Shared Tree Wireless Network Multicast *

Ching-Chuan Chiang, Mario Gerla and Lixia Zhang
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

In this paper, we propose a multicast protocol for a multihop, mobile wireless network with cluster based routing and token access protocol within each cluster. The multicast protocol uses a shared tree which is dynamically updated to adjust to changes in topology and membership (i.e. dynamic joins and quits). Two options for tree maintenance have been simulated and evaluated: "hard state" (i.e. each connection must be explicitly cleared) and "soft state" (each connection is automatically timed out and must be refreshed). For the soft state policy, the performance of different choices of timeout and refresh timers is first analyzed for a range of node mobility values. Next, soft state and hard state policies are compared based on throughput, join delay, and control overhead criteria.

1 Introduction

1.1 Multihop, Mobile Wireless Networks

Wireless networks provide mobile users with ubiquitous communicating capability and information access regardless of the location. If we restrict outselves to ground radio networks, we can define two basic types of wireless networks: (a) cellular, and (b) instant infrastructure, multihop. In cellular radio networks [12] mobile users communicate via a single hop wireless channel with a base station which is in turn connected to a wired backbone.

In a multihop wireless network, in contrast, there are no fixed base stations connected to a wireline network. All nodes communicate via the wireless channel with possible multihopping over several mobile stations. The main application of wireless multihop networks is rapid deployment in an area where there is no wired infrastructure (e.g. the battlefield) or where the infrastructure has failed (e.g. earthquake, fire, flood relief, etc). Examples of such networks are ad-hoc networks [15] and packet radio networks [5, 14].

Multihopping poses new challenges in wireless network protocol design. For example, mobile-IP routing protocols developed for cellular-type networks [13] cannot be directly applied to the multihop case since there is no fixed home agent to maintain routing information. A particularly challenging problem is multicasting. Again, traditional wired network multicast protocols [3, 7] cannot be directly transferred to this environment. For example, in the Internet multicast backbone (MBone) application, the multicast protocol DVMRP [7] uses the reverse path forwarding (RPF) protocol to deliver multicast packets. In reverse path forwarding, a router forwards a broadcast packet originating at source S if and only if it has arrived via the shortest path from S. If source S moves rapidly and its packet arrives after the local routing table has been updated, the router will fail to forward the packet [1]. Also, if the source moves rapidly, the nodes in the tree may not be able to maintain up to date routes to that source. In addition, periodical full broadcast in DVMRP introduces costly overhead on the low bandwidth wireless channel and is not suitable for a sparsely distributed membership.

In general, the following challenges are posed by wireless, mobile multicasting: (a) sources move, making source-oriented protocols inefficient; (b) multicast group members move, thus requiring an easily reconfigurable multicast tree topology; (c) transient loops may form during tree reconfiguration; (d) channel overhead caused by tree reconfiguration updates tends to increase very rapidly with mobility, network size and membership size.

1.2 ST-WIM: a Shared Tree Wireless Multicast Protocol

In this paper, we propose a Shared Tree Wireless Multicast protocol (ST-WIM) inspired by the sparse PIM algorithm [6]. ST-WIM is independent of the underlying wireless routing protocol, thus allowing porting to different wireless platforms. The shared tree is rooted at a rendezvous point (RP) and is dynamically and distributedly reconfigured to account for mobility and multicast membership changes. Each multicast group has its own RP and "grows" its own shared tree. The multicast group is identified by a multicast address. Each member receives packets sent to that address. Many-to-many casting is assumed; senders do not need to know the membership of the group.

*This work was supported by the U.S. Department of Justice/Federal Bureau of Investigation, ARPA/ITO under Contract J-FBI-93-112 Computer Aided Design of High Performance Network Wireless Networked Systems, and by Intel under project "QoS Wireless Networks"

The intermediate routers in the shared tree are responsible for forwarding the multicast data to members. Routers will keep track of the downstream members in order to decide whether or not to forward the multicast data on the corresponding downstream links. Namely, a link has the state of "downstream" link if it leads to active group members. Members may dynamically join and leave the multicast group.

A critical issue in the design of a wireless shared tree protocol is tree maintenance and reconfiguration in the face of mobility and membership change. To this end, two different tree maintenance schemes, hard state and soft state, are proposed and evaluated.

While ST-WIM is compatible with any of the underlying wireless network infrastructure, it can be evaluated (via simulation) only in conjunction with a specific multihop architecture. In this paper, the chosen architecture is based on cluster routing and token access protocol within each cluster [4, 10].

The paper is organized as follows. Section 2 describes the multihop infrastructure. Section 3 introduces the multicast tree maintenance schemes. Section 4 defines the experimental environment. Section 5 presents the performance results. Section 6 concludes the paper.

2 Multihop Network Infrastructure

The multihop architecture considered in this study is a clustered multihop network [4, 10]. The aggregation of nodes into clusters under clusterhead control provides a convenient framework for the development of important features such as code separation (among clusters), channel access, bandwidth allocation and routing [10]. Using a distributed clustering algorithm, specific nodes are elected as clusterheads. All nodes within transmission range of a clusterhead belong to the same cluster. That is, all nodes in a cluster can communicate directly with a clusterhead and (possibly) with each other. Nodes belonging to more than one cluster are called gateways. Gateways support communications between adjacent clusters. In a mobile network, an important criterion in cluster algorithm design is stability. Frequent clusterhead changes adversely affect the performance of other protocols such as scheduling and resource allocation. We use the Least Clusterhead Change (LCC) clustering algorithm [4], where only two conditions cause the clusterhead to change: (a) two mobile clusterheads come within range of each other (and therefore one clusterhead must be "demoted"), and; (b) a node becomes disconnected from any cluster (and therefore becomes its own clusterhead). This is an improvement (in stability) over two previous algorithms, lowest-ID [8] and highest-connectivity (degree) [10], where a new clusterhead may be elected every time the cluster membership changes. The LCC algorithm uses either lowest-ID or highest-connectivity for initialization. After that, a non-clusterhead node moving into an already established cluster cannot challenge the current clusterhead. If, on the other hand, a clusterhead moves into an existing cluster, then it may take over (or be taken over) based on ID, connectivity or some other well defined priority [4].

Clustering provides an effective way to allocate wireless channels among different clusters. Across clusters, we permit spatial reuse by using different spreading codes and thus reduce intercluster interference [11]. Additional procedures are required to maintain code separation across clusters. For example, a common control code must be used for initialization and for reconfiguration [10]; orthogonal codes must be selected in adjacent clusters, etc. Specific solutions are reported in [9].

Cluster maintenance protocols run continuously in the background in order to dynamically reconfigure the cluster structure in the face of mobility. Average convergence time of the clustering algorithm is $O(1)$, that is, it does not depend on network size N [10]. In fact, the clusters reform as quickly as the links are added/deleted. This properly implies that the convergence of the routing algorithm (which operates above clustering) is not "slowed down" by the presence of clusters.

Within a cluster, the clusterhead polls the nodes to allocate the channel. Polling was chosen here for several reasons. First, polling is consistent with the IEEE 802.11 standard (centralized nodes). Secondly, polling gives priority to the clusterhead, which is desirable since routes are forced to go through clusterheads in cluster oriented routing. Thirdly, in our experiments each cluster has on average six neighbors (which is the optimal value in uniform multihop architecture); thus polling latency is not of concern. Forth, polling permits easy support of real time connection (which can be scheduled at periodic intervals by the clusterhead). In a more uniform network (e.g. slow and fast nodes, or high power and low power nodes) optimal cluster size may be much higher than 6 [9]; in which case, the polling scheme may be replaced by a polling/random access scheme, to reduce latency. This latter scheme is also consistent with IEEE 802.11. In this paper we only consider clusterhead controlled polling. The results, however, are applicable also to polling/random access as well as to more general access schemes.

The routing protocol used in this experiment is hierarchical. Namely, packets travel from source to destination through an alternation of clusterheads and gateways [4] It is based on an extension of the DSDV (Destination-Sequenced Distance-Vector) scheme [15]. DSDV is a "distance vector" type algorithm with the same complexity as Bellman-Ford or RIP, but with better protection against loops.

3 Hard State versus Soft State Tree Maintenance

In ST-WIM, the tree maintenance protocol must keep track of downstream and upstream links at each node, for each multicast group. When a node in the tree receives the multicast packet, it will forward it only to downstream links, if any (except, of course, the link the packet came from). Thus each intermediate node must keep the state of its downstream members. Efficient updating of link states has critical impact on the performance of multicasting. For graft-based multicast [3], nodes wishing to join the group send a JOIN_REQUEST message to the RP. All nodes traversed by the JOIN_REQUEST will save the join state to maintain the downstream links. There are two major schemes to maintain the shared tree. One is "hard state", and the other is "soft state".

3.1 Hard State Protocol

In the hard state protocols, when a node wants to join a multicast group, it must send an explicit JOIN_REQUEST and wait until it is acknowledged to become a member. A member will keep its membership until it intends to quit or the connection is broken. Namely, the upstream node in the multicast tree keeps a link in "downstream" state until it receives an explicit QUIT_REQUEST from the downstream node or when the downstream link is disconnected. The hard state protocol relies on the underlying MAC protocol to provide the link connectivity information. Only if the MAC layer provides reliable, periodic link state information, can the hard state protocol adjust to connectivity change (i.e. it can efficiently prune broken links and establish new connections to the tree). For example, in the mobile wireless networks, when a downstream member of node i moves out of the transmission range of node i, it will send a new JOIN_REQUEST to setup a new link to the tree. The original downstream link is disconnected.

Since hierarchical routing is used, the internal nodes of the multicast tree are all clusterhead or gateway types. A regular node type (i.e. neither gateway or clusterhead) can be found only at the leaves of the tree. In this cluster infrastructure, the multicast tree structure in hard state will be reconfigured only in the following cases: (1) The member of a host group moves and changes node type. (2) Tree links break (potentially creating loops).

3.1.1 Member migration It is necessary to reconfigure the multicast tree any time a group member moves or changes node-type. A group member can detect changes in the multicast tree by monitoring its connectivity to upstream and downstream members (as mentioned before, this must be done by MAC layer). A member node reconnects to the tree by sending a JOIN_REQUEST to the RP when its upstream path becomes disconnected (e.g. the upstream node moves out of range or changes node type from clusterhead/gateway to regular node). For example, a clusterhead member will send a JOIN_REQUEST to the RP in order to reconstruct the tree, if its upstream member (a gateway) becomes a regular node, or becomes disconnected. When a regular node member (a leaf) moves from cluster C_i to cluster C_j, the clusterhead of C_i will drop it from its descendant list. The regular node will send a JOIN_REQUEST to its new clusterhead in C_j. The clusterhead of C_i will send a QUIT_REQUEST to its upstream node if it has itself become a leaf.

3.1.2 Multicast Tree Recovery In the multicast tree, an internal node has one parent and one or more children. If the parent link is disconnected, the internal node can recover either by resending JOIN_REQUEST message to connect to the tree via a new parent, or by sending FLUSH_TREE message to its children. The FLUSH_TREE message propagates to the entire subtree, forcing all downstream leaf-members to rejoin the tree individually. Rejoin by the internal node will be quicker than by leaf-members, but it may occasionally create a temporary loop (rejoin to its own downstream node). In a mobile environment, we prefer to use the flush scheme to avoid temporary loops.

3.2 Soft State Protocol

In the soft state protocol, a node which wants to remain in a multicast group must periodically send a JOIN_REQUEST to the RP. No ACK is required. The node receiving a JOIN_REQUEST from its neighbor will store a state for this neighbor as a downstream member and will attach a time stamp to it. The upstream node will update the state timer when it receives another join request (state-driven refresh). The downstream node is automatically expelled from membership when its timer expires. There is no need to keep track of the upstream node. Every node receiving the JOIN_REQUEST just forwards it to the RP using the underlying routing scheme. This is different from Hard State, where the state of the upstream node must also be kept. Soft state provides a fail-safe method for a dynamically changing environment and relaxes the link connectivity dependency on MAC layer.

Note that reliable link connectivity maintaining at the MAC layer is also required for cluster and route maintenance. However, there are nodes (e.g. low power or high mobility) which would never be used as clusterheads or gateways in the hierarchical routing scheme. Thus, the connectivity of these nodes will be updated only sporadically (to save power for example). The sporadic MAC layer connectivity maintaining of low power nodes, say, causes potential problems for Hard State, but, is compensated by the periodic refresh in Soft State.

The time periods for refresh ($T_{refresh}$) and timeout ($T_{timeout}$) must be carefully chosen taking into account

mobility and channel access overhead. In a highly mobile network, a short refresh period is desirable, but it will increase the channel overhead. If the timeout period is long, there will be stale links and wasted duplicate transmissions on such links. If the timeout period is short, branches are prematurely cut off and data may be lost. In order to achieve low overhead and yet maintain connectivity, it is very important to judiciously (possibly, dynamically) select the time period for refresh and timeout. These tradeoffs are explored in the experiments described in the following sections.

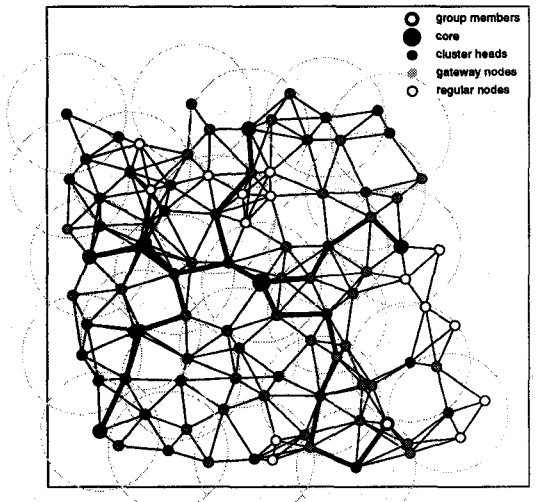


Fig. 1: Initial multicast tree

4 Simulation Environment

A multihop, mobile wireless network simulator was developed using the parallel simulation language Maisie [2]. The network consists of 100 mobile hosts roaming randomly and uniformly at a preset average speed in a 1000x1000 meter square. Radio transmission range is 120 meters. Data rate is 2 Mb/s. Packet length is 10 kbit for data; 2 kbit for routing tables, and 500 bits for MAC control packets. Routing tables and control messages have higher priority over data. Channel overhead (e.g, code acquisition time, preamble, etc.) is factored into packet length. Transmission time is 5 ms for data packet, 1 ms for routing table, and 0.25 ms for control packet.

The RP is hand-picked and does not change throughout the experiment. Dynamic relocation of the RP could improve the efficiency of the tree algorithm. This option, however, is not considered in our study since it would not affect the Hard State vs. Soft State tradeoffs. Members are randomly selected to join and quit the multicast group. On average, seven members are part of the group. Figure 1 shows a typical multicast tree configuration. There is a single source of multicast traffic, placed at the RP. Traffic input rate is high enough to fully load the network. Unlimited buffering is assumed at the nodes. Packets are dropped only if no route is available to the designated destination. Total simulation time for each experiment is $2x10^6$ simulation ticks. One simulation tick corresponds to 10 μs. Thus, each run represents 20 seconds of simulated time.

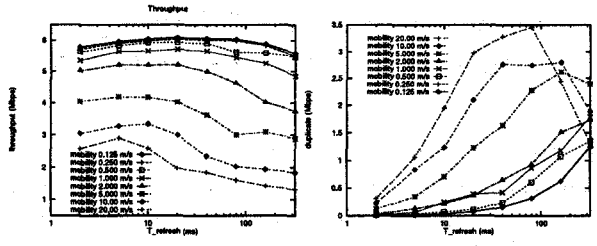


Fig. 2: Soft State: impact of $T_{refresh}$ and mobility

5 Performance Evaluation

In this section, we first evaluate the soft state scheme and select the values $T_{refresh}$ and $T_{timeout}$ which optimize its performance (for the given systems parameters). Then, using these values, we compare the performance of hard state and soft state using as criteria throughput, join latency, and control overhead.

5.1 Soft State Parameter Optimization

The performance of the soft state scheme depends critically on the selection of refresh and timeout intervals. We will evaluate the effect of $T_{refresh}$ and $T_{timeout}$ for various parameter settings. To this end, we first define total throughput performance as the total traffic received by members. Some of the received packets however may be duplicates, as described in section 3.2. Thus, we define throughput as (total received packets) - (duplicate traffic).

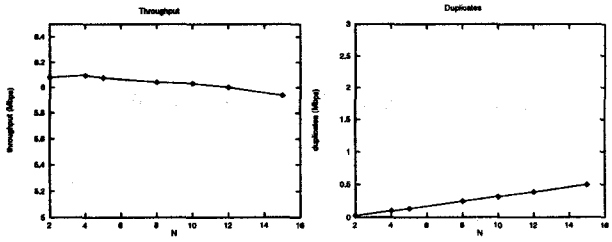


Fig. 3: $T_{timeout}$ = N * $T_{refresh}$ @ mobility = 0.125 m/s

5.1.1 Mobility vs. $T_{refresh}$ First we study the relationship between $T_{refresh}$ and mobility. Recognizing that for stability the timeout must be larger than the refresh period, we set $T_{timeout} = 10 * T_{refresh}$. We then evaluate throughput and duplicates for various values of $T_{refresh}$ and mobility. From figure 2 we note that high mobility causes more duplicates and lower throughput. To improve higher throughput at high mobility, $T_{refresh}$ must be reduced so as to adapt to the rapidly changing topology. However, as $T_{refresh}$ is reduced, $T_{timeout}$ is also becoming

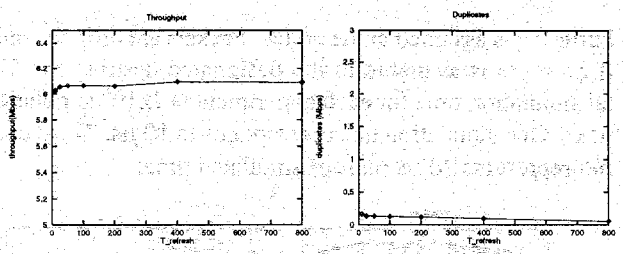


Fig. 4: $T_{refresh}$ @ mobility = 0.125 m/s

smaller, eventually causing throughput degradation due to frequent timeout and tree disconnects. When $T_{refresh}$ is large, $T_{timeout}$ tends to become too long, thus increasing the duplicate overhead. The system is particular sensitive to $T_{refresh}$ at high speed.

5.1.2 $T_{refresh}$ vs. $T_{timeout}$ Next, we vary $T_{refresh}$ and $T_{timeout}$ simultaneously to find the best combination. Starting with low mobility, figure 3 shows throughput and duplicates with mobility = 0.125 m/s and $T_{refresh}$= 400 ms. The throughput is not affected by large timeout since the topology does not change too rapidly and thus few duplicates are created anyway. In figure 4 we fix the timeout interval to 1600 ms and vary $T_{refresh}$. Again, we note that throughput and duplicates are not very sensitive to refresh interval.

Next we study the high mobility case (20 m/s). Now, throughput and duplicates are much more sensitive to $T_{refresh}$ and $T_{timeout}$ (see figure 5 and 6).

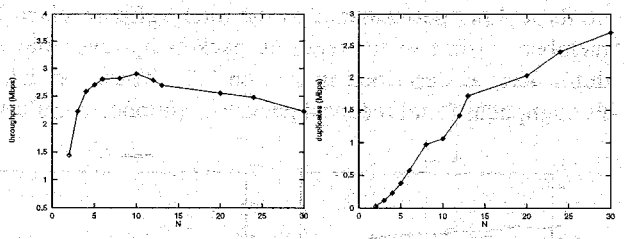


Fig. 5: $T_{timeout}$ = N * $T_{refresh}$ @ mobility = 20 m/s

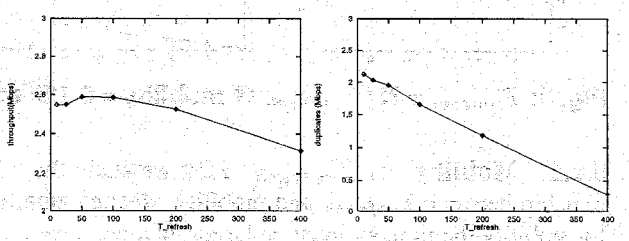


Fig. 6: $T_{refresh}$ @ mobility = 20 m/s

From the above results it is clear that both $T_{refresh}$ and $T_{timeout}$ must be modified as mobility varies, in order to optimize throughput. Table 1 presents the optimal results for various mobility values obtained via repeated simulations. We note that the optimal value of $T_{refresh}$ is inversely proportional to speed. For example, a 100 fold reduction of speed (from 20 m/s to 0.25 m/s) requires an increase of $T_{refresh}$ from 25 ms to 2000 ms. This was expected since the higher the speed, the lower the refresh period to track the changes in topology. The timeout also must decrease as speed increases. However, it should not decrease so rapidly as $T_{refresh}$, in order to avoid unnecessary tree disconnects. We use the parameters in table 1 for soft state scheme to compare the performance evaluation with hard state.

Table 1: parameters of higher throughput

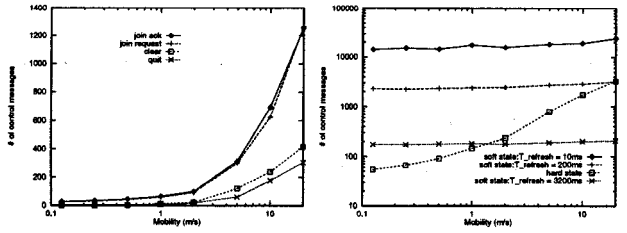
Mobility (m/s)	$T_{refresh}$(ms)	$T_{timeout}$
20	25	250
10	50	300
5	100	450
2	250	600
1	500	850
0.5	1000	1100
0.25	2000	2100
0.125	4000	4100

5.2 Join Latency

Next, we evaluate the join latency, namely, the time required for a new member to join. For soft state, there is no explicit JOIN_ACK like in the hard state protocol. Thus, we measure the join latency as the delay time between the first JOIN_REQUEST and the first multicast data arrival. For hard state, we define two measures for join latency, namely: ACK delay (time between JOIN_REQUEST and JOIN_ACK) and data delay (time between JOIN_REQUEST and first multicast data arrival). Figure 8(a) shows the average join latency for various mobility values. As expected, the join latency of soft state is lower than the data delay of hard state. This is because the hard state protocol requires the issue of JOIN_ACK before data is sent. Conversely, soft state join latency is higher than hard state ACK delay because the ACK packet is much shorter than the data packet.

5.2.1 Control Packet Overhead In hard state, the control messages required to maintain the multicast tree are: JOIN_REQUEST, JOIN_ACK, QUIT, and CLEAR messages. For soft state protocol, only JOIN_REQUEST is required. Figure 7(a) shows the individual control traffic components of hard state , and figure 7(b) compares the total control traffic of hard state and soft state. The control traffic of hard state increases with node mobility, because higher mobility causes more tree disconnects and therefore triggers more join requests. For soft state, control traffic is independent of mobility as long as $T_{refresh}$ is a constant.

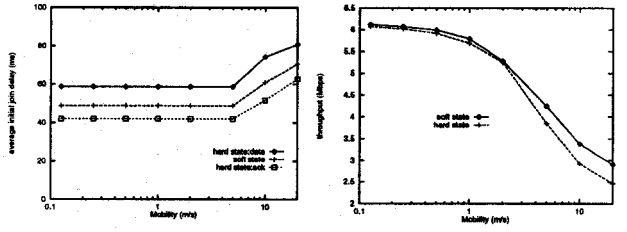
The explicit control messages in hard state are much fewer than in soft state, but the hard state protocol requires the underlying MAC layer protocol to continuously probe link connectivity.



(a) hard state (b) total control messages

Fig. 7: Control Messages

5.2.2 Throughput Comparison Figure 8(b) compares the throughput of hard state and soft state. The soft state experiment uses the best choice of refresh and timeout timers found in our simulations. Soft state performs better than hard state at high mobility This is mainly due to two reasons: (a) in soft state, the join delay is lower than in hard state, thus, fewer packets are dropped during disconnect. (b) when the tree becomes disconnected, hard state suffers the additional delay of flushing the subtree, before the members have the chance to create a new subtree.



(a) Join Latency (b) Throughput

Fig. 8: Performance Evaluation

6 Conclusion

In this paper we have proposed ST-WIM, a shared tree wireless multicast protocol which is inspired by the sparse PIM scheme. The main contribution of the paper is the performance evaluation of ST-WIM as a function of mobility. Two tree maintenance schemes have been proposed, namely soft state and hard state. For soft state, we have investigate the impact of $T_{refresh}$ and $T_{timeout}$ on performance, as a function of mobility. We have found that the two parameters are interdependent and must be jointly optimized in order to improve throughput and reduce duplicates. For hard state, we have shown that performance degrades very rapidly with mobility because of the delays involved in detecting tree disconnections and in restoring the tree. Packets are dropped while the tree is disconnected. A preliminary comparison of hard state and soft state reveals that the two schemes are comparable at low speed. As mobility increases, soft state outperforms hard state because of the high reconnect delays suffered by the latter.

In summary, ST-WIM appears to be reasonably efficient for low values of mobility (say, up to 10 km/hr). For higher values of mobility, performance degrades rapidly. Work is now in progress to compare ST-WIM with other multicast protocols (such as flooding) which are more robust to mobility.

References

[1] A. Acharya, A. Bakre, and B. R. Badrinath. Ip multicast extensions for mobile internetworking. In *INFOCOM*. IEEE, 1996.

[2] R. Bagrodia and W. Liao. Maisie: A language for the design of efficient discrete-event simulations. *IEEE Trans. on Software Engineering*, Vol. 20, No. 4:225–238, 1994.

[3] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (cbt) an architecture for scalable inter-domain multicast routin. In *SIGCOMM*, pages 85–95. ACM, 1993.

[4] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *The IEEE Singapore International Conference on Networks*, pages 197–211. IEEE, 1997.

[5] M. S. Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *ACM Journal on Wireless Networks*, Vol. 1, No. 1, 1995.

[6] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Wei. The pim architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, Vol. 4 No. 2:153–162, 1996.

[7] S. E. Deering and D. R. Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems*, pages 85–111, 1990.

[8] A. Ephremides, J. Wieselthier, and D. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. In *Proc. IEEE 75*, pages 56–73. IEEE, 1987.

[9] M. Gerla and C.-C. Chiang. Multicast routing in multihop, mobile wireless networks. Technical report, UCLA-CSD, May 1997.

[10] M. Gerla and J. T.-C. Tsai. Multicluster, mobile, multimedia radio network. *ACM Journal on Wireless Networks*, Vol. 1, No. 3:255–265, 1995.

[11] K. Gilhousen, I. M. Jacobs, and et al. On the capacity of a cellular cdma system. *IEEE Trans. Veh. Tech.*, Vol. 40:303–312, 1991.

[12] D. J. Goodman. Cellular packet communications. *IEEE Trans. on Communications*, Vol. 38, No. 8, 1990.

[13] J. Ioannidis, D. Duchamp, and G. Q. M. Jr. Ip-based protocols for mobile internetworking. In *SIGCOMM*, pages 235–243. ACM, 1991.

[14] J. Jubin and J. D. Tornow. The darpa packet radio network protocols. *Proc. of the IEEE*, Vol. 75, No. 1, 1987.

[15] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *SIGCOMM*, pages 234–244. ACM, 1994.