ANTS: Efficient Vehicle Locating Based on Ant Search in ShanghaiGrid

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

***Abstract*—Intelligent transportation systems (ITSs) have become increasingly important for public transportation in Shanghai, China. In response, ShanghaiGrid (SG) aims to provide abundant intelligent transportation services to improve traffic conditions. A fundamental service in SG is to locate the nearest desirable vehicles for users. In this paper, we propose an innovative protocol called ANTS to locate a desirable vehicle close to the querying user. The protocol finely mimics the efficient searching strategy adopted by a lost ant searching for its nest. Taking query locality into account, ANTS can retrieve the closest vehicles satisfying the query with high probability but incurs small query latency and modest network traffic. ANTS is a fully distributed and robust protocol and, therefore, has good scalability. Extensive simulations based on the real road network and the trace data of vehicle movements in Shanghai demonstrate the efficacy of ANTS.**

***Index Terms*—Distributed applications, peer-to-peer (P2P) network, radio-frequency identification (RFID) system, vehicle locating.**

I. Introduction

INTELLIGENT transportation systems (ITSs) have rapidly been evolving in the past two decades, leveraging advanced computing and communication technologies. Shanghai, which is the largest metropolis in China, covers an area of 5800 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, the Shanghai government has established the ShanghaiGrid (SG) project [1] since 2005, with the ambitious goal of building a metropolitan-scale traffic information system. This project will construct the basic infrastructure, which is composed of a great number of traffic information collectors and information-processing nodes connected through the Internet to build diverse applications to facilitate public transportation. According to the blue paper of the project, wireless access points (APs) and radio-frequency identification (RFID) readers will be deployed throughout Shanghai. Exploiting the pervasive deployment of these devices, the location and status information of vehicles can actively be captured and logged into a large number of distributed local nodes.

It is a fundamental service in SG to locate a desirable vehicle that is preferably close to a user's position. A wide spectrum of applications can be implemented on top of this service. For example, a user who wants to find a vacant taxi has to passively wait on the side of a road. Ideally, the user can actively locate a nearby vacant taxi and inform it in a timely manner. As another example, a user wants to report an accident to a police car or an ambulance. It is obvious that it is more preferable that such a responding car is close to the accident location. Ultimately, the user may be in an urgent situation and, hence, desires to access to the car as soon as possible. On the other hand, if the vehicle is far away from the user, it is very possible that the vehicle will change its status before reaching the user and, thus, fails to meet the user's requirement.

To realize this service, a centralized scheme is straightforward, where the location information of all vehicles is maintained and retrieved in a centralized database. However, it is infeasible for the large-scale system. 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 [2]. This produces in total about 12 000 events/s. Such a large volume of location-updating data can easily overwhelm the centralized server. Therefore, distributed schemes are inevitable for SG. Given that the vehicle information is stored in local nodes, a search scheme using flooding is intuitive and can always find the nearest vehicle. Nevertheless, the flooding search incurs a large amount of network traffic and, hence, has poor scalability. Another search scheme based on random walks introduces little network traffic, but it does not guarantee that a returned vehicle is close to the user. This is because each step of random walks randomly chooses a neighbor to forward a query and, therefore, has no consideration about the query locality. As a result, there is no existing successful solution, to the best of our knowledge, to locating the nearest vehicle in a large-scale network for transportation services.

In this paper, we propose a novel decentralized scheme called ANTS to solve this problem. Interestingly, we find that such a search for the nearest vehicle is very similar to a lost ant searching for its nest. Inspired by the efficient search strategy of lost ants, a query would generate an agent (like the ant) that searches the overlay network of local nodes by performing a number of loop searches of ever-increasing size, starting and ending at the

Manuscript received October 30, 2008; revised December 4, 2008, February 6, 2009, and March 4, 2009. First published May 19, 2009; current version published October 2, 2009. This work was supported in part by the National Basic Research Program (973 Program) of China under Grant 2006CB303000, by the National Science Foundation of China under Grant 60473092, Grant 90612018, and Grant 60533110, by the Science and Technology Commission of Shanghai Municipality under Grant 05DZ15005, and by the Hong Kong RGC under Grant HKUST617908 and Grant HKBU 1/05C. The review of this paper was coordinated by Prof. Y.-B. Lin.

M. Li, H. Zhu, and Y. Zhu are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200030, China (e-mail: mlli@sjtu.edu.cn; hongzi@sjtu.edu.cn; yzhu@cs.sjtu.edu.cn).

L. M. Ni is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Kowloon, Hong Kong (e-mail: ni@cse.ust.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2009.2023324

origin and pointing at different azimuthal directions each time. The distinctive features of ANTS are twofold. First, without relying on any costly updating strategy for vehicle information, it is lightweight and introduces minimal communication overhead. Second, taking query locality into account, it can retrieve the nearest vehicles satisfying the query with high probability but incurs small query latency and modest network traffic. As a fully distributed and robust protocol, ANTS is scalable to support hundreds of thousands of users and vehicles, as well as the continuous expansion of the city of Shanghai.

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

1) We propose ANTS, which is a novel protocol for locating the nearby vehicle in the large-scale SG. ANTS locates the nearest vehicles with high probability with minimal cost to network traffic and query latency.
2) We model the distribution of the expected distance of the target vehicle and draw the optimal configuration for the key parameters of the ANTS protocol.
3) We have built a small-scale prototype system to track the experimental vehicles in the campus, which covers an area of 322 acres. This experiment system matches our design of ANTS and demonstrates its practical implementation.
4) We evaluate the performance through trace-driven simulations. We base our simulations on the real-road network and trace the data of vehicle movements in Shanghai.

The rest of this paper is structured as follows. In Section II, we introduce SG and present the system model for ANTS design. Section III compares ANTS with related work. Section IV describes the design of ANTS and its theoretical analysis. Several design issues that may be encountered in practice are discussed in Section V. Section VI describes our prototype implementation of the nearest-vehicle locating system realizing the ANTS protocol. In Section VII, we introduce the trace-driven methodology that we employ to evaluate the performance of ANTS and present the simulation results. Finally, we draw conclusions and outline the directions for future work in Section VIII.

II. Related Work

In the literature, there exist many solutions to locating moving objects under different application scenarios.

The Globe system [10] constructs 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 the database community, indexing techniques have been proposed for tracking moving objects, but they are based on the assumption of the existence of a centralized database [11], [12]. Despite the large number of existing methods, there is no applicable method for update-intensive applications, where it is infeasible to continuously update the index and process queries at the same time. ANTS needs no such dedicated directory servers or centralized database and achieves good scalability and flexibility.

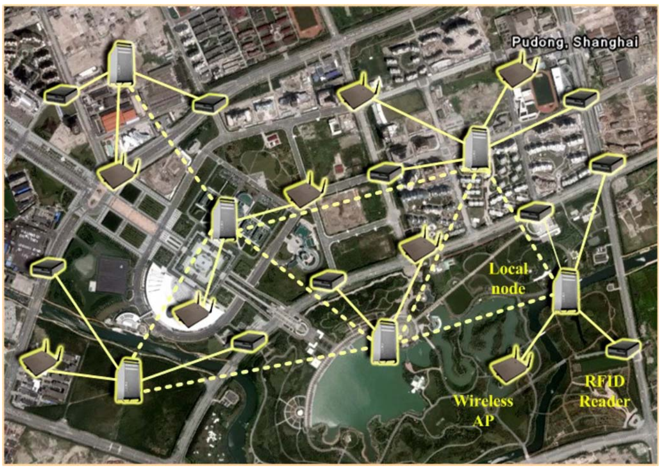


Fig. 1. Infrastructure of SG. A small part of the Pudong District of Shanghai is shown.

In structured peer-to-peer (P2P) networks, various distributed hash table (DHT) schemes were proposed to map objects to peers in a decentralized way and enabled the system to very efficiently satisfy queries [13], [14]. However, DHTs uniformly cast objects using consistent hash functions into the whole network, which cannot handle query locality. In unstructured P2P networks, the most typical query methods are based on flooding [15]. Using flooding is not scalable as described in Section I. Several randomized approaches, such as random walks [9], [16] and randomized gossip-based methods [17], were introduced to distribute and locate objects. These methods are resilient to node failures but have no guarantee of query locality. ANTS introduces minimal updating cost and small query routing overhead to locate the nearest vehicles with high probability.

III. System Model

As the RFID technology continuously evolves, it has widely been used in tracking various mobile objects. The SG project exploits the promising RFID and local-area wireless communication technologies. The infrastructure of the SG, which is still underway, is illustrated in Fig. 1. RFID readers and wireless APs, which are typically installed at crossroads, will be deployed throughout the urban area of Shanghai. 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 that connects to the Internet through a dedicated connection or a cheap asymmetric digital subscriber line connection provided by Shanghai TeleCom [3].

In SG, the vehicles' information is gathered both actively and passively. In the initial prototype, a vehicle is passively captured by using an active RFID. An active RFID tag is able to emit its ID at a fixed interval and has an effective communication range of about 30–80 m. The battery can sustain the operation of an active RFID tag for about two years [4]. A moving vehicle attached with an active RFID tag can be captured if the emitted signal reaches some reader. In addition to 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 Mb/s [5]. The vehicle can actively push important vehicle status information to local nodes. For example, a taxi can tell local nodes about its current availability.

As another pilot effort in Shanghai, certain vehicles (around 6850 taxies and 3620 buses) are equipped with 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., general packet radio service (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 accurately work without any other assistant devices. It is often the case that the reported GPS position of a vehicle can be deviated by more than 100 m from its actual location. To make things worse, a large number of major roads are covered by viaducts, which prevent satellites from seeing the vehicles running under them. Second, the intervals of location information reports can notably be 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 min. Third, the expense of a GPS receiver, as well as the 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 are very valuable for the study of traffic conditions. We evaluate ANTS using real trace data.

IV. Design of ANTS

In this section, we first introduce the search behavior of a lost ant, which has inspired the design of ANTS. Then, we describe the design details of the ANTS algorithm. Finally, we give the optimal configuration of the ANTS algorithm.

A. *Introduction to Ant Search*

The process of a lost ant searching for its nest was studied in [6]. After finding itself lost, an ant employs a very effective strategy to search for its nest. The core idea is that the ant believes that its nest is near its current position. With this belief, the ant uses a search pattern consisting of a number of loops of ever-increasing size. Each loop starts and ends at the *origin*, i.e., the spot where the ant starts the search, and points at different azimuthal directions. On each successive excursion, the ant reaches further away from the *origin*. Nevertheless, even after a number of excursions, the searching trajectory is precisely centered at the *origin*. During the search, the tangential component of the ant's motion randomly varies. This component also tends to be high when the ant is far from the *origin*, and the ant tends to radially run at the beginning and end of each excursion (i.e., near the *origin*). We precisely simulate the ant search pattern as described in [6]. Fig. 2 depicts a trajectory of the simulated search pattern. Fig. 3 depicts the spatial distribution of the searching density of ten simulated ants, where each dot presents the position at the end of each discrete search step. It can obviously be seen that a region closer to the *origin* is more densely searched.

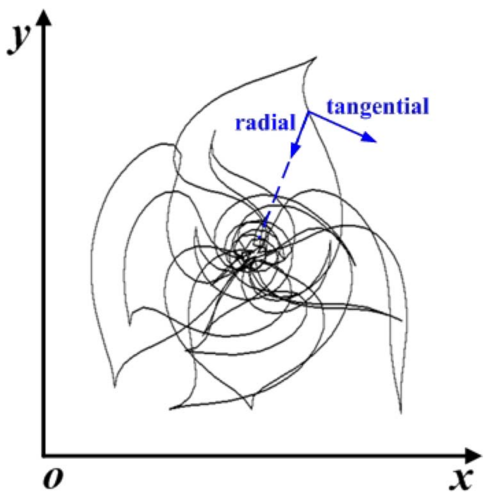


Fig. 2. Trajectory of the simulated search pattern with 500 discrete steps.

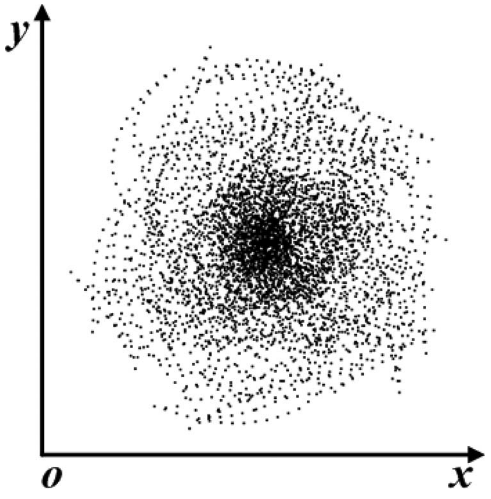


Fig. 3. Spatial distribution of the searching density of ten ants with 100 steps.

After carefully studying the rationality behind the ant search, we are excited to find that it is very similar to the situation that a user tries to locate the nearest vehicle that meets the user's requirement in the city. When the user injects a query into the overlay network to search for a certain vehicle, the query can be forwarded to different nodes in the overlay network. If the search takes a short period of time, then the motions of vehicles can be negligible considering the wide RFID coverage of a local node. We can compare the search process of the query on the overlay network with a lost ant running for its lost nest. The query also expects that the target vehicle is near its current position. In addition, a vehicle closer to the user is more preferable. This highly suggests the exploitation of the ant search strategy in searching for the nearest vehicle. This has inspired the design of ANTS.

B. *ANTS Search Algorithm*

Before discussing the strategy details, we first study the distribution of the expected position of the target vehicle

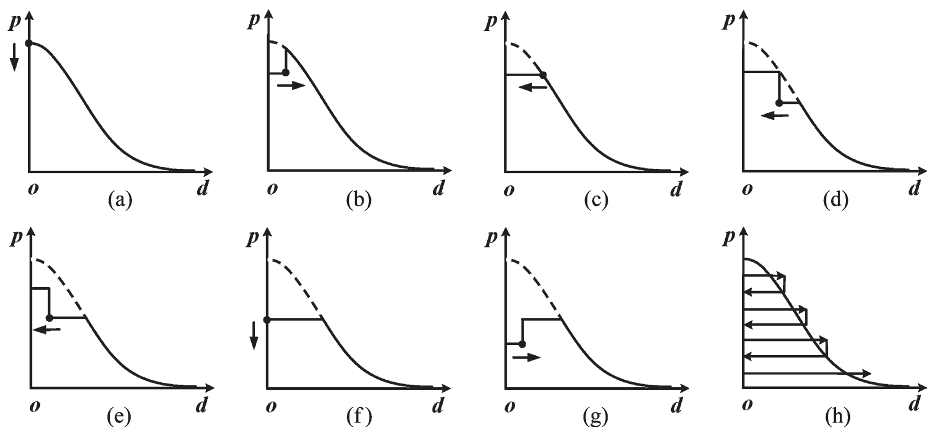


Fig. 4. (a)–(g) Depression of probability density at the visited regions. The search is always conducted at the most promising region. (h) Whole search pattern.

relative to the querying user. This distribution property is of great importance to the design of the search strategy. As discussed, it is desirable for the user that the retrieved vehicle is closest to the user. We describe the relative position of the desirable vehicle with a 2-D probability distribution. We find that this distribution is 2-D Gaussian. To prove the following lemma, we can follow the Herschel–Maxwell derivation of the normal distribution [19].

Lemma 1: If the probabilities of the expected position in orthogonal directions are independent, and the probability is independent of the orientation of the coordinate system, then the distribution of the expected position is a circular symmetric Gaussian.

For a homing ant who is returning to its nest, because of the accumulation of navigational errors, it often gets lost. The error distribution for a homing run of a given length can be assumed as Gaussian according to the theory of errors and the central limit theorem [6]. Based on the Gaussian distribution, the basic search strategy is illustrated in Fig. 4 [6], where the ordinate p denotes the distribution probability, and the abscissa d denotes the distance from the origin spot, which is denoted as o, where the ant begins to search. The search starts at the peak of the Gaussian probability density function (pdf), i.e., at the *origin*, as shown in Fig. 4(a). If the nest is not found at the *origin*, then the most promising region is the annular region surrounding the *origin*. In this situation, the new pdf for the nest has a depression at this region. Therefore, the search should move to the annular region, as shown in Fig. 4(b) and then Fig. 4(c). Then, the most promising region is now inside the annular region, and therefore, the search returns back toward the *origin*, as shown in Fig. 4(d), and later as shown in Fig. 4(e) and (f). Without success, the search process should move further outwards, as shown in Fig. 4(g). In conclusion, the search is always performed in the most promising region, and the whole search pattern is illustrated in Fig. 4(h).

This search pattern has the feature that the distance from the *origin* increases and cyclically decreased, with each outward excursion having a larger distance than the previous, as specified by the profile of the Gaussian pdf. In addition to the distance from the *origin*, to cover all azimuthal angles around the *origin*, each excursion starting from the *origin* has a randomly selected initial azimuthal direction. The trajectory of the excursion also has a tangential component in addition to the radial component, as discussed before.

In ANTS, the local nodes are first organized into an overlay network mapping the real underlying road network in Shanghai (as depicted in Fig. 1, dashed lines present the overlay connections of local nodes). This can easily be realized by checking the fine-grained electronic map of the city. This way, the locality of the vehicles' movements is preserved and mapped onto the overlay network. We emphasize that ANTS does not perform location updating of vehicles, i.e., at each node, all captured information is locally maintained. Fig. 5 shows the pseudocode of the complete ANTS search algorithm executed on each local node. The detailed algorithms are described as follows.

When trying to find the nearest car, a user contacts a local node that is close to the user and issues a query through the local node. We specially call this local node *origin* to indicate the querying location. Upon receiving a query from a user, the *origin* first checks itself whether it contains a desirable vehicle. If not, then it creates an *ant agent* and initializes the first outward-excursion distance of the query packet called an *outward-excursion query packet*. It then randomly selects a neighbor and sends the packet to it.

Upon receiving an *outward-excursion query packet*, a node checks whether it has a desirable vehicle. If not, then the node computes its distance from the *origin*. If the distance is less than the excursion distance of the query packet, then the node randomly selects one of its neighbors whose location is further to the *origin* than itself and forwards the query to that neighbor. Otherwise, the node changes the query packet to an *inward-excursion query packet*. It then randomly selects a neighbor whose location is closer to the *origin* than itself and forwards the packet.

If the *origin* receives an *inward-excursion query packet*, then it continues to conduct the next round excursion on the overlay network by changing the excursion direction and increasing the excursion distance of this packet. Over the overlay network, we refer to a search step of an *ant agent* as forwarding a query packet from one node to the next despite the direction of excursions. Finally, either there are targets found on some node or the *ant agent* dies after reaching the maximum number of search steps.

Algorithm ANTS search

```
Input: Local node nd receives a query packet pkt(last_hop, origin, direction, excursion_distance, TTL)
1.  if nd has the result then
2.     report the result to the origin node;
3.  /* ant agents initially have certain MAXINUM search steps */
4.  else if pkt.TTL ≥ 0
5.     compute the distance d between node nd and the origin node;
6.     /* an outward excursion */
7.     if pkt.direction == outward then
8.        if d ≤ pkt.excursion_distance then
9.           randomly select a neighbor nodes m which has farther distance from the origin;
10.          create a new query packet pkt'(nd, origin, outward, excursion_distance, TTL−1);
11.          send pkt' to node m;
12.       else   /* need to change the excursion direction */
13.          randomly select a neighbor nodes m which has nearer distance from the origin;
14.          create a new query packet pkt'(nd, origin, inward, 0, TTL−1);
15.          send pkt' to node m;
16.    /* an inward excursion */
17.    if pkt.direction == inward then
18.       /* node nd is the origin */
19.       if nd == pkt.origin then   /* need to start next round outward excursion */
20.          randomly select a neighbor nodes m;
21.          create a new query packet pkt'(nd, origin, outward, excursion_distance+Δ, TTL−1);
22.          send pkt' to node m;
23.       else
24.          randomly select a neighbor nodes m which has nearer distance from the origin;
25.          create a new query packet pkt'(nd, origin, inward, 0, TTL−1);
26.          send pkt' to node m;
```

Fig. 5. ANTS search algorithm.

For design simplicity, the *origin* increases the excursion distance of a query packet each time by a constant value instead of variable increments specified by the profile of a certain Gaussian pdf. This simplification does not seriously change the basic characteristics of the search strategy.

C. Protocol Parameter Optimization

We now discuss the optimal configuration of the protocol parameters during the oscillating-fashioned search. We expect that the retrieved vehicle has the nearest distance from the *origin* under a given distribution density of vehicles in the network.

For analysis simplicity, we assume that each excursion only moves in radial directions. For each excursion, the search moves out along a straight line and turns back along another line when it reaches its excursion distance. The real random search trajectories of an excursion can approximately be seen as constant times longer than straight lines. The abstract search model is plotted in Fig. 6, where the *origin*, which is denoted by red dots, is logically spread on the start and end of each excursion.

We further assume that the vehicle distribution is uniform in a large-scale city. Let p denote the probability that a node has a desirable vehicle, x denote the increment of excursion distance, and L denote the maximum steps of the abstract search. Thus, the first outward and inward excursions are x long in distance, and the nth outward and inward excursions are nx. An *ant agent* steps by unit one while moving in both outward and inward excursions. A search starts the nth excursion only when all the previous steps in the trajectory have missed the chance

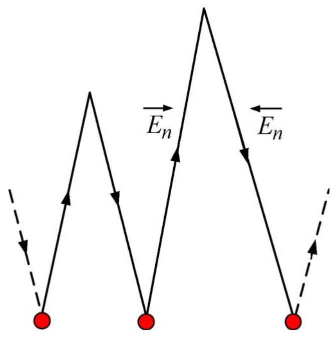


Fig. 6. Abstract search model. $\overrightarrow{E_n}$ and $\overleftarrow{E_n}$ denote the nth outward and inward excursions, respectively.

to find a desirable vehicle. Denote P_{n-1} as the probability of missing a desirable vehicle in all previous $n-1$ outward and inward excursions, which can be expressed as follows:

$$\begin{aligned} P_{n-1} &= (1-p)^{2x} \cdot (1-p)^{4x} \cdot \ldots \cdot (1-p)^{2(n-1)x} \\ &= (1-p)^{n(n-1)x}. \end{aligned} \tag{1}$$

Therefore, the probability of finding the vehicle at the ith step in the nth outward excursion is $P_{n-1} \cdot (1-p)^{i-1} \cdot p$ and the distance. Thus, the mean distance of a desirable vehicle retrieved in the nth outward excursion from the *origin* is

$$\begin{aligned} \overline{d}(\overrightarrow{E_n}) &= P_{n-1} \cdot \sum_{i=1}^{nx} (1-p)^{i-1} \cdot p \cdot i \\ &= (1-p)^{n(n-1)x} \cdot p \cdot \sum_{i=1}^{nx} (1-p)^{i-1} \cdot i. \end{aligned} \tag{2}$$

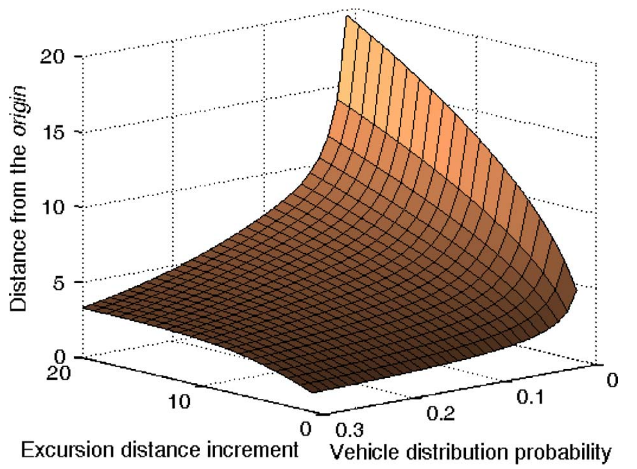


Fig. 7. Mean distance from the *origin* of all steps in a search trajectory with a maximum of 500 steps.

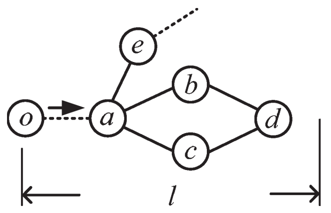


Fig. 8. Search trap example in the outward excursion. Node o denotes the *origin*.

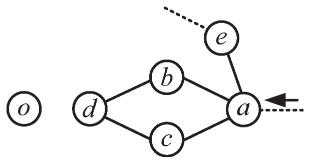


Fig. 9. Search trap example in the inward excursion. Node o denotes the *origin*.

Similarly, the mean distance in the nth inward excursion is

$$\begin{aligned}\overline{d}(\overleftarrow{E_n}) &= P_{n-1} \cdot (1-p)^{nx} \cdot \sum_{i=1}^{nx} (1-p)^{i-1} \cdot p \cdot i \\ &= (1-p)^{n^2 x} \cdot p \cdot \sum_{i=1}^{nx} (1-p)^{i-1} \cdot i. \qquad (3)\end{aligned}$$

We assume that the maximum steps L of a search contain M excursions in all. Thus, $(M+1)M \cdot x = L$. Therefore, the mean distance of the whole search is $\overline{d} = \sum_{n=1}^{M} \overline{d}(\overrightarrow{E_n}) + \overline{d}(\overleftarrow{E_n})$.

Fig. 7 plots the mean distance from the *origin* of a search with a maximum of 1000 steps as a function of the increment of excursion distance x and the vehicle distribution probability p. It can be seen that, given x, the mean distance decreases as p increases. This is obvious because as p is high, a search has less chance to go further without encountering a target. It can also be seen that, given p, the mean distance also increases as x increases. The results indicate that to optimize the mean distance of a search, the increment of excursion distance should take the minimum value at any vehicle distribution probability.

In ANTS, however, where an ant agent searches on the overlay network, the search walks on discrete nodes. Therefore, the minimum increment of excursion distance on the overlay network should be a hop. In the implementation, taking the average geographical distance of two adjacent nodes deployed in Shanghai as the increment is optimal in ANTS.

V. Design Issues

In this section, we discuss some design issues that ANTS may encounter in practice.

A. Detection and Avoidance of Search Traps

According to the ANTS search strategy, an outward excursion would not terminate before the query packet reaches its excursion distance, and an inward excursion would not end before the query packet returns back to the *origin*. However, there exist some circumstances under which the query cannot continue its excursions without extra assistant schemes. We call such a situation a *search trap*.

Fig. 8 illustrates an example of a search trap in an outward excursion, where node a receives an outward-excursion query packet of an *ant agent* with excursion distance l. Suppose l is larger than the distance between node d and the *origin*. Assume that all the illustrated nodes have no desirable information. At first, a selects b to send the packet; b selects d as the query's successor; and d can only choose b or c to forward the query, but no matter which one d picked, for example, c, c will choose d again to forward the query since d's distance is larger than its own. Consequently, the *ant agent* is set in a search trap because it can never get out from b, c, and d until it exhausts all left steps.

To solve search traps, for an *ant agent*, a node needs to remember the direction and the excursion distance of the packet and which neighbors it has chosen to send the query packet. More specifically, when a node resees the same query packet received from a neighbor it has once chosen to send the packet, the node marks this neighbor as unusable and selects a neighbor from unmarked neighbors to forward the packet. Then, it logs down this neighbor and the direction and excursion distance of the packet. In the example illustrated in Fig. 8, if d chooses b, then b resees the packet and marks d as unusable, and now, b can only choose a to send the packet; a marks b as unusable and sends the packet to the best candidate c; and it is easy to see that a finally marks c as unusable too. It is also easy to check if d chooses c, and finally, a also marks b and c as unusable. Therefore, the *ant agent* gets out the search trap. Fig. 9 illustrates an example of search trap in inward excursion. The solution is similar and a little easier since nodes need not log the excursion distances for inward excursions.

With this solution, future excursions will ignore those neighbors marked as unusable and avoid search traps. This saves the *ant agent*'s search steps.

B. Multiple Ant Agents

The ANTS search strategy has the property that the closer a node is to the *origin*, the higher probability that it will be examined by an *ant agent*. To reduce the query latency,

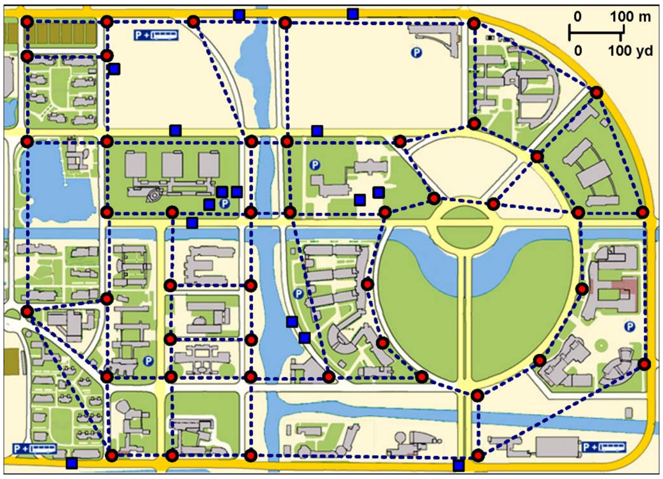


Fig. 10. Layout of the prototype implementation consisting of 45 nodes denoted by red spots.

the *origin* may generate multiple simultaneous *ant agents* and let them search the overlay network in concurrence. Multiple *ant agents* would also improve the hit rates of searches for rare vehicles. In addition, an *ant agent* might die with single node failure. However, the network traffic also increases as the number of *ant agents*. We examine the tradeoff between query locality and network traffic in the next section.

Two methods, i.e., time-to-live (TTL) and "checking," to terminate multiple random walkers are discussed in [18]. TTL refers to each random walk terminating after a certain number of hops. "Checking" refers to a walker periodically checking with the original requestor whether the requestor has obtained an answer before walking to the next node. We also evaluate these two methods and adopt "checking" to terminate multiple *ant agents*.

C. 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. ANTS does not specify any elaborate updating strategy for the moving vehicles' information. This is good for minimizing the communication overhead for data updating. Nevertheless, we need some data replication strategy to improve the data availability. We simply duplicate the vehicles' information to immediate neighbor nodes.

VI. Prototype Implementation

To validate the ANTS design and prove its practical implementation, we have built a prototype system in the campus to track the experimental vehicles. This prototype system contains 45 local nodes distributed in our campus. As shown in Fig. 10, the local nodes (denoted by red spots) are deployed at the 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. 10.

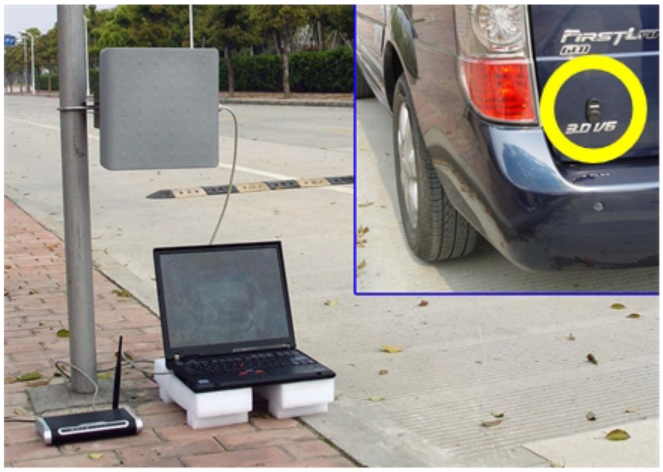


Fig. 11. Local node with an RFID reader and a wireless AP. The highlighted area in the inset shows an active RFID tag attached to a vehicle.

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. 11 shows a typical local node, which is associated with an SP-D300 RFID reader [18], as well as an IEEE 802.11g wireless AP. The inset in Fig. 11 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 Mb/s. 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 of up to 125 mi/h. In addition to the RFID system, the wireless communication technology is also investigated in our prototype implementation. The ANTS protocol runs on Red Hat Fedora 5 and uses POSIX.1 socket API to communicate with each other. User Datagram Protocol (UDP) packets are adopted for location updating and query routing. The size of all packets is 40 B, which includes 20 B of the Internet Protocol packet header, 8 B of the UDP packet header, and 12 B of data.

With this prototype implementation, we conduct a variety of experiments and compare ANTS with a broadcast scheme using a spanning tree for location information updating. We set the search steps of ant agents (i.e., the TTL field in a query packet) to be 100. We deploy active RFID tags to 15 cars of faculty volunteers. A distribution of these experimental vehicles at 10 A.M. on June 30 is depicted by the blue lattices in Fig. 10. As the vehicles enter an RFID reader's field and are captured by the reader, the associated node accordingly logs the location information. During our experiments, we let each node randomly generate 100 queries in half an hour.

Among all the 4500 queries, the maximum query latency is 972 ms, which takes 54 steps. We also notice that the average search steps are about 17. There is no network traffic for location updating among all nodes. In contrast, the network traffic for location updating using broadcast on a spanning tree is about 810 kB. The network traffic for query routing is about

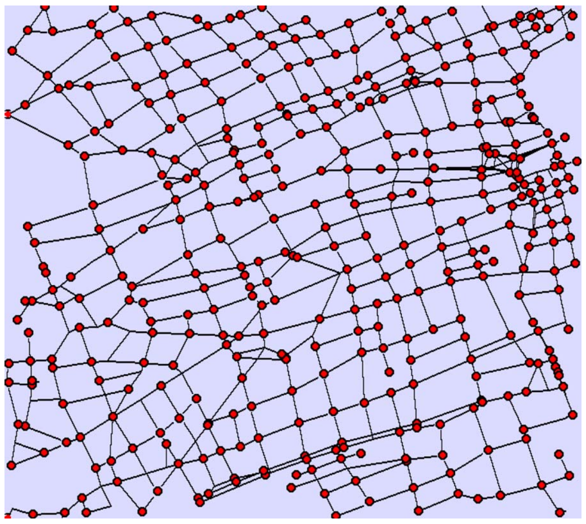


Fig. 12. Topology of the downtown area of Shanghai with 520 nodes deployed at crossroads of this area.

3.06 MB using ANTS. In contrast, using spanning tree to flood query costs 8.1 MB. We can imagine that, in a large-scale network (e.g., in a large city), using flooding for query routing will generate a tremendous network traffic and, therefore, has very limited scalability. Moreover, in our experiments, about 85% of the vehicles returned by ANTS are identical to the answers returned by using query flooding.

The lesson from our prototype implementation is that, with appropriate configuration of the protocol parameters, the overall network traffic overhead, which was introduced by location updating and query routing, can well accommodate a large number of queries. ANTS introduces limited network traffic while still finding the nearest results. To further investigate the performance of ANTS in a large-scale setting, we conduct trace-driven simulations, which are detailed in the following section.

VII. Performance Evaluation

A. *Methodology*

In the simulations, the ANTS protocol is implemented on ns2 [7]. The simulations are solely based on the real complex road network of Shanghai and driven by real trace data of vehicle movements in the Shanghai urban area. We adopt the large volume of trace data of the vehicle movements, which was obtained with GPS technology from August 2006 to October 2006. The information we can directly obtain from GPS reports is very limited: a vehicle's location coordinates, timestamp, and optional speed and heading. The underlying physical network topology of Shanghai is generated with 10 000 routers using Brite [8]. The overlay network is created where local nodes are deployed on every crossroad. The overlay topology of 520 local nodes employed in our simulations is depicted in Fig. 12, where the mean distance of two adjacent nodes is 293 m.

We compare ANTS with two other alternative schemes that might be used in locating the nearest desirable vehicles in SG.

1) Random walks. To search for the vehicle, the query is carried out by simultaneous random walkers. A walker checks with the querying node every certain steps and terminates either if the querying node has already retrieved the result or if the maximum steps are reached.
2) Expanding flooding. The query is flooded in the overlay network. At the beginning, the TTL of the query is small. If this is not successful, then the query will be flooded again with an increased TTL. This process is repeated until the vehicle is found.

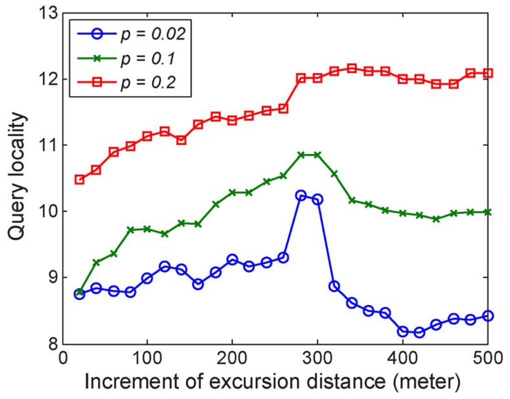


Fig. 13. Query locality versus excursion distance increment.

In ANTS, query locality is the main concern of the search strategy, which presents that a query near a target should make it easy to find the target. This means that not only the distance of retrieved targets from the *origin*, but the network traffic aroused during the search as well, should be the factor to evaluate query locality. We need to design a utility function to evaluate the query locality so that both the distance from the *origin* and the network traffic overhead contribute. We define the metric of query locality as follows:

$$m_{QL} = \frac{c}{k \cdot \log d + \log \tau} \tag{4}$$

where c is a scale coefficient, d is the distance of a retrieved target from the *origin*, k is the weight coefficient of d, and τ is the total network traffic aroused during the search. If the distances of the target d and the aroused network traffic τ take the same weight to the query, then k should take 1. The reason we take the logarithmic forms of d and τ in (4) is that the logarithmic functions are the only continuous isomorphisms from the multiplicative group of positive real numbers to the additive group of real numbers. In addition, we use addition to remove the significance order of preferences from the utility function. We also take the reciprocal form to make the utility function monotonically increasing with d and τ [20]. Therefore, a smaller d and τ will enlarge m_{QL}.

B. *Increment of Excursion Distance*

We first consider the increment of excursion distance, which is denoted as x. We limit the maximum query latency by specifying the TTL in an ant agent. For a particular vehicle distribution probability, we randomly select ten nodes as the *origins* and, for each *origin*, vary x from 20 to 500 m in increments of 20 m. For each value, we repeat the experiment 20 times. Fig. 13 plots the mean m_{QL}, whereas p is equal to

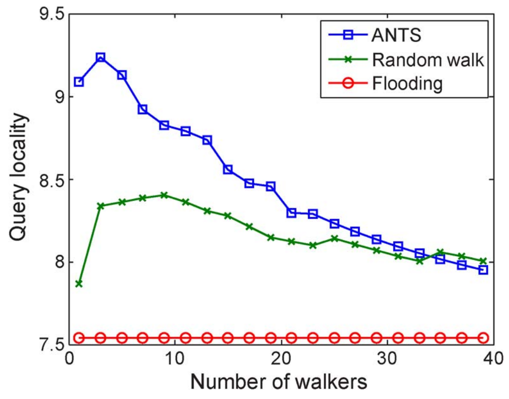


Fig. 14. Query locality versus number of *ant agents*.

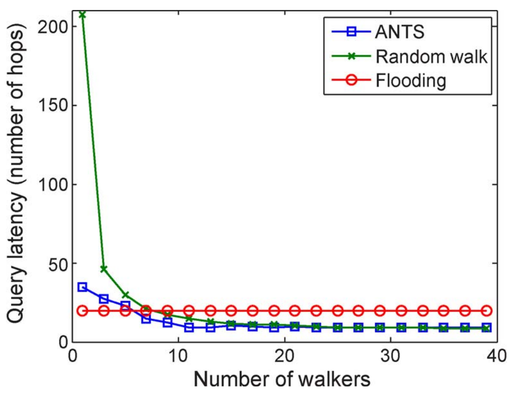


Fig. 15. Number of walkers versus query latency.

0.02, 0.1, and 0.2, respectively. The metric m_{QL} reaches the maximum when x is around the mean distance of two adjacent nodes, i.e., 293 m. Two reasons account for this result. First, the search would generate more traffic at the already-visited nodes when x is less than the mean distance. Second, the search would prefer to select further targets when x is larger than that. This result matches our analysis in Section III. It also indicates a choice of the mean distance of two adjacent nodes in the network as x is optimal for any vehicle distribution density.

C. Number of Ant Agents

By having multiple ant agents, the system can reduce the search latency and improve the resilience against node failures. We use "checking" to terminate multiple ant agents since it is more adaptive than TTL [9]. In this experiment, we set the distance of excursion distance equal to 293 m. The vehicle distribution probability is set to 0.02. Fig. 14 plots m_{QL} over the number of ant agents and that of random walkers with checking periods equal to every four hops. The metric m_{QL} reaches the maximum when the number of ant agents is 3 in ANTS. The results clearly show that ANTS needs to conduct a few ant agents for a search while keeping good performance. In addition to the query locality, Fig. 15 plots the query latency. It can be seen that a few number of ant agents can have less query latency than that of expanding flooding. Fig. 16 also plots m_{QL} over checking periods. In general, choosing eight hops for checking is appropriate.

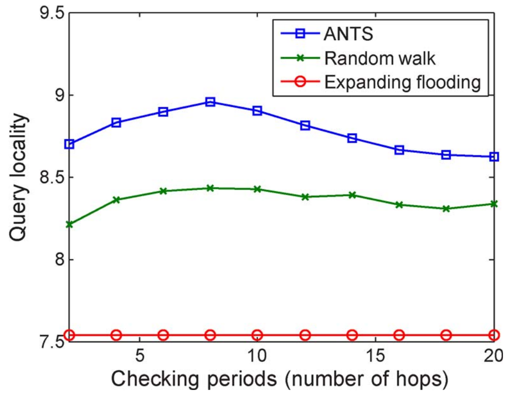


Fig. 16. Checking periods versus query locality.

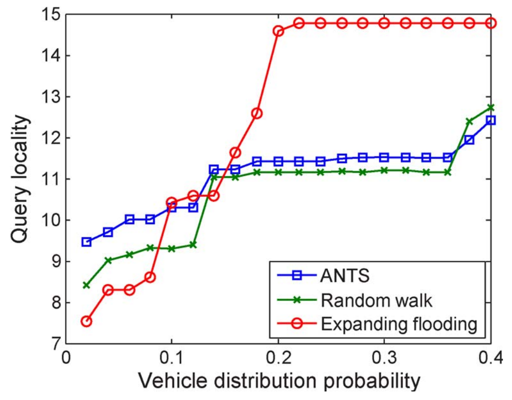


Fig. 17. Vehicle distribution probability versus query locality.

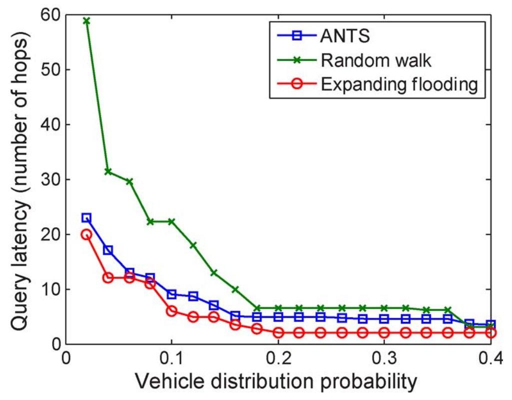


Fig. 18. Vehicle distribution probability versus query latency.

D. Vehicle Distribution Density

In this experiment, we examine the performance of ANTS under different vehicle distribution densities. Three simultaneous ant agents are generated to conduct the search with checking periods set to every eight hops. Fig. 17 plots m_{QL} over the vehicle distribution probability on a node p. When p is relatively large (e.g., greater than 0.2), expanding flooding has a great query locality because it can find the nearest targets before the flooding scale grows very large. Fig. 18 plots the query latency. It can be seen that the query latency of ANTS is less than that of expanding flooding. These results indicate that ANTS outperforms random walks and expanding flooding while searching for common and rare vehicles.

VIII. Conclusion and Future Work

In this paper, we have presented ANTS for locating a nearby vehicle based on the smart search employed by lost ants. ANTS can retrieve the nearest desirable vehicles with high probability but introduces modest network traffic and query latency. It is truly scalable to the number of users, the number of vehicles, and the system scale. Prototype implementation and comprehensive simulations based on the real road network and trace data of vehicle movements demonstrate the efficacy of ANTS.

This is an ongoing research and system effort in locating the nearest 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 locating personal vehicles all the time. The government will guarantee to protect the individual privacy by authorizing legal individuals and corporations with different privileges to access appropriate vehicles. Next, we will delve into designing better schemes such that the query routing overhead can be reduced as much as possible. Based on our realistic prototype testbed, we will validate our design and study its performance under real complex environments. Improvements will be made based on realistic studies before it comes to be deployed in the large-scale SG system.

References

[1] M. Li, M.-Y. Wu, Y. Li, J. Cao, L. Huang, Q. Deng, X. Lin, C. Jiang, W. Tong, Y. Gui, A. Zhou, X. Wu, and S. Jiang, “ShanghaiGrid: An information service grid,” *Concurr. Comput.: Pract. Experience*, vol. 18, no. 1, pp. 111–135, Jan. 2006.

[2] Shanghai City Comprehensive Transp. Planning Ins. [Online]. Available: http://www.scctpi.gov.cn

[3] Ltd. Shanghai Telecom Co. [Online]. Available: http://www.shanghaitelecom.com.cn/

[4] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, “LANDMARC: Indoor location sensing using active RFID,” in *Proc. PerCom*, 2003, p. 407.

[5] *Cisco Aironet 1240AG Series 802.11A/B/G Access Point Data Sheet*, Cisco Syst. Inc., San Jose, CA. [Online]. Available: http://www.cisco.com

[6] R. Wehner and M. V. Srinivasan, “Searching behaviour of desert ants, genus cataglyphis (formicidae, hymenoptera),” *J. Comp. Physiol. A*, vol. 142, no. 3, pp. 315–338, Sep. 1981.

[7] *The Network Simulator*. [Online]. Available: http://www.isi.edu/nsnam/ns/

[8] BRITE. [Online]. Available: http://www.cs.bu.edu/brite/

[9] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, “Search and replication in unstructured peer-to-peer networks,” in *Proc. Supercomput.*, 2002, pp. 84–95.

[10] 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,” in *Proc. USENIX*, 2000, pp. 141–152.

[11] D. Pfoser, C. S. Jensen, and Y. Theodoridis, “Novel approaches to the indexing of moving object trajectories,” in *Proc. VLDB*, 2000, pp. 395–406.

[12] M. Pelanis, S. Saltenis, and C. S. Jensen, “Indexing the past, present, and anticipated future positions of moving objects,” *ACM Trans. Database Syst.*, vol. 31, no. 1, pp. 255–298, Mar. 2006.

[13] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for Internet applications,” in *Proc. ACM SIGCOMM*, 2001, pp. 149–160.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, “A scalable content-addressable network,” in *Proc. ACM SIGCOMM*, 2001, pp. 161–172.

[15] *The Gnutella Protocol Specification v0.6*, 2005. [Online]. Available: http://rfc-gnutella.sourceforge.net

[16] C. Gkantsidis, M. Mihail, and A. Saberi, “Random walks in peer-to-peer networks,” in *Proc. IEEE INFOCOM*, 2004, pp. 120–130.

[17] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Gossip algorithms: Design, analysis, and applications,” in *Proc. IEEE INFOCOM*, 2005, pp. 1653–1664.

[18] Ltd. Shanghai Super Electron. Technol. Co., 2007. [Online]. Available: http://www.superrfid.net/english/

[19] E. T. Jaynes, *Probability Theory: The Logic of Science*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[20] R. G. Bartle, *The Elements of Real Analysis*. New York: Wiley, 1976.

Minglu Li (M’05–A’05–M’08) received the Ph.D. degree in computer software from Shanghai Jiao Tong University, Shanghai, China, in 1996.

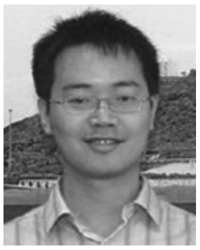
He is a full Professor and the Vice Dean of the School of Electronics Information and Electrical Engineering and the Director of the Grid Computing Center, Shanghai Jiao Tong University. He has published more than 100 papers in academic journals and international conferences. His current research interests include grid computing, services computing, and sensor networks.

Dr. Li is a member of the Executive Committee of the Technical Committee on Services Computing of the IEEE Computer Society.

Hongzi Zhu (M’08) received the M.S. degree in computer science from Ji Lin University, Changchun, China, in 2004. He is currently working toward the Ph.D. degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.

His research interests include peer-to-peer computing, vehicular ad hoc networks, pervasive computing, distributed systems, and network security.

Mr. Zhu is a Student Member of the IEEE Communication Society.



Yanmin Zhu (M’08) received the B.Eng. degree in computer science from Xi’an Jiao Tong University, Xi’an, China, in 2002 and the Ph.D. degree in computer science from the Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2007.

He is currently an Assistant Professor with the Department of Computer Science and Technology, Shanghai Jiao Tong University, Shanghai, China. His research interests include ad hoc sensor networks, mobile computing, grid computing, and resource management in distributed systems.

Dr. Zhu is a member of the IEEE Communication Society.

Lionel M. Ni (S’78–M’80–SM’87–F’94) received the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, 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, Kowloon, Hong Kong. His research interests include parallel architectures, distributed systems, wireless sensor networks, high-speed networks, and pervasive computing.

Dr. Ni has chaired many professional conferences and has received a number of awards for authoring outstanding papers.