A Scenario-Based Performance Evaluation of Multicast Routing Protocols for Ad Hoc Networks

Manoj Pandey and Daniel Zappala
Computer Science Department
Brigham Young University
Provo, UT 84602-6576
{manoj, zappala}@cs.byu.edu

Abstract

Current ad hoc multicast routing protocols have been designed to build and maintain a tree or mesh in the face of a mobile environment, with fast reaction to network changes in order to minimize packet loss. However, the performance of these protocols has not been adequately examined under realistic scenarios. Existing performance studies generally use a single, simple mobility model, with low density and often very low traffic rates. In this paper we explore the performance of ad hoc multicast routing protocols under scenarios that include realistic mobility patterns, high density and high traffic load. We use these scenarios to identify cases where existing protocols can improve their performance. Based on our observations, we make a series of recommendations for designers of multicast protocols.

1 Introduction

Mobile ad hoc networks have numerous practical applications, such as emergency and relief operations, military exercises and combat situations, and conference or classroom meetings. Each of these applications can potentially involve different *scenarios*, with movement pattern, density and traffic rate dependent on the environment and the nature of the interactions among the participants. For example, in a search-and-rescue operation, individuals may fan out to search a wide area, resulting in a fairly regular pattern of movement, low density, and low traffic rate. In a battlefield scenario, the movement of soldiers may be heavily influenced by the movements of their commander, with higher density and a higher traffic rate. In other cases, the environment itself may give rise to movement patterns and density, such as patrons visiting an exhibit hall and moving among a selected group of displays. In addition, depending upon the communication need, applications can be very demanding, requiring the system to support very high traffic rates.

To enable group communication in these scenarios, a number of ad hoc multicast routing protocols have been proposed [15, 10, 22, 12, 14, 5, 18, 24, 3]. In making the transition from wired to wireless networking, protocol designers have focused on the obvious challenge of designing a multicast routing protocol that can cope with a mobile environment. As a result, the main goal of most ad hoc multicast protocols is to build and maintain a multicast tree or mesh in the face of a mobile environment, with a fast reaction to network changes so that packet loss is minimized.

While most ad hoc multicast protocols have met this basic design goal, their performance has not been adequately examined under realistic scenarios. Existing performance studies in this area suffer from three common flaws:

- *Simplistic mobility models.* Existing studies use either a Uniform mobility model [16] or the Random Waypoint model [10, 22, 12]. It is well known that because these models utilize random, independent movements they do not reflect realistic usage patterns. A number of researchers have evaluated unicast routing performance under a variety of mobility patterns [13, 9, 1], using more elaborate models and metrics to capture their effect on routing performance. Our study is the first to apply this methodology to examine multicast routing performance.

- *Low density.* Many evaluations use only 50 mobile nodes in a $1000m^2$ square field and a $250m$ radio range, which often leads to a density of less than 10 nodes within radio range [16, 12, 14, 5]. However, there are many common scenarios in which a network may have many more users in a small area – any situation in which there is a planned gathering or a crowd. This density will likely result in congestion and packet loss; because some protocols use packet loss as an indicator for mobility they may react in precisely the wrong way. Moreover, multicast protocols are often built on top of broadcast, such as for source discovery or tree repair,

Proceedings of the Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM'05)
0-7695-2342-0/05 $20.00 © 2005 IEEE

and these mechanisms will cause excessive overhead in high density scenarios.

- *Low traffic load.* Current evaluations generally employ a very low data rate of 2 to 20 kbps [10, 14, 18, 24, 3], with only a few using rates as high as 80 to 200 kbps [12, 19, 15]. Clearly, a protocol that performs very well at lower traffic rates may not be able to handle higher traffic rates efficiently. In addition, while some theoretical work has examined the capacity of ad hoc networks [8, 17], no one has examined the fundamental question of whether multicast routing protocols can actually reach these limits.

In our study, we try to rectify these shortcomings by examining the performance of two common multicast routing protocols – ODMRP [15] and ADMR [10] – under scenarios that include *realistic mobility patterns*, *high density*, and *high traffic load*. We use a simulation-based performance evaluation so that we can thoroughly examine protocol behavior in a controlled environment. Our goal in this work is to impact how future multicast routing protocols are designed by identifying general cases where existing protocols can improve their performance. Accordingly, we identify specific mechanisms that cause performance bottlenecks, then generalize our experiences into a set of recommendations for multicast protocol designers.

We first explore multicast routing performance using realistic mobility models, similar to those described in the IMPORTANT framework [1] and other unicast evaluations. To the best of our knowledge, this is the first time that multicast routing protocols have been examined over a wide range of mobility patterns. Our goal is to sort through the overwhelming number of models and metrics and identify the key areas where protocol designers should focus their attention.

In studying this scenario, we show that multicast performance is largely predicted by two key mobility metrics – the frequency of link breaks and the density of the mobility pattern. This enables protocol designers to focus on optimizing performance with respect to these metrics, regardless of the particular usage scenario. We find that ODMRP does not react quickly to link breaks because it must rely on a periodic broadcast from the source in order to find members that have moved. ADMR reacts more quickly, but at the cost of much higher transmission and control overhead. There is certainly room for a protocol that achieves better tradeoffs.

Our second scenario examines multicast performance under high density, which we believe will be a common case for ad hoc networks. While ODMRP handles this scenario very well, ADMR throughput is severely impacted by density. In this scenario, ADMR experiences congestion collapse at about 25 kbps, performing even worse than flooding, while ODMRP can achieve rates in excess of 200 kbps. We analyze ADMR's poor performance and show that it is due to its use of explicit acknowledgments and its assumption that all packet loss is a sign of mobility. We design several modifications of ADMR that correct this problem and dramatically increase throughput in this scenario without affecting its performance in other situations.

Our last scenario examines multicast performance under high traffic load. We find that ODMRP performs significantly better than ADMR, but both experience high packet loss as the traffic rate increases. Packet size plays an important role in reaching high capacity – sending fewer large packets is generally better than sending many small packets. We then extend recent theoretical results for the capacity of ad hoc networks to the case where multicast traffic is transmitted, and find that the number of hops in the multicast tree is a critical factor. We argue that for this scenario it is better to use a multicast tree rather than a mesh.

2 Background

We have chosen to study the performance of ODMRP and ADMR in these scenarios because they operate *on-demand* rather than proactively maintaining routes. Several performance studies indicate that these protocols perform well [16, 10]. In addition, ODMRP is mesh-based whereas ADMR is tree-based, providing us a perspective on both types of protocols.

2.1 ODMRP

ODMRP is a mesh-based demand-driven multicast protocol, similar to DVMRP [23] for wired networks. A source periodically builds a multicast tree for a group by flooding a control packet throughout the network. Nodes that are members of the group respond to the flood and join the tree. Nodes that are on the tree use *soft state*, meaning their status as forwarders for a given group times out if not refreshed. Because the forwarding state is shared among all sources for a given group, the set of forwarders for a group forms a mesh, providing robustness for the mobile receivers.

In particular, an active ODMRP source periodically floods a JOIN QUERY message throughout the entire network. Each node receiving this message stores the previous hop from which it received the message. When a group member receives the JOIN QUERY, it responds by sending a JOIN REPLY to the source, following the previous hop stored at each node. Nodes that forward a JOIN REPLY create soft forwarding state for the group, which must be renewed by subsequent JOIN REPLY messages. If the node is already an established forwarding member for that group, then it suppresses any further JOIN REPLY forwarding in order to reduce channel overhead.

The basic trade-off in ODMRP is between throughput and overhead. A source can increase throughput by sending more frequent JOIN QUERY messages. Each message

rebuilds the multicast mesh, repairing any breaks that have occurred since the last query, thus increasing the chance for subsequent packets to be delivered correctly. However, because each query is flooded, increasing the query rate also increases the overhead of the protocol. ODMRP can also control redundancy via the soft-state timer for node forwarding state. A longer timer will increase the size of the mesh and hence provide more redundant paths for packets to be delivered. Of course, increasing the soft-state timer also increases overhead since many of the links in the mesh will result in duplicate packets being delivered.

2.2 ADMR

ADMR creates source-specific multicast trees, using an on-demand mechanism that only creates a tree if there is at least one source and one receiver active for the group. Unlike ODMRP, receivers must explicitly join a multicast group. Sources periodically send a network-wide flood, but only at a very low rate in order to recover from network partitions. In addition, forwarding nodes in the multicast tree may monitor the packet forwarding rate to determine when the tree has broken or the source has become silent. If a link has broken, a node can initiate a repair on its own, and if the source has stopped sending then any forwarding state is silently removed. Receivers likewise monitor the packet reception rate and can re-join the multicast tree if intermediate nodes have been unable to reconnect the tree.

To join a multicast group, an ADMR receiver floods a MULTICAST SOLICITATION message throughout the network. When a source receives this message, it responds by sending a unicast KEEP-ALIVE message to that receiver, confirming that the receiver can join that source. The receiver responds to the KEEP-ALIVE by sending a RECEIVER JOIN along the reverse path.

In addition to the receiver's join mechanism, a source periodically sends a network-wide flood of a RECEIVER DISCOVERY message. Receivers that get this message respond to it with a RECEIVER JOIN if they are not already connected to the multicast tree.

Each node begins a repair process if it misses a defined threshold of consecutive packets (2 for our simulations). Receivers do a repair by broadcasting a new MULTICAST SOLICITATION message. Nodes on the multicast tree send a REPAIR NOTIFICATION message down its subtree to cancel the repair of downstream nodes. The most upstream node transmits a hop-limited flood of a RECONNECT message. Any forwarder receiving this message forwards the RECONNECT up the multicast tree to the source. The source in return responds to the RECONNECT by sending a RECONNECT REPLY as a unicast message that follows the path of the RECONNECT back to the repairing node.

Nodes on the multicast tree also maintain their forwarding state. They expect to receive either passive acknowledgments (if a downstream node forwards the packet) or an EXPLICIT ACKNOWLEDGMENT if it is a last hop router in the tree. If a defined threshold of consecutive acks are missed (15 for our simulations), then the forwarding node expires its state.

Finally, a receiver keeps track of how many times it has had to initiate a repair due to a disconnection timeout. If this number reaches a certain threshold then the receiver asks the source to switch to flooding in its next RECEIVER JOIN message. If enough receivers ask, then the source switches to flooding for a limited time. During flooding, all the data packets are sent as network-wide flood and all repair messages are suppressed.

3 Simulation Methodology

For our simulations we use GloMoSim-2.03 [26]. We fixed the ODMRP implementation provided in the GloMoSim distribution because it contained several major bugs that prevented packets from being delivered. Since GloMoSim did not include ADMR, we wrote our own implementation based on the original ADMR publication [10] and a specification published as an Internet draft [11]. We did not implement source pruning (where the source stops sending data if there are no receivers) so that we could study the effects of partitioning on packet loss. We also wrote a simple flooding protocol for GloMoSim.

One important implementation detail for multicast routing protocols is the use of randomization (or jitter) to avoid collisions due to protocol synchronization. Each node that forwards a multicast message adds a random delay between 0 and 10 ms before forwarding the packet. Likewise, at the application layer, we avoid starting sources at the same time, since they use CBR and would thus remain synchronized for the duration of the simulation. Finally, for ADMR we exclude any startup delay caused by buffering packets before sufficient receivers have joined the group.

To verify our protocol implementations, we ran simulations identical to those reported in [10] that compare ADMR and ODMRP. Our results are very close, with slightly higher delay due to the jitter we have added at each node. The results reported in this paper differ more substantially from [10] because we are using a different field size.

In our simulations we collect the following metrics:

- **Throughput (%)**: The ratio of the number of packets received to the number of packets sent.
- **Throughput (rate)**: The rate at which group members receive data, in kilobits per second.
- **Transmission Overhead**: The ratio of the number of data messages transmitted (originated or forwarded) and the number of data messages received. This metric is a measure of the efficiency of a routing protocol

– a lower value for transmission overhead indicates that fewer forwarders were needed.

- **Control Overhead**: The ratio of the number of control messages originated or forwarded over the combined total of data and control messages originated or forwarded. This metric indicates the percentage of all messages that are control messages.

- **Delay**: The difference between the time when the packet is sent by the source and when it is received.

Note that previous studies have combined transmission overhead and control overhead into a single metric called normalized overhead, but this can obscure the differences between the two.

For ODMRP, control packets consist of JOIN QUERY, JOIN REPLY, and ACKNOWLEDGMENTS. For ADMR, control packets consist of RECEIVER DISCOVERY, MULTICAST SOLICITATION, KEEP-ALIVE, RECEIVER JOIN, REPAIR NOTIFICATION, RECONNECT, RECONNECT REPLY, and also ACKNOWLEDGMENTS sent by the end receivers in order to maintain the tree. Since ODMRP JOIN QUERIES and ADMR RECEIVER DISCOVERY messages have data piggybacked with them, we count these packets as both data and control messages.

Except where noted, our simulations use 50 nodes, randomly placed over a square field whose size is $1000m^2$. Nodes communicate using IEEE 802.11 for the MAC protocol, with a $250m$ radio range, free-space radio signal propagation, and a maximum data rate of 2 Mbps.

4 Mobility Scenarios

Our first scenario explores the effects of realistic mobility patterns on multicast routing performance. We use mobility models similar to the IMPORTANT framework [1], but apply this methodology for the first time to multicast. We consider a wide range of mobility models and explain how they impact multicast routing performance. As a result, we are able to identify two key mobility metrics that predict multicast performance. This enables protocol designers to focus on optimizing performance with respect to these metrics, regardless of the particular usage scenario.

4.1 Mobility models and connectivity metrics

A number of researchers have developed mobility models to capture the movement patterns of wireless network users. We have chosen a set of models from three different classes of motion – random, path-based, and group-based movements:

- *Uniform*: Each node starts at a random position and moves in a random direction with a constant velocity [16]. This models independent movement with high temporal dependency.

- *Random Waypoint*: Each node chooses a random destination within the simulated field, moves to the destination at a randomly chosen speed, pauses for a fixed period of time, and then chooses a new destination. This model has been widely used in the literature, allowing us to validate our results with other research.

- *Manhattan*: Each node moves along a set of pre-defined streets, which are arranged in a grid pattern. All nodes use the same speed, and each node may choose any direction when reaching an intersection. This models path-based motion with low spatial dependence [1, 6].

- *Exhibition*: Each node chooses a destination from among a fixed set of exhibition centers and then moves toward that center with a fixed speed. Once a node is within a certain distance of the center it pauses for a given time and then chooses a new center. This model is similar to the event scenario described by Johansson et al. [13] and represents independent movement but with high node density.

- *Battlefield*: Each node follows a group leader by choosing a destination close to where the leader is currently located and then moving there. The group leader uses the Random Waypoint model. This is similar to the RPGM model [9] and represents group-based mobility with high spatial dependence.

Model	Parameters
Random Waypoint	speed: between max and $max - 5m/s$ pause time: 30 seconds
Manhattan	grid size: 150 meters
Exhibition	centers: 10 minimum distance to center: 20 meters pause time: 30 seconds
Battlefield	leaders: 16 minimum distance to leader: 20 meters pause time for leader: 30 seconds

Table 1. Parameters used in Mobility Models

Table 1 lists the default parameters we use for each of these models. For each of these models we vary the speed at which nodes move. For all models we avoid sharp turns and sudden changes in velocity by using acceleration and deceleration vectors [2]. As part of this work we extended GloMoSim to include the Uniform, Manhattan, Exhibition, and Battlefield mobility models.

To differentiate among these models, we use a set of mobility and connectivity metrics. Of the mobility metrics defined in the IMPORTANT framework [1], we found that spatial dependency and the average number of link changes are

able to differentiate between our mobility models and help to explain multicast routing performance. These are defined informally as:

- *Spatial dependency:* The degree to which two nodes are moving in a similar direction with similar speed.
- *Number of Link Changes:* The number of link changes seen during the course of a simulation. A link change is counted when two nodes come within radio range when previously they had not been able to communicate directly, and vice versa.

In addition, because we are studying multicast routing, we introduce two new metrics:

- *Neighbor Density:* The number of nodes which are within radio range of a given node. We do not distinguish between active and inactive nodes, because with multicast a node that is not transmitting its own data can still act as a forwarder.
- *Reachability:* The number of nodes that are reachable via forwarding through the ad hoc network. We measure this using a recursive coloring algorithm, so that nodes in the same partition have the same color.

We used a set of simulations to confirm that each of these metrics is able to distinguish between the mobility models. Of the most importance to us, the number of link changes clearly differentiates among the models (Figure 1). This behavior was not seen with the IMPORTANT framework [1], but it is significant because the number of link changes can have a significant impact on a multicast routing protocol – each link change may potentially break the multicast tree or mesh and cause a receiver or intermediate node to attempt a repair. Both spatial dependency and density are higher for the group-based mobility models (Exhibition and Manhattan), as is expected, though the difference is clearer with density (Figure 2). Note that density is initially low for Battlefield and Exhibition because at low speeds the nodes do not have time to form groups. Another significant difference among the protocols is that all of them maintain high reachability except for Battlefield [20]. This means that the Battlefield model is a good test for determining how well a protocol reacts to a partition in the network.

4.2 Mobility metrics explain routing performance

For the mobility scenarios, we try to isolate the pattern of node movement from other factors, such as the traffic rate. Hence we use only three multicast groups, each consisting of 1 source and 7 receivers. The multicast groups are not overlapping, which means that with the 3 senders and 21 receivers combined about half of the nodes are participating

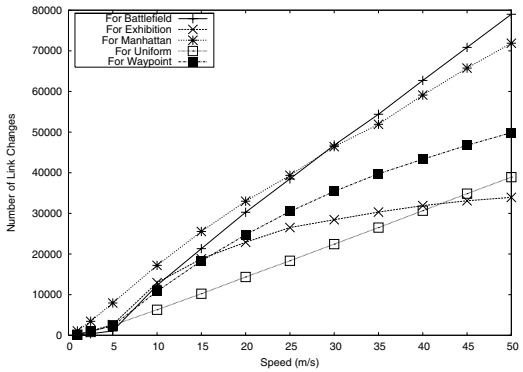

Figure 1. Mobility model link changes

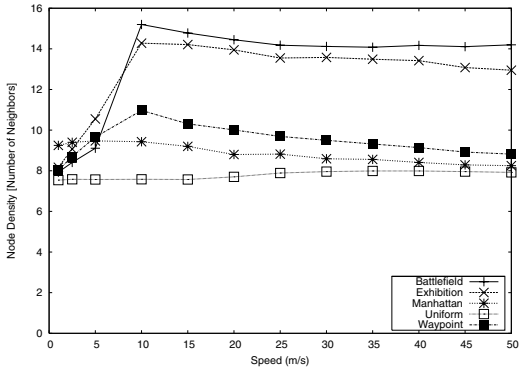

Figure 2. Mobility model density

in a multicast session. Each multicast source uses a Constant Bit Rate (CBR) flow, transmitting a 64 byte packet every 250 milliseconds (6 kbps). We run each simulation for 600 seconds and we average the results of 25 simulations.

For both ODMRP and ADMR, throughput depends on the model (Figures 3 and 4) and the ordering from worst to best can be explained by both the number of link changes and the density. Throughput is worst for Battlefield because it creates the most link changes, to the point that the network frequently becomes partitioned. At moderate to high speeds, approximately 10 nodes are separated from the rest of the network with the Battlefield model. The Manhattan model has a similar large number of link changes and also low density, so it is the next worst. Uniform and Random Waypoint result in similar performance; although Uniform has fewer link changes this is balanced by Random Waypoint having higher density. Finally, the Exhibition model results in the best performance because the number of link changes is relatively low and it creates high density.

Note that for ODMRP our results show lower throughput than with previous ODMRP simulations [16] because these previous simulations are at low speed ($20m/s$), use a lower data rate, and try only the Uniform mobility model. Likewise, we show lower throughput for ADMR than with previous ADMR simulations [10] because the previous simulations use a elongated field ($1500m\ x\ 300m$) that results

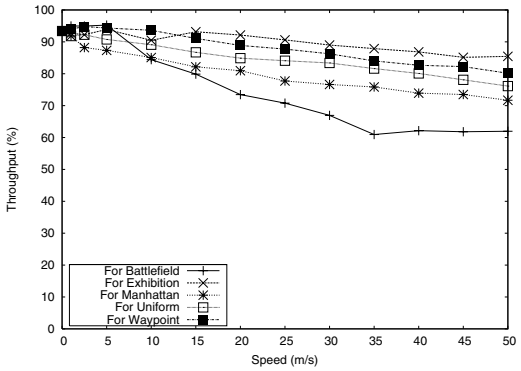

Figure 3. ODMRP throughput

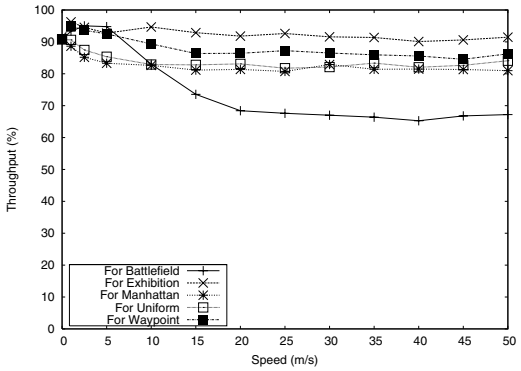

Figure 4. ADMR throughput

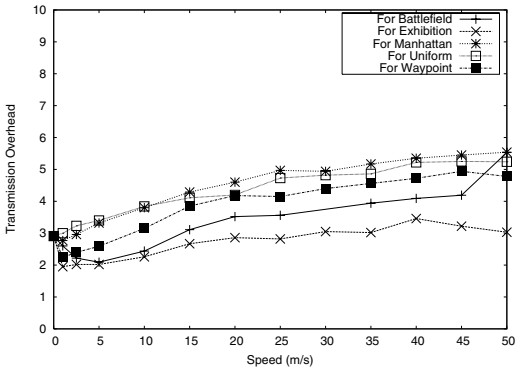

Figure 5. ADMR transmission overhead

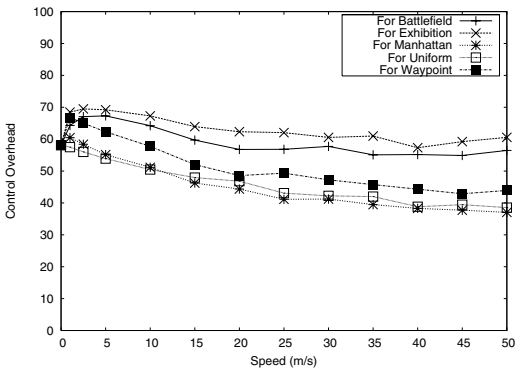

Figure 6. ADMR control overhead

in higher density and are limited to the Random Waypoint model.

Regardless of the mobility model, ODMRP performance degrades as speed increases, whereas ADMR is able to maintain throughput greater than 80%. Based on our results, we calculated Pearson's correlation coefficient and confirmed that there is a strong negative linear relationship between the number of link changes and throughput for ODMRP. Note that ODMRP could obtain better performance by sending JOIN QUERY messages more frequently, and hence reacting to broken links more quickly, but this would in turn incur more overhead. Recent revisions of ODMRP use GPS to track location and adaptively increase the refresh interval as mobility increases. ADMR is able to maintain high throughput because (a) forwarding nodes are able to initiate local repair of the multicast tree and (b) receivers experiencing high packet loss can ask ADMR to switch to flooding. However, the cost of these actions are higher control and transmission overhead.

One of our important findings is that the density created by each of the mobility models explains the performance of the rest of our performance metrics. For both ODMRP and ADMR, the transmission overhead, control overhead, and delay varies according to the mobility model, with the ordering among the models correlates exactly with the ordering for density. We show representative graphs for ADMR in Figures 5 and 6. Group-based mobility models, which lead to higher density, result in a greater chance that multicast group members will be located near the source. This leads to a savings in transmission overhead and delay. High density also decreases control overhead for ODMRP, since JOIN REPLY messages travel fewer hops. For ADMR, however, control overhead increases with density (Figure 6). This happens because ADMR switches to flooding more frequently when density is low [20]; during these times ADMR has no control overhead.

These results confirm that characterizing link variations and density fluctuations for any user movement is crucial towards understanding routing performance. We also include reachability as a key metric since partitioning is a special case of a link breaking. Additional metrics defined in the IMPORTANT framework, such as temporal dependence, spatial dependence, and relative velocity, are important only in that they induce link changes and density. It is possible, for example, to construct a model that creates spatial dependence and yet few link changes. Each node in this strawman model simply vibrates in place, with the amplitude of vibration significantly smaller than the radio range. Given sparse placement, density variations are negligible and since movements are small no link breaks occur. However, depending upon the alignment of the vibrations and relative velocities, spatial dependence between two nodes can be varied arbitrarily.

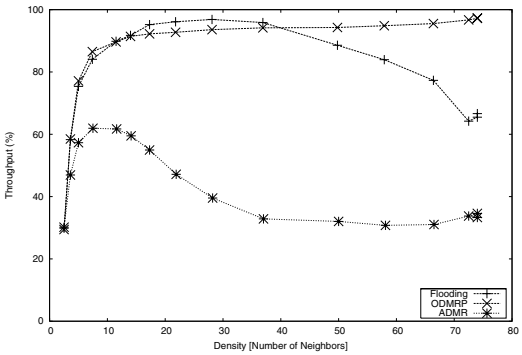

Figure 7. Variation of throughput with density

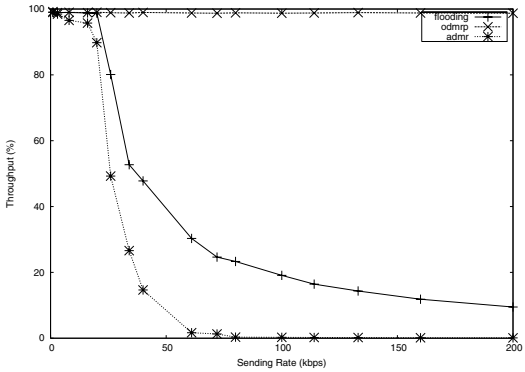

Figure 8. High density throughput

Hence, we can conclude that protocol designers should concentrate on optimizing multicast protocols for both frequent mobility and density.

5 High Density Scenario

Our second scenario explores the effects of high density on multicast routing performance. For this scenario, we study density without any mobility. Accordingly, we statically place a set of 75 nodes on a field and vary the density by varying the size of the field. We use three multicast groups, each having 1 source and 22 receivers. To see the effects of density, we increase the traffic rate of each source slightly; each source sends a 100 byte packet every 100 milliseconds (24 kbps). We run each simulation for 100 seconds and we average the results of 25 simulations.

Surprisingly, ADMR performs very poorly in this scenario (Figure 7). Both ODMRP and ADMR initially receive low throughput as the density of the network is so small that it is partitioned. As the density increases, ODMRP achieves very high throughput once the network is connected, while ADMR never delivers more than 60% of the packets. In fact, ADMR does much worse than a simple flooding protocol.[1] This indicates that ADMR's ability to switch to flooding is not causing this problem.

To explore this problem further, we fix the density at a high value and then vary the traffic rate to determine when ADMR encounters a problem. To accomplish this, we randomly place 50 nodes within 20 meters of a central point, again with no mobility. We use only a single multicast group, with 1 sender and 30 receivers. We vary the traffic rate for this source by keeping the packet size fixed at 100 bytes and adjusting the number of packets sent per second. We run each simulation for 60 seconds and we average the results of 25 simulations.

In this scenario, ADMR's throughput begins to collapse when the sending sending rate is only about 25 kbps (Figure 8). This is even worse than flooding, which must forward each packet 50 times! Because of its simplicity, ODMRP operates extremely well, even at extremely high rates. ODMRP's only control traffic is a periodic JOIN QUERY; while this message is flooded, it is transmitted once every 3 seconds, regardless of the sending rate. The subsequent JOIN REPLY messages, one per receiver, form a tree shaped like a star. This means that each data packet is simply broadcasted by the source and then immediately received by all group members.

To pinpoint ADMR's problems with high density, we plot a trace of the major network activity when the traffic rate is 200 kbps (Figure 9). The graph is divided into three sections, with each section being a different version of ADMR. The bottom section of this plot shows the standard version of ADMR, before we have fixed any of ADMR's problems under high density. Each symbol on the graph represents a packet being transmitted, with the y-axis indicating which node sent the packet. Node 0 is the sender for the group. To make the trace readable, we show only the first 7 seconds of network activity; the rest of the 60 second trace continues with exactly identical patterns to those shown here.

The first problem evident from this trace is that ADMR suffers from RECEIVER JOIN implosion. When the source starts at approximately 1300 ms, it transmits a RECEIVER DISCOVERY message. All existing group members immediately respond with a RECEIVER JOIN message, which overwhelms the source. The source responds with a separate UNICAST KEEP-ALIVE message for each receiver, but these messages are delayed due to congestion.

However, the primary problem in this case is that ADMR sets a timer for each RECEIVER JOIN message that is based on the inter-packet gap advertised by the source. Because the inter-packet gap is so small in this simulation (4 ms), the timer fires faster than the source can respond and ADMR assumes the join attempt has failed. Hence, when the source sends its RECEIVER DISCOVERY message, each receiver sends three RECEIVER JOIN messages, then gives up and sends a MULTICAST SOLICITATION message. Eventually, the source delivers a UNICAST KEEP-ALIVE message to a

[1] For our implementation of flooding, we use a simple protocol in which each node receiving a packet for a group first checks whether it is a duplicate and, if not, forwards the packet by retransmitting it.

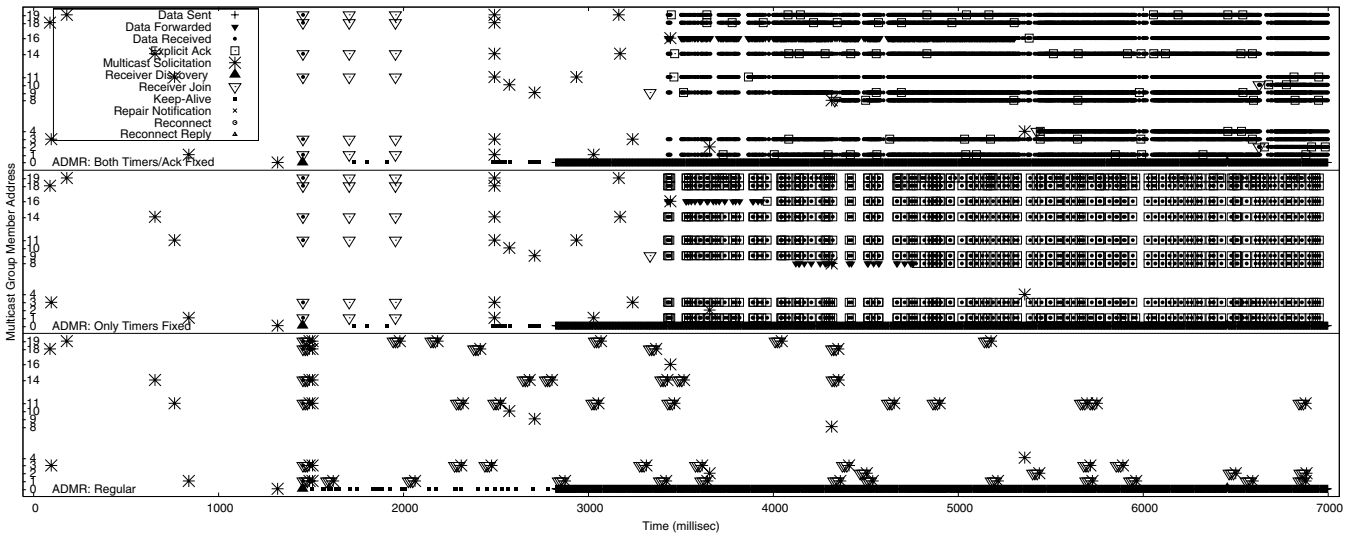

Figure 9. High density ADMR packet trace

receiver, and this causes another round of triple RECEIVER JOIN messages followed by solicitations. Because the solicitations are flooded and sent using broadcast at the MAC layer, they cause severe congestion; 10 solicitations in a second result in 500 control packets transmitted during that time. This congestion forces the source to continually backoff. Hence, even though the source queues packets in the MAC layer at about 2800 milliseconds, many of them are never sent. Even when packets are sent, few of the group members successfully join for any length of time – the throughput is 0.07%. Later in the trace (not shown), Repair timers have the same problem as the Join timers, resulting in additional MULTICAST SOLICITATION messages.

Fixing this problem requires only a small adjustment to the ADMR Join and Repair timers. We establish a *RepairWaitTime*, so that the calculated timer can never be less than this value. At large inter-packet gaps (low sending rates), the original timer value is used, but at higher sending rates the timers are set to their minimum values. By setting the *RepairWaitTime* to a reasonably short time (e.g. 500 ms), we can ensure that the timer never fires too soon (this is the equivalent of 125 missed packets for our packet trace), but is fast enough to adapt to mobility-induced losses.

The middle section of the packet trace in Figure 9 shows network activity for ADMR with the Join and Repair timers fixed, using a value of $500ms$ for the *RepairWaitTime*. This dramatically reduces the number of RECEIVER JOIN messages that are sent, which allows data to be transmitted. Note that most group members can now receive data, though some members that join later have difficulty joining because of the congestion in the network.

This reveals a second serious problem with ADMR in the high density scenario: each receiver transmits explicit acknowledgments to the source, resulting in ack implosion. Like any ADMR forwarder, the source must receive either a passive or active acknowledgment to maintain its forwarding state. Since the forwarding tree is actually a star at high density, all receivers are the last hop in the tree and all must send an EXPLICIT ACK for each packet. Hence, the throughput for this case improves to only 12.92% as many packets are either delayed or lost due to the congestion from explicit acks. Moreover, some nodes actually become forwarders for a short time, as seen for nodes 8 and 16 in the middle section of the trace. This occurs when an intermediate node receives a MULTICAST SOLICITATION before the source.

Our solution for the ack implosion problem is based on the observation that this problem is very similar to the ack implosion problem in reliable multicast [7]. In the wireless case, a source (or any other forwarder in a dense region) only needs to receive one acknowledgment for a packet in order to maintain its forwarding state. Even better, if the source (or forwarder) can receive one ack for every k packets, then it can maintain its forwarding state with a minimum of overhead.

Damping the explicit acks in ADMR is simple, since each ack is broadcasted. Each group member sets an ack timer to a random value between zero and *MaxAckTime*. If the timer expires and the member has received data during this interval then it sends an EXPLICIT ACK. However, if the group member hears an EXPLICIT ACK during this interval, it cancels its timer and then waits the remainder of *MaxAckTime* before it sets a new ack timer. The source (or forwarder) sets its forwarding state timer to *AckWaitTime*, which is equal to $m * MaxAckTime$. This ensures that the source does not time out its forwarding state unless it misses m acks in a row.

The top section of the packet trace in Figure 9 shows network activity for ADMR with both explicit acks and the Join and Repair timers fixed. We use a value of 66 ms for *MaxAckTime*, 2 seconds for *AckWaitTime*, and $500ms$ for the *RepairWaitTime*. As can be seen from this trace, explicit

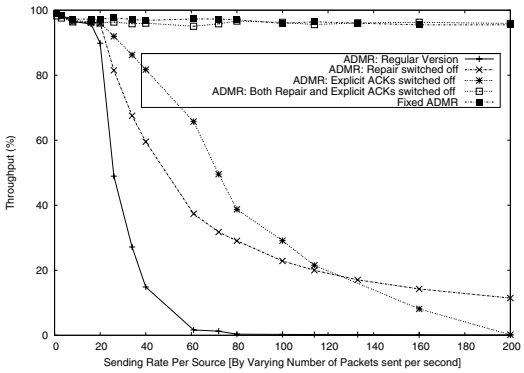

Figure 10. ADMR fixed for high density

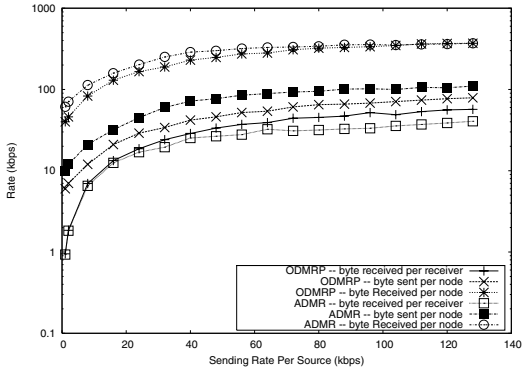

Figure 11. High traffic: packet size

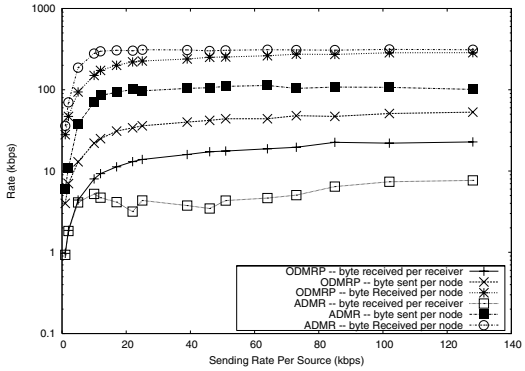

Figure 12. High traffic: inter-packet gap

acks are sent regularly, but at a much reduced rate. This relieves the congestion in the medium, allowing a throughput of 95.86%!

To verify that our two solutions work, we repeated the simulation of Figure 8 using five versions of ADMR: the regular version, ADMR with the Join and Repair timers disabled, ADMR with explicit acks disabled, ADMR with timers and explicit acks disabled, and finally ADMR with our two solutions. As shown in Figure 10, our solutions enable ADMR to scale with offered load during the high density scenario. In fact, they work just as well as completely disabling the problematic mechanisms, though this is not a viable alternative.

We do not claim that our solutions enable ADMR to obtain high throughput for all scenarios. We did, however, test our solutions with the mobility scenario in the previous section and found that we were able to obtain nearly identical results in these cases. We do note that ADMR's performance is sensitive to the timer settings we have adjusted. Using larger values for the *RepairWaitTime* cause ADMR to react too slowly to high mobility conditions, and using larger values for *AckWaitTime* cause excessive pruning and low throughput for high mobility conditions.

Our primary purpose in this exercise is to demonstrate the pitfall of testing multicast routing performance in only low density situations. Our experiments with high density have identified several flaws in ADMR that can be generalized to any multicast routing protocol. Explicit acknowledgments should be avoided if possible, and control messages that are flooded should be rate limited to avoid broadcast storms. While we have implemented changes to ADMR that solve these problems for a single group, dampening these messages across multiple groups is a more difficult problem. Our results also question the practice of building a routing protocol that reacts to packet loss. If the protocol interprets all packet loss as a sign of mobility, then it will misinterpret congestion as a sign of mobility and potentially cause a congestion collapse.

6 High Traffic Scenario

Our third scenario explores the effects of high traffic rates on multicast routing performance. To isolate the effects of load from other factors we keep density and mobility low. In these simulations we use the same parameters as the mobility scenario, except we keep the speed constant at 1 m/s. We then vary the traffic rate in two ways: using a fixed packet size of 64 bytes and varying the inter-packet gap, and using a fixed inter-packet gap of 250 ms and varying the packet size. For this scenario use the Exhibition model with 10 centers; results for other mobility models are similar.

Our results indicate that ODMRP is able to achieve a maximum throughput of about 55 kbps, while ADMR can receive at most 40 kbps when varying the packet size (Figure 11). Both protocols obtain significantly lower throughput when varying the inter-packet gap (Figure 12). In both figures we show the average rate at which data is received by the group members, as well as the average transmission rate (at the MAC layer) for each node in the network and the average rate at which packets are received by each node (also at the MAC layer). The data reception rate indicates how successful the routing protocol is; for example, when the packet size is varied, ODMRP receives about 30 kbps when the sources send at 50 kbps, or about 60% of the throughput. The rate transmitted per node and the rate received per

node indicate the load imposed on the network by the routing protocol. The rate received is always higher than the rate transmitted because each node has more than one neighbor.

Our second purpose in exploring this scenario is to examine a fundamental question: how effectively can a multicast routing protocol use the available capacity of an ad hoc network? It clear from our results that multicast can use more of the available capacity by transmitting large packets rather than small packets. For example, when varying the packet size ODMRP can receive about 55 kbps, but can only receive about 20 kbps when varying the inter-packet gap. But how much of the available capacity can a multicast routing protocol actually use?

From work on the capacity of ad hoc networks [8, 17], we know that the unicast sending rate, λ_u available to a node is bounded by:

$$\lambda_u < \frac{C/n}{\overline{L_u}/r}, \quad (1)$$

where C is the capacity of the network, n is the number of nodes that are transmitting, $\overline{L_u}$ is the expected path length in meters, and r is the radio range in meters. Because $\overline{L_u}/r$ is the expected unicast path length in hops, we can translate this equation into an available sending multicast rate by substituting the expected number of hops in a multicast tree. Hence, for multicast:

$$\lambda_m < \frac{C/n}{\overline{L_m}}, \quad (2)$$

where $\overline{L_m}$ is the expected number of hops in the multicast tree. This bound indicates that one of the critical factors for a multicast tree is its efficiency, with a lower L_m enabling a higher transmission rate. This is exactly what the bandwidth efficient multicast protocol [19] tries to accomplish, by having receivers join along the shortest path to the existing tree, rather than using the shortest path to the source. This indicates that in a high traffic scenario it is better for a multicast protocol to use a tree instead of a mesh.

To determine L_m for our scenarios, we ran an experiment that measured this value for both ODMRP and ADMR as we vary the number of group members for a single group with one source (Figure 13). In both cases, this measurement excludes any forwarding done by flooding of control or data packets. The average L_m increases slowly for both protocols when the group includes only a single source. This measurement appears to follow the Chuang-Sirbu law, in which $L_m/\overline{L_u} = m^k$, where m is the number of receivers [4, 21]. This law fits our data when $\overline{L_u}$ is 3 (the measured value for our simulations) and k is 0.58. When we allow all members to be sources for the multicast group, ADMR is unaffected (since it builds a tree per source), but L_m grows significantly for ODMRP since it uses a mesh. The number of links in the mesh falls at very large group sizes for ODMRP, but this is

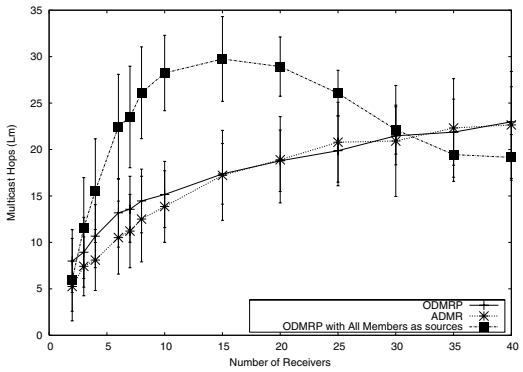

Figure 13. L_m as a function of group size

only because the number of nodes and the area of the network is limited. Hence, we would expect ODMRP to not be bandwidth efficient as the number of sources grows.

Returning to Equation 2, we can now calculate the sending rate available to a node. We use a measured L_m of 12 for a group size of 7 (Figure 13), 2000 kbps for C [17], and we have 3 active sources. This yields $\lambda_m = 55kbps$ for ODMRP, which is what it does obtain when using large packets. Since ADMR has a lower L_m, it should be able to achieve a higher sending rate. However, ADMR switches to flooding once packet loss begins to occur, which is exactly the wrong thing to do during a period of congestion. ADMR misinterprets packet loss as a sign of mobility and thus makes the situation even worse. This problem is particularly acute when varying the inter-packet gap, because increasing the sending rate in this case increases the number of packets that are flooded.

It is an open question as to whether other multicast routing protocols better utilize network capacity. We believe a tree-based protocol (with lower overhead than ADMR) could do better than ODMRP, particularly for multiple sources, because its L_m will be lower. On the other hand, it is possible that the theoretical limits are too high, and that ad hoc networks using IEEE 802.11a are simply unable to support high levels of multicast traffic. We are currently designing a new multicast routing protocol that uses a tree and reliable broadcast at the MAC layer in an attempt to achieve high utilization.

7 Conclusions

Our goal in this work has been to identify cases where ad hoc multicast routing protocols can improve their performance, resulting in a set of recommendations to multicast protocol designers.

Our first recommendation is that designers focus on optimizing for both mobility and density. Protocols should react to link breaks, but with lower overhead than ADMR does. Designers should also be aware of the pitfalls associated with high density situations, avoiding problems such as ack im-

plosion and broadcast storms. While these problems have been seen in wired networks, it is apparent that these lessons have not always carried over into wireless networking. Protocols should instead take advantage of the density created by group-based mobility patterns. Recent work by Yi et al. does this by treating groups of users as a team and then multicast a single packet to each team [25].

We strongly recommend that routing protocols should not attempt to monitor packet loss and repair routes when loss is high. As we have seen with ADMR, such loss may in fact be due to congestion and the increased repair traffic can lead to congestion collapse. Rather, loss monitoring should be done by the transport layer, which can then use input from the MAC layer to determine if the loss is due to congestion or mobility. The transport layer can then request that the routing protocol take appropriate action when the loss is due to mobility (and suspend sending any more data until a new route is found).

To achieve high capacity, protocols should use a bandwidth-efficient tree rather than a mesh and should have very low control overhead. Mesh-based multicast protocols increase the number of hops in the multicast tree and hence cannot support high traffic rates. This argues for a simple, end-to-end protocol design, with receivers in charge of joining groups and reacting to route changes. Asking multicast forwarders to maintain the tree results in too much control traffic, particularly when broadcast is used to repair the tree.

Finally, we suggest that future multicast protocol evaluations – both in simulations and in testbeds – need to be more comprehensive. Evaluations should consider a range of realistic mobility models and should include special cases, such as high density and high traffic rates.

References

[1] F. Bai, N. Sadagopan, and A. Helmy. IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc Networks. In *IEEE INFOCOM*, 2003.

[2] C. Bettstetter. Smooth is Better than Sharp: A Random Mobility Model for Simulation of Wireless Networks. In *ACM MSWiM*, July 2001.

[3] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications & Mobile Computing (WCMC)*, 2002.

[4] J. Chuang and M. Sirbu. Pricing Multicast Communications: A Cost-Based Approach. *Telecommunication Systems*, 17(3):281–297, July 2001.

[5] S. K. Das, B. S. Manoj, and C. S. R. Murthy. A Dynamic Core Based Multicast Routing Protocol for Ad Hoc Wireless Networks. In *ACM MobiHoc*, June 2002.

[6] V. Davies. Evaluating Mobility Models Within an Ad Hoc Network. Master's thesis, Colorado School of Mines, Mathematical and Computer Sciences, 2000.

[7] A. Erramilli and R. Singh. A Reliable and Efficient Multicast Protocol for Broadband Broadcast Networks. In *ACM SIGCOMM*, August 1987.

[8] P. Gupta and P. Kumar. The Capacity of Wireless Networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, March 2000.

[9] X. Hong, M. Gerla, G. Pei, and C. Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. In *ACM MSWiM*, August 1999.

[10] J. Jetcheva and D. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks. In *ACM MobiHoc*, October 2001.

[11] J. Jetcheva and D. B. Johnson. Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks, July 2001.

[12] L. Ji and M. S. Corson. Differential Destination Multicast - A MANET Multicast Routing Protocol for Small Groups. In *IEEE INFOCOM*, 2001.

[13] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario Based Peformance Analysis of Routing Protcols for Mobile Ad Hoc Networks. In *IEEE MobiCom*, August 1999.

[14] S. Lee and C. Kim. Neighbor Supporting Ad Hoc Multicast Routing Protocol. In *ACM MobiHoc*, Aug 2000.

[15] S. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks. In *ACM/Baltzer Mobile Networks and Applications*, 2000.

[16] S. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *IEEE INFOCOM*, 2000.

[17] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Mobile Computing and Networking*, pages 61–69, 2001.

[18] J. Luo, P. Eugster, and J.-P. Hubaux. Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks. In *INFOCOM*, 2003.

[19] T. Ozaki, J. B. Kim, and T. Suda. Bandwidth-Efficient Multicast Routing for Multihop, Ad-Hoc Wireless Networks. In *IEEE INFOCOM*, 2001.

[20] M. Pandey and D. Zappala. The Effects of Mobility on Multicast Routing in Ad Hoc Networks. In *Technical Report, UO-CIS-TR-2004-2*, March 2004.

[21] G. Phillips, H. Tangmunarunkit, and S. Shenker. Scaling of Multicast Trees: Comments on the Chuang-Sirbu Scaling Law. In *ACM SIGCOMM*, 1999.

[22] E. Royer and C. Perkins. Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol. In *Mobile Computing and Networking*, 1999.

[23] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. RFC 1075, November 1988.

[24] C. Wu, Y. Tay, and C. Toh. Ad hoc Multicast Routing Protocol Utilizing Increasing id-numberS (AMRIS) Functional Specification. In *Internet-Draft (work in progress)*, Nov 1998.

[25] Y. Yi, M. Gerla, and K. Obraczka. Scalable Team Multicast in Wireless Networks Exploiting Coordinated Motion. *Ad Hoc Networks Journal*, August 2003.

[26] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *Workshop on Parallel and Distributed Simulation*, May 1998.