CAB: A CAB BOOKING APPLICATION USING SHORT-RANGE WIRELESS COMMUNICATION

Sweety[1], Aastha Bathla[2], Sweta Kumari[3], Indu Khatri[4]
[1,2,3]Students of Computer science, BMCEM, Sonipat
[4]Guide

Abstract: Cab is a proof-of-concept ubiquitous computing application that allows people to book nearby cabs using their cell phones or PDAs equipped with short-range wireless network interfaces. Cab discovers and books free cabs using mobile ad hoc networks of vehicles. We have implemented a Cab prototype on top of Smart Messages, a middleware architecture based on execution migration, which we had developed to provide a common execution environment for outdoor ubiquitous computing applications. The experimental and simulation results have demonstrated the feasibility of Cab.

I. INTRODUCTION

For many years, people have discussed about mobile ad hoc networks (MANET) and have proposed numerous routing algorithms [23, 31], but the technology has not been mature enough to support the deployment of such networks in the real world. Recently, shortrange wireless networking has started to live up to its expectations, and we see a large deployment of products based on either IEEE 802.11 standards or Bluetooth (a low-cost, low-power alternative to IEEE 802.11 family of protocols). While short-range wireless technology is on its way to become ubiquitous in the near future, little has been done to develop real-life services or applications over mobile ad hoc networks. This paper presents Cab, a real-life ubiquitous computing application built over MANET, which allows people to book nearby cabs in densely populated urban areas using their cell phones or PDAs equipped with short range wireless network interfaces. Current cab booking systems rely on centralized schemes for cab dispatching short messages (SMSs) to a certain server over cellular links [1, 2]. Although under the traditional centralized solution cab dispatching is guaranteed, this solution is not scalable due to: 1) all requests have to go through one or multiple cab dispatchers, which introduces waiting time for the clients, especially during periods of peak cab requests, and 2) in order to dispatch the nearest cab to the client, all cabs in the city have to be monitored to find the closest one to the client's location.

The Cab dispatching system, on the other hand, is simpler, faster, and more scalable since it works in a completely decentralized fashion, and there is no need to gather the locations of all the cabs in real-time. Cab provides all these benefits because it defines a system architecture in which the clients and vehicles communicate using only short-range wireless network interfaces. This design decision, however, makes Cab a "best effort" service. Clients can switch to the standard centralized methods to book a cab if they fail to get a cab using Cab within a short period of time. Hence, the Cab system can be incrementally deployed and coexist with current centralized systems. Cab is useful in cities with high density of cabs, such as New York or Tokyo, where the contention to get cabs during certain periods (e.g., people getting out from a show) is very annoying.

Cab is made possible by two recent technology trends. The first is the transformation of PDAs (e.g., HP iPAQ [5], Toshiba Pocket PC [10]) and cell phones (e.g., Ericsson P900 [7], Motorola A760 [6]) into relatively powerful mobile computers equipped with short-range wireless capabilities. The second is the increasing presence of powerful embedded systems, GPS receivers, and even wireless network interfaces in modern vehicles. For instance, GPS has been successfully used to track vehicles and provide accurate position information [11, 21]. A taxi service, based on real-time GPS information collected by a centralized dispatching centre over cellular networks, has also been implemented in Singapore [28]. We have implemented a Cab prototype on top of Smart Messages (SM) [13, 24], a middleware architecture based on execution migration, which we developed to provide a common execution environment for outdoor distributed applications. An SM carries its own routing code and routes itself at each node in the path toward a node of interest. To perform routing, SMs store routing information in a memory addressable by names at nodes. The SM self-routing mechanism [12] is especially useful for Cab because it allows on-demand deployment of new routing algorithms and changing the routing algorithm during execution. This feature enables Cab to adapt to highly dynamic network configurations. The testbed for Cab consists of ad hoc wireless networks composed of HP iPAQs running Linux and communicating through IEEE 802.11 network interface cards. The experimental results show that Cab is a viable solution for booking cabs in densely populated urban areas. We have also evaluated and compared through simulations multiple mechanisms for discovering free cabs under different traffic scenarios. The rest of this paper is organized as follows. describes the Cab prototype, its SM-based routing algorithms, and the experimental results. Section 4 presents the simulation results comparing different routing algorithms.

II. DESIGN

Cab consists of a mobile ad hoc network of computers embedded in taxis and client handheld devices, which communicate using short-range wireless network interfaces such as IEEE 802.11. Instead of booking cabs through a centralized dispatcher, Cab clients book free nearby cabs by

communicating directly with other Cab nodes (i.e., taxis) over a mobile ad hoc network. This decentralized architecture provides a simple, cheap, and scalable solution to a real-life problem. However, Cab presents new challenges which do not exist in traditional systems based on centralized dispatching centre. For instance, we need a distributed protocol to ensure that at most one cab arrives at the site of the client who requested the cab.

Furthermore, we must ensure that any free cab accepts only one client request at any point in time. To provide an automatic booking mechanism, we also need accurate location information for both clients and cabs (e.g., the client's street address has to be present in the booking request). Finally, Cab needs to provide a mechanism for the client and driver to authenticate each other when they meet. This section presents the Cab system architecture and its protocol that satisfies these requirements.

2.1 System Architecture

Cab consists of two types of entities: client stations and driver stations. A client station is PDA (or cell phone), while a driver station is a system embedded in the cab. We assume that all driver stations communicate with each other using IEEE 802.11, the de facto standard for high bandwidth, short-range wireless networking. Additionally, each driver station has a GPS receiver that can report its current location when needed. All driver stations that are within the radio range of each other form an ad hoc network of nodes which communicate and perform the distributed computation necessary to book a free cab. The client stations have to join such a network to inject their requests for free cabs. A client station communicates directly with cabs in its transmission range, and the cabs in the network forward the request until a free cab is discovered. Unlike driver stations which are homogeneous, client stations can have different capabilities. Besides powerful PDAs equipped with IEEE 802.11 wireless network interfaces and GPS receivers, cell phones equipped with lowpower Bluetooth interfaces and without GPS receivers can also be used as client stations. In such a case, however, the client station needs to connect to a gateway station that forwards the request into the network of cabs and sends the answer back to the client. A gateway station is a computer equipped with a GPS receiver and multiple network interfaces (i.e., Bluetooth and IEEE 802.11), which can be co-located with Hotspots (i.e., IEEE 802.11 access points) at public places such as restaurants, stores, theaters, bus stops, telephone booths, and building lobbies. The role of a gateway station is to "stamp" location information on requests received from lighter clients that cannot incorporate a GPS receiver (i.e., given the range of Bluetooth, the location of the gateway is a good approximation for the location of the client).

2.2 Cab Protocol

The Cab application starts when a client sends out a request and ends with a validation at the client's location. The Cab protocol consists of two phases: Cab Booking and Validation. Figure 2 shows the Cab Booking phase. During this, the client and driver stations collaborate with each other to forward the client's request through the network until a free cab is discovered. To accommodate various network configurations determined by density and mobility of the cabs, Cab assumes an extensible routing layer which supports different routing algorithms as plugins (similar to Active Networks [3]). Sections 3 and 4 will present our experiences with developing routing algorithms specific to Cab. Cab Booking is a three-way handshake protocol which ensures that a nearby free cab is booked and no more than one client books a cab simultaneously. It starts by sending a request containing the client's information (e.g., current location, destination location) into an ad hoc network of cabs. This request is routed to a free cab by the routing layer.

Once the driver of a free cab agrees to the client's requirements, the cab status is changed from free to occupied. The expected arrival time (estimated based on the distance between the client and the cab, the time of the day, and previous experiences of the cab driver) and the license plate of the cab are sent to the client. The handshake protocol completes with an acknowledgment from the client station to the driver station. Once the driver station receives the acknowledgment, the cab driver is notified to pick up the client.

III. PROTOTYPE IMPLEMENTATION

We have implemented Cab using Smart Messages [13, 24], a middleware architecture based on execution migration, which we developed to provide a common execution environment for outdoor ubiquitous computing applications. This section presents a short overview of Smart Messages, the Cab prototype, the routing algorithms used by Cab to find free cabs, and experimental results over ad hoc networks of PDAs.

3.1 Smart Messages

Smart Messages (SM) define a middleware architecture, similar to mobile agents, for programming distributed applications over mobile ad hoc networks of resource constrained devices. An SM is a distributed application consisting of code, data, and a lightweight execution state. Instead of transferring data among nodes involved in computation, SMs migrate the execution to each of these nodes. An SM carries routing code and routes itself at each node in the path toward a node of interest. Each node cooperates to support the SM execution by providing a virtual machine (VM) for execution over heterogeneous platforms, a shared memory addressable by names (tag space) for inter-SM communication and synchronization, and a code cache for storing frequently executed code. An SM calls explicitly for migration when it needs to execute on a different node. Upon an SM arrival at a new node, its execution is resumed from the next instruction following a migration invocation. During its execution an SM can spawn or create new SMs. Additionally, an SM can interact with the host or other SMs using tags stored in the tag space. Essentially, the tags are (name, data) pairs. Corresponding to their functionality, there are two types of tags: application tags for "persistent" memory across SM executions which

can store applicationspecific data, and I/O tags for interaction with the host's operating system and I/O. Tags can also be used for synchronization between SMs. An SM can block on a tag until another SM performs a write on that tag (i.e., updatebased synchronization). SMs name nodes by tag names (i.e., content-based naming) and migrate to nodes of interest using a high level migration function that implements routing [12]. The routing is executed at each node on the path toward a node of interest; hence, SMs are self-routing applications. The implementation of routing uses information stored by SMs in the tag space and a systemprovided primitive for one-hop migration. This primitive captures the current execution control state and migrates it to the next hop along with the code and data. The SM platform is developed by directly modifying Sun Microsystem's
Kilobyte Virtual Machine (KVM); KVM features a small memory footprint (160K) suitable for most embedded devices. The tag space and SM operations are available to applications as a Java API.

3.2 Cab Prototype
We have implemented the Cab prototype on top of the SM platform installed on HP iPAQs running Linux.
The iPAQs use Orinoco's 802.11 cards for wireless communication, and each of them is connected to a Geko 201 GPS receiver. We have demonstrated in a different project [18] that cars can be mapped with high accuracy on the roads despite the low accuracy provided by raw GPS data (e.g., raw GPS data provides on average an accuracy of 10 meters). Figure 3 illustrates an Cab driver station.

IV. PERFORMANCE EVALUATION

To compare the performance of the three Cab routing algorithms under different scenarios, we have simulated Cab using the ns-2 simulator [9], enhanced with the CMUwireless extensions [8]. Different scenarios are used to test sensitivities of the algorithms to various application and network parameters. In this section, we describe our experiments and present the corresponding results.

4.1 Scenario Generator
We have developed our own scenario generator tool based on set, a generator tool for random-way point mobility model, developed at Carnegie Mellon University to generate traffic scenario for cities with grid roads (e.g., Manhattan). The scenario generator accepts as parameters the simulation time, the width and length of the city grids in meters, number of horizontal roads, number of vertical roads, number of lanes per road, average speed of the cabs in meters/sec, average gap distance between cabs on the same lane, and the number of clients in the city. In our traffic model, we use an even number of alternated single direction roads along each dimension of the grid. Cabs can change their lanes within the same road independently of each other. The probability of staying on the same lane is 0.6, whereas the probability of changing the lane is
0.4 either to the left or to the right. Similarly, cabs can switch their roads at intersections. With equal probabilities, a cab chooses either to stay on the same road or change to the road it intersects with. When a cab reaches the last intersection on a road, it is forced to change to the road it intersects with. Clients behave similarly to the cabs at the intersections, except that they can move on both directions of a road. Therefore, clients at intersections can choose uniformly between the three new directions. The cabs select their speed as [average speed ±(0 .25×average speed× rand())], where rand() returns a uniformly distributed random number from the range [0,1]. The clients select their speed uniformly from range [0,1] meters/sec. Initially, each cab moves toward a random destination along its road using its currently selected speed. Once a cab reaches its destination, it selects another random destination along its road as well as a new speed. If the selected destination is behind the next intersection on the road, the cab sets its destination to the intersection. Clients select their new destinations in a similar manner. For all the simulations, we fixed the width of the grids to 2,000 meters, while the length is 3,000 meters. The roads are distributed uniformly as 6 vertical roads and 10 horizontal roads with 2 lanes per road. The average gap between successive cabs on the same lane is 185 meters. The scenario generator places [number of lanes× (city width gap × number vertical roads+ city length gap × number horizontal roads)] = 410 cabs and 200 clients evenly distributed on the roads. We used IEEE 802.11b as the wireless media, with a data transmission rate of 11Mb and a transmission range of 250 meters. During our outdoor experiments, we found out that the wireless transmission range is less than 250 meters. However, we have been able to restore this transmission range using external antennas.

4.2 Simulation Results
For all the simulation runs, the first 200 seconds represent a warm up period (i.e., no cab request occurs within this period). Each client sends a cab request (Book once during the simulation period, at a time
chosen randomly after the warm up period. Book is unicasted for the on-demand and proactive routing mechanisms, while it is broadcasted for the flooding mechanism. Note that in case a client uses the Probabilistic On-demand or Probabilistic Proactive routing mechanisms and has no routing table information, it will broadcast the Book or Discover (to simplify the exposition, we will refer only to Book). The maximum number of hops a request can propagate (i.e., TTL) searching for a free cab is set to 20 hops. Once a free cab receives a cab request, it sets its status to reserved for a period chosen randomly between 2 and 5 seconds and sends back a Report SMS. Once a confirmation (Confirm SMS) is received from a client, the cab status is set to booked (occupied) for a random period chosen uniformly over the range [300, 1800] seconds. Regardless of the routing mechanism, a client re-broadcasts a cab request if it does not receive a reply from any free cab within a random period chosen uniformly over the range [2, 5] seconds. The average cab speed is set to 15 meters/second with zero pause time. We set the fudge factor L for the on-demand mechanism to 3. For the proactive mechanism, we set the maximum entries in the routing table (max) to 20, and the

periodic update interval (UPD) for Update SMS to 5 seconds. The maximum number of hops (i.e., TTL) for Update SMS generated by free cabs is set to 3. Each entry in the routing table expires after 2.5×UPD seconds of the last update.

4.2.1 Effects of Number of Free Cabs

We first look at the effect of the initial number of free cabs. An initial free cab can be booked during the simulation and then switches its status back to free after the booking period. On the other hand, all the cabs initialized as booked at the beginning of the simulation remain booked during the simulation period. The initially booked cabs act only as relays. For these runs, we fix the simulation time to 1500 seconds, with the 200 seconds warm up period. The clients request cabs uniformly over the next 1200 seconds, leaving the last 100 seconds for any unfinished requests. mobility model, developed at Carnegie Mellon University to generate traffic scenario for cities with grid roads (e.g., Manhattan). The scenario generator accepts as parameters the simulation time, the width and length of the city grids in meters, number of horizontal roads, number of vertical roads, number of lanes per road, average speed of the cabs in meters/sec, average gap distance between cabs on the same lane, and the number of clients in the city. In our traffic model, we use an even number of alternated single direction roads along each dimension of the grid. Cabs can change their lanes within the same road independently of each other. The probability of staying on the same lane is 0.6, whereas the probability of changing the lane is 0.4 either to the left or to the right. Similarly, cabs can switch their roads at intersections. With equal probabilities, a cab chooses either to stay on the same road or change to the road it intersects with. When a cab reaches the last intersection on a road, it is forced to change to the road it intersects with. Clients behave similarly to the cabs at the intersections, except that they can move on both directions of a road. Therefore, clients at intersections can choose uniformly between the three new directions. The cabs select their speed as [average speed ± (0 .25×average speed× rand ())], where rand () returns a uniformly distributed random number from the range [0,1]. The clients select their speed uniformly from range [0,1] meters/sec. Initially, each cab moves toward a random destination along its road using its currently selected speed. Once a cab reaches its destination, it selects another random destination along its road as well as a new speed. If the selected destination is behind the next intersection on the road, the cab sets its destination to the intersection. Clients select their new destinations in a similar manner. For all the simulations, we fixed the width of the grids to 2,000 meters, while the length is 3,000 meters. The roads are distributed uniformly as 6 vertical roads and 10 horizontal roads with 2 lanes per road. The average gap between successive cabs on the same lane is 185 meters. The scenario generator places [number of lanes× (city width gap × number vertical roads + city length gap × number horizontal roads)] = 410 cabs and 200 clients evenly distributed on the roads. We used IEEE 802.11b as the wireless media, with a data transmission rate of 11Mb and a transmission range of 250 meters. During our outdoor experiments, we found out that the wireless transmission range is less than 250 meters. However, we have been able to restore this transmission range using external antennas.

The average cab speed is set to 15 meters/second with zero pause time. We set the fudge factor L for the on-demand mechanism to 3. For the proactive mechanism, we set the maximum entries in the routing table (max) to 20, and the periodic update interval (UPD) for Update SMS to 5 seconds. The maximum number of

hops (i.e., TTL) for Update SMS generated by free cabs is set to 3. Each entry in the routing table expires after 2.5×UPD seconds of the last update. 10 horizontal roads with 2 lanes per road. The average gap between successive cabs on the same lane is 185 meters. The scenario generator places [number of lanes× (city width gap × number vertical roads + city length gap × number horizontal roads)] = 410 cabs and 200 clients evenly distributed on the roads. We used IEEE 802.11b as the wireless media, with a data transmission rate of 11Mb and a transmission range of 250 meters. During our outdoor experiments, we found out that the wireless transmission range is less than 250 meters. However, we have been able to restore this transmission range using external antennas.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented Cab, a reallife ubiquitous computing application for booking cabs in cities. Cab discovers and books free cabs using only vehicle-to-vehicle short-range wireless communication. Cab is easy to deploy, cost effective, and scalable since it works in a completely decentralized fashion. Cab is just one of the outdoor distributed computing applications that we are developing on top of mobile ad hoc networks using Smart Messages as a common middleware architecture. The experimental results over ad hoc networks of nodes running our prototype have demonstrated the feasibility of Cab. The simulation results using different scenarios show that Cab with a Probabilistic Proactive routing yields the best response time and finds the closest cab to the client. We are considering several extensions to Cab as future work. In this paper, we assumed the cabs are cooperative and willing to propagate data between each other. We plan to investigate the performance of Cab when it uses different classes of cabs in which communication happens only between cabs belonging to the same class. Cab classes model the situation of different cabs companies, where each company is not willing to route messages for a competing company. Another extension will allow occupied cabs to act as free candidate cabs based on their scheduled drop-off location and time. For example, it would be useful to consider a cab that is scheduled to drop-off a client a mere 10 feet from the new client within a minute, which is far better than booking a 15 minutes away free cab. We also plan to study the use of priorities for the free cabs based on, for example, their distances to the client, how long they have been idle, or the number of clients served in the last hour. The goal of such priority system is to guarantee fairness and optimality.

REFERENCES

[1] http://www.citycab.com.sg/services/ net/sms booking.html.
[2] http://www.comforttransportation.com.sg/ booking svcs.html.
[3] http://nms.lcs.mit.edu/activeware/.
[4] http://www.auriga.co.uk/.
[5] HP iPAQ. http://h71016.www7.hp.com.
[6] http://www.motorola.com.
[7] Sony Ericsson P900, http://www.sonyericsson.com/p900/.
[8] The Monarch Group at Rice University. http://www.monarch.cs.rice.edu/.
[9] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns/.
[10] Toshiba Pocket PC. http://www.csd.toshiba.com.
[11] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. Personal and Ubiquitous Computing, 7(5):275–286, October 2003.
[12] C. Borcea, C. Intanagonwiwat, A. Saxena, and L. Iftode. Self-Routing in Pervasive Computing
[13] Environments using Smart Messages. In Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (Per Com 2003), pages 87–96, Dallas-Fort Worth, TX, March 2003.
[14] C. Borcea, D. Iyer, P. Kang, A. Saxena, and L. Iftode. Cooperative Computing for Distributed Embedded Systems. In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002), pages 227– 236, Vienna, Austria, July 2002.
[15] J. Broch, D. A. Maltz, D. B. Johnson, Y.- C. Hu, and J. Jetcheva. A performance comparison of multi- hop wireless ad hoc network routing protocols. In Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, pages 85–97, ACM Press New York, NY, USA, 1998.
[16] Z. Chen, H. Kung, and D. Vlah. Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways. In Proceedings of the 2nd ACM International Symposium on Mobile Ad-hoc Networking and Computing, pages 247–250, Long Beach, CA, Oct 2001.
[17] Chisalita and N. Shahmehri. A Peerto-Peer Approach to Vehicular Communication for the Support of Traffic Safety Applications. In Proceedings of the 5th IEEE International Conference on Intelligent Transportation Systems, Singapore, Sept 2002.
[18] D. Wether all. Active Network Vision Reality: Lessons from a Capsule-based System. In Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP 1999), pages 64–79, Charleston, SC, December 1999. ACM Press, New York, NY.
[19] S. Dashtinez had, T. Nadeem, B. Dorohonceanu, C. Borcea, P. Kang, and L. I ftode. Traffic View: A Driver Assistant Device for Traffic Monitoring based on Carto-Car Communication. In Proceedings of the 59th IEEE Semi annual Vehicular Technology Conference, May 2004.
[20] R. Gray, G. Cybenko, D. Kotz, and D. Rus. Mobile Agents: Motivations and State of the Art. In J. Bradshaw, editor, Handbook of Agent Technology. AAAI/MIT Press, 2002.
[21] H. Hartenstein, B. Bochow, A. Ebner, M. Lott, M. Radimirsch, and D. Vollmer. Position-aware ad hoc wireless networks for inter-vehicle communications: the Fleetnet project. In Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pages 259–262, Long Beach, CA, 2001.

