Multicasting in Mobile Ad-Hoc Networks: Achieving High Packet Delivery Ratios

Thomas Kunz
Systems and Computer Engineering
Carleton University

Abstract

Multicasting is intended for group-oriented computing. There are more and more applications where one-to-many or many-to-many dissemination is an essential task. The multicast service is critical in applications characterized by the close collaboration of teams. Many applications, such as audio/video distribution, can tolerate loss of data content, but many other applications cannot. In addition, even loss-tolerant applications will suffer a performance penalty: an audio stream may experience a short gap or lower fidelity in the presence of loss. This paper describes our experience with implementing a multicast routing protocol that delivers packets to all intended recipients with high probability. Due to a number of reasons, 100% packet delivery cannot be achieved, but packet delivery ratios in excess of 99% are possible in most cases.

1 Introduction

Multicasting is the transmission of datagrams (packets) to a group of zero or more hosts identified by a single destination address and is intended for group-oriented applications. There are more and more applications where one-to-many or many-to-many dissemination is an essential task. The multicast service is critical in applications characterized by the close collaboration of teams (e.g. rescue patrol, battalion, scientists, etc) with requirements for audio and video conferencing and sharing of text and images. In the Internet (IPv4), multicasting facilities were introduced via the Multicast Backbone (MBone), a virtual overlay network on top of the Internet. Support for multicasting is an integral component of IPv6, so we expect that multicasting applications will become even more popular with the increased spread of IPv6. However, the acceptance and use of group-related applications is not only based on technological criteria. [8] for example discusses some sociological issues relevant to the design and use of group applications.

Typically, the membership of a host group is dynamic; that is, hosts may join and leave groups any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. In open multicast protocols, a host does not have to be a member of a group to send packets to it. The use of multicasting within a network has many benefits. Multicasting reduces the communication costs for applications that send the same data to multiple recipients [31]. Instead of sending via multiple unicasts, multicasting minimizes the link bandwidth consumption, sender and router processing, and delivery delay. In addition, multicasting provides a simple yet robust communication mechanism whereby a receiver's individual address is unknown or changeable transparently to the source.

Maintaining group membership information and building an efficient multicast distribution structure (typically in the form of a routing tree) is challenging even in wired networks. A detailed survey of the work done in that area and a discussion of various design trade-offs can be found in [17]. However, nodes are increasingly mobile. One particularly challenging environment is a mobile ad-hoc network (MANET). A MANET consists of a dynamic collection of nodes with sometimes rapidly changing multi-hop topologies that are composed of relatively low-bandwidth wireless links. There is no assumption of an underlying fixed infrastructure. Nodes are free to move arbitrarily. Since each node has a limited transmission range, not all messages may reach all the intended hosts. To provide communication through the whole

network, a source-to-destination path could pass through several intermediate neighbour nodes. MANET routing protocols therefore must address a diverse range of issues. The network topology can change randomly and rapidly, at unpredictable times. Since wireless links generally have lower capacity, congestion is typically the norm rather than the exception. The majority of nodes will rely on batteries, thus routing protocols must limit the amount of control information that is passed between nodes. Also, multicast group members and other nodes move, thus precluding the use of a fixed multicast topology. [12] provides an overview of some best-effort IP multicast routing protocols for fixed networks and the evolution of these protocols as hosts and finally all nodes (including intermediate routers) become mobile.

The goal of MANETs is to extend mobility into the realm of autonomous, mobile, wireless domains, where a set of nodes form the network routing infrastructure in an ad-hoc fashion. The majority of MANET applications are in areas where rapid deployment and dynamic reconfiguration are necessary and the wireline network is not available. These include military battlefields, emergency search and rescue sites, classrooms, and conventions where participants share information dynamically using their mobile devices. These applications can therefore benefit from a multicast service. In addition, within a MANET, it is even more crucial to reduce the transmission overhead and power consumption. Multicasting can improve the efficiency of the wireless link when sending multiple copies of messages by exploiting the inherent broadcast property of wireless transmissions.

RFC 3170 [26] describes the challenges involved with designing and implementing multicast applications. The document lists a number of multicast applications and derives unique multicast service requirements for various groups of applications. While many applications, such as audio/video distribution, can tolerate loss of data, many other applications cannot. In addition, even loss-tolerant applications will suffer a performance penalty: an audio stream may experience a short gap or lower fidelity in the presence of loss. Among the loss-intolerant application categories are file distribution and caching, monitoring applications (stock prices, sensor readings, etc.), synchronized resources (directories, distributed databases, etc.), concurrent processing, collaboration/shared document editing, and online auctions. A similar discussion of multicast applications and their requirements can be found in [31]. Some of the loss-intolerant applications discussed in these documents are relevant in a MANET environment as well (such as the collaboration, caching/file distribution, or monitoring applications). In addition, MANET-specific applications such as military command-and-control applications also require a high degree of reliability.

This paper describes our experience and insights in developing a multicast routing protocol that delivers packets to all multicast receivers with a high probability. It is organized as follows. Section 2 reviews the definition of "reliable multicasting". Section 3 outlines why reliable multicasting is difficult to achieve by reviewing results published in the literature and reporting on some simulation results we collected. Section 4 discusses design alternatives to achieve a reliable multicast protocol; Section 5 describes our reliable multicast routing protocol and its performance. Section 6 discusses related work and section 7 concludes the paper with a summary and discussion of future work.

2 Defining Reliable Multicasting

As expressed by RFC 2357, "the meaning of reliability varies in the context of different applications" [25]. Consequently, we need to define the meaning of "reliable multicast". As discussed for example in [17], at least three different levels of reliable data packet delivery can be distinguished (listed in increasing order of difficulty):

- all data packets are delivered,
- the causal order between data packets is maintained, or
- a total order of data packet delivery is achieved.

All three possible definitions require that all multicast receivers will (eventually) receive all data packets. However, there is no guarantee that for multiple data packets sent by the same or different senders, a specific order of reception is maintained. For example, if sender S sends two packets M1 and M2, it is okay for receiver R1 to receive M1 before M2, while receiver R2 may receive those same two packets in the reverse order (M2 before M1).

While delivery of all packets is a minimum requirement for any reliable multicast

mechanism, some applications have more stringent requirements. In particular, the order in which multicast packets are received may matter. An example could be a replicated database, where packets trigger update operations on the data. If two receivers receive and process these update messages in different orders, the replicated data easily becomes inconsistent. The difference between the second and third definition of reliable multicast is in the constraints imposed by the reception of different packets. The most rigid definition is the third definition, where the multicast protocol determines an ordering among ALL multicast packets (originating from the same or a different sender), and enforces that every receiver receives all packets in exactly that order. The order in which multicast packets are delivered to all receivers is called their "total order".

A slightly more relaxed constraint is the second definition, which enforces only what is called a "partial order" on the delivery of multicast packets. In a nutshell, this definition says that if M1 could have influenced the content of M2, all receivers will have to receive M1 before M2. If, however, M1 could not have had any impact on M2, than the order in which these two packets can be received is left open. A packet M1 in this scheme can impact a packet M2 (or, as it is also called, "causally precede it"), if either:

- M2 is sent after M1 by the same sender
- M2 is sent by a sender after it received M1

and the transitive closure of these two cases. In essence, M2 "causally follows" M1 if there is a chain of packets from M1 to M2 that are sent in a sequence. Since this relationship does not enforce an ordering on all messages, it is also called a "partial order" relationship.

In this work, we will focus on ensuring that all packets are delivered, anticipating that this will be a difficult enough problem in MANETs. If additional constraints on packet delivery are necessary, they can be implemented based on reliable packet delivery by adding logical timestamps to messages, as defined in [6, 19].

However, even guaranteeing 100% packet delivery is not a realistic goal. In mobile networks, receivers can be disconnected from the network for unpredictable amounts of time. As will be discussed later, implementing reliable protocols involves buffering packets to service retransmission requests. Since buffer space is finite on any concrete computer, we cannot expect to be able to support arbitrarily long disconnections, with the ensuing requirement to buffer copies of all packets until the disconnected receiver(s) reconnect. The only realistic goal therefore is to provide as high a packet delivery ratio as possible with finite resources.

3 Reliable Multicasting is Hard

3.1 Literature Results

Achieving high packet delivery ratios in a MANET is not trivial. A few simulation studies have explored the performance of MANET multicast routing protocols such as the multicast extensions for AODV and ODMRP [3, 5, 34]. The results show that the packet delivery ratio does drop below 20% (i.e., only 1 out of every 5 packets is, on average, received by a multicast receiver) for a large number of senders [5]. In environments with one or a few senders, packet delivery ratios as low as 25% were observed when the mobility rate increased (nodes moved constantly, with speeds up to 20 m/s) [3]. Possible improvements, such as pro-actively predicting link breakage and maintaining the multicast distribution data structure before links break (and therefore packets get lost) can increase the packet delivery ratio, but under high mobility scenario, it is still below 90% [34]. In a nutshell, all these protocols exhibit intolerably high packet loss rates under moderate to high mobility rates. Similarly poor results are shown in [14] for a number of other multicast routing protocols.

Building and maintaining a multicast distribution structure (typically a tree or mesh) in a MANET with its highly dynamic topology introduces its own complexities and overheads (control messages, data structures at intermediate nodes, etc.). Based on the above studies, this effort does not necessarily result in good performance; it therefore becomes questionable whether it is indeed worth the effort. Following this line of thought, some researchers have explored whether broadcasting/flooding a MANET with packets could be a viable alternative to ensure high packet delivery ratios. The results in [22, 23] indeed show that flooding results in higher packet delivery ratios then multicast routing protocols. But in the scenarios studied in these papers, flooding could result in packet delivery ratios as low as 70%, leading the authors to conclude "*even flooding is insufficient for reliable multicast in ad hoc networks when mobility is very high*" [23]. In addition, the

simulation scenarios were all based on the assumption that every node in the MANET was interested in the data packets (i.e., a broadcast scenario). More generally, only a subset of nodes will be interested in any specific multicast group, flooding the data to all nodes may induce a high overhead, negating one of the stated advantages of multicasting. As discussed in [32] and [18], more efficient broadcast algorithms can be implemented. However, even those algorithms will suffer from low packet delivery ratios (60%-80%) as the severity of the network environment (mobility rate, traffic load, etc.) increases.

In conclusion, flooding/broadcasting data in a MANET is not sufficient to ensure high packet delivery ratios. While flooding is attractive due to its absence of a multicast distribution structure (and its ensuing maintenance), it may lead to high network traffic when propagating data packets to nodes that are not interested in it. However, to explore this issue further, we compared some best-effort multicast approaches by running extensive simulations, as discussed next.

3.2 Simulation Results

We studied the performance of a number of best-effort multicast routing protocols in NS2, the most widely used simulator for research on ad-hoc networking. This subsection summarizes the protocols, the simulation environment, and the results. The goal of these experiments is two-fold. First, we wanted to confirm that achieving high packet delivery ratios is indeed non-trivial. Second, we wanted to identify a good starting point for a reliable multicast protocol by identifying a protocol that already achieves relatively high packet delivery ratios.

Three best-effort multicast protocols were used in this study. Two protocols do *not* build and maintain any distribution structure. Rather, they broadcast the data packets from the sender to each node in the network; multicast receivers pick up the packets as they arrive at the node. The third protocol is a true multicast routing protocol, building and maintaining a multicast distribution structure.

The first, and simplest protocol is FLOOD. It implements standard flooding: each node, upon receiving a packet for the first time, will re-broadcast it over its wireless interface (i.e., using MAC-layer broadcasting). To reduce the chance of packet collisions, re-broadcasts are randomly jittered by 10 ms.

The second protocol, BCAST, implements a scalable broadcast algorithm similar to the algorithm described in [18, 32]. BCAST uses 2-hop neighbor knowledge that is exchanged by periodic "Hello" messages. When node B receives a broadcast from node A, B knows all neighbors of A. If B has neighbors not covered by A, it schedules the broadcast packet with a random delay. If, during the delay, B receives another copy of this broadcast from C, it can check whether its own broadcast will still reach uncovered neighbors. If this is no longer the case, it will drop the packet. Otherwise, the process continues until B's timer goes off and B itself rebroadcasts the packet.

To determine an appropriate random delay, we utilize the greedy strategy proposed by the original authors. Each node searches its neighbor table for the maximum neighbor degree of any neighbor node, MAX. If its own node degree is N, it calculates the random delay as MAX/N. This is a greedy strategy: nodes with the most neighbors usually broadcast before others. In our implementation we delay a re-broadcast by randomly choosing a delay of up to MAX/N * 100 ms. This may result in reordering successive broadcasts, which may cause problems when using this as a basis for a reliable multicast protocol: packets will arrive out of order with high probability. Before a packet is rescheduled, we therefore check whether earlier packets from the same sender are already pending. In this case, we increase the delay for the new packet to maintain the packet order on a per-sender basis.

The multicast protocol that builds and maintains a multicast distribution structure is ODMRP: On-Demand Multicast Routing Protocol [15]. ODMRP is mesh based, and uses a forwarding group concept (only a subset of nodes forwards the multicast packets). In ODMRP, group membership and multicast routes are established and updated by the source on demand. When a multicast source has packets to send, but no route to the multicast group, it broadcasts a Join-Query control packet to the entire network. This Join-Query packet is periodically broadcast to refresh the membership information and update routes. When the Join-Query packet reaches a multicast receiver, it creates and broadcasts a "Join Reply" to its neighbors. During this phase, nodes construct (or update) the routes from sources to receivers and build a mesh of nodes,

the forwarding group. These meshes are source-based, so in an environment with many senders a number of these meshes will have to be built and maintained. On the other hand, no work is required to update the mesh as the topology changes or nodes join/leave the multicast group: such changes get reflected the next time the mesh is rebuilt. In the results reported below, we used the implementation of ODMRP provided by the Monarch group [20], including the patches until March 2003.

The overall goal of the simulations reported here is to show that achieving high packet delivery ratios is difficult. To this end, we set up a rather challenging simulation environment, using the following parameters (similar to other setups reported in the literature): Area of 1500 x 300 meters, 50 nodes, simulation length 910 seconds, 10 repetitions for each scenario. As Physical/Mac layer we choose IEEE 802.11 radios with 2 Mbps data rate and a 250-meter transmission range. The nodes move around based on the random waypoint model with no pause time, maximum speed either 20 m/s (high mobility scenario) or 1 m/s (low mobility scenario). All nodes are in constant movement in our experiments. The only traffic is the multicast traffic. We study a range of multicast senders (1, 2, 5, or 10), sending to a number of multicast receivers (10, 20, 30, 40, or 50). We keep the sender and receiver sets disjoint. For example, in a scenario with 10 senders and 30 receivers, nodes 0 through 9 are the senders and nodes 20 through 49 are the receivers. Only in scenarios with 50 multicast receivers (a broadcast) will some nodes act as both sender and receiver. In these scenarios, we expect the packet delivery ratio to be slightly better, since packet delivery within a single node is not subject to network problems.

All receivers join the single multicast group at the beginning of the simulation, the sender(s) start sending data 30 seconds into the simulation (so where appropriate, the routing protocol can start building its multicast distribution structure or gain initial neighborhood knowledge). After 900 seconds, all senders stop transmitting data, during the remaining 10 seconds, data packets still in flight have a chance to be delivered. Each sender sends 2 packets per second; each packet is 256 bytes long. In total, 400 NS2 simulations are run per protocol, exploring each combination of number of senders and number of receivers (20 combinations) at high and low mobility. Each table entry represents the average over 10 runs, where each run differs in the scenario input file (different initial location and movement of the 50 nodes).

The basic performance metrics are *Packet Delivery Ratio* (PDR) and *Packet Latency*. Packet delivery ratio is defined as the percentage of received packets, relative to the total number of packets ideally received. Packet latency is the elapsed time between a packet being transmitted and ultimately received.

An ideal protocol will achieve high packet delivery ratio and low packet latency with little overhead. Traditionally, counting the number of control messages and relating them to the number of received packets measures protocol overhead. However, FLOOD does not generate any dedicated control messages. And the overheads in any broadcast protocol are not only related to any control messages, but also to the waste of delivering packets to nodes not interested in this data. So we define a metric that captures the "network efficiency of the protocol", the *Packet Send Ratio* (PSR). The PSR is the number of packet transmissions (at the MAC layer) per data packet received by a multicast receiver and captures the normalized total traffic in the network.

	1 Sender		**2 Sender**		**5 Sender**		**10 Sender**	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.998	0.023	0.991	0.027	0.956	0.051	0.828	2.084
20 Receiver	0.998	0.025	0.991	0.029	0.956	0.052	0.827	2.099
30 Receiver	0.996	0.025	0.989	0.029	0.954	0.052	0.826	2.091
40 Receiver	0.996	0.026	0.989	0.029	0.954	0.052	0.826	2.078
50 Receiver	0.996	0.025	0.990	0.028	0.955	0.051	0.831	2.041

Table 1: PDR and Latency for FLOOD, 1 m/s maximum speed

	1 Sender		2 Sender		5 Sender		10 Sender	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.999	0.023	0.993	0.029	0.965	0.053	0.815	1.684
20 Receiver	0.999	0.023	0.993	0.028	0.965	0.052	0.815	1.683
30 Receiver	0.999	0.023	0.993	0.029	0.965	0.052	0.815	1.687
40 Receiver	0.999	0.023	0.993	0.028	0.965	0.052	0.815	1.686
50 Receiver	0.999	0.022	0.993	0.028	0.965	0.051	0.818	1.652

Table 2: PDR and Latency for FLOOD, 20 m/s maximum speed

Tables 1 and 2 show the performance of FLOOD. Under both low and high mobility, the packet delivery ratio stays high (at or above 99%) for one or two multicast senders, dropping to 95.5% to 96.5% for 5 multicast senders and to 81.5% to 82.7% for 10 multicast senders. The packet latencies are small (a few tens of milliseconds) for most scenarios, but increase drastically for the 10 multicast sender scenarios. The number of multicast receivers, as expected, does not impact the protocol performance.

The network efficiency is largely independent of the number of multicast senders and mobility rate. In FLOOD, every packet is transmitted up to 50 times. Therefore, the PSR is 5 (50 packet transmissions for 10 packet receptions) for 10 multicast receivers and 1 (50 packet transmissions for 50 packet receptions) for 50 multicast receivers.

In summary, this simple protocol only delivers packets with high probability in limited scenarios: few (one or two) multicast senders (each sending a relatively low bandwidth data stream). In these cases, the protocol achieves packet delivery rates above 99%. As the number of senders and the amount of data per sender increases, the delivery ratios deteriorate rapidly. The main reason is network congestion: the MAC queues will slowly build up to their maximum capacity; new incoming packets (i.e., packets passed down from the network layer) will be dropped. As expected, neither the speed of node movement nor the number of multicast receivers impacts the protocol performance.

To improve the packet delivery ratio, two distinct approaches are possible:

1. Reduce the number of packet re-broadcasts: this is the BCAST approach. Fewer re-broadcasts will result in the wireless media being less busy, increasing the chance that the MAC layer will successfully transmit a packet, rather then dropping it. On the downside, this may decrease the chances of successful packet delivery, since we reduce the number of ways a packet can reach a destination. In addition, BCAST exchanges control information (the periodic "Hello" messages) that increases the network traffic.
2. Reduce the number of target nodes: rather than broadcasting the data to every node, send the data to all intended recipients, maybe via intermediate nodes where necessary, as in ODMRP. However, such protocols introduce their own overheads, so it is not obvious whether the overall protocol performance will improve.

Tables 3 and 4 summarize the performance for BCAST under the best parameter settings we found in a limited number of experiments (HELLO interval 2 seconds, missing a single HELLO message results in the neighbor entry being dropped):

	1 Sender		2 Sender		5 Sender		10 Sender	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.997	0.116	0.996	0.108	0.987	0.118	0.970	0.126
20 Receiver	0.997	0.119	0.996	0.112	0.987	0.121	0.970	0.129
30 Receiver	0.995	0.121	0.994	0.113	0.985	0.121	0.968	0.129
40 Receiver	0.995	0.121	0.994	0.113	0.984	0.121	0.967	0.129
50 Receiver	0.995	0.118	0.994	0.110	0.985	0.118	0.968	0.126

Table 3: PDR and Latency for BCAST, 1 m/s maximum speed

	1 Sender		2 Sender		5 Sender		10 Sender	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.993	0.104	0.992	0.108	0.986	0.114	0.968	0.126
20 Receiver	0.992	0.103	0.991	0.107	0.986	0.114	0.968	0.127
30 Receiver	0.993	0.103	0.992	0.107	0.986	0.113	0.968	0.127
40 Receiver	0.993	0.103	0.992	0.107	0.986	0.113	0.968	0.126
50 Receiver	0.993	0.101	0.992	0.105	0.986	0.110	0.969	0.124

Table 4: PDR and Latency for BCAST, 20 m/s maximum speed

BCAST achieves similar performance to FLOOD for one or two multicast sender, but with significantly better PDR for 5 and 10 multicast senders. For 1 multicast sender, the PDR is slightly below the PDR in FLOOD, in particular for high mobility scenarios. This is due to the fact that BCAST has less redundancy, dynamically selecting only a subset of nodes to re-broadcast a packet. This shows in the packet send ratio as well: PSR ranges from 2.5 to 0.5 for 10 and 50 multicast receivers, respectively. The resulting lower network traffic is beneficial in the 2 and 5 sender cases, where fewer collisions occur at the MAC layer, resulting in improved PDRs, compared to FLOOD. And scenarios with 10 multicast senders benefit the most, with PDRs of about 97%. Overall, the protocol performance is very consistent across all scenarios, with PDR ranging from 96.7% to 99.7% and packet latency ranging from 104 ms to 129 ms.

Packet latency is rather high, consistently above 100 ms, since packet re-transmission is delayed by an amount of time randomly chosen from [0, MAX/N * 100 ms], as explained above. Running experiments with a smaller scaling factor, such as 10 ms, shows that a large scaling factor is beneficial, achieving high packet delivery ratio at the expense of some latency. With a 10 ms scaling factor, PDR for the 1-sender cases is around 99.6% to 99.7%, dropping to 98.3% to 98.6% for the 2 sender cases. It drops even more drastically to 91% to 92% for the 5 sender cases, and to 69% for the 10 sender cases. For BCAST to achieve a high packet delivery ratio, a relatively large delay factor is advantageous.

In summary, BCAST can achieve the same or better performance as FLOOD for all scenarios, with significantly lower overheads. Overall, the protocol performance follows a pattern similar to FLOOD: the performance is independent of the number of multicast receivers and node mobility, and deteriorates as the number of multicast senders increases.

	1 Sender		2 Sender		5 Sender		10 Sender	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.995	0.009	0.991	0.010	0.973	0.015	0.937	0.031
20 Receiver	0.995	0.010	0.991	0.011	0.974	0.017	0.936	0.039
30 Receiver	0.993	0.011	0.990	0.012	0.972	0.017	0.934	0.043
40 Receiver	0.992	0.011	0.990	0.012	0.972	0.018	0.932	0.049
50 Receiver	0.973	0.011	0.971	0.012	0.953	0.018	0.915	0.056

Table 5: PDR and Latency for ODMRP, 1 m/s maximum speed

	1 Sender		2 Sender		5 Sender		10 Sender	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.987	0.010	0.992	0.011	0.977	0.016	0.940	0.033
20 Receiver	0.991	0.010	0.992	0.011	0.979	0.017	0.937	0.039
30 Receiver	0.991	0.010	0.994	0.012	0.980	0.017	0.935	0.046
40 Receiver	0.993	0.010	0.994	0.012	0.979	0.018	0.934	0.052
50 Receiver	0.973	0.010	0.974	0.012	0.961	0.018	0.916	0.060

Table 6: PDR and Latency for ODMRP, 20 m/s maximum speed

Tables 5 and 6 summarize the ODMRP results. ODMRP performs worse for few multicast senders than FLOOD or BCAST, and slightly better than FLOOD for the 5 and 10 multicast sender scenarios, in terms of packet delivery ratio. Mobility seems to have little impact on overall performance, with PDR and latency roughly similar across comparable scenarios. One advantage of ODMRP is the lower packet latency, ranging from 10 ms to about 60 ms only.

In terms of network efficiency, ODMRP is not more efficient than the two flooding protocols. In general, the higher the number of multicast receivers, the lower the PSR. The highest PSR value is observed for 10 senders and 10 receivers, at about 5.4 (i.e., for every successful packet delivery, 5.4 packet transmissions occurred at the MAC layer). This can be explained by the need to build and maintain 10 multicast meshes. As the number of receivers increases, these costs are amortized over an increased number of successfully delivered packets, reducing the PSR to 1.4 for 50 receivers. For a single multicast sender, the PSR ranges from 2.7 (10 multicast receivers) to 0.88 (50 multicast receivers).

3.3 Summary

Both the results published in the literature and our own experiments show that achieving a high packet delivery ratio is hard. Even in the absence of network congestion, the major cause of packet loss in most of our experiments, multicast receivers will not receive all packets. We therefore need to explore ways to

1. Increase the packet delivery ratio for low traffic levels.
2. Increase the network capacity to support more high-traffic scenarios.

The second objective will require work along two lines: reducing the routing protocol overhead and reducing the MAC protocol overhead/increasing the link capacity, which is one of our items of future work. However, the remainder of this paper focuses on the first approach, increasing the packet delivery ratio. Since BCAST gave us, overall, high packet delivery ratio with relatively low overhead, we will use it as a starting point.

4 Design Alternatives

A reliable multicast protocol has to address a number of key issues. The two overriding and orthogonal issues are what mechanism should be used to ensure reliability and whether the protocol is implemented as a transport layer protocol or as a network layer protocol. This section discusses the alternatives and justifies our high-level design decisions.

4.1. Reliability Mechanism

The previous section showed that best-effort multicast approaches do not always deliver a sufficiently high number of data packets to be considered suitable. So we have to assume that data packets will be lost in transmission. To recover from this loss, two (complementary) approaches are possible: the use of forward error correction codes and packet retransmissions.

Protocols based on forward error correction (FEC) codes take a data packet, split it up, and add additional, redundant information. These sub-packets are then transmitted to all multicast receivers. Assuming that these smaller packets are lost independently, a receiver can re-assemble the original data packet as long as a sufficient subset of smaller packets is received. In general, the more redundant information the sender adds, the more packet losses can be tolerated. Example protocols based on this idea are described in [4, 29].

Protocols based on packet retransmission involve some form of automatic repeat request (ARQ) scheme. A receiver has to be able to identify that it missed a packet transmission, which is typically achieved by assigning packets sent by the same sender a unique and monotonically increasing sequence number. When receivers detect a gap in the packet sequence, they will ask for retransmission of this packet, with retransmission requests either directed to the sender or some intermediate node that is known or suspected of having stored a copy of previously transmitted packets. Protocols based on this idea are described in [7, 10 , 13 24]

FEC-based approaches are attractive when the communication links are unidirectional, since they do not require any feedback to be sent back to the sender. They also, in general, have lower recovery latencies than retransmission-based protocols. However, most reliable multicast protocols are based on the idea of packet retransmissions, for the following reasons:

- FEC-based protocols need to know the worst-case packet loss to generate enough redundant data to ensure correct packet

delivery.

- FEC-based protocols typically increase network traffic, even when packet loss rates are low.

Since the use of forward error correction codes alone is often insufficient, and most wireless media allow for bi-directional communication, packet retransmission is often employed even in protocols that are primarily based on the use of FEC, for example [4]. Therefore, we will only explore the use of packet retransmission to achieve reliable multicasting.

A number of specific issues arise when considering packet retransmission. First, should the receivers explicitly request the retransmission of missed packets in the form of negative acknowledgements (NACKs) or should the receiver indirectly communicate this information by only sending positive acknowledgements (ACKs), similar to TCP's acknowledgement mechanism? In a multicast scenario with one or a few senders and many receivers, overwhelming nodes with status information from many receivers (control information implosion) is a serious concern. Under the assumption that losses occur less often than packet receptions, NACKs are our preferred choice.

Second, whom should the receiver contact for retransmission? Ultimately, it could be the original sender, but the sender can be far away, leading to a long recovery time. Many protocols try to speed up the recovery latency and distribute the load by having other nodes buffer data packets and retransmit them on demand. This works particularly well in cellular networks, where base stations/access points are often used for this purpose, servicing all mobile nodes in their area [2, 13, 24]. In a MANET, no such special nodes exist, so either potentially every node provides this functionality or the multicast protocol has to provide mechanisms to identify alternative sources of missed packets. In our protocol, every node will support packet retransmission by buffering the most recent packets.

Third, every node that will retransmit packets on demand will have to buffer them. Since nodes only have a finite amount of buffer space, buffer management becomes important. Ideally, the multicast protocol needs to allow such nodes to determine when all receivers received a given packet, at which point they can discard the packet from their buffer. ACKs (rather than NACKs) do provide this information, which is utilized by some multicast protocols. However, this buffer management introduces another issue: as introduced above, senders send data packets to an IP multicast address, not knowing and usually not caring as to which senders are currently joined. However, the buffer management strategy described above implies that a node/sender knows when ALL receivers received the data packet, i.e., such nodes know the current group membership information. Many reliable multicast protocols for fixed networks therefore provide reliable group membership protocols that allow senders to have an accurate knowledge about the current multicast group status. Since, in a MANET, nodes may be disconnected from the others for unpredictable amounts of time, protocols that do not require accurate knowledge of the group status are preferred.

4.2 Transport Layer vs. Routing Layer

Another major issue concerns whether the reliable protocol should be implemented at the transport layer or at the network layer (i.e., as part of the multicast routing protocol). As stated in [21], more recent multicast protocols tend to follow a design model referred to as "application-layer framing": they are designed for specific applications, rather than as general-purpose protocols. As such, this concept does not necessarily imply that specific functionality is implemented at the transport or application layer. But as more protocols are developed, there is some benefit to separating functionality into multiple layers, providing more generic support at lower layers and more specific functionality at higher layers.

Reliable multicast protocols exemplify such an application-specific approach, and based on the preceding logic therefore should exist higher in the protocol stack than the general-purpose IP routing protocols. This also is one approach currently favored by the IETF who chartered a Working Group on Reliable Multicast Transport Protocols (http://www.ietf.org/html.charters/rmt-charter.html). Examples of reliable multicast protocols designed for specific applications, exploiting application-specific features, are described in [1, 35].

Separating the functionality into different protocol layers has quite a history within the IETF, the most obvious example being the split between IP and TCP. IP provides best-effort

connectivity between hosts, TCP provides true end-to-end communication between applications running on these hosts and provides the abstraction of a reliable data stream over a best-effort IP network. To achieve this, TCP adds port numbers to the IP addresses to uniquely identify specific senders and receivers on each host, and an ARQ mechanism to retransmit lost IP packets.

Separating routing and reliability into two layers has a number of advantages. It allows the reliable multicast protocol to be independent of the underlying multicast routing protocol. This in turn allows the transport layer protocol to benefit from any improvement of the routing protocol. Furthermore, changes to the routing protocol will be confined to the routers, not requiring changes to the end hosts. However, in a MANET, it is questionable whether these advantages still hold. Since all nodes are end hosts and routers, changes to either the reliable multicast protocol or the multicast routing protocol will result in updates to all hosts. In addition, as the experience with TCP shows, a true end-to-end transport layer protocol performs rather poorly in a wireless environment. Following the end-to-end argument of Salzer et al. [28], pushing functionality down the protocol stack often results in better performance. To efficiently deal with the characteristics of the wireless links and the mobility of all nodes, per-hop actions may be preferred. For example, for faster recovery from packet loss, intermediate nodes in the multicast distribution structure may be asked to retransmit the packet. A transport layer protocol is only defined between the communicating peers, so packet retransmissions will have to be requested from the sender in this case. Since a sender is potentially multiple hops away, this can easily result in either the retransmission request or the retransmitted packet to be lost again. Furthermore, even if the retransmitted packet is successfully received, this may take a relatively long time. Therefore, we will implement the reliability mechanism together with the routing protocol at the network layer.

5 Reliable BCAST

The basic mechanism to improve the packet delivery ratio in BCAST is fairly straightforward: every node buffers the last X packets. X can be any arbitrary number, to keep the memory requirement at each node low, we set X to a small number (10 packets) in our code. The buffer is implemented in round-robin fashion, storing the last X unique packets a node received (from the same or a different sender, in order or out of order).

When a node receives a packet with sequence number S from source node SRC, it checks whether it also received packet S-1 from the same source. If not, a node issues a 1-hop broadcast to the neighbors, asking for retransmission of this packet (the NACK message). Each neighbor, upon receiving the NACK packet, checks its local cache and retransmits the packet (assuming it has it in its local buffer). To reduce collisions, the NACKs and the packet retransmissions are jittered randomly by 10 milliseconds. In addition, NACKs have a timeout mechanism associated with them, so even if a NACK or retransmission is lost, packets can be recovered. NACKs are re-issued up to a certain maximum number of attempts (up to 3 times in our implementation).

To reduce network traffic, nodes with pending packet retransmissions will cancel their retransmission if they overhear another node X re-broadcasting this packet. This is based on the assumption that the requesting node will receive this packet as well, satisfying the NACK. This is arguably not guaranteed to be the case: node X could be out of reach of the requesting node, broadcasting packet N not because it received a NACK but because packet N is delivered out of order. However, with multiple NACK attempts (spaced apart multiple seconds), eventually only nodes that received a NACK will attempt to re-transmit a packet. Since they received the NACK, and packets are retransmitted with little additional delay, it is reasonable to assume that the requesting node, in turn, will receive their transmission.

If a sequence of packets is lost, our NACK mechanism recovers from this by backtracking. Assume that packets 3-6 from source S are lost. When a node receives packet 7 from S, it will trigger a NACK for packet 6. Once packet 6 is received, a NACK for packet 5 is triggered, etc. Finally, upon receiving a re-transmitted packet 3, the node determines that it already received packet 2 and the backtracking will stop. Note that this backtracking will also terminate once a packet cannot be recovered. For example, if the node is unable to recover packet 4, it will not check for receipt of packet 3, and

therefore not trigger a NACK for packet 3. All else being equal, this is not unreasonable: since nodes buffer the most recent packets, being unable to retrieve packet 4 from a neighbor is a strong indication that older packets (such as packet 3) may also not be available from those neighbors.

With this NACK based scheme, the packet delivery ratio can be improved. For example, in the 1-sender/10 receiver scenarios, the packet delivery ratio increases from 99.653% to 99.822% under low mobility (as will be discussed in more detail later, this improvement in PDR is statistically significant at a 99% confidence level). In 8 out of 10 simulation scenarios, the packet delivery ratio was a perfect 100%. However, there are still sources of packet loss that a NACK-based scheme cannot completely avoid: problems due to the NACK mechanism itself and long-lived network partitions.

1. *Persistent loss of NACK/retransmission*
   Due to collisions, the NACK or packet re-broadcast may be lost. In this case, the mechanism fails. We improve the reliability of the NACK mechanism by using a timer and re-issuing NACKs if needed. Since, for reasons discussed below, not every packet is recoverable, we keep the number of NACK retransmission attempts to a small number (3) to avoid inducing a high load on the network.
2. *Network partitions*
   Due to the dynamic nature of MANETs, the network may partition for a lengthy period of time. The two scenarios for the 1 sender/10 receiver cases that do not achieve 100% packet delivery ratio exemplify this situation, although they also highlight two important variations. In these two scenarios, node 0 is the multicast sender and nodes 40 through 49 are the multicast receivers. In the first scenario, all receivers receive all packets except for node 44, which misses out on packets 13 to 246. Closer examination of this scenario reveals that up to time 36.55 seconds node 44 is within communication range of node 36, receiving packets 1 through 12. At 36.55 seconds, the two nodes have moved more than 250 meters (the communication range) apart; node 44 becomes disconnected from the rest of the network (together with the non-receiver nodes 4, 20, and 30). At time 156.6 seconds, node 20 has moved sufficiently close to node 7 to establish a link, node 44 is within range of node 20, and packets are delivered again to this receiver. Node 44 recovers, by backtracking, from a number of missed packets, but with nodes only buffering up to the 10 most recent data packets, the recovery is limited by the buffer size and the number of multicast senders. In essence, under low traffic and with a single multicast sender, 10 packets allow to bridge network partitions of up to 5 seconds (2 packets/second traffic rate). To recover from a network partition of 120 seconds, the nodes would have to buffer approximately 240 packets, more if there was more than one multicast sender.

   Even if nodes were to buffer many packets, this does not guarantee 100% reliable data delivery. In the second scenario, the partition occurs towards the end of the simulation, at time 865.2 seconds. At that time, receiver 42 is more than 250 meters away from its closest neighbor. This partition remains until the end of the simulation, so node 42 missed out on the last 81 packets. In the absence of a new packet with higher sequence number, this situation is indistinguishable from the case of the sender having stopped transmitting, so no NACKs are triggered (nor would this help, since the node remains disconnected from the network).

We conducted a first set of simulations based on the mechanisms described above: every node buffers the 10 most recent packets, NACKs are re-issued if we fail to receive a packet for up to three times, with a NACK timeout value of 1 second.

The results for small numbers of multicast senders confirmed our expectations: packet delivery ratios went up to over 99.6% for 1 and 2 sender and consistently above 99.2% for five senders under low mobility. For high mobility scenarios, the PDR was 99.9% for 1 sender, 99.8% for two senders, and 99.5% for five senders. These results show a trend we noticed consistently: high mobility scenarios achieve better performance than low mobility scenarios. The explanation, based on the above discussion, is rather straightforward: under high mobility scenarios, potential network partitions may be more frequent, but are also more short-lived, resulting in fewer packet losses (and therefore a greater chance of being recoverable from a

limited-size cache). However, the results for the 10 sender scenarios were actually worse then the best-effort protocol, dropping to as low as 70%.

Upon closer examination of the results, we discovered that the NACK mechanism potentially results in many NACKs. In the 10 sender scenarios, close to 225,000 NACKs were issued on average per simulation run. This number is significantly higher than the number of NACKs issued in scenarios with fewer multicast senders: around 75 NACKs were issued on average in the 1 sender cases, around 200 NACKS in the two sender cases, and around 2,000 NACKS in the five sender cases. This high number of NACKs (plus the packet retransmission they trigger) substantially adds to the network load, resulting in congestion and packet losses. It is clearly apparent from these numbers that NACKs have to be rate controlled: if the network is busy, aggressively asking for packet retransmissions makes a bad situation only worse.

After exploring a number of options, we settled in the following mechanism to limit the number of NACKs: each node monitors the traffic density by keeping track of when it sends a packet or receives one. This is done in a sliding window of fixed size. Before sending a NACK, the node checks how long ago the earliest packet in that sliding window was send/received. If this period is too short (indicating a high network traffic load), the NACK is suppressed. The size of the sliding window (30) and the minimum required time difference (0.4 secs) are derived at experimentally and balance the need for allowing many NACKs to go ahead with the need to prevent congestion a heavily loaded network. With these parameter settings, few to no NACKs are suppressed in the 1, 2, and 5 multicast sender scenarios, but only 5,300 NACKs are transmitted for 10 sender scenarios, reducing the number of NACKs by over 97%.

	1 Sender		**2 Sender**		**5 Sender**		**10 Sender**	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.998	0.117	0.998	0.109	0.995	0.126	0.972	0.130
20 Receiver	0.998	0.120	0.998	0.113	0.995	0.129	0.972	0.134
30 Receiver	0.997	0.122	0.997	0.114	0.993	0.129	0.970	0.134
40 Receiver	0.996	0.123	0.996	0.114	0.992	0.129	0.970	0.134
50 Receiver	0.997	0.119	0.997	0.111	0.993	0.126	0.970	0.131

Table 7: PDR and Latency for reliable BCAST, 1 m/s maximum speed

	1 Sender		**2 Sender**		**5 Sender**		**10 Sender**	
	PDR	Latency	PDR	Latency	PDR	Latency	PDR	Latency
10 Receiver	0.999	0.109	0.999	0.113	0.996	0.125	0.970	0.132
20 Receiver	0.999	0.108	0.999	0.113	0.995	0.126	0.970	0.132
30 Receiver	0.999	0.108	0.999	0.113	0.996	0.122	0.970	0.132
40 Receiver	0.999	0.107	0.999	0.112	0.996	0.122	0.970	0.132
50 Receiver	0.999	0.105	0.999	0.110	0.996	0.120	0.971	0.129

Table 8: PDR and Latency for reliable BCAST, 20 m/s maximum speed

Tables 7 and 8 summarize the PDR and packet latency under low and high mobility for our reliable BCAST protocol, using NACK rate control. The packet delivery ratio is consistently high (well above 99%) for 1 through 5 multicast senders. As discussed above, higher rates of mobility are good for achieving high packet delivery ratios, since potential network partitions are shorter and can more likely be recovered from with a limited-size cache. For 10 multicast receivers, the packet delivery ratio is significantly lower at 97%, but still improved over the unreliable version by about 0.2%. Under low mobility, packet latency increases only slightly, usually only by about 1 or 2 ms. Under high mobility, average packet latency increases more significantly, by about 5-6 ms for the 1, 2, and 10 multicast sender scenarios, and over 10 ms in the 5 multicast sender scenarios. These increases are in line with the number of packet losses our NACK-based scheme recovers from, which is highest for the 5 multicast sender scenarios, adding to the average packet latency in those scenarios.

Tables 7 and 8 show that reliable BCAST improves PDR and causes slightly increased packet latency, compared to the best-effort version (Tables 3 and 4). However, are these differences statistically significant? For each scenario, 10 simulation runs with different initial node positions and node mobility patterns were conducted. As we use the same mobility patterns for the best-effort version of BCAST and the reliable version, we perform paired t-tests on the average PDR and packet latency for each group of 10 runs. In all scenarios the protocol performance differences are statistically significant at a 99% confidence level. That is, with high probability, the observed increased packet delivery ratios and packet latencies are a direct consequence of the protocol changes.

6 Related Work

A number of researchers have proposed reliable multicast protocols for fixed networks (a detailed survey can be found in [17]) and cellular networks [2, 13, 24]. The latter protocols typically assign a special role to the base stations/access points, and are therefore not applicable in a MANET environment. Relatively few papers have addressed the issue of reliable multicasting in a MANET. [30] proposes RALM (Reliable Adaptive Lightweight Multicast), a multicast transport layer protocol that achieves relatively high packet delivery by throttling traffic using a window mechanism based on congestion experienced by a feedback receiver. [7] describes RMA, a Reliable Multicast Protocol for MANETs. The protocol is based on the assumption that senders know the identities of all receivers and achieves reliability by explicit ACKs from all receivers. The protocol achieves high packet delivery ratio with a data packet overhead similar to MAODV. Finally, [29] discusses how to assure packet delivery in MANETs using error correction codes: packets are encoded and split into smaller pieces, which are transmitted independently to the receiver. Assuming a certain threshold of pieces arrives at a receiver, the packet can be re-assembled. No performance evaluation is provided in the paper, however.

Our approach is different from these attempts: we increase packet delivery at the routing layer, unlike [30], which described work at the transport layer, for reasons discussed above. Our protocol does not require the senders to know the identity of all senders, unlike [7], and it is based on packet retransmission and not forward error correction codes, as in [29].

7 Conclusions and Future Work

This paper summarizes our experiences with implementing a reliable multicast protocol. Based on a study of the performance of various alternatives, we selected a broadcast protocol, BCAST, and extended it with a NACK mechanism to increase the packet delivery ratio. As the discussion shows, designing a reliability mechanism to improve packet delivery ratio is non-trivial. Due to the dynamic nature of MANETs, achieving 100% packet delivery is unrealistic, since nodes can and will become partitioned from the multicast senders. In particular under low mobility, such network partitions can exist for lengthy periods of time, recovering from them is therefore not trivial, and would require substantial buffer space at all MANET nodes. It is also probably not desirable from an application perspective: in a VoIP application, for example, a user will assume that the call is disconnected and terminate the application, rather than waiting for 100 seconds to continue the conversation. Similarly, delays of 100 seconds will probably cause a user to terminate interactive downloads and web browsing sessions.

Our protocol achieves consistently over 99% packet delivery ratio under both low and high mobility for a small number of multicast senders. We also noted that higher mobility rates are actually beneficial for achieving high packet delivery ratios, as potential network partitions tend to be more short-lived and therefore can more easily be recovered from. As the number of senders (and therefore the overall traffic load) increases to 10 senders, congestion starts to set in and packet delivery drops to around 97%. Under this high traffic load, the number of NACKs and packet retransmissions has to be carefully controlled to not increase the overall rate of congestive losses. Our protocol therefore implements a mechanism to rate control NACKs based on traffic load.

Future work will progress along a number of lines. First, we will explore ways to fine-tune the protocol parameters, such as the NACK throttle mechanism or “Hello” interval.

Ideally, appropriate parameters could be derived automatically, based on observed network traffic and rate of neighborhood change. Second, as discussed above, we will explore ways to increase the network capacity by modifying the MAC protocol. As more traffic can be carried, packet delivery ratios should increase, in particular for scenarios with many multicast senders and/or more data per sender. To increase the link capacity, we can either replace the current MAC protocol, IEEE 802.11, or modify this protocol. Replacing the MAC protocol with a scheduling-based protocol is a more radical change. Since IEEE 802.11 is widely used in practice, we will concentrate on ways to increase its efficiency, along the ideas presented in [11], [27], and [33]. Finally, we will further explore whether broadcast protocols are indeed the most efficient starting point for a reliable multicast routing protocol by evaluating the performance of other multicast protocols such as ADMR [9, 20].

Acknowledgements

This work is funded by the Defense Research and Development Canada (DRDC) and benefited from various discussions with members of the CRC RNS mobile networking group.

About the Author

Dr. Thomas Kunz is a faculty member in the Department of Systems and Computer Engineering at Carleton University. His main research interests are in adaptive mobile applications and wireless networks, including ad-hoc networks. In the latter area, he is interested in routing protocols and QoS issues at the MAC, routing, and application layer.

References

[1] B. Adamson, G. Borman, M. Handley, and J. Macker, *NACK-Oriented Reliable Multicast Protocol (NORM),* IETF INTERNET-DRAFT draft-ietf-rmt-pi-norm-06.txt, expires September 2003 (work in progress).

[2] G. Anastasi, A. Bartoli, and F. Spadoni, *A Reliable Multicast Protocol for Distributed Mobile Systems: Design and Evaluation*, IEEE Trans. on Parallel and Distributed Systems, pp. 1009-1022, Vol. 12, No. 10, Oct. 2001.

[3] E. Cheng, *On-demand Multicast Routing in Mobile Ad Hoc Networks,* M.Eng. Thesis, Carleton University, Ontario, Canada. January 2001.

[4] P. Chumchu and A. Seneviratne, *HMoM: A Reliable Mobile Multicast Scheme Using Hybrid ARQ: Proposed Architecture and Performance Analysis*, in Proc. of Wireless 2002, pp. 464-473, Alberta, Canada, July 2002.

[5] W. Ding, *Multicast Routing in Fixed Infrastructure and Mobile Ad Hoc Wireless Networks with a Multicast Gateway,* M.Sc. (ISS) Thesis, Carleton University, Ontario, Canada, July 2002.

[6] C. Fidge, *Logical Time in Distributed Computing Systems*, IEEE Computer, pp. 28-33, Vol. 24, No. 8, August 1991.

[7] T. Gopalsamy, M. Singhal, D. Panda, and P. Sadayappen, *A Reliable Multicast Algorithm for Mobile Ad Hoc Networks,* in Proc. of the Distributed Computing Systems Workshops, pp. 563-570, Vienna, Austria, July 2002.

[8] J. Grudin, *Group Dynamics and Ubiquitous Computing*, Communications of the ACM, pp. 74-78, Vol. 45, No. 12, December 2002.

[9] J.G. Jetcheva and D.B. Johnson, *Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad Hoc Networks*, in Proc. of the 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc 2001), pp. 33-44, ACM, Long Beach, CA, October, 2001.

[10] M. -Y. Jiang and W. Liao, *Family ACK Tree (FAT): A New Reliable Multicast Protocol for Mobile Ad Hoc Networks*, in Proc. of the IEEE Int. Conf. on Communications, pp. 3393-3397, NY, USA, April 2002.

[11] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. Knightly, *Distributed Multi-hop Scheduling and Medium Access with Delay and Throughput Constraints*, in Proc. of the 7th Conf. on Mobile Computing and Networking, pp. 200-209, Rome, Italy, July 2001.

[12] T. Kunz, *Multicasting: From Fixed Networks to Ad Hoc Networks*, in Handbook of Wireless Networks and Mobile Computing, pp. 495-507, John Wiley & Sons 2002, ISBN 0-471-41902-8.

[13] J. Kuri and S.K. Kasera, *Reliable Multicast in Multi-Access Wireless LANs,* Wireless Networks, pp. 359-369, Vol. 7, 2001.

[14] [S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, *A Performance Comparison Study*

of Ad Hoc Wireless Multicast Protocols, in Proc. of the 19th Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM 2000), Vol. 2, pp. 565-574, Tel Aviv, Israel, Mar. 2000.

[15] S.-J. Lee, W. Su, and M. Gerla, *On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks*, Mobile Networks and Applications, pp. 441-453, Vol. 7, No. 6, December 2002.

[16] J. Li, C. Blake, D. De Couto, H.I. Lee, and R. Morris, *Capacity of Ad Hoc Wireless Networks*, in Proc. of the 7th Conf. on Mobile Computing and Networking, pp. 61-69, Rome, Italy, July 2001

[17] V. Li and Z. Zhang, *Internet Multicast Routing and Transport Control Protocols*, in Proc. of the IEEE, pp. 360-391, Vol. 90, No. 3, March 2002.

[18] W. Lou and J. Wu, *On Reducing Broadcast Redundancy in Ad Hoc Wireless Networks*, IEEE Transactions on Mobile Computing, pp. 111-122, Vol. 1, No. 2, April 2002.

[19] F. Mattern, *Virtual Time and Global States of Distributed Systems*, in Proc. of the Workshop on Parallel and Distributed Algorithms, pp. 215-226, North Holland / Elsevier, 1989.

[20] The Monarch Project Software Distributions, http://www.monarch.cs.rice.edu/ software.html

[21] K. Obraczka, *Multicast Transport Protocols: A Survey and Taxonomy*, IEEE Communications Magazine, pp. 94-102, January 1998.

[22] K. Obraczka, G. Tsudik, and K. Viswanath, *Pushing the Limits of Multicast in Ad Hoc Networks,* in Proc. of the 21st Int. Conference on Distributed Computing Systems, pp. 719-722, Phoenix, USA, 2001.

[23] K. Obraczka, K. Viswanat, and G. Tsudik, *Flooding for Reliable Multicast in Multi-Hop Ad Hoc Networks,* Wireless Networks, pp. 627-634, Vol. 7, No. 6, 2001.

[24] H. Omar, T. Saadawi, and M. Lee, *An Integrated Platform for Reliable Multicast Support in the Regional Mobile-IP Environment*, Mobile Computing and Communications Review, pp. 37-54, Vol. 6, No. 2, April 2002.

[25] A. Mankin, A. Romanow, S. Bradner, V. Paxson, *RFC 2357: IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols*, June 1998. (Status: INFORMATIONAL)

[26] B. Quinn, K. Almeroth, *RFC 3170: IP Multicast Applications: Challenges and Solutions*, September 2001. (Status: INFORMATIONAL)

[27] R. Rozovsky and P. R. Kumar, *SEEDEX: A MAC Protocol for Ad Hoc Networks*, in Proc. of the Second Symposium on Mobile Ad Hoc Networking and Computing, pp. 67-74, Long Beach, USA, October 2001.

[28] J. Salzer, D. Reed and D. Clark, *End-to-end Arguments in System Design*, ACM Transactions on Computer Systems, pp. 277-288, November 1984.

[29] L. Shu and D. Poppe, A*ssuring Message Delivery in Mobile Ad Hoc Networks with Packet Erasure Recover*y, in Proc. of the Distributed Computing Systems Workshops, pp. 14-19, Vienna, Austria, July 2002.

[30] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla, *A Reliable, Congestion-Controlled Multicast Transport Protocol in Multimedia Multi-hop Networks*, in Proc. of the 5th International Symposium on Wireless Personal Multimedia Communications, pp. 252-256, Honolulu, USA, October 2002.

[31] U. Varshney, *Multicast over Wireless Networks*, Communications of the ACM, pp. 31-37, Vol. 45, No. 12, December 2002.

[32] B. Williams and T. Camp, *Comparison of Broadcast Techniques for Mobile Ad Hoc Networks*, in Proc. of the Third Symposium on Mobile Ad Hoc Networking and Computing, pp.194-205, Lausanne, Switzerland, June 2002.

[33] X. Yang and N. H. Vaidya, *Priority Scheduling in Wireless Ad Hoc Networks*, in Proc. of the Third Symposium on Mobile Ad Hoc Networking and Computing, pp. 71-79, Lausanne, Switzerland, June 2002.

[34] Y. Zhu, *Pro-Active Connection Maintenance in AODV and MAODV,* M.Sc. (ISS) Thesis, Carleton University, Ontario, Canada, August 2002.

[35] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J.D. Kubiatowicz, *Bayeaux: An Architecture for Scalable and Fault-tolerant Wide-are Data Dissemination,* in Proc. of the Int. Workshop on Network and Operating System Support for Digital Audio and Video, pp. 11-20, Port Jefferson, USA, June 2001.