

bold, 16pt)

A Location Management Protocol for Hierarchically Organized Multihop Mobile Wireless Networks*

Kamal K. Kasera[†]

Dept. of Computer Science, University of Massachusetts, Amherst, MA
email: kkasera@cs.umass.edu

Ram Ramanathan, *Senior Member, IEEE*

Advanced Networking Dept., BBN Systems and Technologies, Cambridge, MA
email: ramanath@bbn.com

Abstract

We consider the problem of location management in a large, hierarchically organized multihop mobile network, where the switches comprising the infrastructure as well as the endpoints (terminals) may be mobile. Such networks are common in applications such as battlefield networks, disaster management, law enforcement, etc. We present a novel network architecture and location management protocol that accommodates switch mobility and, using a new concept called “roaming level”, allows the updating procedure to be controlled on a per endpoint basis. We discuss simulation results on the variation of performance with the speed, roaming level and call frequency of endpoints. Our results show that at high endpoint speeds or low call frequencies a significant decrease in the control overhead can be obtained in comparison to traditional schemes.

1 Introduction

Technological advances are continuing to move the concept of universal personal communications from the realm of speculation to that of reality. A key piece in the ability to provide ubiquitous untethered communications is *location management*, which is the set of mechanisms used to determine where an endpoint is with respect to the network infrastructure. In abstract terms, location management provides a time-varying mapping between the endpoint *identifier*, that is, *what* an endpoint is called, and the endpoint *address*, that is, *where* an endpoint is located.

In this paper, we consider the problem of location management in the context of a hierarchical *multihop mobile* network containing an arbitrary number of hierarchical levels. By a multihop mobile network we mean a network in which not only the endpoints (terminals) *but also the switches (routers, base stations) may be mobile*. In our architecture, endpoints have one-hop wireless connectivity to some switch, and the switches have multihop wireless connectivity to one another. This is a hybrid of the cellular, and ad hoc [1] or packet radio network [2] architectures.

Switch mobility allows for instant-infrastructure and rapidly-deployable networking, which is crucial for applications such as battlefield networks [3], disaster management, law enforcement, and intelligent vehicular highway systems. Although several problems such as routing, self-organization, link control etc. must be solved in order to establish communications in a multihop mo-

bile wireless network, our focus in this paper is on the problem of keeping track of the endpoint location, in particular the switch to which the endpoint is affiliated. We have addressed some of the other issues as part of a larger ongoing effort at BBN for multimedia support in mobile wireless networks (MMWN), and refer the interested reader to [4] for details. In turn, the MMWN effort is part of the Global Mobility (GloMo) program, involving a number of organizations sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA) whose objective is to make available technologies for rapidly deployable information systems for defense requirements [5].

Previous approaches to location management in wireless networks may be broadly classified into static and dynamic strategies. In a static strategy, there is a predetermined set of locations (e.g., cells) at which location updates may be generated [6, 7]. In a dynamic strategy, an endpoint determines when an update should be generated based on its movement [8, 9]. A hierarchical location updating strategy is described in [10]. Location management based on these and other ideas exist in the Personal Communication Services (PCS) systems such as IS-41 and GSM, and in the Internet, e.g., the Mobile-IP protocol. For a survey of existing location management strategies, and routing in mobile networks in general, the reader is referred to [11].

Our work differs from most previous location management works in three significant ways. First, it is designed to operate in a hierarchical network whose infrastructure is itself mobile and constantly re-organizing in response to mobility. This environment brings up new scenarios not considered in previous works – for instance, an endpoint’s address may change even if the endpoint does not move because of autonomous network reorganization; and location servers, being mobile, may move and become permanently inaccessible. We note, however, that all of our results are applicable to static infrastructure networks as well, since they are a special case when there is no switch mobility. Second, it employs an updating procedure that is controlled based on the hierarchical levels rather than geographical boundaries (such as location areas). Third, it allows for the selection of updating versus paging tradeoffs on a per endpoint basis without using geographical boundaries.

2 Network Architecture

Our architecture consists of two functional elements, or *nodes*: *switches* and *endpoints*. Only switches can forward (or route) packets and only endpoints can be sources of or destinations for packets. Switches as well as endpoints can be mobile.

*Partially funded by United States Defense Advanced Research Projects Agency Contract no. DAAB-07-95-C-D156

[†]Work done during summer internship at BBN Systems and Technologies

The clustering hierarchy used in our architecture is illustrated in Figure 1. At the lowest level, endpoints affiliate with switches to form a *cell*. At higher levels, switches may be clustered to an arbitrary number of levels. The hierarchy imposes a straightforward parent-child relationship between clusters depicted on the right (switches are omitted). The *level* of a cluster is the maximum, taken over all switches, of the number of parent relations a switch has to invoke to reach that cluster. By definition, a switch and the endpoints affiliated with it form a level-0 cluster. At the highest level of the hierarchy, all clusters are contained in an imaginary “universal” cluster. For example, in Figure 1, clusters C,D,E,F are at level 1, clusters A,B at level 2 and the universal cluster U at level 3. Although the example in the figure shows a balanced and binary tree, neither balanced nor binary is a requirement.

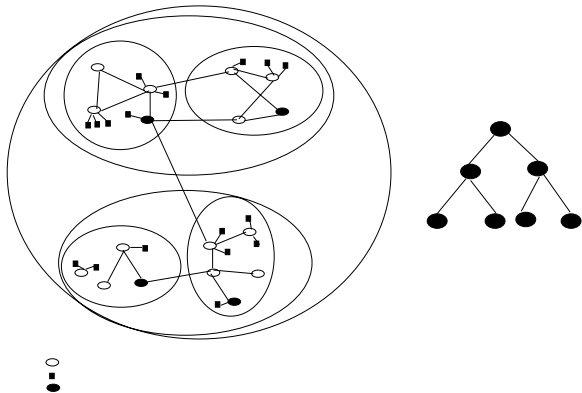


Figure 1: The clustering hierarchy used in our architecture.

Every switch and endpoint is assumed to have a globally unique identifier, referred to as the *switch-id* and the *endpoint-id* respectively. Every cluster except the zeroth level clusters (switches) acquire a *cluster-id* unique among its siblings. By definition, the cluster id of a zeroth level cluster is the corresponding switch id. The *hierarchical address* of a cluster C_k is the sequence of cluster ids $C_1, C_2 \dots C_k$, where cluster C_{i-1} is the parent of cluster C_i , $1 \leq i \leq k - 1$. We will use the dotted notation for hierarchical addresses. For example, if cluster C is a child of cluster B and cluster B is a child of cluster A, then the hierarchical address of C is A.B.C. In our descriptions, we often omit the word “hierarchical” and simply use *address*. A *switch address* is the address of the cluster to which a switch belongs, suffixed by the switch id. The *endpoint address* of an endpoint is the same as the address of the switch with which the endpoint is affiliated. The node identifiers are configured and do not change while the node addresses are autonomously acquired and may change with time.

A *prefix (suffix)* of an address containing n terms is a portion of the address containing the first (last) k terms, $1 \leq k \leq n$, of the address. For example, given an address A.B.C.D, A.B and A.B.C are among its prefixes, and B.C.D and D are among its suffixes.

In order to provide location management, each cluster (including clusters at levels greater than one) has a *location manager* (LM). The LM for a cluster is a switch, elected from among all of the switches within the cluster. For instance, in our current implementation, the LM is the switch with the lowest id amongst all switches in the cluster. A switch can perform the LM functionality at multiple levels. For example, in Figure 1, there is an LM in cluster F that is a location manager for both F and B.

3 A Hierarchical Location Management Protocol

Location management involves the maintenance of a dynamic distributed database whose entries are *associations* between endpoint identifiers and endpoint addresses. In our context, an entry in this database changes if and only if one of the following three events occur:

1. An endpoint reaffiliates, thereby changing its address. This may involve a change in any suffix of its address. For example, endpoint q in Figure 1 has an address of U.A.C.R. If it moves to switch V, then suffix R will change; if it moves to switch T in cluster D, suffix C.R will change to D.T.
2. The switch to which the endpoint is affiliated moves to a different cluster, but the endpoint continues to be affiliated to the switch. Here, although the endpoint does not change cells, its address changes because its switch’s address changes.
3. Cluster reformation occurs, that is, a cluster splits into two or merges with another cluster. Since the cluster address itself may change, there may be a change in the endpoint address.

For the first part of this section and for all of our experimental results, we focus on responses to reaffiliation (item 1 above). Items 2 and 3 will be discussed briefly in section 3.3.

We divide location management functionality into two components: *location updating*, that is, updating the distributed database with the new association, and *location finding*, that is, given an endpoint id, obtaining the corresponding address. A key feature of our location management scheme is that it allows for control on updating versus finding on a *per endpoint basis*, thereby accommodating endpoints with diverse call frequencies and endpoint speeds. In our approach, each endpoint is associated with two parameters as defined below.

Definition 3.1 *The roaming cluster (RC) of an endpoint is the lowest-level cluster containing the endpoint such that an update is triggered if and only if the endpoint exits this cluster.*

Definition 3.2 *The roaming level (RL) of an endpoint is the hierarchical level of its roaming cluster.*

For example, in Figure 1, if the RC of endpoint m is cluster E, then no updates will be sent as long as m remains within E, even if it reaffiliates to another switch, for example, switch P. If it goes outside of E, say, to cluster F, then a location update will be triggered. The RL of m in this case is 1.

For convenience’s sake we have defined the roaming cluster first, but it is in fact the roaming level that is configured for an endpoint. The roaming cluster of an endpoint is derived from its roaming level and its current location. In particular, the RC is the cluster that contains the endpoint, and is at a level equal to the roaming level of the endpoint. For instance, suppose endpoint q in Figure 1 is configured with a RL of 2. Then its RC is cluster U.A. Suppose now that q moves to within cluster E. Then its RC would be cluster U.B. Thus, when an endpoint exits its current RC, it will get a new RC based on its RL. An endpoint’s RL may be changed dynamically in response to its call frequency and speed. In general, the more mobile the endpoint, the higher should be its RL. For this paper, however, we shall restrict ourselves to statically configured RL and focus on the updating and finding schemes and their performance.

We note that if the RL of an endpoint is 0, then its RC is the current cell, and an update will be triggered for every reaffiliation

– this is equivalent to an “always update” scheme. Similarly, if the RL of an endpoint is greater than or equal to the level of the universal cluster, then no updates are triggered – this is equivalent to a “never update” scheme.

3.1 Location Updating

If an endpoint changes its address due to reaffiliation then a location *UPDATE* message will be sent by the endpoint whenever it exits the current roaming cluster. An *UPDATE* is also sent when the endpoint is activated and affiliates for the first time. As part of the (re)affiliation process, an endpoint obtains its new hierarchical address. By comparing its old address and its new address, an endpoint determines whether or not it has exited its roaming cluster. In particular, if the length of the differentiating suffix is greater than its roaming level, then the endpoint has exited its roaming cluster. For example, in Figure 1, suppose that the RL of endpoint m is 1. Its current address is U.B.E.S, and RC is U.B.E. If m moves and reaffiliates with nearby switch P, thereby obtaining a new address U.B.E.P, then the differentiating suffix is P (of length 1) and hence the endpoint concludes that it has not exited the RC and does not send an update. However, if m reaffiliates with switch Q in cluster F, thereby obtaining a new address U.B.F.Q, then the differentiating suffix is F.Q (of length 2) and hence the endpoint concludes that it has exited its RC and sends an update.

The *UPDATE* message is generated by the endpoint itself, and contains four fields: its own endpoint id, old address, new address, and roaming level. The *UPDATE* is sent to its cellhead, which initiates the forwarding of the message to the appropriate location managers as follows. An LM receiving an update message first trims the last l terms of the address, where l is its level, from the old and the new addresses carried in the update message and compares the resultant addresses. If they are not equal, then the message is forwarded on to the parent LM, which repeats this check, and so on until it reaches an LM such that the comparison results in equality. Such an LM is in the *lowest common cluster* (LCC) in which the movement took place.

Each LM that receives an *UPDATE* message creates an association entry for the endpoint if there is none, or changes the pointer for the association to point to the LM from which it received the *UPDATE*. The LM in the LCC additionally sends a *CANCEL* message to the LM pointed to previously. The purpose of the cancel message is to delete the now invalid entries for the endpoint’s location. The cancel message trickles down the tree, following the previously installed pointers, deleting an association entry for the endpoint at each LM, if found. Entries that are not deleted expire and are flushed.

3.2 Location Finding

Location finding is the process of obtaining the address of an endpoint, given its endpoint id. Our location finding procedure has two components: *location query*, which involves the following of location pointers in LM’s to reach the LM containing the association, and *location paging*, which involves the polling of switches within a given cluster in search of an endpoint. Paging is required because (and only because)¹ an endpoint may have an RL > 0 .

A node wishing to obtain the address of an endpoint originates a *QUERY* message which contains three fields: the id of the endpoint whose address is required, its own id, and its own address. If

the *QUERY* message is originated by an endpoint, then it is sent to the switch with which the endpoint is affiliated. Thus, we begin the description of the finding process from a switch in possession of the *QUERY* message.

The switch (which is also an LM for its own cell) first searches its affiliation list to see whether the target endpoint is in its own cell. If it is, then the location finding procedure terminates. If not, the switch forwards the *QUERY* to its parent LM. An LM receiving a *QUERY* message searches its location database for an association corresponding to the target endpoint. If an entry is found, the *QUERY* is forwarded to the LM in a child cluster/switch as pointed to in the entry. If no entry is found, then it is forwarded on to its parent LM. Thus, the *QUERY* makes its way up the tree until it finds an entry for the endpoint and then down the tree until it reaches a level-0 LM, i.e., the final switch. There are two possibilities:

1. The final switch contains the endpoint. This is the case if the target endpoint has an RL of 0, forcing an update for every movement. In this case, the switch in question creates a location *REPLY* message containing the id and address of the target endpoint and sends it to the originator (requestor) address present in the *QUERY* message.
2. The final switch does not contain the endpoint. This is the case if the target endpoint has an RL of at least 1. The location *QUERY* has followed the pointers from its most recent update but the endpoint has since moved to a new location but not updated. In this case, a location *PAGE* message is initiated by the switch. The *PAGE* contains the same information as the *QUERY* and is flooded throughout the level- r cluster containing the endpoint, where r is the roaming level of the endpoint being paged. In other words, the paging is done in the current roaming cluster of the endpoint. Note that the roaming level of the endpoint is part of the database entry in the LM, and hence r is easily obtained. Upon receiving a page message, a switch checks to see if it contains the endpoint. If it does, then it sends a *REPLY* message to the requestor as described in item 1.

3.3 Switch Mobility

Thus far, we have ignored the effects of a switch being mobile. Mobility of switches causes two problems that must be addressed. First, a switch and the endpoints affiliated with it may all move to a new cluster. Second, the clustering hierarchy may autonomously undergo changes such as two clusters merging into one or a cluster splitting into two. In both cases, there is no reaffiliation but nonetheless an update needs to be sent. The cluster splitting and merging mechanisms themselves are beyond the scope of this paper, and can be found in [4].

Cluster Changing. When a switch changes clusters, it obtains, through mechanisms not described here due to space constraints (see [4] for details), a new hierarchical address. It then prepares an “aggregated” update message that contains the new address and the list of endpoints affiliated with it. The handling of this message is similar to the update message generated due to reaffiliation (section 3.1), with the difference that the entry for each of the endpoints in the list is changed at the appropriate LM(s).

Cluster Splitting. When a cluster splits into two, the LM of the original cluster will remain in one of the clusters and the

¹Paging may also be required if the movements are so rapid that by the time the updating process is completed the endpoint has already moved to a new location. Handling such cases requires predictive techniques, which are beyond the scope of this paper.

other cluster will get a new LM. This new LM, however, will be “empty” to begin with, i.e., it will not have the associations for the endpoints in its cluster. In our protocol, the old LM sends a *DB-INIT* message to the new LM. This message contains the old LM’s entire association database. The new LM thus has the associations for switches in its cluster, and more. The excess information will be flushed at the next expiration timer since they will not be refreshed.

Cluster Merging. When a cluster merges with another cluster, one of the LM’s will give up its role (since a cluster has only one LM). However, the surviving LM will only have partial information about the new cluster, i.e., it will only have the associations for endpoints that were in the portion of the cluster before the merge. In our protocol, the resigning LM sends a *DB-HANDOFF* message to the surviving LM which contains the associations that it holds. These associations will be added to the associations already present in the surviving LM.

4 Comparative Analysis

In this section, we derive an expression for the conditions under which our scheme will perform better than an “always-update” scheme (i.e. our scheme with $RL = 0$).

Let Θ_i denote the total overhead per update associated with the location management of an endpoint using scheme i . Now, $\Theta_i = n_u \cdot U_i + n_c \cdot Q_i$, where U_i is the overhead associated with a single update, Q_i is the overhead associated with a single query, and n_c and n_u are the number of calls and cell affiliation changes respectively. Let *AL* denote the always-update scheme, and let *RC* denote our scheme where updates are sent only when the roaming cluster is exited. The always-update scheme will perform worse than our scheme if $\Theta_{AL} > \Theta_{RC}$, i.e.,

$$n_u \cdot (U_{AL} - U_{RC}) + n_c \cdot (Q_{AL} - Q_{RC}) > 0 \quad (1)$$

The update and query overheads depend on many factors. Let

RL = Roaming level of the endpoint.

p_e = Probability of a move (reaffiliation) exiting RC

$\Phi(h)$ = Overhead of paging in a cluster at level h .

$\Upsilon(h)$ = Overhead of updating LM’s upto a level h .

The difference in update overhead between the two schemes is due to the fact that the always-update scheme records all updates while the RC based scheme only records when the endpoint exits RC. An update can be due to an intra-RC move with probability $1 - p_e$, and thus,

$$U_{AL} - U_{RC} = (1 - p_e) \cdot \Upsilon(h') \quad (2)$$

Here, h' is the average height through which the update travels within a roaming cluster of height RL . Let $p(i)$ denote the probability that the update travels up i levels. Assuming, for some constant k that $p(i) = k \cdot p(i + 1)$, $i = 1, \dots, RL - 1$, h' is given by

$$h' = 1 + \frac{1}{k - 1} - \frac{RL}{k^{RL} - 1} \quad (3)$$

The difference in querying overhead is due to the fact that our scheme has to page for the endpoint within an RC. Thus,

$$Q_{AL} - Q_{RC} = -\Phi(RL) \quad (4)$$

Substituting equations 2 and 4 in equation 1 and rearranging, our scheme performs better if and only if

$$\frac{n_u \cdot (1 - p_e)}{n_c} > \frac{\Phi(RL)}{\Upsilon(h')}, RL > 0 \quad (5)$$

where h' is as given in equation 3. To apply this inequality to real-life situations, an estimate for p_e , Φ , and Υ would be required. A network designer can often make these estimates based, for instance, on historical data for p_e and on the network topology and clustering hierarchy for Υ and Φ .

5 Experimental Results

Our simulations were done in the Maisie simulation language, a C-based simulation language that can be used for sequential and parallel execution of discrete-event simulation models.

An instance of a multihop mobile wireless network is generated by placing a set of S switches and E endpoints in a L meters by L meters square area. The position of each switch and endpoint in this area is randomly and independently chosen, with uniform distribution along each axis. Each switch and endpoint is given a communication range of R meters. Free-space propagation is assumed with threshold cut off, i.e., two nodes can communicate with each other if and only if their euclidean distance is less than or equal to R . We note that a shortcoming of this model is its inability to capture the effects of barriers, foliage, multipath interference etc. – nonetheless, we have adopted this model because of its simplicity and the fact that the use of real-life propagation models make the simulation very time-consuming.

The mobility of each node follows a novel *Extended Random Walk* (ERW) model. Here, each node moves in the same direction with a constant speed for D seconds, and then picks another direction randomly, moves in this direction with a constant speed for D seconds, and so on. For our study, we have assumed a 2-level hierarchical clustering. Thus, the roaming level RL can be 0, 1, or 2, with 2 being a trivial case where no updates are sent. We let switches as well as endpoints to be mobile. All switches are assumed to have the same speed V_s and all endpoints are assumed to have the same speed V_e . Calls are generated randomly between endpoint pairs with a frequency of C per minute.

Each experiment must choose a point in the 9-dimensional parameter space of simulation model parameters S , E , L , R , V_s , V_e , RL , C , and D identified above. Since it is prohibitively time-consuming to study the dependence on all of the 9 parameters, we keep all but V_e , RL and C constant and let these three vary as follows.

- *Constant parameter values.* $S = 20$; $E = 4$; $L = 2000$ meters; $V_s = 1$ m/s (3.6 km/hr); $D = 100$ sec; $R = 800$ m.
- *Variable parameter ranges.* $V_e = 5$ m/s (18 km/hr) ... 20 m/s (72 km/hr); $C = 0.1$... 1.4 calls per minute; $RL = 0, 1$.

The *total overhead* incurred by the location management mechanism is plotted against call frequency C in Figures 2 and 3, and against endpoint speed V_e in Figures 4 and 5. The total overhead is the total number of messages transmitted anywhere in the network due to updating or finding procedures². The *DB-HANDOFF* and *DB-INIT* messages are not counted as part of the overhead. Not surprisingly, the plots show that the overhead increases with both call frequency and speed, with ranges in which $RL=0$ is better or worse than $RL=1$.

Figure 2 shows that for call frequencies of less than approximately 0.25 calls per minute the use of $RL=1$ incurs less overhead. At 20 m/s speeds (Figure 3), the cutoff point increases to 0.75 calls

²For instance, if an update message takes 4 hops to reach its LM then the overhead due to the update message is 4.

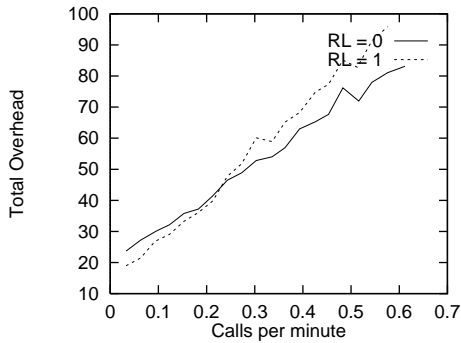


Figure 2: Overhead vs call freq, speed = 15 m/s

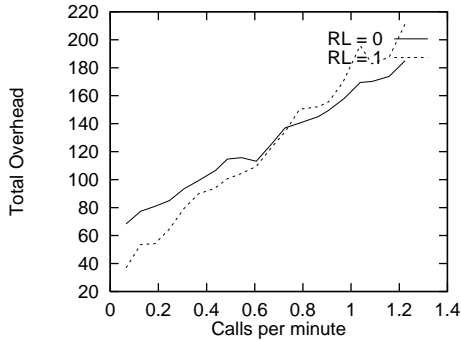


Figure 3: Overhead vs call freq, speed = 20 m/s

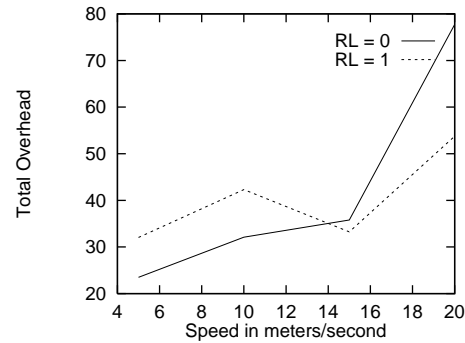


Figure 4: Overhead vs speed, call freq = 0.15 per min

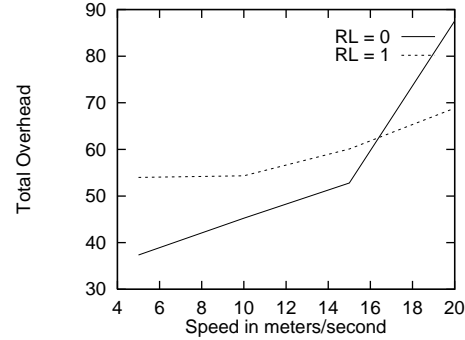


Figure 5: Overhead vs speed, call freq = 0.3 per min

per minute, and at 0.1 calls per minute, use of RL=1 can result in a savings of over 38 % over an “always update” scheme (from Figure 3).

Figure 4 shows that at speeds greater than about 14 m/s, using RL=1 incurs less overhead. The cutoff point increases to about 17 m/s for call frequencies of 0.3 per minute (Figure 5). At 20 m/s, using RL=1 can result in a savings of over 31 % over an “always update” scheme (from Figure 4).

6 Concluding Remarks

We have presented a network architecture for a multihop mobile wireless network, one in which switches as well as endpoints may be mobile. In this context, we described a location management scheme that provides for the update versus paging tradeoff on a per endpoint basis using the concepts of roaming level and roaming cluster. We also presented simulation results that validate the approach and show significant reduction in overhead in relation to the traditional “always-update” scheme.

As wireless devices continue to decrease in size and increase in transceiver capabilities, an increasing number of future wireless networks are likely to be large, dense, and dynamic. Our architecture and protocol are well-suited to provide an efficient and robust location management scheme for applications in this context.

Acknowledgments

We are grateful to Regina Hain (BBN) for her assistance in the simulations, and to Gregory Lauer (BBN) and Martha Steenstrup (BBN) for their guidance and support.

References

[1] C.E. Perkins, P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers,” in *Proc. of ACM SIGCOMM* 1994.

- [2] N. Shacham, J. Westcott, “Future Directions in Packet Radio Architectures and Protocols,” *Proceedings of the IEEE*, Vol. 75, No. 1, Jan. 1987, pp. 83-99.
- [3] Capt. J.A. Brandler, “Tactical Military Communications”, *IEEE Communications Magazine*, Jan. 1992.
- [4] M. Bergamo, R. Hain, K. Kasper, D. Li, R. Ramanathan, M. Steenstrup, “System Design Specification for Multimedia Support for Mobile Wireless Networks (Draft)”, available from anonymous ftp.bbn.com:/pub/ramanath/mmw-n-design.ps.
- [5] B.M. Leiner, R.J. Ruth, A.R. Sastry, “Goals and Challenges of the DARPA GloMo Program”, *IEEE Personal Communications*, December 1996.
- [6] J.W. Ketchum, “Routing in cellular mobile radio communications networks”, in: *Routing in Communication Networks*, ed. M. Steenstrup, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [7] A. Bar-Noy, I. Kessler, “Tracking Mobile Users in Wireless Communications Networks,” *IEEE Transactions on Information Theory*, Vol. 39, No. 6, Nov. 1993.
- [8] H. Xie, S. Tabbane, D.J. Goodman, “Dynamic location area management and performance analysis,” *Proc. 43rd IEEE Vehicular Tech. Conf.*, pp. 536-539, 1993.
- [9] Y. Birk, Y. Nachman, “Using direction and elapsed-time information to reduce the wireless cost of locating mobile units in cellular networks,” *Wireless Networks I* (1995), pp. 403-412.
- [10] J.Z. Wang, “A fully distributed location registration strategy for universal personal communication system”, *IEEE JSAC*, Vol. 11, No. 6, Aug. 1993.
- [11] S. Ramanathan, M. Steenstrup, “A Survey of Routing Techniques for Mobile Communications Networks”, *Mobile Networks and Applications (MONET)* 1 (1996) 89-104.