Routing and Multicast in Multihop, Mobile Wireless Networks *

Ching-Chuan Chiang and Mario Gerla
University of California, Los Angeles

Abstract

In this paper we present a multicast protocol which builds upon a cluster based wireless network infrastructure. First, we introduce the network infrastructure which includes several innovative features such as: minimum change cluster formation; dynamic priority token access protocol, and; distributed hierarchical routing. Then, for this infrastructure we propose a multicast protocol which is inspired by the Core Based Tree approach developed for the internet. We show that the multicast protocol is robust to mobility, has low bandwidth overhead and latency, scales well with membership group size, and can be generalized to other wireless infrastructure.

I. Introduction

Wireless networks provide mobile users with ubiquitous communicating capability and information access regardless of location. In this paper we address a particular type of wireless networks called "multihop" networks. As a difference from "single hop" (i.e. cellular) networks [12] which require fixed base stations interconnected by a wired backbone, multihop networks have no fixed based stations nor wired backbone [10]. The main motivation for mobile wireless multihopping is rapid deployment and dynamic reconfiguration. When the wireline network is not available, as in battlefield communications and search and rescue operations, multihop mobile wireless networks provide the only feasible means for ground communications and information accesses. Examples of multihop wireless networks are ad-hoc networks [13, 16] and packet radio networks [5, 14]. Multihopping poses new challenges in wireless network protocol design. For example, routing protocols developed for single-hop networks [12] cannot be applied to multihop networks since there is no fixed home agent to maintain routing information. Another challenging problem is multicasting. Traditional multicast protocols [3, 6] are not suitable for this environment. For example DVMRP [6] 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 the router back to S. If the source S moves, the reverse path algorithm will not forward packets correctly [1]. In general, the following challenges are posed by wireless, mobile multicasting: (a) multicast servers (the sources) move, making source-oriented protocols inefficient; (b) multicast group members move, thus precluding the use of a fixed multicast topology; (c) transient loops may form during tree reconfiguration; (d) the tree reconfiguration scheme should be simple to keep channel overhead low.

To address these challenges, we propose a modified version of the Core Based Tree (CBT) multicast algorithm which was recently developed for wired networks such as the Internet [3]. CBT multicast routing protocol uses a single-core tree to improve scalability. CBT is more suitable than DVMRP for multihop mobile networks. Yet, it too is sensitive to node mobility. For example, the JOIN_ACK, which creates the tree branch, and the JOIN_REQUEST may traverse different paths.

Referring to the wireless network protocol stack, the multicast protocol is on top of the routing protocol, which in turn sits on top of the MAC protocol. Below the MAC protocol resides the clustering protocol. The various layers are in principle modular and independent of each other. Thus, a correctly defined multicast protocol should work on top of any routing protocol, MAC scheme and clustering algorithm. The performance of the multicast protocol, however, is dependent on the implementation of the lower layers. Since one of our goals is multicast performance evaluation, we will define in this paper the network infrastructure on which the multicast protocol will be tested. The multicast protocol, however, is generalizable to any wireless, multihop infrastructure.

The infrastructure itself is inspired by that of recent multihop, mobile, network proposals, such as Cluster TDMA [10] and Cluster Token [15]. It does, however, introduce many novel elements which improve performance. Thus a brief description of the infrastructure is provided prior to the introduction and evaluation of the multicast algorithm.

The paper is organized as follows. Section 2 presents the Cluster and MAC layers. Section 3 introduces the various multihop, mobile routing algorithms. Section 4 compares their performance. Section 5 describes the proposed multicast scheme. Section 6 evaluates multicast performance. Section 7 concludes the paper.

II. Cluster and token infrastructure

A. Clustering

In multihop, mobile wireless networks, the aggregation of nodes into clusters controlled by a clusterhead provides a convenient framework for the development of important features such as code separation (among clusters), channel access, routing, and bandwidth allocation [10, 11]. Using a distributed clustering algorithm, specific nodes are elected to be clusterheads. All nodes within transmission range of a clusterhead belong to the same cluster. That is, all nodes in a cluster can communicate with a clusterhead and (possibly) with each other. The most important criterion in cluster algorithm design is stability. Frequent clusterhead changes adversely affect the performance of other protocols such as scheduling and resource allocation which rely on it. We proposed a novel clustering algorithm (Least Clusterhead Change (LCC) clustering algorithm) [4], where only two conditions cause the clusterhead to change: (a) two clusterheads come within range of each other, and; (b) a node becomes disconnected from any cluster. This is an improvement (in stability) over two previous algorithms, lowest-ID algorithm [7] and highest-connectivity (degree) [10], which reelect the clusterhead every time the cluster membership changes. The LCC algorithm uses either lowest-ID or highest-connectivity for initialization and routine maintenance. However, the key difference here is that when a non-clusterhead node moves into an already established cluster, it cannot challenge

*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 current clusterhead. If, on the other hand, a clusterhead moves into an existing cluster, then it may or may not prevail based on ID, connectivity or some other well defined priority. More details are reported in [9].

Figure 5 shows an example of clustering output using LCC with lowest-id among 100 nodes. It should be pointed out that there are many issues must be addressed in the design of a clustering algorithm with code separation across clusters. For example, as described in [10], a common control code must be used for initialization and for reconfiguration; orthogonal codes must be selected in adjacent clusters, etc. Specific solutions to these problems are omitted here for brevity. The interested reader is referred to [9].

B. MAC layer

Clustering provides an effective way to allocate wireless channels among different clusters. Across clusters, we enhance spatial reuse by using different spreading codes (i.e. CDMA [11]). Within a cluster, we use a clusterhead controlled token protocol (i.e. polling) to allocate the channel among competing nodes. The token approach allows us to give priority to clusterheads in a way which maximizes channel utilization and minimizes delay.

Various token scheduling schemes can be used to improve routing efficiency. One way to do this is to give higher priority to neighbors from which a packet was recently received. Here is a simple way to implement priority-token-scheduling (PTS).

- Initially every node in the cluster has the same priority to receive the token from the clusterhead.
- When a data packet is transmitted by node i, the clusterhead increases the priority of node i.
- When the token returns from an empty queue at neighbor j, the clusterhead decreases the priority of node j.

More generally, priority token scheduling allows us to forward high priority traffics with the least delay. Moreover, dynamic scheduling permits us to reserve a portion of the channel by offering more transmission opportunities to real time and multimedia sources.

Previous cluster oriented schemes, such as cluster TDMA [10] and cluster token [15], did not take full advantage of clusterheads. In our clusterhead oriented token scheme, the clusterhead plays an important role both in clustering and in dynamic channel scheduling. As a result, LCC clustering is more stable than previous clustering schemes, and token scheduling is more flexible.

C. Gateways

We define a node to be a *gateway* if it belongs to more than one cluster. To communicate within a cluster, a gateway must select the code used by that cluster. We assume that a gateway can change its code after it returns the permission token or it receives a message. When a clusterhead issues the permission token to a gateway which is tuned to a different code, the token will be lost (i.e. a code conflict occurs). Clearly, code scheduling will affect the message delivery performance. The simplest type of scheduling is random code selection. However, simple heuristics can considerably improve performance. One such heuristic, denoted GCS (Gateway Code Scheduling), assumes that when a gateway transmits a packet to the downstream clusterhead, there will be soon another packet coming in from the upstream clusterhead. Thus, the gateway, after the downstream transmission, goes back to the upstream cluster code. Likewise, the gateway, after receiving a packet from upstream, promptly returns its receiver to the downstream code, to receive the token and thus forward the packet downstream as quickly as possible.

III. Routing

Routing is a critical component in any multihop wireless network. It is also a key element of multicasting. Thus, particular attention was given to routing in our research. One important requirement in mobile networks is the avoidance of loops which are caused by stale routing tables. Several adaptive, loop free routing schemes have been recently proposed specifically for wireless, mobile networks [16, 8]. In our proposed scheme we use as a basis the Destination Sequenced Distance Vector (DSDV) routing scheme [16] which was recently implemented also in cluster TDMA [10] and cluster token [15] schemes. DSDV stamps increasing sequence numbers on routing updates relative to a given destination. This way, stale updates can be easily detected and loops avoided.

In our project, we modify the DSDV scheme by exploiting the clusterheads. Namely, we use hierarchical routing to route packets. Each node maintains a cluster member table which records the destination clusterhead for each node, and broadcasts it periodically. A node will update its cluster member table when it receives a new one from its neighbor. Here again we use destination sequence numbers as in DSDV to avoid stale tables. There are two tables for each node to route packets. One is the cluster member table which is used to map destination address to the destination clusterhead address, and the other is the routing table which is used to select the next node to reach the destination cluster. We call this cluster (hierarchical) routing scheme DSCR.

There are ways to improve the efficiency of DSCR by optimizing the interaction between routing and MAC layer. The first strategy we consider consists of routing packets alternatively through clusterheads and gateways. That is, a packet will be routed via C_1, G_1,C_2,G_2,C_3,G_3.., where C_i are clusterheads and G_i are gateways. We call this routing strategy Clusterhead-Gateway Switch Routing (CGSR). Figure 1 shows routing examples for DSDV, DSCR, and CGSR. Node 1 is the source and node 11 is the destination. The main difference with respect to the previous schemes is that the packet is forced to pass through the clusterhead, avoiding gateway to getaway shortcuts as from node 5 to node 7 in figure 1. At first glance, this may seem to be a drawback rather than an advantage since it increases path length. However, recalling that clusterheads have more chances to transmit than other nodes, and that a gateway-to-gateway transmission requires that both gateways rendezvous on the same code, we realize that the presence of a clusterhead between two gateways is well worth the cost of the extra hop. Experiments verify this conjecture. We

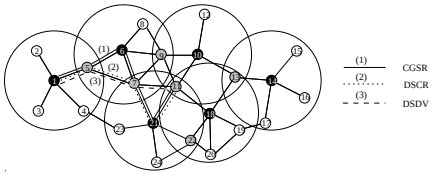

Fig. 1. Routing examples (from node 1 to node 11)

can further reduce packet delay by combining CGSR with priority token scheduling (CGSR+PTS), as discussed in section 2. We can go one step further and also add gateway code scheduling (CGSR+PTS+GCS). In the two latter cases, the delay improvement is due to MAC layer features, rather than routing features. In both case, the improvement is obtained by exploiting the knowledge that steady traffic exists on certain paths in the network, and by assuming that this traffic will persist in the future. However, in a mobile situation, the paths change continuously, nullifying the advantage of traffic pattern driven schedules and priorities. To keep the traffic pattern more stable, we may attempt to reserve the path for a connection (in a virtual circuit fashion) until it becomes disconnected, instead of selecting the new shortest path after each move. Once the first packet selects the path, all the subsequent packets will follow this path until it breaks. We call this path

reservation scheme CGSR+PTS+GCS+PR.

IV. MAC and Routing Experiments

The MAC and routing strategies described in the previous section have been evaluated via simulation. To this end, a multihop, mobile wireless network simulator was developed using an existing process-oriented, parallel simulation language called Maisie [2, 4]. The environment consists of 100 mobile hosts roaming uniformly in a 1000x1000 feet square. Each node moves randomly at a preset average speed. Radio transmission range is 100 feet.Data rate is 2 Mb/s. Packet length is 10 kb for user data and most control packets. It is 2.5 kb for token maintenance packets. Channel overhead (e.g, code acquisition time, preamble etc) is factored in packet length. Thus, data packet transmission time is 5 ms.

The experiment consists of transmitting a file of 100 packets from one source to one destination (using a free-wheeling protocol such as UDP), and measuring the effective throughput (i.e. bits transmitted/total transfer time) for various routing and MAC layer options, with mobility ranging from 0 to 20 m/s. Figure 2 reports the results. It is clear that the combination of cluster (hierarchical) routing, clusterhead/gateway alternation, traffic pattern driven token scheduling and gateway code scheduling (i.e., CGSR+PTS+GCS) yields a remarkable throughput improvement (typically, between 3 and 4-fold) with respect to the "flat" routing scheme (DSDV), for a broad range of node speeds. Path Reservation, on the other hand, does not appear to improve performance in a consistent manner. Furthermore, path reservation is not easy to implement, since it requires saving the "state" of each connection. For these reasons, in the sequel we use CGSR+PTS+GCS (referred to as CGSR for brevity) as the basic routing algorithm. Figure 2 also permits us to assess the throughput degradation caused by mobility. For zero mobility, the CGSR throughput is 450 kbps (i.e., less than one fourth of maximal channel speed, 2 Mbps). Here, the degradation is attributed to single tx/rcv radio multihop, token overhead and code switching overhead. At 20 m/s, CGSR throughput has dropped to 60 kbps. At high mobility, additional throughput loss is caused by delays and link level retransmissions due to path changes. Average end to end delays were also monitored. The delay results are correlated with throughput results (high throughput $\rightarrow$ low delay). In particular, for the CGSR+PTS+GCS case we observed 0.229 s delay for zero mobility. Since the average number of hops was 13 in this case, the average delay per hop is 17.6 ms, which accounts for transmission delay, token latency and code switching. At 20 m/s, the average end to end delay was 2.7 s. The main delay contribution in this case is link retransmission delay.

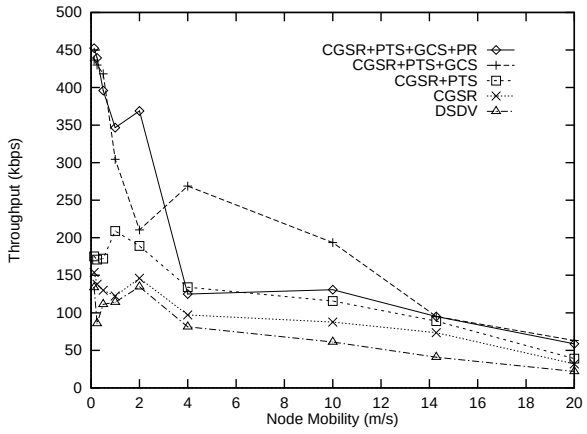

Fig. 2. Throughput of CGSR$*$ and DSDV

V. Multicast Routing Protocol

The multicast protocol is inspired by the Core Based Tree (CBT) scheme [3]. Each multicast group has a unique multicast identifier (Mid). Each multicast address identifies a host group, the group of hosts that should receive a packet sent to that address. Each multicast group is initialized and maintained by a multicast server (MS) which becomes the *core* of the CBT for this multicast group. Initially the multicast server broadcasts the Mid and its own node id (MSid) using a flooding algorithm. When a node receives this information, it records the pair Mid and MSid into its multicast database which can be used to join or quit this multicast group. Alternatively to avoid flooding, the multicast server registers the Mid on a directory server. Any node which wants to join a particular multicast group can query the directory server.

A. Multicast Tree graft and prune

The construction and maintenance of the core-based tree is receiver-oriented. When node i wants to join a multicast group G, it first gets the corresponding Mid and MSid either from its database or from the directory server. Then, it sends a JOIN_REQUEST to MSid. The JOIN_REQUEST will be routed to MSid (core) , using CGSR, until it reaches any node j which is already a member of the host group of G. Node j terminates the JOIN process by sending a JOIN_ACK back to node i. A node joins the multicast group and grafts a branch to the multicast tree (core-based tree) upon being traversed by JOIN_ACK Since CGSR routing is used, the internal nodes of the multicast tree are all clusterheads and gateways. Regular nodes can be found only at the leaves of the tree.

When internal node n (a clusterhead or gateway) is traversed by JOIN_ACK, it records the upstream and downstream node of JOIN_ACK. This information will be used to reconstruct the tree when the links in the tree break due to mobility or crash. The clusterhead of node i will record node i as a member of G after it forwards JOIN_ACK to node i.

When a leaf node wants to quit the group G, it sends a QUIT_REQUEST to its clusterhead. The clusterhead will update its membership information and then acknowledge this request with a QUIT_ACK. A leaf clusterhead leaves G and sends a QUIT_REQUEST to its upstream member when all of its downstream members have quit G. A nonleaf node cannot quit until it becomes a leaf.

The above scheme is somewhat different from the CBT scheme proposed in [3], where JOIN_ACK must follow the same path as JOIN_REQUEST. We allow JOIN_ACK to follow a different path (from JOIN_REQUEST), if so provided by routing tables; and use JOIN_ACK to graft links into the tree. The JOIN_ACK strategy is more adaptive to a higher mobile situation where routes may change between REQ and ACK. In this case, we want to choose the most current route. Figure 3 compares the performance of the two schemes. The tree created by ACK messages achieves higher throughput performance (throughput: number of packets received by members) under high mobility. The improvement is relatively small, however, since the mobility is not high enough to cause significant route changes during the REQ-ACK round trip.

B. Multicast Tree reconfiguration

The core-based tree is not static since the core and the host group may move. The multicast tree will be reconfigured in the following cases: (1) The member of a host group moves and changes node type. (2) Tree links break and transient loops are created.

B.1. Member migration It is necessary to reconfigure the multicast tree when its group members move or change node-type. A group member can detect the change of its multicast tree by monitoring its connectivity to upstream and downstream members. A member node reconnects to the tree by sending a JOIN_REQUEST to its multicast server (core) when its upstream member moves out

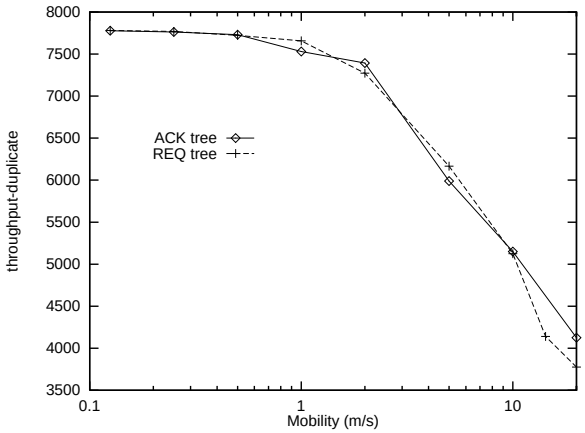

Fig. 3. Performance comparison of ACK tree and REQ tree

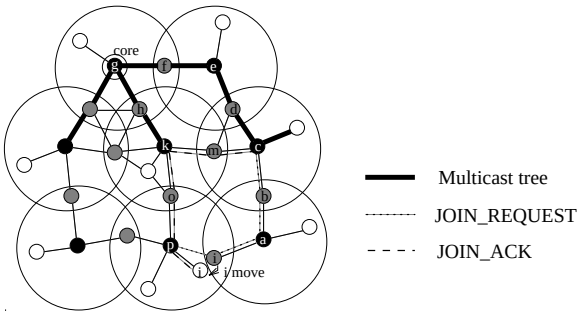

Fig. 4. Loop example

of range or changes node type. For example a clusterhead member will send a JOIN_REQUEST to the MSid in order to reconstruct the tree, if its upstream member (a gateway) changes to a regular node, or becomes disconnected. When a regular node member (a leaf) moves out of a cluster C_i and enters into a 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 of C_j. The clusterhead of C_i will send a QUIT_REQUEST to its upstream member if it has become itself a leaf.

B.2. Loops When a node i wants to join a multicast tree G, it sends a JOIN_REQUEST to the core. The JOIN_REQUEST will be acknowledged by the first member in G, which sends back a JOIN_ACK to node i. If node i has moved in the interim, the JOIN_ACK may trace a different path than the JOIN_REQUEST. Thus a loop may be formed. Figure 4 shows an example of loop caused by the move of node i. Node i, before the move, sends the JOIN_REQUEST to the core on the path a, b, and c. Node c, the first member in G on the path to the core , returns the JOIN_ACK to node i. However, since node i has moved, the new path m, k, o, and p is traced, thus forming the loop c, d, e, f, g, h, k, and m. To avoid loops, it is required that an established group member, upon receiving a JOIN_ACK, return a QUIT_REQUEST to the node which sent this JOIN_ACK while forwarding the JOIN_ACK on the new path. In figure 4, for example, node k will send a QUIT_REQUEST to node m after it receives a JOIN_ACK. At the same time node k will forward JOIN_ACK to node o on the new path creating a loop-free branch to node i. We can generalize this loop avoidance method as follows: a group member already connected to the multicast tree will acknowledge a JOIN_ACK with a QUIT_REQUEST, if this JOIN_ACK does not come from its upstream member. Since each group member in a tree can only have one upstream member, a necessary and sufficient condition for loop avoidance is to allow only one upstream member.

VI. Multicast experiments

We have implemented the multicast protocol in our wireless simulator in order to evaluate its performance in terms of : (a) control packet overhead; (b) robustness to mobility; (c) scaling properties with respect to multicast group membership, and; (d) response time (i.e. JOIN latency). The environment consists of 100 mobile hosts roaming in a 1000x1000 ft square (as described in Section 4). The wireless network operates using the LCC clustering algorithm and the cluster token access protocol. As for routing, CGSR is used, unless otherwise specified.

Figure 5 shows the initial multicast tree layout, with 7 members plus core. The core is hand-picked. Based on CGSR and clustering properties, the core is a clusterhead and never gives up this role. That is, the core will not change to a non-clusterhead node. Unless otherwise specified, we assume that membership is fixed. As members move, they leave one branch of the multicast tree and join another. Furthermore, as nodes move, the routes change, thus causing a dynamic reconfiguration of the tree topology. It is thus important to measure the control packet overhead caused by these reconfigurations as a function of node speed. In these experiments, the main focus is on algorithm response time and control packet overhead. Thus, the network does not carry any user traffic (only control traffic) to avoid interference between user packets and control packets.

Figure 6 shows total number of tree reconfigurations during the experiment lifetime as a function of node speed (up to 9 m/s). We note that the number of reconfigurations (i.e., changes in the tree) grows about linearly with speed. In figure 6 we also report the number of JOIN, ACK and QUIT packets. While the first two grow almost linearly with speed, the third is not very speed sensitive. Also, the QUIT event is much less frequent than JOIN/ACK. The reason is that QUIT is issued by a clusterhead or a gateway only when it has no members below it.

Figure 7 shows the number of temporary loops detected and removed. This number grows with speed, but in an erratic fashion due to the very small sample size. In any event, loop detection and recovery does not cause significant overhead. Figure 8 reports the reconfiguration and control packet measurements as a function of membership size. Node speed is assumed fixed at 5 m/s. Control traffic grows with membership size, as expected, but less than linearly, since the tree route reconfiguration is independent of membership size. Furthermore, the number of JOIN/ACK packets generated when a member moves from one cluster to another decreases when size increases since there are more members in the tree and the JOIN packet must traverse fewer hops up the tree. On the other hand, the number of moves from one cluster to another increases linearly with the number of members. In balance, we have slightly increasing control packet (ACK/JOIN) traffic for membership ranging from 7 to 80. The number of QUIT packets decreases with membership size, since, when almost all nodes participate in the multicast, no clusterhead or gateway will ever become childless and thus be forced to quit. In summary, the control packet overhead required to maintain the multicast tree does not increase significantly with member group size.

Figure 9 shows packets O/H growth with node speed. Packet overhead is computed using the formula $(T \times NP)/(ST \times NC)$, where NP = number of control packet transmission (ACK, JOIN, QUIT); T = control packet transmission time ($5ms$); NC = average number of clusters ($\simeq$ 27), and; ST = total simulated time ($62.5s$ for our experiments). Thus, the overhead represents the fraction of bandwidth used up by control packets. We note that the growth is less than linear, consistent with figure 6 results. Furthermore, the overhead is only a few percent, even at top speed.

The responsiveness of a multicast scheme with dynamic join

can be measured by the latency of a JOIN operation, i.e. the time between the issue of a JOIN request by a new member and the receipt of an ACK. Intuition suggests that latency should increase with speed. The results in figure 10, however, seem to indicate that latency is rather insensitive to speed, at least for the range of speeds considered in our study. For example, using the CGSR routing scheme, JOIN latency is less than 600 ms for the entire range of speeds. Figure 10 also reports latency under the conventional DSDV routing scheme. It is interesting to notice that CGSR performs considerably better than DSDV. The latency reduction in CGSR can be attributed to the clever token and code scheduling heuristics.

In summary, the result show that the proposed multicast scheme meets the target performance goals. Namely, it is robust to mobility (latency is insensitive to speed; overhead increases less than linearly with speed); it has low overhead (less than a few percent at top speed); it scales well with member group size, and; it has very low latency (less than 1 s).

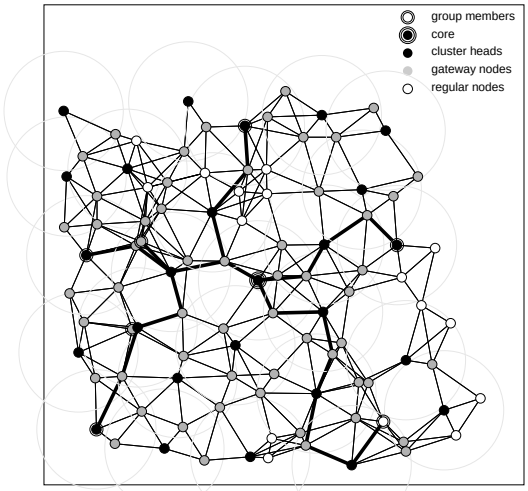

Fig. 5. Initial multicast tree

VII. Conclusion

The two principal contributions of this paper are:(1) the multicast protocol, and; (2) the wireless network infrastructure which supports it. The multicast strategy is inspired by the CBT (Core Based Tree) Internet scheme. It is robust to mobility(low JOIN latency up to 10 m/s); it introduces extremely low control overhead (typically, less than 1%); it scales well with number of nodes and with multicast group size (up to the hundreds). By virtue of the use of CBT, it can be easily interfaced with an Internet CBT.

While the proposed multicast strategy is independent of the particular wireless infrastructure (ie, routing, MAC and cluster layers) in use, it has been developed on top of a novel wireless, multihop infrastructure for the purpose of evaluating its performance. The underlying infrastructure itself is innovative and in many aspects improves upon existing architectures. In particular, the LLC clustering algorithm was proven to be more robust to mobility than existing schemes. The clusterhead controlled token MAC layer allows flexible priorities and powerful heuristics. The hierarchical routing scheme provides a solution with low overhead and potential for scalability to very large networks.

Future research directions include: (1) the dynamic relocation of the CORE;(2) the extension of the Internet (or ATM) multicast tree solutions to the wireless segments, and; (3) QoS multicasting.

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, 1996.

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

[7] 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.

[8] J. J. Garcia-Luna-Aceves. A unified approach to loop-free routing using distance vectors or link states. In *SIGCOM*, pages 212–223. ACM, 1989.

[9] M. Gerla and C.-C. Chiang. Multicast routing in multihop, mobile wireless networks. Technical report, UCLA-CSD, Sept. 1996.

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

[13] D. B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proc. of Workshop on Mobile Computing and Applications*, 1994.

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

[15] C. R. Lin and M. Gerla. A distributed architecture for multimedia in dynamic wireless networks. In *IEEE International Conference on Communications (ICC'95)*, pages 1468–1472. IEEE, 1995.

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

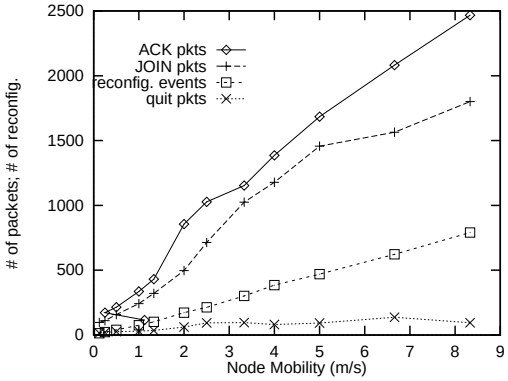

Fig. 6. Reconfig. & control pkts vs group size

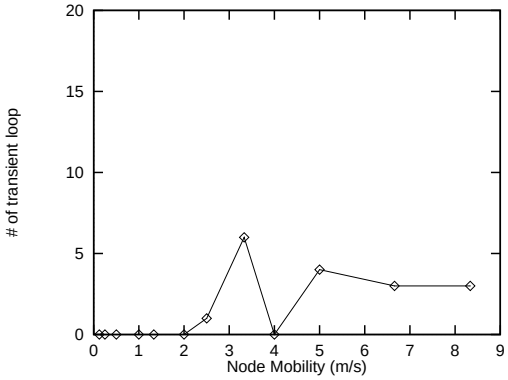

Fig. 7. Total # of transient loops

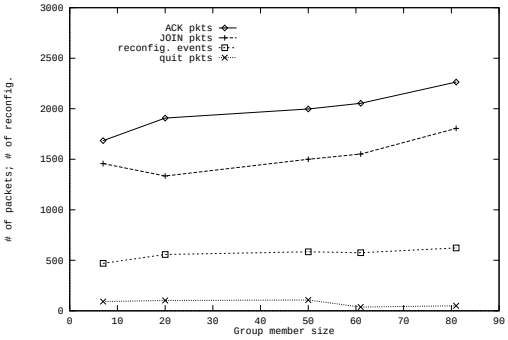

Fig. 8. Reconfig. & control pkts vs group size

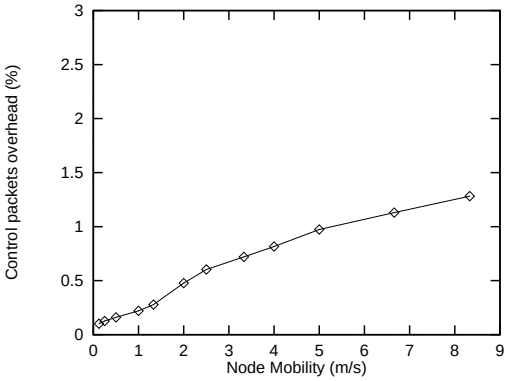

Fig. 9. Control packet O/H

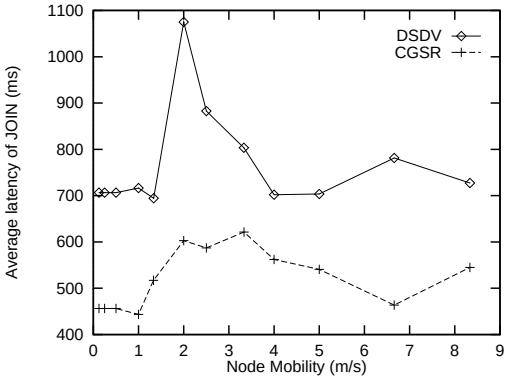

Fig. 10. Average JOIN latency