Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks*

Ranveer Chandra   Venugopalan Ramasubramanian   Kenneth P. Birman
Department of Computer Science
Cornell University, Ithaca, NY 14853, USA
{ranveer, ramasv, ken}@cs.cornell.edu

Abstract

In recent years, a number of applications of ad-hoc networks have been proposed. Many of them are based on the availability of a robust and reliable multicast protocol. In this paper, we address the issue of reliability and propose a scalable method to improve packet delivery of multicast routing protocols and decrease the variation in the number of packets received by different nodes. The proposed protocol works in two phases. In the first phase, any suitable protocol is used to multicast a message to the group, while in the second concurrent phase, the gossip protocol tries to recover lost messages. Our proposed gossip protocol is called Anonymous Gossip(AG) since nodes need not know the other group members for gossip to be successful. This is extremely desirable for mobile nodes, that have limited resources, and where the knowledge of group membership is difficult to obtain. As a first step, anonymous gossip is implemented over MAODV without much overhead and its performance is studied. Simulations show that the packet delivery of MAODV is significantly improved and the variation in number of packets delivered is decreased.

1. Introduction

Ad-hoc networks are mobile networks that operate in the absence of any fixed infrastructure, employing peer-to-peer communication to establish network connectivity. These networks have a wide range of applications such as disaster relief and field operations, war front activities, and communication between automobiles on highways. Group communication or multicast is a natural requirement for many of these applications and the reliability of the multicast protocol could affect their performance significantly. Ad-hoc networks function under severe constraints such as mobility of nodes, insufficient power and memory on mobile devices, and bandwidth restriction of the wireless medium. These restrictions make the existing multicast routing protocols such as MAODV very unreliable even in moderately sized networks. This paper discusses our protocol, Anonymous Gossip, a scalable method for providing probabilistic guarantees to multicast reliability in mobile ad-hoc networks.

2. Multicast Reliability in Ad-Hoc Networks

Several protocols have been designed in recent years to address the issues of multicast routing in ad-hoc networks. Approaches range from simple ideas such as selective flooding[13] to more complex protocols that maintain knowledge of the network connectivity or dynamically gather route information. Many multicast protocols approach the problem of multicast in ad-hoc networks by building and maintaining multicast trees or meshes to establish connectivity among group members. MAODV[2] and AMRIS[8] are protocols that maintain multicast trees while ODMRP[10] and MCEDAR[6] are mesh-based. The mesh-based protocol, ODMRP[10] provides better packet delivery than tree-based protocols but pays an extra cost for mesh maintenance. However, these protocols do not attempt to ensure packet delivery and packet loss is a problem during mesh/tree reconfiguration, a frequent repair activity. Further, the number of packets received by different members of a group is highly variable, with some members receiving very few packets while others receive almost all the packets even though the network may not be partitioned at that time.

Multicast reliability in wired networks has received a lot of attention in the past years. Protocols such as SRM[3], RMTP[4] and PGM[5] focus on best effort reliability but are very scalable and easy to implement. These protocols, which are NACK-based (meaning that the receiver has the onus for initiating recovery) depend on the multicast routing tree constructed by the Internet group multicast protocol, IGMP. In ad-hoc networks, routes changes very rapidly and the methods used by these protocols are consequently not available to us. Bimodal Multicast[1] is another scalable system, which uses gossip to provide probabilistic reliability in wired networks. Our premise is that gossip is well matched to the needs of ad-hoc networks because it is a controlled form of flooding - messages are slowly prop-

*The authors were supported in part by DARPA/AFRL-IFGA grant F30602-99-1-0532 and in part by NSF-CISE grant 9703470, with additional support from the AFRL-IFGA Information Assurance Institute, from Microsoft Research and from the Intel Corporation.

agated through the network without congesting the wireless medium - and is independent of topology. This paper proposes a new method of gossip called anonymous gossip(AG), which does not require a group member to have any knowledge of the other group members.

A multicast protocol based on anonymous gossip would proceed in two phases. In the first phase, packets are multicast to the group using any unreliable multicast protocol. In the second phase, periodic anonymous gossip takes place in the background and ensures that most of the reachable members receive the packets. This method can be implemented on top of any of the tree-based and mesh-based protocols with little or no overhead, and without affecting the scalability of the underlying protocol. In this paper, we discuss our implementation of AG using MAODV as the underlying protocol. Simulation results show that using gossip over MAODV significantly increases the packet delivery, while the variation in the number of packets received by different nodes decreases.

3. Overview of AODV Multicast

MAODV is a reactive protocol that dynamically creates and maintains a multicast tree for each group. It is an adaptation of AODV, a unicast routing protocol. Due to constraints of space, we present in this section a brief overview of only those aspects of MAODV relevant to our implementation. A detailed description of MAODV can be found in [2][11].

Each node running MAODV maintains two routing tables: Route Table(RT) and Multicast Route Table(MRT). The Route Table is used for recording the next hop for routes to other nodes in the network. Each entry in RT contains a destination IP address, a destination sequence number, hop count to the destination, IP address of next hop, and the lifetime of this entry. The destination sequence number tracks the freshness of the route to that destination. A source node S trying to send a message to a node B, first looks for a route to B in its RT. If a valid route is not found, S broadcasts a route request message called RREQ. A node receiving this RREQ message can unicast a route reply RREP to S if it is the destination node or if it has a fresh enough route to B. Otherwise the node broadcasts the RREQ to its neighbors. The source node S selects the shortest among the freshest routes from the received RREPs and adds the entry in the Route Table. Nodes relaying the RREQs and the RREPS add the reverse and forward route entries into their Route Table respectively.

The Multicast Route Table contains entries for multicast groups of which the node is a router (i.e., a node in the multicast tree). Each entry in this table contains the multicast group IP address, the group leader IP address, the group sequence number, hop count to the group leader, the next hops, and the lifetime. The next hops are the nodes in the multicast tree to which this node is connected. Each next hop entry has an *enabled* flag to indicate a potential but not yet activated entry. The next hop that is closer to the group leader is called the upstream node.

A node S that is not a part of the multicast tree can join the multicast group by broadcasting a RREQ message with the *join* flag set. Any node in the multicast tree can respond to a Join RREQ by unicasting an RREP back to S. These RREQs and RREPs are processed similar to unicast routing. In addition, nodes receiving Join RREQs also add entries with *enabled* flag false in their MRT. The node S selects a suitable route from the RREPs and sends an activation message called MACT along this route. All nodes receiving the MACT message change the *enabled* flag to true in their entries.

Any group member, which is a leaf node in the multicast tree can leave the group by sending a MACT message to its upstream node with the *prune* flag set. A node receiving a Prune MACT deletes the sender from its next hop table. If it is a non-group member that has now become a leaf node, it leaves the group by sending a Prune MACT to its upstream node. Non-leaf nodes can leave a multicast group but must continue to function as routers in the multicast tree. When a link breakage occurs between two nodes U and D of a multicast tree, only the downstream node D attempts to repair this link. This restriction is necessary to prevent formation of loops. D sends an RREQ with an extension containing the hop count to the group leader. Any multicast tree member closer to the group leader than D can reply to this RREQ. In case D receives no replies within a certain time even after a few rebroadcast of the RREQ, the network is assumed to be partitioned and a new group leader is selected in the downstream sub-tree. The details of how the new group leader is selected and how two partitioned trees can rejoin are described in [2][11].

4. Anonymous Gossip Protocol

Gossip as a general technique has been used to solve a number of problems such as network news dissemination(NNTP), replicated data management [14] and failure detection [15]. Bimodal multicast [1] uses gossip as a technique to achieve probabilistic reliability of multicast in wired networks. This protocol achieves a bimodal guarantee i.e., all or no delivery with very high probability and partial delivery with very low probability, without sacrificing scalability and stable throughput(low jitter). We also use gossip to address the problem of reliable multicast in mobile ad-hoc networks and provide all or no delivery with very high probability and partial delivery with very low probability. However, we use a different method of gossip to provide the same guarantees.

A gossip based reliable multicast protocol involves two phases. In the first phase, any suitable unreliable protocol is used to multicast the message m, to be sent to the group. In the second phase, gossip is used to recover lost messages from other members of the group that might have received it. This phase consists of periodically repeated gossip rounds in the background as more and more messages are multicast. A single gossip round can potentially recover many lost messages. A single round of gossip consists of the following steps,

1. Node A randomly chooses another member of the group, say B.
2. A sends B the information about messages it has received or not received.
3. B checks to see if it has received any of the messages listed by A.
4. Then A and B could exchange messages which are not a part of each other's message history.

Posing such a broad algorithm in the form of a protocol has to be done with meticulous care. The constraints of the environment have to be taken into account. As we have mentioned earlier, one of our primary goals is to provide reliability without increasing the message overhead of the already congested mobile wireless environment. The following issues have to be answered keeping in mind the above constraints.

1. How does node A know who the other members of the group are?
2. How does A select which member to gossip with?
3. What should be the direction of information exchange?
4. How is the message history maintained?

The following subsections describe these design issues in detail.

4.1. Anonymous Gossip

In our opinion, the first of the above questions is the most crucial issue. Bimodal multicast requires each participating node to have partial or total knowledge of group membership. This usually involves each node sending periodic heart beat messages to other nodes to keep the membership information current. In wired networks, where the nodes are in a domain sub-domain hierarchy, group membership can be maintained with limited overhead. However, in the wireless environment maintaining even partial group membership is extremely expensive and would significantly reduce the throughput of the network. We propose a method called *anonymous gossip(AG)* to overcome this problem. AG does not require any member to know the other members of the multicast group. The node attempting to send a gossip message does not even know the identity of the node with which it will gossip until the other node sends back a gossip reply.

We add a new type of message called *gossip_message*. This message has the following five fields:

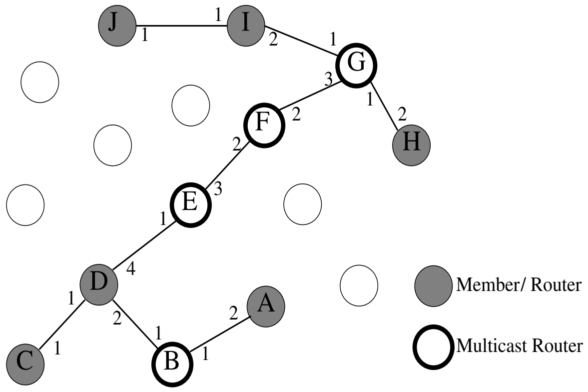

Figure 1. Local Anonymous Gossip in MAODV

- Group Address: the address of the multicast group
- Source Address: the address of the node sending the gossip message
- Lost Buffer: an array of fixed size, that carries sequence numbers of messages that the source node believes it has lost
- Number Lost: the size of the Lost Array.
- Expected Sequence Number: the sequence number of the next message that the source expects.

Each node randomly selects one of its neighbors and sends a gossip message to it. Any node receiving a gossip message randomly selects one of its neighbors (excluding the neighbor which sent the message) and propagates the message to it. If the receiving node is itself a member of the multicast group then it randomly decides to either accept the gossip or propagate it. The accepting node then unicasts a gossip reply to the initiator of this gossip request. The reply is described in section 4.4. In a general multicast protocol of an ad-hoc network, the nodes themselves participate as routers. Also, only a subset of these nodes/routers would participate in routing any messages meant for a given multicast group. In such cases, only participating routers are to be considered while propagating the gossip message. For example, in the implementation of this protocol on MAODV, only the routers in the multicast tree participate in propagating the gossip. As have seen in section 3, each router maintains a multicast route table which constitutes the nexthops for this router. While propagating the gossip message, one of these nexthops is randomly selected. Propagation along the multicast tree prevents gossip messages from reaching the same node twice.

4.2. Locality of Gossip

Periodic propagation of gossip messages generates continuous traffic on the network. Choosing nearer members to gossip with will reduce the network traffic, however gossiping with a distant node is extremely important because sometimes message loss could affect an entire locality. Hence we need a scheme that would gossip locally with a very high probability and with distant nodes occasionally. Our AG protocol is augmented to achieve this optimization. Here, we assume some familiarity with MAODV, because the constraint of brevity prevents us from explaining MAODV in detail, and yet our algorithm builds upon it. We require each node participating as a router in the multicast tree to maintain one additional field called *nearest_member*. In the implementation over MAODV, the multicast route table is augmented to have this extra field associated with each nexthop entry. The *nearest_member* field associated with a nexthop contains the distance to the nearest group member from this node by taking the link through this nexthop node. This adds very little overhead to the existing multicast route table. Whenever a gossip message is received, a nexthop node is chosen so that a nexthop with a smaller *nearest_member* value is chosen with higher probability than a nexthop with a larger *nearest_member* value. Thus with higher probability the gossip message is propagated to a closer node than to a distant node. Figure 1 illustrates an AODV Multicast Tree, where $\{A, C, D, H, I, J\}$ are the group members and $\{B, E, F, G\}$ are the other participating routers. The numbers shown on their edges give the values of the *nearest_member* field. As an example, for the router E, the nearest group member through D is at a distance 1 and through F is at a distance 3.

Maintaining the validity of the *nearest_member* field in each router can be done with limited overhead. This value needs to be modified when one of the following events occurs - a new member joins the group, an existing member leaves the group, or the mobility of nodes creates a topology change. Whenever a new member joins the group or an existing member leaves the group, the routers nearest to this node realize the event first. For example in MAODV a member sends a MACT message to establish itself while joining (see section 3 for details). A member leaving the group sends a prune message to its nearest routers. The nearest router adds this new nexthop to its multicast route table with value of *nearest_member* field set to one or deletes this entry from its list of nexthops. Then for each of the nexthops present in its table, it recomputes the value of *nearest_member* and sends this modified value to that nexthop. For example, if $\{B, C, E\}$ are the next hop entries for node D, and $\{b, c, e\}$ are the associated *nearest_member* values for these nexthop entries, then D sends $1 + \min(c, e)$ to B, $1 + \min(b, e)$ to C and $1 + \min(b, c)$ to E. This value needs to be sent only if it is different from its previous value. Also, this value can be piggy backed on any other messages bound for the same router. When a router receives this new value, it updates this entry in its table. It then sends a modify message to its nexthops in the same way as described above. Since we are propagating the minimum of a set of values, any change in topology or group membership will only affect the routers in its locality. These modify messages will not propagate far and hence will not increase congestion significantly.

4.3. Cached Gossip

AG is done only using the routers participating in the multicast tree. If these routers are not well distributed in the topology of the network, there is a possibility that a gossip message reaches a node along a route longer than the shortest existing route between these two nodes. For example, when using AG over MAODV, the gossip message propagates only along the multicast tree, although other, potentially shorter, routes may exist between the gossiping members (note that the return path is unaffected by this phenomenon, because gossip replies are unicast). It is efficient to use the unicast routes to gossip with those nodes whose membership is already known. Further, these unicast routes may be available for gossip even when the multicast tree is being repaired.

We incorporate this by introducing a *member_cache* in all the member nodes of each multicast group. The *member_cache* is a bounded buffer containing entries which are 3-tuples $(node\_addr, numhops, last\_gossip)$. The *node_addr* field contains the address of a group member, the *numhops* field contains the shortest distance between the nodes and the *last_gossip* contains the time at which last gossip occured between these two nodes. This information itself is collected at no extra cost. The *member_cache* table is updated each time a message is received from a group member. This message could be a data packet meant for this group, a gossip reply, or any other maintenance packet. For example, in MAODV, each route request(RREQ) for a group join generates replies from a number of nodes, many of which are members of the group(see [11][2] for details). Whenever such a message is received, we add the member information to the *member_cache*. If this table is full and a new member has to be added, a member with a greater numhops is deleted from this table. If there are no members with greater numhops then the member with most recent *last_gossip* is replaced with the new entry. This is done to avoid frequent gossips with the same members.

In each gossip round, the node chooses to do anonymous gossip with probability p_{anon}. If AG is chosen, then a gossip message is constructed with the address of this node as the initiator and this message is propagated as described in section 4.1 and 4.2. If cached gossip is chosen, then a member is selected randomly from the *member_cache* and a gossip message is unicast to this member. When a gossip reply is received from a member, the member information is updated in the *member_cache*. If this member already exists in this table, then the numhops and last_gossip fields are updated. Otherwise, the member is added to the table as described above.

4.4. Push vs Pull

The importance of direction of information exchange is explained in [14]. Our protocol implements a pull mode of message exchange that may be described as follows. Each node maintains a table called *lost_table*, for every multicast group that it belongs to. This contains the sequence numbers of all the messages this node believes itself to be lacking. An entry in this table will be made whenever a message is received with a sequence number greater than the expected sequence number. Note that the sequence number is a 2 tuple including the sender address and a sequence number, because different senders send messages to the multicast group. Each node also maintains a bounded FIFO buffer, called *history_table* containing the most recent messages received. The most recent entries of the *lost_table* are placed in a *lost_buffer*. Whenever a node prepares a gossip message, the *lost_buffer* and a list of expected sequence numbers is added to the gossip message. When a node receives a gossip message, it compares the *lost_buffer* and the expected sequence number list to see if its *history_table* has a copy of any message sought by the gossip initiator. It then unicasts any message found back to the gossip initiator as the gossip reply.

5. Simulation and Performance

We simulated our protocol on GloMoSim[17], which uses a parallel, event-driven simulation language called PARSEC[18]. MAODV was implemented as described in version 5 of the IETF draft[11] and was adapted to implement our gossip protocol. We understand that more recent drafts of MAODV are available, but we believe that the extensions would not impact performance for the scenarios investigated.

5.1. Simulation Environment

A fixed area of 200m$\times$200m was used in the simulation, and all the nodes were initially placed randomly within this area. The simulations were performed using the Random-Waypoint scheme to model the mobility of the nodes in the network. In this model, each node has a predefined minimum and maximum speed. It then travels towards a random spot at a speed chosen randomly from this interval. After reaching the destination, it rests for a period chosen from a uniform distribution between 0 and 80 seconds, before continuing this mobility pattern. Each simulation was set to run for 10 minutes. The MAC layer protocol used was IEEE 802.11 and the bandwidth of the wireless medium was assumed to be 2Mbps.

The network consisted of a single multicast group with one-third of the nodes being group members. All the nodes joined the group at the beginning of the simulation and remained in it throughout the run period. One of the members was selected as the source of data packets. The source node generated packets of length 64 bytes at regular intervals of 200ms with the destination as the multicast group. In the 600second simulation time, the source node continued to send packets starting at 120 seconds and ending at 560 seconds. The initial delay of 120sec was allowed for MAODV to build the initial multicast tree. The source node generated 2201 data packets during the period of interest.

Every group member sends one gossip message per second. Each gossip message could request for at most 10 lost messages. The size of the membership cache was also set to 10. Each member could keep track of at most 200 lost messages, in the *lost_table*, and remember the most recent 100 messages received in the *history_table*. For MAODV, the hello interval was set to 600ms, the allowed hello loss to 4, and the group hello interval to 5sec.

We studied the performance of our protocol while varying three parameters: the transmission range, the maximum speed and the number of nodes. The transmission range is the distance from a node within which another node can receive packets from it. All the nodes are assumed to have the same transmission range. The minimum speed was set to 0 for all the runs. We varied each parameter over a wide range, for different fixed values of the other two parameters. For each set of parameters, we measured the number of packets received by each node in the multicast group. Each simulation was carried out 10 times with different values for the random-seed.

The performance graphs measure the number of packets received by the group members during each run. Each data point in the graph corresponds to the average of the number of packets received by each group member. The range of measured data values obtained for the full set of receivers is shown in the form of error bars, while a line connects the average across receivers. Notice that reception rates can vary widely for different receivers because of differences in network connectivity. Perfect reception rates would be graphed as 2201: the number of packets multicast by the source.

5.2. Packet Delivery vs. Transmission Range

The transmission range was varied from 45m to 85m in steps of 5m, keeping the total number of nodes constant at 40. This experiment was carried out for 6 values of the maximum speed. Fig. 2 and Fig. 3 show the variation of packet delivery for maximum speeds 0.2m/sec and 2m/sec respectively for both MAODV and our gossip protocol.

It is interesting to note that the gossip protocol consistently performs better than the underlying multicast protocol. Not only is the average packet delivery higher but the variation in the number of packets received by different group members is also significantly lower. As the transmission range increases, the connectivity of the network improves leading to fewer link failures. Further, the length

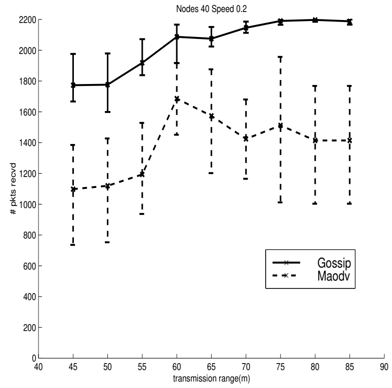

Figure 2. Packet Delivery vs Transmission Range when the max speed = 0.2m/sec

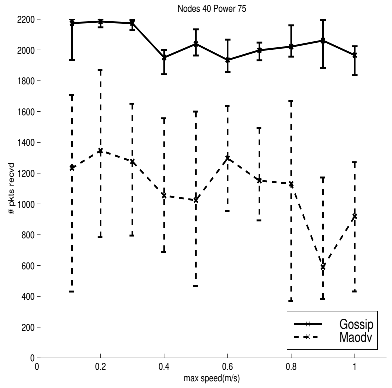

Figure 4. Packet Delivery vs Maximum Speed when the transmission range is 75m

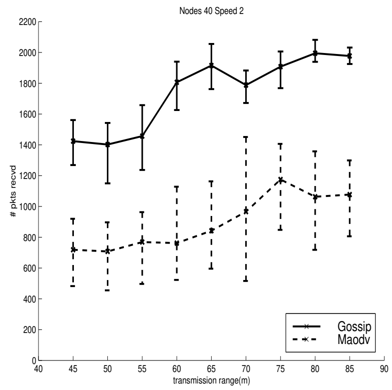

Figure 3. Packet Delivery vs Transmission Range when the max speed = 2m/sec

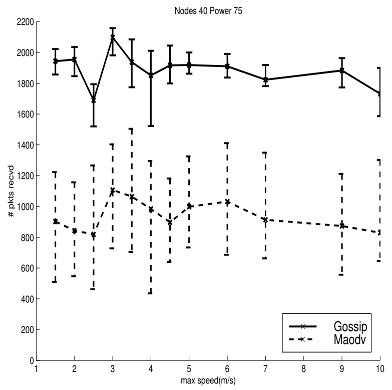

Figure 5. Packet Delivery vs Maximum Speed when the transmission range is 75m

of the network routes becomes shorter, decreasing the route discovery and setup time. The effect of this is reflected by the steady improvement in packet delivery of both gossip and MAODV with increase in the transmission range. However, this runs a risk of increasing the congestion in the network, which could affect the performance of the protocols.

5.3. Packet Delivery vs. Maximum Speed

The maximum speed of the nodes was varied in two phases, from 0.1m/sec to 1 m/sec in steps of 0.1m/sec and from 1m/sec to 10m/sec in steps of 1m/sec. The transmission range was set constant at 75m and the number of nodes was fixed at 40. Fig. 4 and Fig. 5 show variation of packet delivery with speed for both the phases. These graphs continue to illustrate that our protocol achieves better packet delivery with a decreased variation in the number of packets received by the group members. At very low values of maximum speed, upto 0.3m/sec, our protocol gives near 100% packet delivery. At higher speeds, the data delivery varies from 90% to 80%. We can see a gradual decrease in data delivery as the maximum speed increases in this range. Increase in the maximum speed causes the nodes to move faster and more frequently in the random waypoint model. This causes an increase in the frequency of link breakages, affecting the overall performance of the system.

5.4. Packet Delivery vs. Number of Nodes

The number of nodes was varied from 40 to 100. In one experiment, the transmission range was adjusted in such a way that the average number of neighbors of a node remained approximately the same. As the number of nodes in the network increases, the routing distance between the nodes goes up, and hence the frequency of link failures in the network also increases. Fig. 6 shows the variation of packet delivery with the increase in the number of nodes. As the number of nodes increases, the packet delivery rate tends to decrease gradually because of the above mentioned reasons. In another experiment, the transmission range was

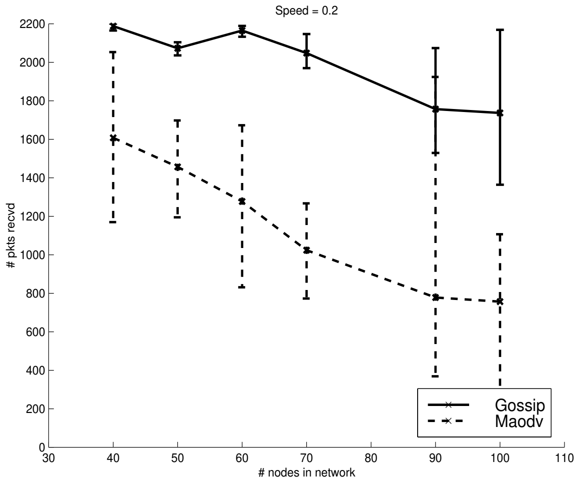

Figure 6. Packet Delivery vs Number of Nodes keeping the average number of neighbors for a node approximately constant

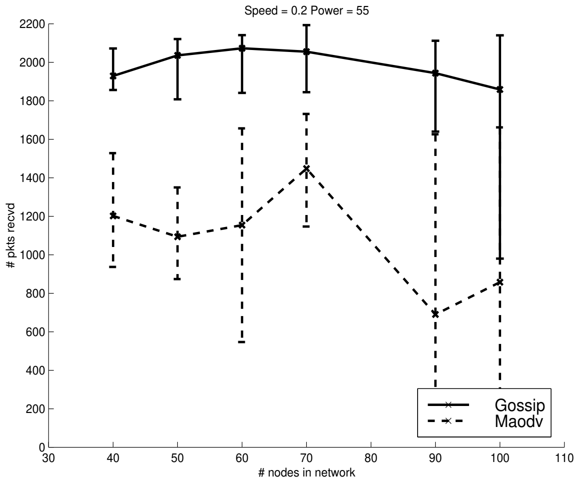

Figure 7. Packet Delivery vs Number of Nodes when the transmission range is constant at 55m and maximum speed is 0.2m/sec

kept constant at 55m and the number of nodes was varied from 40 to 100. Fig. 7 shows the results of this experiment. As the number of nodes is increased, the connectivity of the network increases improving packet delivery. However, the traffic in the network also increases and beyond a point this leads to congestion decreasing the number of packets delivered.

5.5. Performance Analysis

The simulation results show that AG can improve the reliability of multicast routing protocols without the use of acknowledgements, and without adding significant overhead to them. The overhead analysis of gossip is presented in Fig. 8. Goodput is defined as the percentage of non-duplicate messages received through gossip replies to the total number of messages received through gossip replies. In other words goodput gives a measure of the redundant traffic - more the goodput, more the number of useful messages carried by gossip replies and lesser is the redundancy. In our simulations goodput is measured at different group members for two values each of transmission range and maximum speed. Fig. 8 shows that the goodput is close to 100 percent for most of the cases under which the experiments were run. It implies that nearly all the gossip replies were useful and very few duplicates were received. This can be attributed to the parameters that were chosen for gossip. For a different messaging rate these parameters would have to be adjusted to give a comparable goodput. In general the gossip rate should be tuned so that the network does not get congested and the goodput is nearly 100 percent.

AG works even when the multicast tree is broken, recovering packets dropped during the maintenance of the tree. The effectiveness of anonymous gossip depends on the values chosen for the size of the *history_table* and the *lost_table*, besides the gossip interval. We are currently doing more experiments to study the performance of AG under different parameters. Implementing anonymous gossip with other multicast protocols, such as ODMRP[10] and AMRIS[8], could also be done in a similar manner without adding extra complexity. We believe that AG would be able to improve the reliability of these protocols as well.

6. Related Work

Despite the importance of reliable multicast in mobile ad-hoc applications, not much work has been done on it. In [9], Pagani et al propose a protocol that guarantees validity, agreement, integrity and termination for multicast in such environments. The protocol is characterized in two phases: the *scattering phase*, in which a message is diffused from the source to all its destinations, and the *gathering phase*, in which the source collects all the acknowledgements messages from the destinations. Most of the communication is done on a forwarding tree, that is an on demand source-centered routing tree, with the lifetime of at most the time needed to scatter and gather a single message. In case, the tree links break, no effort is made to repair it, and flooding is used for further communication. For reliability, a message is buffered at each of the clusterheads along the tree, as long as its ack from the downstream nodes is not received. Using this mechanism, the protocol provides strong guarantees for message delivery as long as the conditions of eventual subsidence, liveness and clusterhead stability is satisfied. While this protocol provides these guarantees in a slow moving network where the failures are rare, the per-

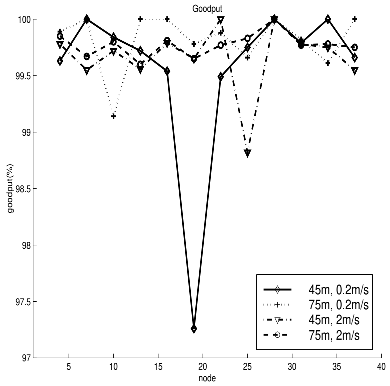

Figure 8. Goodput at different group members for 2 values each of transmission range and maximum speed

formance decreases dramatically with the increasing size of the network. There is also degradation in performance with increase in mobility. The reasons for the poor performance can be attributed to two main aspects of the protocol. Firstly, since buffer size, in practice, is bounded, a situation could arise when old messages are still stored in most of the node buffers. This gives rise to a situation where newer messages cannot be accommodated since older messages have to be stored to provide the strong guarantees. The second reason for decrease in performance is the use of ack messages. This proves to be very expensive in wireless networks where the physical layer is bandwidth constrained.

Another approach to achieve reliability in ad-hoc networks is proposed in [13]. This protocol guarantees best effort delivery using an adaptive flooding scheme called hyper-flooding. The nodes do not keep any state regarding the multicast routes. Whenever a sender has to multicast a message, it broadcasts the message to all its neighbors who in turn re-broadcast the packet. Some optimizations are made to prevent packets from looping forever in the network. The advantage of this protocol is the reliability it provides for high-speed networks. On the other hand this protocol is extremely expensive since it generates a large number of messages, and may easily congest the network. Especially when the multicast group is sparsely distributed in a dense network or when all the nodes are not moving at high speeds, the protocol decreases the throughput considerably.

7. Concluding Remarks and Future Work

Many of the exciting applications for wireless ad-hoc networks will require reliable multicast although the degree of reliability demanded by them may vary. Few applications might demand special properties from the protocol, many may be satisfied with best effort delivery and others could be satisfied with an unreliable multicast as long as most of the messages reach the destination. While [9] describes a protocol for the first case, we do not address that problem here. At the same time we would like to emphasize that stronger reliability guarantees can be obtained by developing protocols that function over the underlying multicast and our gossip protocol. Work is being done on the Spinglass project[16] in Cornell University, where protocols for leader election, consensus and other advanced properties are being developed over bimodal multicast. As a future work, we would like to develop efficient algorithms to do the same over probabilistic multicast protocols in mobile ad-hoc networks.

In this work, we have not tried to improve upon the efficiency and scalability of MAODV itself. Our protocol could be used with any existing multicast protocol, without significantly changing the existing protocol. AG inherits limitations from the underlying protocol; for example, in the work described here, AG would not work if the links were unidirectional. On the other hand AG introduces little overhead beyond that associated with the underlying mechanisms. In the future, with much better and efficient multicast protocols coming up, we would like to improve upon their reliability by using the gossip protocol. Our work with MAODV is the first effort in improving reliability of the existing multicast protocols. It is possible to get improvements by using anonymous gossip on the other protocols proposed in [6], [10], [8] and [7].

Acknowledgements

We are extremely grateful to Indranil Gupta and Robert Van Renesse for the useful discussions and Chaitanya Swamy and Sheela Tiwary for reviewing the paper.

References

[1] Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu and Yaron Minsky. "Bimodal Multicast". *ACM Transactions on Computer Systems*, Vol. 17, Issue 2, May 1999, 41-88.

[2] Elizabeth M. Royer and Charles E. Perkins. "Multicast Operation of the Ad-hoc On-Demand Distance Vector Routing Protocol". *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and Networking(MOBICOM).* Aug. 1999, 207-218.

[3] Sally Floyd, Van Jacobson, Steve McCanne, Ching-Gung Liu and Lixia Zhang. "A Reliable Multicast Framework for Light Weight Sessions and Application Level Framing". In *Proceedings of the '95 Symposium on Communication Architectures and Protocols (SIGCOMM).* ACM. August 1995, Cambridge MA.

[4] Sanjoy Paul, K. K. Sabnani, J. C. Lin, S. Bhattacharya. "Reliable Multicast Transport Protocol(RMTP)". *IEEE*

Journal on Selected Areas in Communications, special issue on Network Support for Multipoint Communication, April 97, Vol 15, No. 3.

[5] T. Speakman and D. Farinacci and S. Lin and A. Tweedly. "Pretty Good Multicast". *Internet draft, ftp://ds.internic.net/internet-drafts/draft-speakman-pgm-spec-00.txt*

[6] Prasun Sinha, Raghupathy Sivakumar, Vaduvur Bhargavan. "MCEDAR: Multicast Core-Extraction Distributed Ad-Hoc Routing". *Proceedings of IEEE Wireless Communications and Networking Conference*, September 1999.

[7] L. Ji and M. S. Corson. "A Lightweight Adaptive Multicast Algorithm". *Proceedings of IEEE GLOCECOM*, Sydney, Australia, December 1998, 1036-1042.

[8] Chun-Wei Wu, Y.C. Tay. "AMRIS: A Multicast Protocol for Ad-Hoc Wireless Networks". *Proceedings of MILCOMM '99*, Nov. 1999.

[9] Elena Pagani and Gian PaoloRossi. "Reliable broadcast in mobile multihop packet networks". *Proceedings of the third annual ACM/IEEE international conference on Mobile computing and networking(MOBICOM)*, September 26 - 30, 1997, 34-42.

[10] S.-J. Lee, M. Gerla, and C.-C. Chiang. "On-Demand Multicast Routing Protocol(ODMRP)" *Proceedings of IEEE WCNC'99*, Sep. 1999.

[11] C. Perkins, E. Royer, S. Das "Ad-Hoc On Demand Distance Vector(AODV) Routing". *IETF Internet Draft, draft-ietf-manet-aodv-05.txt*, March 14, 2000.

[12] Elizabeth M. Royer and Charles E. Perkins. "Ad-hoc On-Demand Distance Vector Routing". *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, Feb. 1999, 90-100.

[13] Christopher Ho, Katia Obraczka, Gene Tsudik and Kumar Viswanath. "Flooding for Reliable Multicast in Multi-Hop Ad-Hoc Networks". *MobiCom Workshop on Discrete Algorithms and Methods for Mobility*, Aug 1999.

[14] Alan Demers, Dan Greene, Carl Hauser, Wes Irish and John Larson. "Epidemic algorithms for replicated database maintenance". *Proceedings of the Sixth Annual ACM Symposium on Principles of distributed computing*, 1987, 1-12.

[15] Robbert van Renesse, Yaron Minsky, and Mark Hayden. "A Gossip-Based Failure Detection Service". *Proc. of Middleware*, 1998.

[16] Cornell University. "Spinglass: Adaptive Probabilistic Tools for Advanced Networks."

[17] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks". *Proceedings of the 12th workshop on Parallel and distributed simulation*, May 26 - 29, 1998, Banff Canada, 154-161.

[18] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, and H. Song. "Parsec: A Parallel Simulation Environment for Complex Systems". *IEEE Computer, Vol. 31(10)*, October 1998, pp. 77-85