A Reliable Extension to the ODMRP Ad Hoc Multicast Protocol

Lawrence Klos, Golden G. Richard III (lklos@uno.edu, golden@cs.uno.edu)
Department of Computer Science
University of New Orleans, New Orleans, LA 70148

Abstract

In this paper an enhanced reliability protocol, R-ODMRP, added to the ODMRP multicast ad hoc protocol is described. This NACK based protocol increases overall data packet delivery by adding data storage and retransmission operations coordinated by the multicast source. Storage responsibilities are assigned to individual nodes based on localized 'neighborhoods' with minimal spanning hopcount, within the receiver group. A description is included of the mechanism the source uses to discover and partition all group members into neighborhoods. Each neighborhood stores a sliding window of the full set of data, keeping resend requests and replies localized to one portion of the overall network, reducing network overhead. Simulation results are presented that reflect the enhanced reliability characteristics of the protocol. A discussion is included clarifying the competing metrics and tradeoffs in providing reliability.

Keywords: Ad Hoc, Multicast, Reliable, ODMRP.

1 Introduction

Ad hoc networks consist of sets of mobile wireless nodes communicating with no fixed infrastructure support. Communication occurs on a node to node basis, with the links established between nodes forming the overall ad hoc network. Applications for ad hoc networks range from an informal collection of conference participants in a ballroom, to soldiers organizing in a battlefield or rescuers coordinating efforts on a remote mountaintop. Group communication is a natural extension of ad hoc communication. For group communication, reliability is often an important issue: in situations such as rescue operations, missed messages can have critical consequences.

2 Reliable Group Communication

Reliable group communication research has a relatively long history in wired networks. Reliable communication guarantees in wired networks begin with the notion that all nodes in a group will eventually receive all data packets. The *Dynamic Group Membership* model[1] allowed for nodes in wired networks to join and leave a group at will, with reliable communication guarantees enforced. It is beneficial to understand this model, as the necessity for nodes to unpredictably join and leave an ad hoc group is even greater, due to network partitions caused by node mobility. In wired group communication, research has differentiated between *sender based* (ACK) and *receiver based* (NACK) responsibility for message delivery. In a receiver based model, a clear advantage is that the list of group members required by the *Dynamic Group Membership* model is not necessary. Receivers take on the responsibility for NACKing missed packets themselves. In wired multicast this is an invaluable advantage, avoiding ACK implosions and the large overhead needed to maintain the dynamic membership list. In ad hoc networks this advantage becomes a necessity: resource constraints make it infeasible to maintain and continually update a membership list.

To date, it is not feasible to implement reliable communication protocols in ad hoc networks. It is not even possible to guarantee delivery of all data packets to all nodes. This paper is an initial study of factors involved in eventually reaching 100% reliability. Generally, two factors are responsible for missed messages in ad hoc networks: one is network link contention (whether the contention results from control overhead or data forwarding efficiency overhead), and the second is non existent links. The first factor is more common in dense networks, while the second factor occurs increasingly as density lessens. The reliability of most ad hoc multicast protocols are affected by these two factors, because most protocols rely on best-effort delivery of packets. Some mechanism must then be added to enhance reliability, if the eventual goal is 100% reliability.

A mechanism of storing and resending data packets is one potential solution to both factors. Whether packets are undelivered due to link contention or missing links, future resends of packets allow improved chances of eventual delivery. Since a reliability component will increase network link contention, the design goal of such a component is to increase this contention minimally, causing the least number of additional dropped packets. To achieve this goal, the control and data packet resend traffic overhead of the reliability component must be as low as possible.

In this paper, the strategy taken was to implement a receiver based model for data delivery. ODMRP was taken as the underlying multicast group communication protocol, since it is well documented and simulations [6] have shown it to have a high data delivery ratio in comparison with other ad hoc multicast protocols. Mechanisms for message storage and retransmission were added, distributing these responsibilities to all group members, with the source of each data stream being responsible for storage coordination. As the network grows, replicated data storage is assigned to an expanding number of individual neighborhoods of nodes. This means that NACK requests/replies of data packets will require packet traversal of fewer and fewer hops to fulfill, reducing network overhead and data delivery latency.

The choice of the source node as coordinator for these responsibilities is not intuitive. It is common wisdom that ad hoc multicast protocols are distributed algorithms, and centralized components are to be avoided. However, several factors suggested that this centralized mechanism could be successful:

- For any ad hoc protocol, an implicit centralized component, the data source, is unavoidable. If the central source of data dies or is partitioned, there will be no data to reliably deliver.
- ODMRP, while categorized as non-centralized, still relies on the near-centralized two step process of join query/reply to establish datapaths. This mechanism closely resembles R-ODMRP's mechanism of reliable join query/reply from a network traffic standpoint, with the final step of *nodes adjacent to the source sending a packet back to the source* removed.

Section 3 presents a brief overview of ODMRP, section 4 describes R-ODMRP, and section 5 presents a performance evaluation. Finally, section 6 discusses related work, section 7 covers future work, and section 8 concludes the paper.

3 ODMRP overview

ODMRP[5] is a mesh-based on-demand ad hoc protocol for group communication in ad hoc networks. Simulations [6] have shown it to be very reliable relative to other ad hoc multicast protocols. It performs scoped flooding of data packets to all group members in the network by establishing a 'forwarding group' of network nodes between a source and group members. Timeout driven periodic route refreshes update the broken links due to node mobility or resource changes. Route setup and refresh both consist of two phases: a Request and Reply phase.

3.1 Route Setup – Request Phase

When a source has multicast data packets to send but no knowledge of routes to receivers, it piggybacks the first data packet onto a "Join Query" packet, adds its IP address as source, and broadcasts it to the network. Each node receiving the Join Query packet will store the source IP address and packet ID for future duplicate packet detection, add the IP addresses of the upstream node and originating source to its routing table, add its own IP address into the last hop IP address field, and rebroadcast it downstream. The Join Query packet eventually floods to reach all nodes.

3.2 Route Setup – Reply Phase

A group member, on receiving a Join Query packet, first completes the Join Query processing described above, then initiates a "Join Reply" packet. The node pulls all source and next hop IP addresses for the group from its routing table, add its own IP address into the previous hop field, and broadcasts the Join Reply upstream. Each neighbor node receiving this packet compares its IP address to the series of next hop IP address entries, and if one matches, it is on the forwarding path between source and receiver. The node sets its Forwarding Group Flag, looks into its own routing table entries for the group ID and builds its own Join Reply packet to broadcast upstream.

3.3 Forwarding, Maintenance and Unicasting

When a node receives a multicast data packet, it checks the setting of its Forwarding Group Flag. If the flag is set and not yet expired, the node is a forwarding group member for the group, and will rebroadcast the packet to its neighbors.

Periodically, the source will refresh routes with another Join Query packet. All forwarding groups are then reset to the new network topology. A soft state approach is taken for group membership. Once a source has no data to multicast, it stops sending Join Query packets. All forwarding nodes eventually timeout and revert to non-forwarding status for that source. If a group receiver wants to leave the group it stops sending Join Replies in response to Join Queries.

Using the same Join Query/Reply protocol with a unicast IP address as the destination, a unicast sender can discover a route to a receiver. The forwarding group route created by the unicast operation of ODMRP is a single path rather than a mesh.

4 Reliable ODMRP

In R-ODMRP the responsibility for data packet storage and retransmission is assigned to all members of the multicast group, with the source of each data stream coordinating responsibilities. The full set of group members are partitioned by the source into sets of local neighborhoods, with each neighborhood storing a sliding window of all data packets. The source sets the number of nodes per neighborhood and determines the data cache size required at each node. When receivers Nack missed packets, the local neighborhood nodes with the cached packets will unicast them to the Nacking node.

4.1 Packet Storage

In R-ODMRP, When a source initially sends out a Join Query, it becomes a Reliable Join Query (RJQuery) packet. The RJQuery packet has a timeout value attached. Once the RJQuery packet is sent, each node receiving it will decrement this timer value by a preconfigured "two hop time" before sending the RJQuery downstream. After the RJQuery timer expires at each node, each will send a Reliable Join Reply (RJReply) back upstream. If a node with an expiring timer is not a receiver, it will send an RJReply only if it receives other RJReplies from downstream.

Each RJReply contains a 2D table, known as the Network Datapath table. When a node receives RJReplies from downstream nodes, it stores their Network Datapath table blocks in its own table sorted relative to other received blocks with the topmost block having the longest datapath. On timer expiration, just before the table is sent upstream in an RJReply, each table entry is shifted such that entry (x, y) becomes entry (x, y + 1), emptying the leftmost column, column 0. The node stores an entry for itself in entry (0,0) containing its id, branch count (the number of received RJReplies received), and receiver status. It then forwards the table upstream in its own RJReply. Figure 1 shows the datapath tables for several nodes after processing downstream tables.

The end result of the RJQuery/RJReply phase is that the source obtains a full positional listing of all receivers and forwarding group members in the network. RJQuery/Reply operations occur periodically, but at a lower frequency than the standard Join Query/Reply operation. The source will then determine the "nodes per neighborhood" count, and, use the Network Datapath table to partition all receiver nodes into local neighborhoods with a "Source Neighborhood" Algorithm. The source assigns data packet storage responsibilities so that the set of nodes within any given neighborhood will store the full set of data packets in sliding window fashion.

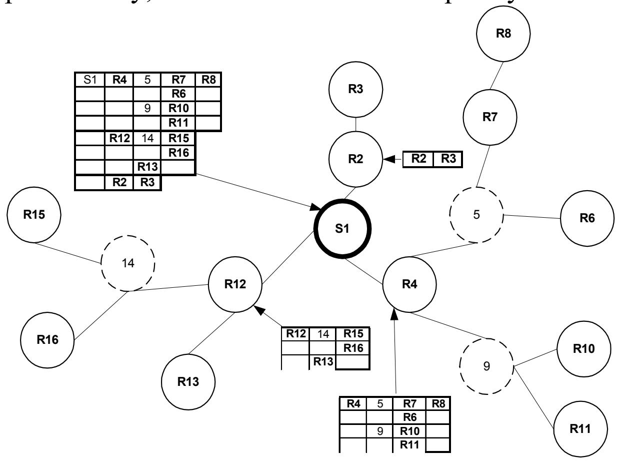

Figure 1: Network Datapath Table creation.

On the next data packet after a Reliable Join Query, the source adds a table defining the range of packet sequence numbers each receiver in each neighborhood is responsible for storing. Each receiver then begins storing its share of data packets. This recovery scheme does not depend on which node stores the packets, only that each neighborhood stores them somewhere.

As nodes leave the group, their storage responsibilities are reassigned on new RJQuery/Reply rounds. However, as more and more nodes join over time, more neighborhoods are created and duplicate storage responsibilities are assigned. The individual neighborhoods storing the duplicate packets become smaller and smaller, relative to the overall network.

4.2 Packet Retransmission

The second responsibility, that of data packet retransmission, will be initiated by a receiver node noticing a gap in received data packets. It will broadcast a Resend Request packet to its neighborhood, with a local time-to-live scope, listing the packets needed by sequence number. The requester will give its IP address for unicast replies. A neighborhood node, upon receiving the packet, will check its storage for the requested sequence numbers, unicasting found data packets back along a single path. If the requesting node receives an incomplete reply or no reply, it retains the gap sequence numbers, sending them out with the next Resend Request.

As the number of group members in a network grows, the size of local neighborhoods (and hopcount of Resend Request/Replies) becomes smaller relative to the overall network.

4.3 Source Neighborhood Algorithm

The Source Neighborhood Algorithm is the mechanism the source node uses to partition receivers into neighborhoods, and to assign data packet storage responsibilities to all receivers in each neighborhood. The neighborhood partitioning algorithm is based on minimizing the hopcount between the receivers within a neighborhood, for each neighborhood in the overall network. The input to this algorithm is a Network Datapath table, with each entry consisting of a structure with the following elements:

- Address: the node's individual id.
- Branch_Count: the number of table rows (i.e. downstream branches) extending from this node. This is the same as the count of received RJReplies.
- Receiver_Status: a boolean indicating if this node is a multicast receiver or forwarder.

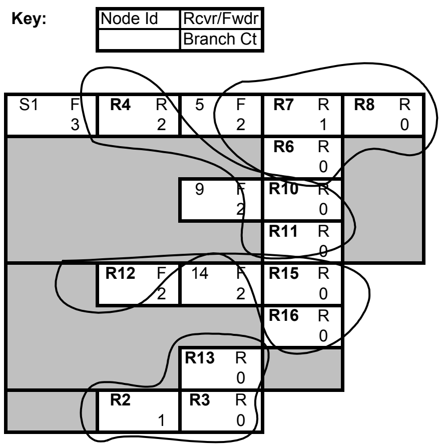

Figure 2: Partitioning of Network Datapath Table.

Figure 2 shows the Network Datapath Table in the source node for the network shown in Figure 1, with receivers partitioned into neighborhoods as a result of this algorithm using[receivers per neighborhood] = 3. The algorithm works by taking the most remote node in the topmost block (the block with the longest datapath to the farthest receiver in the network) and stepping backward along the row, partitioning receivers into the first neighborhood. Once a branch node is reached, the algorithm steps down into the proper branch row, moves to its end, then works back toward the source, again partitioning receivers into the first neighborhood. During this process the algorithm tracks neighborhood hopcount, while remaining within an individual block. Once enough receivers within the first block are partitioned into full neighborhoods, processing moves on to the next block.

After receivers in each block are partitioned into full neighborhoods there will be remainder nodes potentially from each block. Remainder nodes are then partitioned into neighborhoods. Given that remainder nodes will be those closest to the source, remainder node neighborhoods should also have a minimal hopcount between members.

After all receiver nodes are partitioned into neighborhoods, the neighborhoods are entered into a Storage Responsibility table, with each row containing an individual neighborhood, and data packet sequence number storage implicitly assigned by column.

Pkts 1-33	Pkts 34-66	Pkts 67-100	Nbrhd Hopct
R8	R7	R6	3
R4	R10	R11	2
R15	R16	R12	2
R13	R3	R2	4

Figure 3: Storage Responsibilities

Max hopcount per neighborhood is stored in this table at the end of each row; see Figure 3 for the neighborhood partitioning and storage assignments for the Network Datapath table partitioned in Figure 2. This Storage Responsibility Table is then added to the next outgoing Join Query packet and sent out to all receivers. Each receiver, upon obtaining this packet, parses out its neighborhood spanning hopcount and storage responsibilities, and begins storing packets.

5 Protocol Performance Evaluation

R-ODMRP was implemented within the ns-2 network simulator[3], developed by the University of California, Berkeley, and the VINT project, with Carnegie Mellon's Monarch Project mobile and wireless ns-2 extensions [10] incorporated. The ns-2 simulator is publicly available, and commonly used in networking research. [2] provides a comprehensive description of the software layers and the IEEE 802.11 MAC protocol used in these simulations. One reason for ns-2's popularity is the detailed manner in which the layers of the communication hierarchy are modeled, and the configurability at each layer.

5.1 Simulation Details

The ODMRP and R-ODMRP simulations all executed with identical randomly generated baselines of network traffic and node movement files to more accurately compare performance. This baseline consisted of five node movement scenarios and six traffic pattern scenarios. All scenarios established fifty mobile nodes with a single node as multicast source within a 1000m x 1000m area. The radio propagation range for each node was 250 meters, and the channel capacity was 2 Mb/sec. Each simulation executed for 600 seconds of simulated time. Once all nodes joined the group the multicast source started sending 512 byte packets with a constant bit rate of 3 packets per second. The traffic pattern scenarios had 25, 30, 35, 40, 45 and 49 receiver nodes respectively.

30 simulation runs each for ODMRP and R-ODMRP. A total of 60 simulations were performed. This baseline was chosen for initial results because simulations [6] have shown that ODMRP performs best in conditions of relatively good network connectivity, low traffic load, and speed, and any protocol with the goal of increasing reliability would have to outperform it under these conditions. R-ODMRP would seem to have its greatest advantages in sparse networks with frequent partitions, however. For ODMRP and R-ODMRP, the ODMRP parameters were set to 3 seconds for the Join Query flood interval and 9 seconds for the forwarding state timeout, the values used by ODMRP's creators in their simulation studies. R-ODMRP set a flag in every fourth Join Query packet, turning it into a Reliable Join Query packet. The node count per neighborhood for R-ODMRP was 3, and nodes were set to store a maximum of 500 data packets, in Round Robin mode.

5.2 Evaluation Metrics

The performance of ODMRP and R-ODMRP were compared according to the following metrics.

- *Packet Delivery Ratio*: the number of multicast data packets from a source correctly received over the number that should have been received.
- *Ratio of Data and Control Packets per Delivered Data Packet*: measures protocol routing efficiency.
- *Forwarding Efficiency:* the average number of times each data packet was transmitted.

5.3 Initial Experiments

Initial experiments lead to modifications of the basic mechanisms of R-ODMRP to produce better end results. Originally, the time-to-live hopcount for a resend request packet was set to the maximum distance between nodes within a given neighborhood. This produced average results. Some data packets that would have been correctly delivered under ODMRP were dropped due to channel contention with the Resend Requests, causing the R-ODMRP portion of the protocol to work harder to fill the gaps, leading to further network contention. In the end, a TTL of 1 gave best results for Resend Request packets.

5.4 Simulation Results

Figure 4 shows initial results for Packet Delivery Ratio. The ODMRP protocol alone produced a packet

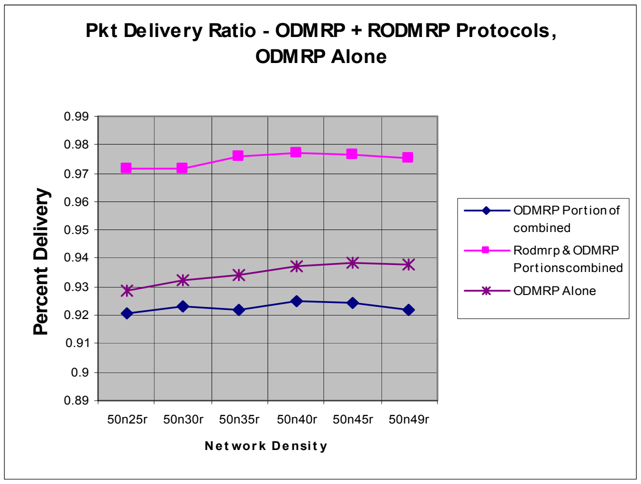

Figure 4: Packet Delivery Ratio.

delivery ratio of 93% to 94%. For the R-ODMRP protocol (the results of which were broken down into the ODMRP portion alone, and the R-ODMRP portion combined with ODMRP), the ODMRP portion operated between 1% and 1½ % worse than its standalone counterpart. This represents the decrease in normal packet delivery attributable to the overhead of the R-ODMRP portion. With the R-ODMRP portion combined however, the combination consistently performed about 4% better overall than ODMRP.

The other metrics show the tradeoffs for R-ODMRP. The Ratio of Data and Control Packets per Delivered Data Packet, shown in Figure 5, shows an increase for R-ODMRP. An increase must exist, since the basic premise is retransmission of previously sent data packets. The increase shown for R-ODMRP is manageable however, scaling similarly to ODMRP.

Figure 6 shows the Forwarding Efficiency of both protocols. Here, R-ODMRP shows worse performance than ODMRP. Again, this is expected given the basic premise. Data delivery latency of the two protocols shows the greatest differential, however. While the average latency of ODMRP, and the ODMRP portion of R-ODMRP averaged about 10 ms across all receiver counts, the extra packets delivered by the Resend

Request/Reply mechanism tended to have a latency of seconds due to several factors. One is the fact that two seconds elapse after a gap is noticed before the Resend Reply packet is triggered, and a second is that a random delay is added to allow nodes to snoop other node's Resend Replies.

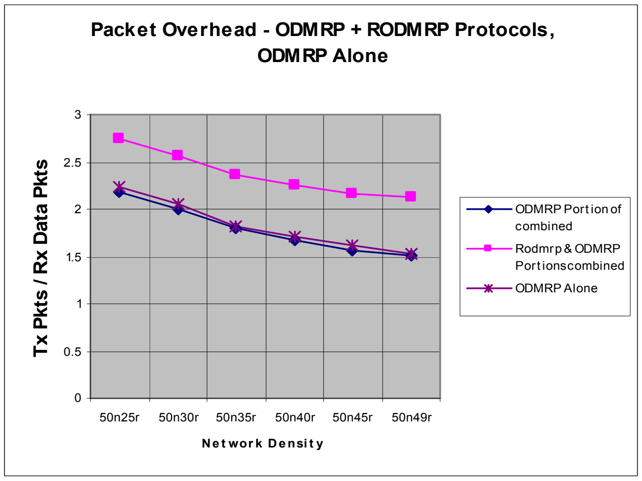

Figure 5: Ratio of Data+Control Pkts per Delivered Data Pkt.

In dense networks future experiments will fine tune the tradeoff between latency and increased channel contention from frequent Resend Requests, to bring this latency down. In sparse networks this is less of an issue, since network partitions will put minimum possible boundaries on latency, making the time required for Resend Requests/Replies not an issue.

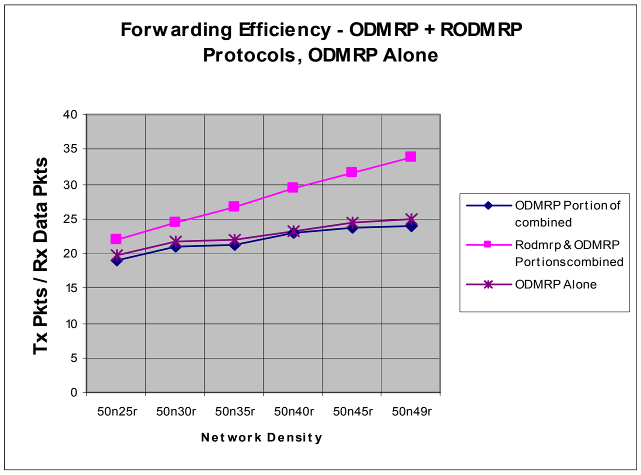

Figure 6: Forwarding Efficiency

The competing metrics involved in enhancing reliability for ODMRP have been clarified as a result of this work. Four central factors balance against each other: Packet Delivery Ratio ("Reliability"), Ratio of Data and Control Packets per Delivered Data Packet ("Control Overhead"), Forwarding Efficiency ("Forwarding Overhead") and Data Packet Latency to all Receivers ("Latency"). When the R-ODMRP portion is added to ODMRP, reliability increases, latency increases and network traffic overhead increases. A successful reliability component will increase reliability, increase overhead by an 'acceptable' amount (i.e. low enough so that the resulting additional dropped packets can be overcome), and increase data packet latency as minimally as possible. Of the three factors, latency can be the least critical when increased reliability is desired, since the overhead metrics are more tightly linked to the reliability metric.In multicast ad hoc protocols, reliability falls off sharply as node density becomes sparser. It is expected that the sparser the network, the more successful a reliability component such as R-ODMRP will be in achieving its goals.

6 Related Work

Research has begun on reliability in multicast ad hoc networks in recent years. Mechanisms can be broken down into three categories; congestion control, data packet retransmission and forward error correction.

RALM[9] is a reliable ad hoc multicast protocol that works by enforcing error and congestion control similar to TCP in wired networks. Reliable data delivery is guaranteed to one group member at a time, round-robin, by requiring the source to select a neighboring group member for data transmission. The neighboring member will reply with either an ACK or NACK. This feedback from the neighbor is used to initiate retransmission and adjust the source window size. Here, decreased throughput is the tradeoff for increased reliability. Another downside is that even with congestion control, if packets are dropped, they are resent from the source rather than locally, requiring more network overhead both for the receiver NACK and the source retransmit, and requiring the source to cache all data.

Packet Erasure Recovery[8] uses error correction codes to provide reliability. Data packets are encoded and split into fragments with replication, and the fragments are then transmitted to receivers. If a certain number of fragments arrive, the data packet is correctly reassembled. The downside of this approach is that forward error correction work best when loss rates are predictable. In ad hoc networks, worst case behavior for a node to drop packets is unpredictable: a node may go out of range of the network and stay out of range. Also, more bandwidth and processing power is consumed with replicated data transmission.

Hyper flooding [7] is an adaptive technique using flooding as a base, with modifications to prevent loops. Received data packets are stored internally by nodes that record neighbors by listening for and sending hello messages. Rebroadcasts of stored data packets occur when a packet is received from a node that is not on the neighbor list, or when receiving a new hello message. In these cases, all packets in the data packet cache are retransmitted. The downside of this approach is that a greater amount of network overhead is used, which leads to more channel contention and data loss. Another disadvantage is the large amount of data caching at each node.

7 Future Work

Near term goals include examining the Resend Request/Reply mechanism to increase scalability for growing receiver counts in dense networks. A closer look at packet ratios for each phase of R-ODMRP should provide information here. Also, examination of the transmission/reception details of the simulations will provide further data to close the gap to full packet delivery. It would be good to examine other network scenario baselines. For example, sparse networks would seem to be an area where R-ODMRP could operate to greatest advantage. A new mechanism for Resend Requests will be needed for this scenario.

8 Conclusion

This paper described R-ODMRP, a reliability protocol overlaid on ODMRP. R-ODMRP consists of operations to store and retransmit sequenced data packets between receiver nodes, with overall functionality coordinated by the source node.

R-ODMRP has been implemented in the ns-2 simulator and run against a baseline scenario of a dense network with increasing receiver count, ideal conditions for the base ODMRP protocol. Results show that R-ODMRP does outperform ODMRP under these conditions, but at a cost of an increase in routing efficiency, forwarding efficiency and data delivery latency. Future work will determine how the two protocols compare in less than ideal conditions.

9 References

[1] K. Birman, "Building Secure and Reliable Network Applications", Manning Publishing Company, Greenwich, CT, and Prentice Hall 1997.

[2] J. Broch, D. Maltz, D. Johnson, Y. Hu and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", Proc. of the Fourth Annual ACM/IEEE International Conference on Mobile Computing And Networking, Dallas, TX, Oct. 98.

[3] K. Fall, K. Varadhan, editors, "The ns Manual", The VINT Project, UC Berkeley, LBL, USC/ISI, and XEROX PARC, April 2002. Available at http://www-isi.edu/nsnam/ns/.

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

[5]S.J. Lee, W. Su and M. Gerla, Internet Draft, "On-Demand Multicast Routing Protocol(ODMRP) for Ad Hoc Networks", Jan. 2000. draft-ietf-manet-odmrp02.txt

[6] S.J. Lee, W. Su, J. Hsu, M. Gerla and R. Bagrodia "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols", Proc. of IEEE INFOCOM 2000, Tel Aviv, Israel, March 2000.

[7] K. Obraczka, G. Tsudik, K. Viswanath, "Pushing the Limits of Multicast in Ad Hoc Networks", Proc. of the 21st International Conference on Distributed Computing Systems, Phoenix, Arizona, April, 2001.

[8] L. Shu, D. Poppe, "Assuring Message Delivery in Mobile Ad Hoc Networks with Packet Erasure Recovery", Proc. of the Distributed Computing Sys. Workshops, pp 14-19, Vienna, Austria, July 2002.

[9] K. Tang, K. Obraczka, S.J. Lee, M. Gerla, "A Reliable Congestion-Controlled Multicast Transport Protocol in Multimedia Multi-hop Networks", The 5th International Symposium on Wireless Personal Multimedia Communications, pp 252-256, Hololulu, USA, October 2002.

[10] "The CMU Monarch Projects wireless and mobility extensions to ns", The CMU Monarch Project, August 1999. Available at http://www.monarch.cs.cmu.edu/.