ANTS: Efficient Vehicle Locating Based on Ant Search in ShanghaiGrid

Minglu Li, Senior Member, IEEE, Hongzi Zhu, Student 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 aims to provide abundant intelligent transportation services to improve the traffic condition. A fundamental service in ShanghaiGrid is to locate the nearest desirable vehicles for users. In this paper we propose an innovative protocol ANTS to locate a desirable vehicle close to the querying user. The protocol finely mimics the efficient searching strategy adopted by a lost ant in searching for its nest. Taking query locality into account, ANTS can retrieve the nearest 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, RFID system, peer-to-peer network, vehicle locating

1 INTRODUCTION

Intelligent transportation systems (ITS) have been evolving rapidly in the past two decades, leveraging advanced computing and communication technologies. Shanghai, the largest metropolis in China, covers an area of 5,800 square kilometers 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, 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.

It is a fundamental service in SG to locate a desirable vehicle which 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 timely. As another example, a user wants to report an accident to a police car or an ambulance car. 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 changes 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 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 thousand events per second. Such a large volume of location updating data can easily overwhelm the centralized server. Therefore, distributed schemes are inevitable for SG. Given that vehicle information is all 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 ANTS to solve this problem. Interestingly, we
find that such a search for a nearest vehicle is very similar to the search of a lost ant for its nest. Inspired by the efficient search strategy of lost ants, a query would generate an agent (like the ant) which searches the overlay network of local nodes by performing a number of loop searches of ever-increasing size, starting and ending at the 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 Shanghai city.

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

- We propose ANTS, a novel protocol for locating the nearby vehicle in the large-scale SG. ANTS locates the nearest vehicles with high probability at minimal cost of network traffic and query latency.

- 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.

- 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 matches our design of ANTS and demonstrates its practical implementation.

- We evaluate the performance through trace-driven simulations. We base our simulations on the real road network and trace data of vehicle movements in Shanghai.

The rest of this paper is structured as follows. In Section 2, we introduce SG and present the system model for the ANTS design. Section 3 compares ANTS with related work. Section 4 describes the design of ANTS and its theoretical analysis. Several design issues that may be encountered in practice are discussed in Section 5. Section 6 describes our prototype implementation of the nearest vehicle locating system realizing the ANTS protocol. In Section 7, 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 8.

2 RELATED WORK

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

The Globe system [10] constructs a static world-wide 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 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 one 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.

In structured peer-to-peer (P2P) networks, various DHT (Distributed Hash Table) schemes were proposed to map objects to peers in a decentralized way and enabled the system to satisfy queries very efficiently [13, 14]. However, DHTs cast objects uniformly 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 1. 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 for query locality. ANTS introduces minimal updating cost and small query routing overhead to locate the nearest vehicles with high probabilities.

3 SYSTEM MODEL

As RFID technology continuously evolves, it has been widely used in tracking various mobile objects. The SG project exploits the promising RFID and local-area wireless communication technologies. The infrastructure of SG, which is still underway, is illustrated in Figure 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 the Internet through a dedicated connection or a cheap ADSL connection provided by Shanghai TeleCom [3].

In SG, the vehicles' information is gathered both actively and passively. In the initial prototype, a vehicle is
captured passively by using 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 meters. The battery can sustain the operation of an active RFID tag for about 2 years [4]. 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 access point working under IEEE 802.11g has an effective outdoor communication range of about 280 meters at the transmission rate of 2Mbps [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 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 meters deviated 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 be notably long. Due to the GPRS communication cost for transmitting the GPS location information back to the data centre, drivers prefer to choose relatively large intervals. The typical value would be from 1 minute 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 ANTS using the real trace data.

4 DESIGN OF ANTS

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

4.1. Introduction to Ant Search

The process of a lost ant’s 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 to 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, 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 varies randomly. This component also tends to be high when the ant is far from the origin and the ant tends to run radially 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. 2 Trajectory of the simulated search pattern with 500 discrete steps

Fig. 3 The spatial distribution of the searching density of 10 ants w 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 certain vehicle, the query can be forwarded to different nodes in the overlay network. The query also expects that the target vehicle is near to 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 of the nearest vehicle. This inspires the design of ANTS.

4.2. ANTS Search Algorithm

Before discussing the strategy details, we first study the distribution of the expected position of the target vehicle 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 two-dimensional probability distribution. We find that this distribution is a two-dimensional 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 towards 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 distribution probability and the abscissa \( d \) denotes the distance from the origin spot, denoted as \( o \), where the ant begins to search. The search starts at the peak of the Gaussian PDF, i.e., at the origin as shown in Fig. 4(a). If the nest is not found at the origin, the most promising region is then the annular region surrounding the origin. At 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, so the search returns back towards the origin as shown in Fig. 4(d), and later as shown in Fig. 4(e) and Fig. 4(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 decreased cyclically, with each outward excursion having a larger distance than the previous one, as specified by the profile of the Gaussian PDF. Besides 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, 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 be easily realized by checking the fine-grained electronic map of the city. By this means, the locality of 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 maintained locally. 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 a 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, 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, the node computes its distance from the origin. If the distance is less than the excursion distance of the query packet, 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 nearer to the origin than itself and forwards the packet.

If the origin receives an inward-excursion query packet, it continues to conduct 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 of 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.

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

### 4.3. Protocol Parameter Optimization

We now discuss the optimal configuration of the protocol parameters during the oscillating-fashioned search. We expect 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 be approximately seen as constant times longer than straight lines. The abstract search model is plotted in Fig. 6, where the origin, 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 \( n \)th 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 \( n \)th excursion only when all previous steps in the trajectory have missed the chance 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,

\[
P_{n-1} = (1 - p)^{x} \cdot (1 - p)^{2x} \cdot \ldots \cdot (1 - p)^{x(n-1)x} = (1 - p)^{x(n-1)x}.
\]

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

\[
\overline{d(E_n)} = P_{n-1} \cdot \sum_{i=1}^{n} (1 - p)^{x(n-1)x} \cdot p \cdot i = (1 - p)^{x(n-1)x} \cdot p \cdot \sum_{i=1}^{n} (1 - p)^{x(n-1)x} \cdot i.
\]

Similarly, the mean distance in the \( n \)th inward excursion is,

---

**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 MAXIMUM search steps */
4. **else if** \( pkt.TTL \geq 0 \)
5. compute the distance \( d \) between node \( nd \) and the origin node;
6. /* an outward excursion */
7. **if** \( pkt.direction == \text{outward} \) **then**
8. **if** \( d \leq pkt.excursion\_distance \) **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** \( pkt.direction == \text{inward} \) **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 \( pkt'(nd, origin, outward, excursion\_distance+\Delta, TTL-1) \);
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 The ANTS search algorithm.
We assume the maximum steps, $L$, of a search contain $M$ excursions in all. Thus, $(1 + M)M \cdot x = L$. So the mean distance of the whole search is $d = \sum_{i=1}^{M} \bar{d}(E_n) + \bar{d}(E_i)$.

Fig. 7 plots the mean distance from the origin of a search with a maximum of 1,000 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 implementation, taking the average geographical distance of two adjacent nodes deployed in Shanghai as the increment is optimal in ANTS.

5 Design Issues

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

5.1 Detection and Avoidance of Search Traps

According to 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 can not continue its excursions without extra assistant schemes. We call such a situation a search trap.

Fig. 8 illustrates an example of search trap in an outward excursion, where node $a$ receives an outward-exursion query packet of an ant agent with excursion distance $l$. Suppose $l$ is larger than the distance between node $d$ and the origin. Assume all illustrated nodes have no desirable information. At first, $a$ selects $b$ to send the packet; $b$ selects $d$ as the query’s successor; $d$ can only choose $b$ or $c$ to forward the query but no matter which one $d$ picked, say $c$, $d$ 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 packet. More specifically, when a node re-sees 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 $b$, $b$ re-sees the packet and marks $d$ as unusable, and now $b$ can only chooses $a$ to send the packet; $a$ marks $b$ as unusable and sends the packet to best candidate $c$; it is easy
to see that $a$ finally marks $c$ as unusable too. It is also easy to check if $d$ chooses $c$, $a$ also marks $b$ and $c$ as unusable at last. 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 to 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.

5.2. Multiple Ant Agents

ANTS search strategy has the property that the nearer a node is from the origin the higher probability it would be examined by an ant agent. To reduce the query latency, the origin may generate multiple simultaneous ant agents and let them search the overlay network in concurrency. 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 the network traffic in the next section.

Two methods, TTL (Time-To-Live) and “checking”, to terminate multiple random walkers are discussed in [18]. TTL refers to each random walk terminates after a certain number of hops. “Checking” refers to a walker periodically checks 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.

5.3. 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 moving vehicles’ information. This is good for minimizing the communication overhead for data updating. Nevertheless, we need some data replication strategy to improve data availability. We simply duplicate vehicles’ information to immediate neighbor nodes.

6 Prototype Implementation

To validate the ANTS 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. 10, 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. 10. An overlay connection is established between two nodes if there is a road 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 a SP-D300 RFID reader [18] as well as an IEEE 802.11g wireless AP. The inset of 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 a RS-485 interface and has a data transfer rate of 1 Mbps. The reader has a configurable operation range from 2 to 80 meters. Each reader can simultaneously detect up to 200 tags in 800 milliseconds. 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 milliseconds and can be detected at a high speed up to 125 miles per hour. Besides the RFID system, 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. 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 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
am on June 30 is depicted by the blue lattices in Fig. 10. As the vehicles enter an RFID reader’s field and is captured by the reader, the associated node logs the location information accordingly. During our experiments, we let each node randomly generate one hundred of queries in half an hour.

Among all the 4,500 queries, the maximum query latency is 972ms, 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 810KB. The network traffic for query routing is about 3.06MB using ANTS. In contrast, using spanning tree to flood query costs 8.1MB. We can imagine that, in a large-scale network (e.g., in a large city), using flooding for query routing will generate tremendous network traffic and therefore has very limited scalability. Moreover, in our experiments, about 85% 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, introduced by location updating and query routing, can well accommodate a large number of queries. ANTS introduces limited network traffic whereas can still find most 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.

7 PERFORMANCE EVALUATION

7.1 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 Shanghai urban area. We adopt the large volume of trace data of vehicle movements which was obtained with GPS technology from August 2006 to October 2006. The information we can obtain directly 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 meters.

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

- **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.

- **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. This process is repeated until the vehicle is found.

In ANTS, query locality is the main concern of the search strategy, which presents that a query near to a target should be easy to find the target. This means not only the distance of retrieved targets from the origin but also the network traffic aroused during the search should be the factor to evaluate query locality. We need to design a utility function to evaluate 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

\[ m_{QL} = \frac{c}{k \cdot \log d + \log \tau}. \]  

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 \( \tau \) is the total network traffic aroused during the search. If the distance of the target, \( d \), and the aroused network traffic, \( \tau \), take the same weight to the query, \( k \) should take 1. The reason we take the logarithmic forms of \( d \) and \( \tau \) in (4) is that 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 \( \tau \) [20]. Therefore, a smaller \( d \) and \( \tau \) will enlarge \( m_{QL} \).

7.2. Increment of Excursion Distance

We first consider the increment of excursion distance, 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 10 nodes as the origins and, for each origin, vary \( x \) from 20 meters to 500 meters in increments of 20 meters. For each value, we repeat the experiment 20 times. Fig. 13 plots the mean \( m_{QL} \) while \( p \) equals to 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 meters. Two reasons account for this result. First, the search would generate more traffic at 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 3. It also indicates that choosing the mean distance of two adjacent nodes in the network as $x$ is optimal for any vehicle distribution density.

### 7.3. Number of ant agents

By having multiple ant agents, the system can reduce the search latency and improve 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 meters. The vehicledistribution 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 4 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 few ant agents for a search while keeping good performance. Besides 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 8 hops for checking is appropriate.

### 7.4. 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 8 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 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 as less as that of expanding flooding. These results indicate ANTS outperforms random walks and expanding flooding while searching for common and rare vehicles.

### 8 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 on-going research and system effort in locating 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 implications of locating 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
schemes such that query routing 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.

ACKNOWLEDGMENT

This research was supported in part by the National Basic Research Program (973 Program) of China (No. 2006CB303000), NSFC (No. 60473092, 90612018 and 60533110), STCSM (No. 05DZ15005), Hong Kong RGC Grants HKUST617908 and HKBU 1/05C.

REFERENCES


Minglu Li received his PhD in Computer Software from Shanghai Jiao Tong University in 1996. He is a full Professor. He is a full professor and the vice dean of the School of Electronic Information and Electrical Engineering, the director of Grid Computing Center in Shanghai Jiao Tong University. Currently, grid computing, services computing, and sensor networks. He has published more than 100 papers in academic journals and international conferences. He is also a member of the Executive Committee of the Technical Committee on Services Computing of the IEEE Computer Society.

Hongzi Zhu received his MS degree in computer science from Ji Lin University in 2004. He is a PhD candidate in the Department of Computer Science and Engineering at the Shanghai Jiao Tong University. 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 and the IEEE Communication Society.

Yanmin Zhu obtained his PhD in computer science from Hong Kong University of Science and Technology in 2007, and BEng in computer science from Xi’an Jiao Tong University in 2002. He is a Assistant Professor in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. 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 earned his Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 1980. He is Chair Professor and Head of Computer Science and Engineering Department at the Hong Kong University of Science and Technology. His research interests include parallel architectures, distributed systems, wireless sensor networks, high-speed networks, and pervasive computing. A fellow of IEEE, Dr. Ni has chaired many professional conferences and has received a number of awards for authoring outstanding papers.