Networked Parking Spaces: Architecture and Applications*

P. Basu and T.D.C. Little
Department of Electrical and Computer Engineering, Boston University
8 Saint Mary's St., Boston, MA 02215
Telephone: (617) 353-8042, Fax: (617) 353-1282
pbasu@bu.edu, tdcl@bu.edu

MCL Technical Report No. 07-01-2002

Abstract– Finding a parking space is a common challenge faced by millions of city-dwellers every day. As common is the revenue generation by fee and fine collection in these municipalities. Wireless ad hoc networking technologies offer a new and efficient means to both simplify the process of parking and find collection as well as extending the convenience for drivers. In this paper we describe a multi-hop wireless parking meter network (PMNET) that, when coupled with a GPS receiver, allows a user (driver) to quickly locate and navigate to an available parking space. Our solution is achieved by equipping existing parking meters with wireless radio frequency (RF) transceivers and auxiliary hardware and software. We believe that this is a compelling application that applies wireless ad hoc networking and low-power, short range RF technologies. The attractiveness of the proposal stems from the fact that such a network of nodes can function without any *fixed* wired or wireless infrastructure such as cellular or satellite networks. In this work, we model a PMNET as a special class of ad hoc networks characterized by a combination of static, immobile nodes (parking meters) and mobile nodes (vehicles). We propose scalable techniques for satisfying a mobile user's query in a distributed fashion. In particular, we make use of the static nature of the parking meters for efficient discovery and location based routing of information between them and users.

* *Proc. IEEE Vehicular Transpotation Conference (IEEE VTC 2002)*, Vancouver, Canada, September, 2002. This work was supported by the NSF under grant No. ANI-0073843. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Motivation

Imagine the frustration of a group of friends trying to locate parking in a crowded city plaza (such as Manhattan in New York City or Harvard Square in Cambridge, Massachusetts) before heading to the theater, ballgame, etc. Each passenger in the car fervently looks for a parking spot, preferably on-street, at lowest cost, and closest to the destination. If this is during a busy period, they might circle the area several times in a growing radius until they find an open spot (or open spots, if they travel in multiple vehicles). In a recent book [1], Calvin Trillin eloquently captures the psyche of a New Yorker, Tepper who enjoys sitting in his car all day reading the newspaper, just for the opportunities to turn other would-be parkers away. The most closely held secret in the story concerns the easy availability of parking meters on Madison Avenue after 7 PM.

Now imagine a scenario in which the occupants have a feature that responds to a query of "Find me the closest open spots within 100 yards of Fenway Park" or "Find me a nearby place where two spots are open" or "Find me a 50c spot near Frank's restaurant which allows parking for more than 2 hours," and the results are presented on an in-car display almost immediately. The user might select and reserve a parking spot of choice, and pay the parking fee electronically before driving to the spot. Finally, the use of GPS allows a personal navigator to show on-screen driving directions to the selected spot.

There is little precedent for our proposal. Subramanian and Katz have briefly mentioned "parking lot networks" as being an example of generic self-organizing systems [2]. In our work, we consider the architectural requirements of this application as well as algorithmic and practical issues. Specifically, we recognize that a standalone PMNET is a static multi-hop ad hoc network with nodes at fixed geographic positions after their installation. Users who have a device in their car (a PDA or a car navigation system) with wireless communication capabilities can query and communicate with the parking meter nodes wirelessly while driving on the streets. We present a scalable parking spot discovery algorithm and a simple unicast routing scheme to realize the proposed application. We also argue for a fair pricing model which is punitive to defaulters.

2 Architecture of a Parking Meter Network

Realization of the application scenario described in Section 1 requires augmentation of existing parking meter function. Parking meters will require a low-cost, low-power embedded processor and operating system. They require an inexpensive short-range infra-red sensor to detect occupancy; and a short-range, low-power RF transceiver to communicate with neighboring meters and nearby vehicles. Candidate technologies that have support for short range, low power wireless connectivity are Bluetooth [4], IEEE 802.11b (in low power mode), and Millenial Net's i-Bean [3]. In this paper, however, we do not focus on specific technologies for wireless connectivity, rather, our focus is on algorithms and techniques for enabling the smart parking space application. We defer discussion of the system hardware for elsewhere. A few kilobytes of programmable memory is also essential to store non-volatile attributes as well as volatile state related to the parking meter (a *pm-node*). A basic schematic of the hardware requirements is shown in Fig. 1.

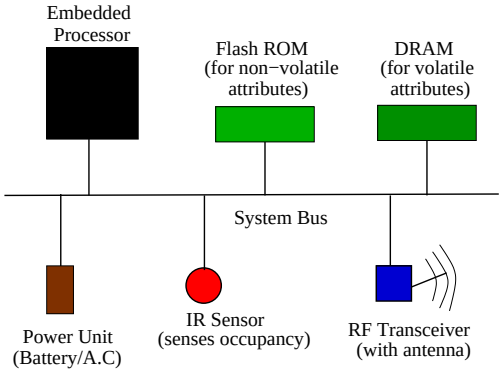

Figure 1: Basic Hardware Modules in a *pm-node*

Because a pm-node is a stationary entity, it has a fixed geographical location attribute which can be exploited during the discovery process. Instead of putting GPS receivers into each of the parking meters, meter deployment can include the use of a portable GPS receiver to permanently fix the geographical coordinates of each pm-node into its memory. Note that this location attribute may not be unique since its accuracy depends on that of the GPS receiver. However, a locally unique ID along with the GPS location attribute can serve as a globally unique ID (GUID). We refer to these and some other attributes such as "street location" as *non-volatile* attributes. Other attributes that are likely to be stored at a pm-node at any time instant include the size of the spot, its current availability, the fee for use, and the time limit. These are referred to a *volatile* attributes. Wirelessly enabling these devices results in their connection to a virtual network of devices thus allowing them to function as resources that can be discovered remotely, instead of physically. Fig. 2 shows a schematic of a wireless multi-hop PMNET that can be envisaged to exist in Manhattan, New York City.

In addition to supporting drivers seeking parking, PMNETs can be used by the municipality to simplify and optimize revenue generation from fees or penalties. Although a detailed game-theoretic analysis of the distinct goals of individuals vs. municipalities will yield a variety of approaches to parking rules, it is beyond the scope of this paper. Some representative queries that might be posed by the fee collector or driver are:

Q0: Which streets in the locality have vacant spots right now?

Q1: Which meters are about to expire within 200 yards of my location (GPS coordinate specified)?

Q2: Is there any parking spot scheduled to become available soon on Main St. until midnight?

Q3: Locate all vehicles that reside in expired spots.

A local query processing software module structures a query in a suitable query language before it is dumped in the network via pm-nodes that are reachable from the user's current location as shown in Fig. 2. Responses to a query can be sorted by user

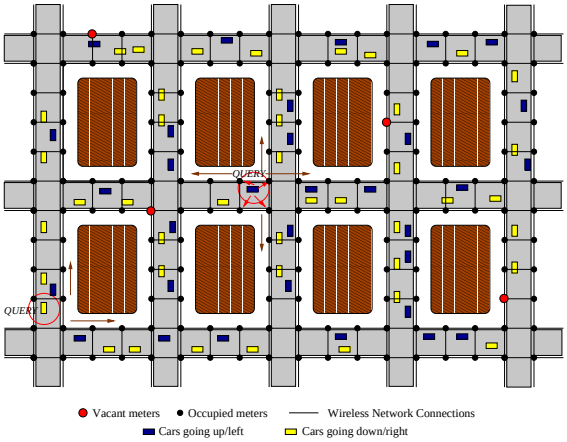

Figure 2: A Parking Meter Network in Manhattan, New York City

defined criterion before presentation to the user. The process can be visualized as querying a *physically distributed* database formed by pm-nodes possessing dynamically changing attributes. In this paper, we do not address the details of structuring a user query or a response since our focus is on algorithms for discovery and scalable updates which are described in the next section.

3 Algorithms for Parking Spot Discovery

The salient nature of a PMNET is that the pm-nodes are immobile but the content served by them changes dynamically. In addition to the static, non-volatile attributes such as geographic location, it is this dynamic content (e.g., is the spot available now?) which is often of great interest to the users. Most mobile user queries are such that both static and dynamic attributes of pm-nodes will be needed to resolve them. The algorithms which we later describe in this section take these factors into account.

Each pm-node in the system will have several components as described in Fig. 3. The *query processing* module receives structured queries from mobile users, determines if a query matches the current values of its relevant attributes and takes appropriate action. If a query can be satisfied locally, the pm-node responds to the user using the underlying unicast routing protocol described in Sec. 3.3. In the basic version of the system, if a pm-node is unable to satisfy the query itself, it *rebroadcasts* the query to its neighbors. In a more sophisticated version of the system as proposed in Sec. 3.1, query processing is performed in a hierarchical fashion on pm-nodes which act as ClusterHeads (CH) for a set of other pm-nodes.

The *status update processing* module receives status updates (values of volatile attributes) from other pm-nodes and updates the local data structures accordingly. These data structures need to be queried repeatedly while responding to user queries. Fundamentally, there are two different approaches for performing dynamic status updates in a PMNET. One approach is to propagate status updates throughout the entire PMNET by flooding or network-wide broadcast. In that case, when a user U queries an adjacent pm-node, the latter has complete

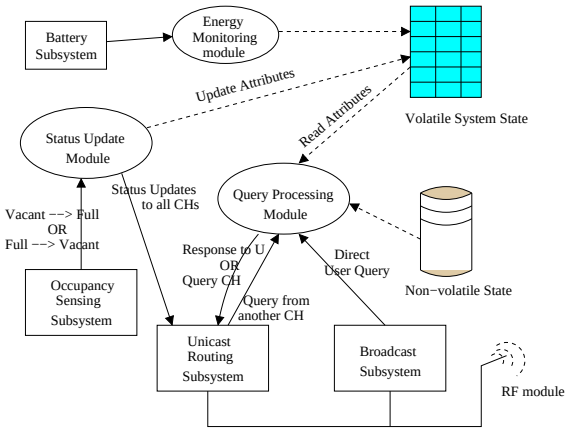

Figure 3: Basic System Architecture

knowledge about the *current* status of all other pm-nodes in the PMNET, and U's query can be satisfied immediately (with minimum delay). Obviously, flooding can consume a large amount of wireless network bandwidth and may not be desirable in dynamic scenarios where status updates are very frequent.

At the other end of the spectrum is an approach which is completely demand driven or reactive, and the pm-nodes do not flood any status updates across the network: U's query is broadcast throughout the PMNET and the suitable pm-nodes reply to U via unicast. This approach can save network bandwidth and storage overhead but increases delays. The third approach is a hybrid scheme which attempts to minimize user latencies as well as bandwidth usage for status updates. In Sec. 3.1 we propose one such scheme to improve scalability of the update/discovery process.

3.1 Scalable Cluster-based Architecture

We propose a hybrid cluster-based approach which utilizes the natural geographical clustering existing in PMNETs for facilitating quick status updates and speeding up the discovery process. Every pm-node located on a particular street in a locality possesses a unique *street* attribute which can be used to *cluster* a set of pm-nodes together. The radio transmission range is assumed to be large enough so that the pm-nodes on opposite sides of a street can communicate with each other.

Election of Clusterheads In our approach, exactly one of all pm-nodes with the same *street* attribute serves as a clusterhead (CH) of the set. The CH can be elected to its role during installation or can automatically be selected by the following leader election process among the relevant pm-nodes: each pm-node broadcasts to its neighbors a $(ID, street, val)$ tuple where *val* corresponds to a measure of its computing resources[1]. Every pm-node then periodically neighbor-broadcasts the maximum *val* from the same *street* that it has heard so far. The pm-node with maximum *val* is chosen as the CH of the particular street by

[1]For example, *val* can be a function of the amount of CPU and memory resources, as well as the remaining battery power (if applicable)

every pm-node therein. Ties are broken by IDs. This process consumes little bandwidth as only local communication happens, and that too after installation, failure, or repair of a pm-node. It can also be triggered when the existing CH's battery is about to run out, i.e., if the pm-nodes run on battery power.

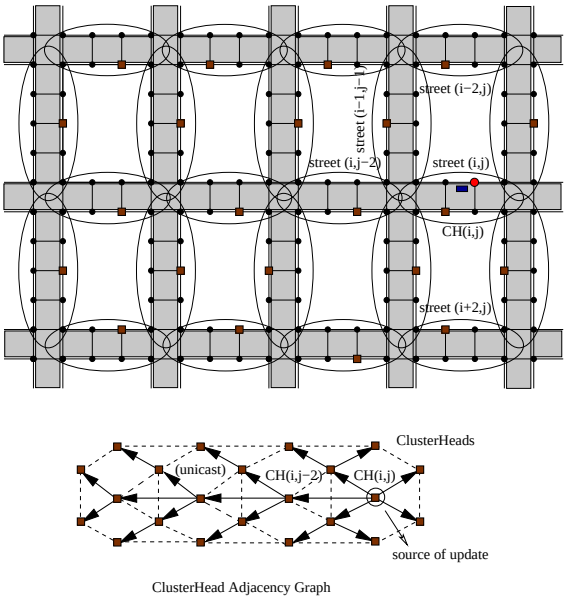

Figure 4: Scalable Dissemination of Status Updates

Dissemination of Status Updates A non-CH pm-node of street X upon hearing about the election of a CH in a neighboring street Y informs its own CH about it. Thus every CH comes to know about the CH's in the neighboring streets. This is useful for efficient dissemination of current status information as the CH nodes in the entire locality form an overlay graph and the dissemination of status updates is performed only from the affected CH to the rest of the CH's by means of unicast. Specifically, each pm-node reports a change in its status (vacant $\rightarrow$ full, or vice versa) to its CH which then floods this incremental information to other CH's in the PMNET. As a result of this, a user node only needs to query a nearby CH about the attributes/status of pm-nodes in other streets while discovering a spot.

Fig. 4 illustrates the process of performing a status update with a specific example. When a car leaves a parking spot on street(i,j), the corresponding pm-node (marked by a red filled circle) informs its clusterhead, CH(i,j) about this change. Now, CH(i,j) needs to send a status update to all other CH's in the locality. This can be accomplished in a number of ways. In the first approach, a CH attempts to send reliable updates to its neighboring CH's over unicast (utilizing the underlying unicast routing substrate described in Sec. 3.3) which then re-forward the updates further downstream. To determine who to forward the updates to, a Clusterhead Adjacency Graph (CHAG) is constructed from the topology of street intersections[2]. Now, unicast forwarding needs to be done only along the

[2]Two CH's are adjacent in a CHAG if and only if their corresponding streets intersect with each other.

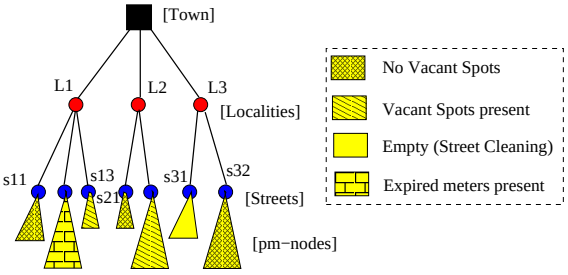

Figure 5: Hierarchical Name Trees

edges of a minimum spanning tree of CHAG rooted at CH(i,j). In the second approach, the dissemination problem is treated as a multicast routing problem where all the CH's are sinks in a multicast group and CH(i,j) is the source. This approach however assumes that all pm-nodes support multicast. The CHAG-based unicast approach is also known as *end-system multicast* [5] as it does not assume any multicast support in the intermediate, non-CH pm-nodes. Moreover, it is easier to guarantee reliability in updates in this approach in contrast with doing that in the pure multicast based solution.

A CH can choose to disseminate only a salient part of the status update to other CH's, e.g., if a pm-node in street S becomes vacant, CH(S) can inform other CH's of only the fact that there is an additional vacant parking spot available, instead of furnishing the details of the spot. Queries similar to Q0 can be readily answered with such condensed information. However, when a user in street X issues a more involved query, the designated clusterhead CH(X) may not be able to answer the query right away from this information, but it can query the relevant CHs (which have reported vacant spots earlier) about the details of the parking spots at the current time. This approach is advantageous as a CH does not need to store the status of every pm-node in the city. Instead it stores information only at a coarse level of granularity. The obvious penalty is that of additional time taken to discover a spot. This demonstrates that there are several time-space trade-offs in the design of such a system. The best approach in a particular scenario would depend on factors such as the predominant nature of queries and their frequency, the frequency of changes, size of the PMNET etc.

3.2 Hierarchical Naming

We propose the use of hierarchical name-trees (similar to MIT's Intentional Naming System [6]) for answering queries such as Q2. Every pm-node has a unique GUID, "Street" and "Town" attributes, and other information such as nearby landmarks, which are programmed set at the time of installation. When CH's exchange information about pm-nodes, they can create tree-like data structures with actual pm-node records as leaf nodes, and "towns", "streets" and "localities" as *virtual* nodes in the hierarchy. Fig. 5 depicts a possible instance of a *name-tree* which is maintained at every clusterhead pm-node in a street. The triangular blocks hanging from the "Street" nodes depict the individual pm-nodes and have been used as a visualization aid. The relative sizes of triangles indicate the numbers of pm-nodes existing in each street. The various visual patterns inside them characterize the status of parking on a particular street as a whole at a particular instant of time. Such hierarchical presentation of information is possible because of the hierarchical geographic clustering of

the pm-nodes.

When the status of a pm-node changes, it informs its CH which reflects the change in its name-tree. It also *floods* the update to the other CH's using the mechanisms described in Sec. 3.1. When a query such as Q2 is issued by the user, it gets routed to the nearest CH pm-node and the query processing module on that pm-node searches under the "Street" metanode in its name-tree. After finding the "Main St." node, it searches for suitable parking meters (children of that node in the attribute tree) which can satisfy the request. Note that the status updates need to occur on a detailed/individual pm-node basis and not just at an aggregated level such as "Main St. has a 2 vacant spots now." If aggregates updates are being used, then the current CH has to query the CH(Main St.) pm-node to get more details.

The name-tree shown in Fig. 5 can also be visually presented to a user of the system when he/she submits a general query "Display the overall status of parking in the area". It can be an invaluable tool for the fee collectors who then would not have to go to each individual parking meter and verify if the meter has expired or not. Ideally, a fee collector can on his/her PDA, zoom in on the streets which have expired meters and navigate to the meters individually. This can save significant effort and resources of the municipality.

Using additional intelligent data structures which are not described in detail in this paper, the number of operations necessary for querying and updating status information can be minimized.

3.3 A Hybrid Unicast Routing Architecture

After a user discovers a set of candidate spots, he/she may initiate communication with one or more of them to check the validity of the status or to reserve a spot, if possible. A unicast routing infrastructure is needed to facilitate this. Because pm-nodes are static and possess a fixed GPS location attribute, the topology of the static part of PMNET changes only when new pm-nodes are installed or when they break down or have a battery outage. Hence, a dynamic MANET routing protocol is unnecessary in its full generality for routing between two pm-nodes.

We propose the use of position based routing schemes [7] for routing between pm-nodes. Such schemes are ideal in static scenarios since they do not need to exchange packets or store full routing table information. In position based routing, packets are routed hop-by-hop in a greedy fashion by forwarding to the neighboring pm-node which is *closest* to the destination location coordinate. In regions of the network where such a greedy path does not exist (i.e., the only path requires that one move temporarily farther away from the destination), position based routing schemes recover by forwarding in *perimeter mode*, in which a packet traverses successively closer faces of a planar subgraph of the full radio network connectivity graph, until reaching a node closer to the destination, where greedy forwarding resumes. An example of a position based routing protocol which behaves in this manner is Greedy Perimeter Stateless Routing [8].

Now, since users of a PMNET are usually mobile, their positions change continuously until they find a parking spot. Hence, the location coordinates of a user at the time of issuing the query may not be exactly the same when a pm-node issues a response. Therefore, static position based routing needs to be augmented slightly in order to facilitate routing in the

reverse path (from pm-node to U). Routing table entries with reverse paths to U are setup at the intermediate pm-nodes during the forward routing (greedy location-based forwarding) phase, thus enabling reverse routing of messages to the user at a slightly later time instant.

However, even the aforementioned protocol augmentation is inadequate for handling routing from a pm-node to a user or between two users at a much later instant of time since the reverse routing table entries may have become stale owing to the movement of the user from his/her earlier location. The former mode of communication may be needed when a particular pm-node wants to inform a user about competitors for a parking spot or if it wants to warn the user about the expiring time. The latter mode can be used by anonymous users for negotiations about a spot. In order to solve this problem, we believe that the use of a standard on-demand MANET routing protocol which does not depend upon any position information is suitable.

3.4 Competition between Vehicles

So far, we have described a system where one user is trying to discover a parking spot. The scenario becomes much more interesting when multiple users are looking for parking in a crowded area. Competition between drivers occurs when multiple vehicles simultaneously receive an indication of an available spot. In a baseline version of the system, vehicles which do not ultimately get the spot, query the system again from their new locations hoping to find a new spot. However, by including more state information, the system can be improved further. For example, if vehicles v_1 and v_2 are competing for the same spot P, the corresponding pm-node can inform one about the other's current location. If v_1 is physically much farther from P than v_2, it cancels its query and searches for another spot. In the process, v_1 saves time by not driving to P.

In a more complex instance of our proposal, we consider the dynamic resource allocation problem with objectives of maximizing utilization and fairness and minimizing delays, under stochastic inputs. This is a very hard problem to solve, especially in an online scenario in which requests arrive frequently, and PMNET nodes must achieve a good allocation quickly in a cooperative manner. We do not address this problem in this paper and leave it as future work.

3.5 Fee Structure and User Behavior

Although we have not addressed fee models or rules in this paper, we fully expect that driver behavior is coupled with the fee structure and the competing goals of the municipality in system design. For example, if a driver violates the time limit, the system might automatically debit his/her account instead of issuing a ticket. Our proposal in amenable to such variants, including an increase in the parking fee in a non-linear fashion after the parking meter expires. For example, the 2 hour fee could be \$2, but if one stays for 3 hours, it could increase to \$10, and then to \$30 after 4 hours. The fee structure could thus determine user behavior and vice versa.

4 Conclusion

In this paper, we proposed an application for a wireless parking meter network in which drivers can quickly locate and navigate to available parking spaces. Our solution requires wirelessly enabling existing parking meters yielding a self-organizing, self-managing, and distributed system relying on techniques in mobile ad hoc networking. In this paper, we have described in detail algorithms for efficient discovery of vacant spots as well as scalable dissemination of status updates which attempt to keep both discovery latency and the bandwidth consumed for status updates low. We believe that the proposed network is both economically and technologically feasible, although the details of a complete solution are not shown here.

References

[1] C. Trillin, "Tepper Isn't Going Out," *New York: Random House*, 213 pp. New York Times book reviews at URLs:
http://www.nytimes.com/2002/01/14/books/14MASL.html
http://www.nytimes.com/2002/02/12/books/12TRIL.html

[2] L. Subramanian and R. Katz, "An Architecture for Building Self-Configurable Systems," *Proc. MobiHoc 2000*, Boston, MA, August 2000.

[3] Millenial Net's i-Bean. *http://www.millenial.net*

[4] Bluetooth SIG. *http://www.bluetooth.com*

[5] Y. Chu, S. G. Rao and H. Zhang, "A Case For End System Multicast," *Proc. ACM SIGMETRICS 2000*, Santa Clara,CA, June 2000, pp 1–12.

[6] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," *Proc. ACM SOSP 1999*, Kiawah Island, SC, December 1999.

[7] M. Mauve, J. Widner, and H. Hartenstein, "A Survey on Position-Based Routing in Mobile Ad-Hoc Networks," *IEEE Network*, November 2001.

[8] B. Karp and H.T. Kung, "Greedy Perimeter Stateless Routing for Wireless Networks," *Proc. ACM MobiCom 2000*, Boston, MA, August, 2000, pp. 243–254.