADSL connections can be measured. Let t_d denote the maximum delay of a query between two adjacent nodes, and t_0 denote the application real-time constraint. We try to minimize the average updating overhead per hop, $\eta(D)/D$, under the constraint $\lceil \log_k(D/r) \rceil \le t_0/t_d$. The average is a function of r and k, and let $g(r,k)$ denote it. Then, we have

$$\begin{aligned} g(r,k) &= \eta(D)/D \\ &= c\left(k\cdot D + 2r(r-k-1) - 6r^2/D\right). \end{aligned} \tag{6}$$

To minimize the traffic overhead, let $\log_k(D/r) + 1 = t_0/t_d$, and $g(r,k)$ can be reduced to

$$g(r,k) = c\left[D^{\frac{t_d}{t_0-t_d}}\cdot r^{-\frac{t_d}{t_0-t_d}}(D-2r) + 2\left(1-\frac{3}{D}\right)r^2 - 2r\right]. \tag{7}$$

Further, let the differentiation of $g(r)$ equal to zero, $dg(r)/dr = 0$. Since it is difficult to derive the exact r and k that produce the smallest network traffic overhead, we develop numerical procedures to compute the approximately optimal value of r and k. Figs. 7 and 8 show the optimal values of r and k using numerical computation, respectively, where t_d is set to 48 ms in the example. It can be seen that the first radius of $\mathcal{R}_1$, where HERO tries to restrain the locating updating, increases very slowly with the network scale.

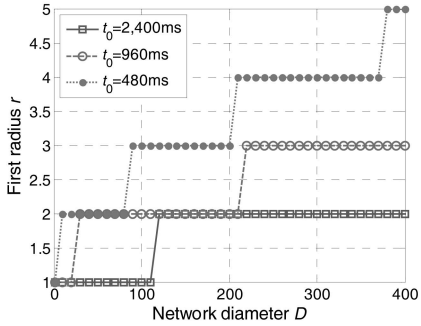

Fig. 7. First radius r as a function of D.

5 Design Issues

This section discusses some design issues that HERO may encounter in practice.

Scalability. HERO is designed to track hundreds of thousands of vehicles in a metropolitan-scale system with a large number of users. Therefore, the system scalability concern in terms of the number of vehicles, the number of users, and the number of local nodes is critical. With HERO, the system needs to maintain a hierarchy for each vehicle. If every movement of a vehicle will introduce a lot of location updating traffic into the system, the cost can be prohibitively expensive. However, this is where HERO comes to help. HERO leverages the inherent locality of vehicle movements and only updates a small number of nodes nearby the vehicle. Therefore, the location updating cost should be small. We can also notice that the query cost is modestly low, which is a logarithmic scale to the size of the network. Moreover, each node in the system only needs to maintain the information of several neighboring nodes. It is a lightweight protocol to join and leave the system. We will further investigate the scalability of HERO by extensive trace-driven simulations in Section 7.

Resilience to unreliable data. It is possible that occasionally a vehicle is not captured by an RFID reader (e.g., when the vehicle is moving too fast). In addition, a local node may also fail from time to time. It is critical to the operations of HERO if a boundary node misses a vehicle passing by. This inaccuracy can be easily detected in the system. At any time, a node in region $\mathcal{R}_i$ $(i \ge 2)$ should have received an update packet from a boundary node of $\mathcal{R}_{i-1}$ before the node itself captures the vehicle. Otherwise, it is aware that the vehicle has escaped from $\mathcal{R}_{i-1}$ and the

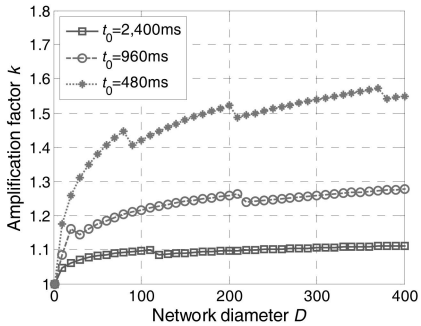

Fig. 8. Amplification factor k as a function of D.

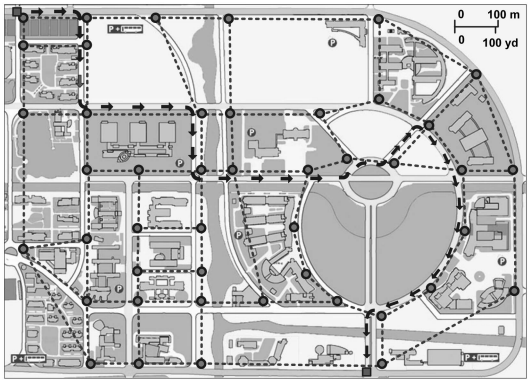

Fig. 9. The layout of the prototype implementation consisting of 45 nodes denoted by red spots.

Fig. 10. A local node with a RFID reader and a wireless AP; the highlight area in the inset shows an active RFID tag attached to a vehicle.

corresponding updating process fails. To solve the problem, we let the node which discovers this inaccuracy take the responsibility over as if it were a boundary node of $\mathcal{R}_{i-1}$ and triggers updating for the reorganization of regions from $\mathcal{R}_1$ to $\mathcal{R}_{i-1}$. Unless the node itself happens to be a boundary node of $\mathcal{R}_i$, it performs updating for the reorganization of regions from $\mathcal{R}_1$ to $\mathcal{R}_i$ instead.

Tracking accuracy. As a vehicle keeps moving, it may run out of the reading range of an RFID reader while still has not entered the territories of others. This causes the system to have inaccurate vision about the current position of the vehicle before the vehicle reenters into the system. It also defines the resolution of tracking accuracy of the system to be the uncovered distance between two adjacent RFID readers. In more practical environments, this inaccuracy can be enlarged when RFID readers fail to capture the vehicle as the vehicle passes. To refine the tracking resolution, more RFID readers can be deployed in the system. In order to reduce the cost, readers can be deployed more densely at those places where more accurate location information of individual vehicles is required and less densely at other places.

Node join and maintenance. In HERO, a single node failure can be discovered in a short time. A local node can periodically check with its neighbors while performing HERO protocol. An unavailable node is then reported to the system administrator. To join the system for tracking vehicles, a new node (or a recovered node) only needs to contact its adjacent nodes. Then, for each vehicle, the node configures its status the same as that of the neighbor which resides in the smallest region among all neighbors in the hierarchy. Thereby, it knows its position in the hierarchy for each vehicle and can perform location updating and query processing properly.

Data replication. The tracking data of vehicles can be of great importance for many applications. It is an important issue for the system to protect these data from node failures and disasters, such as fires or earthquakes. HERO actually has the implicit advantage of protecting important tracking data. Recall that tracking data are replicated in the first region. It implies that the system is still able to track the vehicle even when the chaser node becomes unavailable. If some vehicles are particularly important and need additional protection of tracking data, we can make r relatively large associated with the vehicle. By this means, more data can be replicated in the first region organized for the vehicle.

6 Prototype Implementation

To validate the HERO design and prove its practical implementation, we have built a prototype system in the campus to track experimental vehicles. This prototype system contains 45 local nodes distributed in our campus. As shown in Fig. 9, local nodes (denoted by red spots) are deployed at crossroads of main roads. Every local node has an IEEE 802.11g wireless network interface connecting the local node to the campus Internet. Furthermore, the overlay network formed by the local nodes is illustrated by the dashed lines in Fig. 9. An overlay connection is established between two nodes if there is a road that immediately connects them.

In the prototype system, we employ an active RFID system using "Tag Talk First" technology. Fig. 10 shows a typical local node, which is associated with a SP-D300 RFID reader [10] as well as an IEEE 802.11g wireless AP. The inset in Fig. 10 shows an active RFID tag (in highlighted area) attached to a vehicle. The reader's operating frequency is 2.4 GHz. It connects to the local node via an RS-485 interface and has a data transfer rate of 1 Mbps. The reader has a configurable operation range from 2 to 80 m. Each reader can simultaneously detect up to 200 tags in 800 ms. Each tag has a unique 64-bit ID. Its battery has a life of 6 to 8 years. Tags send their unique ID signal in random with an average of 300 ms and can be detected at a high speed up to 125 mi/hour. Besides the RFID system, wireless communication technology is also investigated in our prototype implementation. The HERO protocol runs on Red Hat Fedora 5 and uses POSIX.1 socket API to communicate with each other. UDP packets are adopted for location updating and query routing. The size of all packets is 40 bytes, which includes 20 bytes of the IP packet header, 8 bytes of the UDP packet header, and 12 bytes of data.

With this prototype implementation, we conduct a variety of experiments. Since we use the campus Internet as the underlying network, real-time guarantee seems to be nontrivial because the jitter (end-to-end round-trip time) can vary largely. To demonstrate this, we randomly choose two local

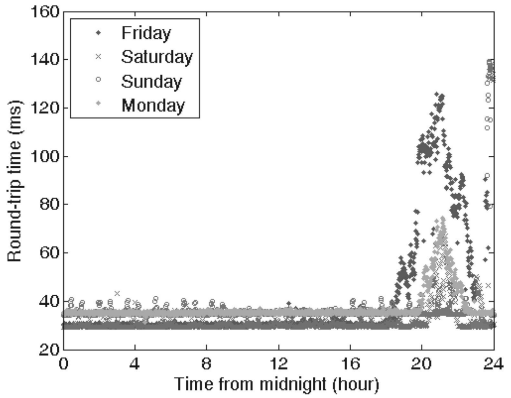

Fig. 11. The round-trip time pinged from two nodes randomly chosen from 45 nodes. The measurement is taken from 27 June (Friday) to 30 June in 2008 (Monday).

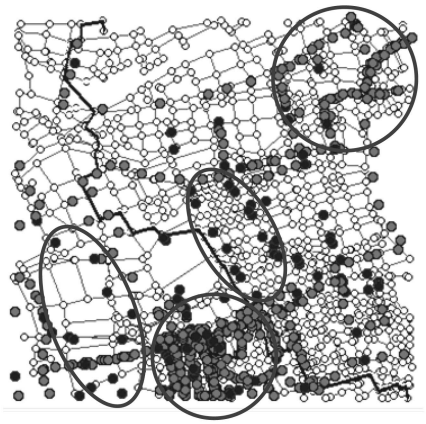

Fig. 12. The topology of the downtown area of Shanghai with 1,000 nodes deployed at crossroads of this area.

nodes to measure the round-trip time by ping. Fig. 11 shows the measured round-trip time from 27 June 27 to 30 June in 2008. It can be seen that the round-trip time increases sharply from 7 p.m. to 11 p.m. Moreover, the peak value can be almost four times larger than that at daytime. Nevertheless, the round-trip time is much more stable during the daytime. Thus, we choose to perform experiments with our prototype system from 10 a.m. to 12 a.m. on 30 June in 2008. We set the real-time constraint to 100 ms and take the maximum transmission delay between two online nodes which is 18.05 ms. Therefore, the resulting r and k are therefore configured to 3 and 1.278, respectively. We let a van carrying an active RFID tag travel at 30 mi/hour along the route as depicted by the dark arrows in Fig. 9. As the van enters an RFID reader's field and is captured by the reader, the associated node performs location updating accordingly. During the journey which lasts about 4 minutes, we let each node randomly generate a hundred of queries.

Among all the 4.500 queries, the maximum query latency is 90.45 ms, which is strictly shorter than the required real-time constraint. We also notice that the average query latency is about 47.93 ms. The network traffic for location updating among all nodes adds up to 13.2 Kbytes. In contrast, the network traffic for location updating using broadcast on a spanning tree is about 28.8 Kbytes. Since the maximum routing hops of a query is bounded (i.e., 5 hops in this experiment), the network traffic for query routing linearly increases with the number of queries in the system.

The lesson from our prototype implementation is that, with appropriate configuration of the protocol parameters, the query latency can be guaranteed to satisfy the real-time constraint requirement in terms of the number of hops that a message has to traverse. In addition, the overall network traffic overhead, introduced by location updating and query routing, can well accommodate a large number of queries. To further investigate the performance of HERO in a large-scale setting, we conduct trace-driven simulations, which are detailed in the following section.

7 Performance Evaluation

7.1 Methodology

In the simulations, the HERO protocol is implemented using ns2 [11]. Since we connect local nodes to the metropolitan-scale ATM network provided by Shanghai Telecom through cheap ADSL connections, the transmission delay between any two local nodes is reliable. Therefore, we can construct the overlay network topology by simply mapping the real complex road network of Shanghai where local nodes are deployed on every crossroad. The typical link transmission delay between two neighbor nodes in the overlay network is 48 ms, measured by ping between two desktop PCs with 1-Mbyte bandwidth ADSL connections. One of the overlay topologies employed in our simulations is depicted in Fig. 12. The topology containing 1,000 nodes (denoted by small hollow dots) covers the geographical downtown area of Shanghai. The dark line shows the network diameter in the topology, which is 55 hops. Upon the ATM network, we use UDP protocol for communicate with the packet size being 40 bytes.

To investigate the impact of the vehicle moving patterns to the HERO design, we use real GPS trace data of taxies which were obtained with GPS technology from August 2006 to October 2006. Taxies can move more randomly and extensively in the whole city and, therefore, have more sense to be considered. A typical trace of a taxi in the downtown area of Shanghai through daytime (on 13 August 2006) is shown by solid dots in Fig. 12. It can be seen that, when the taxi is vacant, it cruises around within an area most the time seeking for passengers, as shown by these solid dots in the circle areas in Fig. 12. This benefits our HERO design best because most of the location updating can be perfectly restricted within small regions. It can also be seen that, when the taxi has a delivery, it runs very fast along the straightest path for its destination, as illustrated by those solid dots in the ellipse areas in Fig. 12. HERO leverages restricted location updating strategy to reduce network traffic while still keeping the whole system up-to-date.

We compare HERO with several alternative schemes:

- **ST-Updating.** In this scheme, whenever a node captures a vehicle, it updates this information to all other nodes. To reduce the network traffic overhead of this update, the system maintains a global spanning tree. Therefore, only $N-1$ update packets are introduced across the whole network for each update when there are N nodes in the network. The

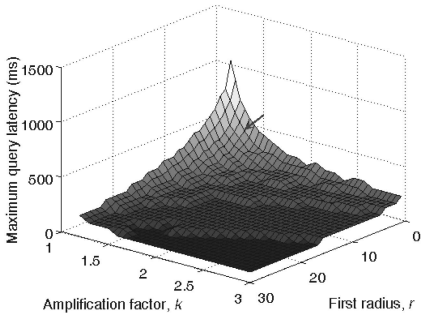

Fig. 13. Maximum query latency versus different protocol parameters.

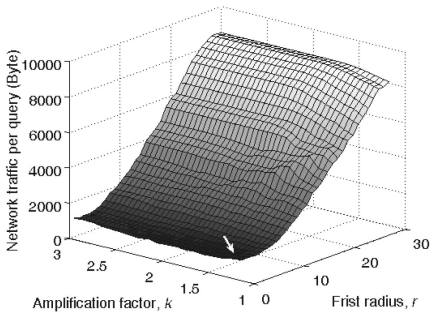

Fig. 14. Updating network traffic versus different protocol parameters.

strength of this scheme is that each node can answer any query locally, providing minimal query response time.

- **ST-Flooding**. This scheme does not perform vehicle information update in the network, and hence, no overhead is introduced for location updating. To search for a vehicle, a query is flooded throughout the network. A global spanning tree is used to broadcast the query to reduce the network traffic overhead.
- **Ex-Flooding**. This scheme does not perform vehicle information update either. Without relying on a global spanning tree, it employs expanding flooding. The query is flooded in the overlay network. At the beginning, the TTL of the query is small. If this try is not successful, the query will be flooded again with an increased TTL (plus 4 hops). This process is repeated until the vehicle is found.
- **Random walks**. Similar to ST-Flooding and Ex-Flooding, this scheme does not perform information update. To search for a vehicle, the query is carried out by five simultaneous random walkers. A walker checks with the querying node every 50 steps and terminates either if the querying node has already retrieved the result or if the maximum number of steps 2,000 is reached.
- **Chord**. In this scheme, each local node joins an overlay network of a logical ring [23]. With a series of indexing pointers maintained in each local node, each local node can update and retrieve the location information of a vehicle within $\log_2 M$ on average, where M is the size of the logical ring. In our implementation, M is set to 2^{32} which is moderate to support a large number of nodes and vehicles in the system.

We propose two important metrics to evaluate the performance of HERO and the above schemes:

1. ***Maximum query latency*** (M_{QL})**.** It refers to the maximum query response time of a successful query. The intention of this metric is to check whether a scheme can guarantee certain real-time requirements.
2. ***Network traffic per query*** (M_{NT})**.** It can be seen that if there were no query then no location updating would need to be carried out at all. Therefore, to answer a query, the system cost should involve two parts of network traffic, i.e., for location updating and for routing query packets. We investigate the communication cost per query by any location updating as well as query processing.

7.2 Effects of Protocol Parameters

We first examine the effects of protocol parameters on the system performance and validate the theoretical analysis. We employ 1-hour extensive trace data of 100 taxies, randomly generate 10^5 queries for different vehicles during this hour and demand any query to be answered within 500 ms. We vary r from 1 hop to 30 hops with an increment of 1 hop, and vary k from 1.2 to 3 with an increment of 0.05. For each pair of r and k, we repeat the experiment 10 times and present the average.

Figs. 13 and 14 plot M_{QL} among all the generated queries and M_{NT} under different configurations of r and k, respectively. It shows that M_{QL} drops dramatically with increasing r and k. It can be seen that M_{NT} increases with both increasing r and increasing k. This is reasonable because either a greater r or a greater k leads to a more aggressive updating strategy. At the extreme, if r equals to D or r equals to one and k equals to D, HERO floods every location updating throughout the whole network. In this experiment setting, according to the numerical computations in Section 3, r and k should take 2 and 1.393, respectively. The arrows in Figs. 12 and 13 show the corresponding positions. It is clear to see that, with this configuration of r and k, HERO can actually guarantee any query to satisfy the real-time requirement and meanwhile minimizing the network traffic overhead per query.

7.3 Impact of Query Quantity

In this experiment, we investigate the impact of the query quantity on the system performance. We take the same setting as the previous experiment. The protocol parameter r and k are set to 2 and 1.393, respectively. We vary the total number of queries from 10^3 to 10^5 with an increment of 400.

Among all queries, M_{QL} of HERO is 480 ms which is strictly shorter than the real-time constraint. In ST-Updating, M_{QL} is about 14 ms which is for local database operations. The other schemes cannot guarantee to satisfy the real-time requirement. M_{QL} of Chord, ST-Flooding, and Ex-Flooding is 1,536, 5,232, and 14,120 ms, respectively. M_{QL} of Random walk is about 10^5 ms due to the search step limitation of 2,000. Fig. 15 plots M_{NT} with different number of queries per vehicle. M_{NT} of HERO is much less than that of other schemes. In addition, it declines as the number of queries increases. It can be seen that, with this experiment setting, HERO has less query overhead than ST-Updating until the number of queries for the same vehicle exceeds 41,400. It is very interesting to find out that the number of queries decides whether ST-Updating or HERO is preferable. However, we argue that it is impractical that a single

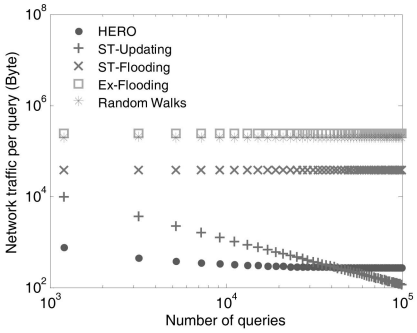

Fig. 15. Network traffic per query versus number of queries.

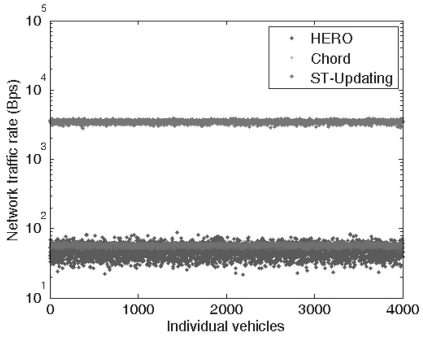

Fig. 17. Network traffic rate versus individual vehicles.

vehicle would be queried so tensely within 1 hour in a region with 1,000 nodes.

To further compare HERO with Chord, we conduct another experiment. We take the same setting as the previous experiment except we distribute all the queries in both uniform and nonuniform manners and collect the total incurred network traffic. To nonuniformly distribute queries, we divide the whole network into 10 areas and assign each area a different probability for a local node to generate a query. For each probability configuration, we repeat the experiment 10 times. Fig. 16 shows the total network traffic rate with different number of queries. It can be seen the difference between the results of uniform distribution of queries and nonuniform distribution of queries is very slight. We notice Chord has incurred much more network traffic than HERO. This is because Chord takes on average 16 hops to forward a query when the size of the logic ring is 2^{32}, while HERO guarantees to route a query within a desired number of hops (i.e., 9 hops in this experiment). HERO takes on average 5.28 hops to forward a query in this setting.

7.4 Impact of Vehicle Quantity

In this experiment, we investigate the impact of the vehicle quantity on the system performance. We use the same network topology and protocol parameter configuration. To gain enough trace data, we take trace data from different dates and treat a taxi at two different dates as two separate taxies. In this way, we gain 20,000 taxi traces of 1-hour extensive GPS data of 1,000 taxies from 12 August to 12 September in 2006.

Fig. 17 shows the network traffic rate with 4,000 different taxies. We can see that different taxies have introduced different network traffic for location updating. This is because different taxies have different moving patterns (for example, a vacant taxi compared to an occupied one), and therefore, the variance of location updating cost can be large. Chord uses consistent hash to uniformly distribute updating traffic among all local nodes, which is less influenced by different moving patterns. Beyond all the facts, we notice HERO has less average updating traffic than Chord. This is because HERO fully leverages the inherent locality of vehicle movements and tries to constrain updating traffic only within nearby nodes. Moreover, we vary the total number of taxies from 500 to 3,000 with an increment of 500. Fig. 18 plots the network traffic rate with different number of vehicles in the system which further confirms our analysis. The average network traffic for location updating of HERO is 47.4 bps, whereas that of Chord is 57.38 bps. It can be seen that HERO has less updating traffic than Chord and has good scalability with the increasing number of vehicles.

7.5 Impact of Network Scale

To evaluate the impact of network scale, we conduct an experiment on multiple topologies. We adopt 1-hour trace data of 100 vehicles, randomly generate 10^5 queries and set the real-time constraint to 500 ms. For each topology, we set r and k according to particular numerical computation results.

In Fig. 19, we plot M_{QL} over different number of nodes in the system. HERO meets the real-time constraint under different network scales. M_{QL} of ST-Flooding increases from 5,232 ms in the 1,000-node topology up to 11,088 ms in the 3,000-node topology. M_{QL} of Ex-Flooding also increases from 14,120 to 31,056 ms. Random walks can always reach the maximum number of search steps. Fig. 20 plots M_{NT}

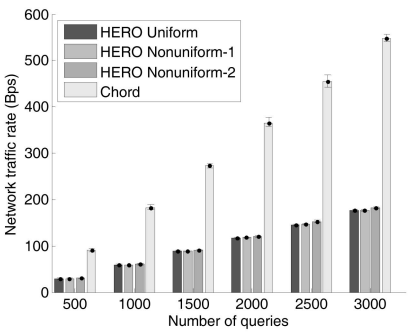

Fig. 16. Network traffic rate versus number of queries.

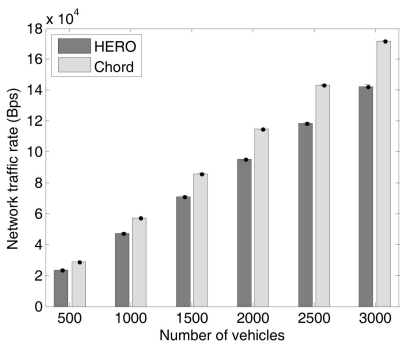

Fig. 18. Network traffic rate versus number of vehicles.

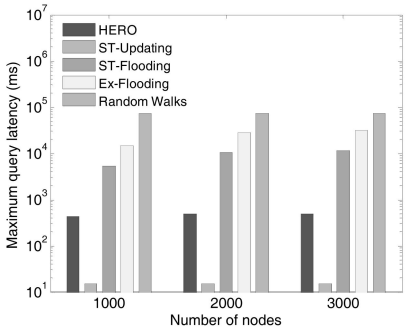

Fig. 19. Maximum query latency versus number of local nodes.

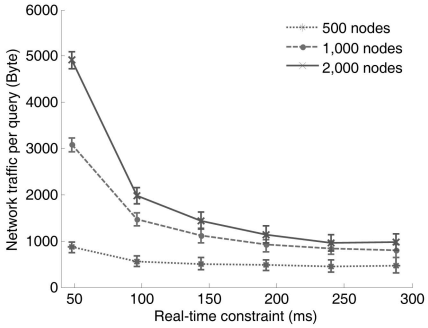

Fig. 21. Network traffic per query versus real-time constraint.

over different number of nodes. As the network scale increases, the network traffic of HERO increases very slowly. The reason is that HERO can constrain the updating traffic within a small region and has little influence on other nodes which are far away from the vehicle in the network.

7.6 Effect of Real-Time Constraint

We conduct an experiment to study the relationship between the network traffic overhead per query and the real-time constraint in HERO. We use the same trace data and randomly generated queries as the experiment described in Section 7.2. We vary the real-time constraint from 50 to 500 ms. Fig. 21 plots M_{NT} over different real-time constraints. M_{NT} first drops rapidly in the beginning and tends to increase slowly with the real-time constraint. This is because the network traffic for routing queries takes more account into the overall network traffic as the real-time constraint increases. This result is valuable for applications to select appropriate real-time constraints to satisfy their requirements while reducing the system overhead.

8 Conclusion and Future Work

In this paper, we have presented the real-time tracking protocol HERO for the metropolitan-scale ITS. Exploiting the locality of vehicle movements in the urban area, HERO adaptively updates the locations of a vehicle according to the innovative hierarchical structure. HERO significantly reduces network traffic while still satisfying the real-time requirement. As a fully distributed protocol, this protocol is highly scalable to the number of users, the number of vehicles, and the system scale as well. Prototype implementation and comprehensive simulations based on the real road network and trace data of vehicle movements demonstrate the efficacy of HERO.

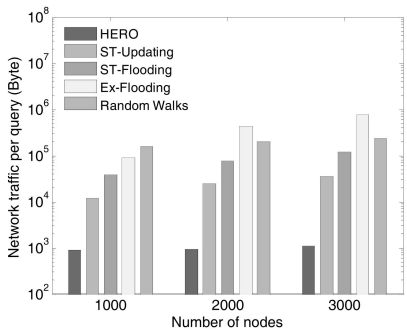

Fig. 20. Network traffic per query versus number of nodes.

This is an on-going research and system effort in tracking various vehicles in the metropolitan-scale system. Following the current work, we have a lot of more exciting yet challenging topics ahead. One of these topics is the privacy implication of tracking personal vehicles all the time. The government will guarantee to protect individual privacy by authorizing legal individuals and corporations with different privileges to access appropriate vehicles. Next, we will delve into designing better location updating schemes such that update overhead can be reduced as much as possible. Based on our realistic prototype test-bed, we will validate our design and study its performance under real complex environments. Improvements will be made based on the realistic studies before it comes to be deployed in the large-scale SG system. Moreover, it is important to ensure the security of chasers. If a malicious chaser does exist, the system may behave abnormally and the system performance would be degraded. However, this paper focuses on the system design for real-time tracking. We will gradually incorporate security measures into the system implementation in the future.

References

[1] European Commission, *The Karen European ITS Framework Architecture,* http://www.frame-online.net/, 2004.
[2] *The Nat'l ITS Architecture Version 5.1.* Dept. Transportation of the US, http://itsarch.iteris.com/itsarch/index.htm, 2005.
[3] Ministry of Internal Affairs and Communication National Police Agency, and Ministry of Land, Infrastructure, and Transport of Japan, *Vehicle Information and Comm. System,* http://www.vics.or.jp/english/index.html, 2006.
[4] Shanghai City Comprehensive Transportation Planning Inst., http://www.scctpi.gov.cn/chn/chn.asp, 2007.
[5] LoJack Corporation, *Stolen Vehicle Recovery System,* http://www.lojack.com/what/stolen-vehicle-recovery-system.cfm, 2007.
[6] iPico Corporation, *Test Report: Single-Lane Vehicle Identification with UHF RFID,* http://www.ipico.com/site/iPico_100/pdf/WP_App_HighSpeedVehicleID.pdf, 2007.
[7] "Transportation Recall Enhancement, Accountability, and Documentation (TREAD) Act," *The 106th United States Congress,* http://www.citizen.org/documents/TREAD%20Act.pdf, 2000.
[8] Cisco Systems Inc., *Cisco Aironet 1240 Series 802.11a/B/G Access Point Data Sheet,* http://www.cisco.com/application/pdf/en/us/guest/products/ps6521/c1650/cdccont_0900aecd8031c844.pdf, 2007.
[9] Shanghai Telecom, http://www.shanghaitelecom.com.cn/, 2007.
[10] Shanghai Super Electronic Technology, http://www.superrfid.net/english/, 2007.
[11] *The Network Simulator,* http://www.isi.edu/nsnam/ns/, 2007.

[12] *The Gnutella Protocol Specification V0.6,* http://rfc-gnutella.sourceforge.net, 2005.
[13] H. Zhu, Y. Zhu, M. Li, and L.M. Li, "ANTS: Efficient Vehicle Locating Based on Ant Search in Shanghai Transportation Grid," *Proc. Int'l Conf. Parallel Processing (ICCP),* 2007.
[14] H. Zhu, Y. Zhu, M. Li, and L.M. Li, "HERO: Online Real-time Vehicle Tracking in Shanghai," *Proc. IEEE INFOCOM,* 2008.
[15] A. Bakker, E. Amade, G. Ballintijn, I. Kuz, P. Verkaik, I. van der Wijk, M. van Steen, and A.S. Tanenbaum, "The Globe Distribution Network," *Proc. USENIX Ann. Conf.,* 2000.
[16] A. Civilis, C.S. Jensen, and S. Pakalnis, "Techniques for Efficient Road-Network-Based Tracking of Moving Objects," *IEEE Trans. Knowledge and Data Eng.,* vol. 17, pp. 698-712, 2005.
[17] D. Pfoser, C.S. Jensen, and Y. Theodoridis, "Novel Approaches to the Indexing of Moving Object Trajectories," *Proc. 26th Int'l Conf. Very Large Data Bases (VLDB),* 2000.
[18] G. Kollios, D. Gunopulos, V. Tsotras, A. Delis, and M. Hadjieleftheriou, "Indexing Animated Objects Using Spatio-temporal Access Methods," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, pp. 758-777, 2001.
[19] D. Lin, C.S. Jensen, B.C. Ooi, and S. Saltenis, "Efficient Indexing of the Historical, Present, and Future Positions of Moving Objects," *Proc. Sixth Int'l Conf. Mobile Data Management (MDM),* 2005.
[20] M. Pelanis, S. Saltenis, and C.S. Jensen, "Indexing the Past, Present, and Anticipated Future Positions of Moving Objects," *ACM Trans. Database Systems,* vol. 31, pp. 255-298, 2006.
[21] J.F. Roddick, M.J. Egenhofer, E. Hoel, and D. Papadias, "Spatial, Temporal and Spatio-Temporal Databases—Hot Issues and Directions for PhD Research," *Proc. ACM SIGMOD,* 2004.
[22] B.Y. Zhao, J. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," Technical Report UCB/CSD-01-1141, Univ. of California at Berkeley, 2001.
[23] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *Proc. ACM SIGCOMM,* 2001.
[24] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proc. 18th IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware),* 2001.
[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. ACM SIGCOMM,* 2001.
[26] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and Replication in Unstructured Peer-to-Peer Networks," *Proc. 16th ACM Int'l Conf. on Supercomputing (ICS),* 2002.
[27] C. Gkantsidis, M. Mihail, and A. Saberi, "Random Walks in Peer-to-Peer Networks," *Proc. IEEE INFOCOM,* 2004.
[28] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip Algorithms: Design, Analysis, and Applications," *Proc. IEEE INFOCOM,* 2005.
[29] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregation Information," *Proc. 44th Ann. IEEE Symp. Foundations of Computer Science (FOCS '03),* pp. 482-491, 2003.
[30] S. Jiang, L. Guo, and X. Zhang, "LightFlood: An Efficient Flooding Scheme for File Search in Unstructured Peer-to-Peer Systems," *Proc. Int'l Conf. Parallel Processing (ICCP),* 2003.

Hongzi Zhu received the MS degree in computer science from Ji Lin University in 2004. He is a PhD candidate in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai. His research interests include peer-to-peer computing, vehicular ad hoc networks, pervasive computing, distributed systems, and network security. He is a student member of the IEEE, the IEEE Computer Society, and the IEEE Communication Society.

Minglu Li received the PhD degree in computer software from Shanghai Jiao Tong University, Shanghai, in 1996. He is a full professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He is currently the vice dean in the Department of Computer Science and Technology and the director of the Grid Computing Center. His current research interests include Web services, grid computing, and multimedia computing. He is a senior member of the IEEE and the IEEE Computer Society.

Yanmin Zhu received the PhD degree in computer science from Hong Kong University of Science and Technology in 2007 and the BEng degree in computer science from Xi'an Jiao Tong University in 2002. He is a research associate in the Distributed Software Engineering Center, Department of Computing, Imperial College London, London. His research interests include ad-hoc sensor networks, mobile computing, grid computing, and resource management in distributed systems. He is a member of the IEEE and the IEEE Communication Society.

Lionel M. Ni received the PhD degree in electrical and computer engineering from Purdue University, West Lafayette, Indiana, in 1980. He is a chair professor and the head of the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. His research interests include parallel architectures, distributed systems, wireless sensor networks, high-speed networks, and pervasive computing. He has chaired many professional conferences and has received a number of awards for authoring outstanding papers. He is a fellow of the IEEE and the IEEE Computer Society.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.