Gradient Routing in Ad Hoc Networks

Robert D. Poor
Media Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
r@mit.edu

***Abstract* -- A wireless *ad hoc* communication network is a collection of wireless links that cooperate to form a complete communication system without the need for centralized control or preexisting infrastructure. This paper presents *Gradient Routing* (GRAd), a novel approach to routing and control in wireless ad hoc networks. A GRAd network attains scalability through a *multi-hop* architecture: nodes that are not within range of one another can communicate by relaying messages through intermediate neighbors. Routing information is established on-demand and is updated opportunistically as messages are passed among nodes.**

Unlike other ad hoc routing techniques, a node in a GRAd network does not single out a particular neighboring node to relay its message. Instead, it advertises its "cost" for delivering a message to a destination, and only those neighboring nodes that can deliver the message at a lower cost will participate in relaying the message. In this way, a message descends a loop-free "gradient" from originator to destination.

Since multiple neighbors can participate in the relaying of messages, GRAd maintains good connectivity in the face of frequently changing network topologies. A node does not need to know the identities of its neighbors and establishes routes on demand, making periodic "hello" beacons unnecessary and increasing the overall security of the network. Because GRAd does not use link to link handshakes, end-to-end latencies remain small.

I. Introduction

Wireless ad hoc networks—characterized by decentralized, self-organizing, multi-hop communications—have recently become the subject of increasing study. The growing interest can be attributed to the confluence of several independent factors: the explosion of portable computing and communication devices; the availability of inexpensive radio systems operating in unlicensed bands; and government funding for research of easily deployed, mobile networks [9][12]. The promise of ad hoc networks is pervasive network connections which are cheap to manufacture and trivial to deploy.

In any wireless ad hoc network, a major challenge lies in the design of routing and network control. Lacking any centralized point of control, nodes in an ad hoc network must cooperatively manage routing and medium access functions. Nodes may be mobile, creating continual changes in the network topology. Also, wireless links are not as robust as their wired counterparts; high bit error rates and packet losses are commonplace.

In the last decade, a number of ad hoc network protocols have been proposed. As an indicator of the amount of activity in this field, the Internet Engineering Task Force (IETF) recently formed the Mobile Ad Hoc Networking (MANET) working group to develop ad hoc protocol specifications and introduce them into the Internet Standards track [5]. At this time, there are eight separate ad hoc routing protocols under consideration by the working group.

This paper introduces GRAd, a new approach to routing and control for ad hoc networks. GRAd falls under the category of *on-demand* routing protocols, in which routes are established only when nodes wish to communicate with one another; no attempt is made to maintain state when there is no data to send.

In other on-demand routing protocols such as the *Ad Hoc On-Demand Distance Vector Routing* protocol (AODV) [10] and the *Dynamic Source Routing* protocol (DSR) [7], a node relays a message by sending to a particular neighboring node. The popular 802.11 MAC layer protocol uses "virtual carrier sensing " as part of its collision avoidance mechanism for such unicast transmissions [4], requiring a *request to send / clear to send* handshake (RTS/CTS) between each pair of wireless links. This exchange contributes to significant delays in the relaying of messages, resulting in long latencies.

By comparison, a node in a GRAd network makes no attempt to identify *which* of its neighbors is to relay a packet. Instead, it includes its "cost to destination" information in the packet and broadcasts it. Of all the nodes that receive the broadcast, only those that can deliver the packet at a lower cost will relay the message. In this way,

This research has been supported in part by the Things That Think Consortium at the MIT Media Laboratory and by the Motorola Fellows Program.

the packet descends a loop-free "gradient" towards the ultimate destination.

Since each transmission is a local broadcast, GRAd does not (and in fact, cannot) use the RTS/CTS handshake associated with unicast transmissions. Consequently, GRAd exhibits very low latencies.

GRAd collects cost information opportunistically: each message carries with it the cost since origination, which is recorded at each node that overhears the transmission, and is incremented when the message is relayed. Thus, the simple act of passing a message quickly and efficiently updates the cost estimates in nearby nodes.

GRAd demonstrates very good immunity to rapidly changing topologies. Since each message reaches a number of neighboring nodes, a single link failure will not cause a break in the communication path as long as another neighbor is available to relay the message.

The essence of GRAd is embodied in its routing and control algorithms—these are detailed in Section II of this document. Section III describes the testing and simulation that demonstrates the viability and robustness of GRAd. Section IV suggests future directions for research into GRAd, and Section V summarizes the work.

II. The GRAd Algorithm

A. Assumptions

GRAd is designed for use in multi-hop wireless networks, and makes relatively few assumptions about the underlying physical medium. It does assume that links are *symmetrical*: if Node A can receive messages from Node B, then Node B can receive messages from Node A. In a practical wireless network, strict symmetry is impossible to guarantee due to the mobility of the nodes and time-varying environmental noise. As will be shown in Section III, GRAd continues to work well in cases where only partial symmetry holds.

GRAd assumes a local broadcast model of connectivity. When a node transmits a message, all neighboring nodes within range simultaneously receive the message.

GRAd provides best effort delivery of messages with the understanding that higher-level protocols will handle retransmission and reordering of packets as needed.

The propagation of a message through the network establishes and updates *reverse path* routing information to the originator of the message. Consequently, GRAd is most efficient when the network traffic has a "call and response" pattern, such as streamed packet data with periodic acknowledgments.

B. GRAd message format

Messages passed among nodes in a GRAd network carry the fields shown in Fig. 1.

TABLE 1 GRAd Message Format

msg type	originator id	seq #	target id	accrued cost	remaining value

- `msg_type`: Takes on one of two values, `M_REQUEST` for a reply request message and `M_DATA` for all others.
- `originator_id`: The id of the node originating this message. This id may be statically assigned, or as suggested in Section 4.?, may be dynamically generated on a per-session basis.
- `seq_#`: A sequence number associated with the originator id, and incremented each time the originator issues a new message. The combination of `[originator_id, sequence_#]` uniquely identifies a message, so a receiving node can distinguish a new message from a copy of a message already received.
- `target_id`: The id of the ultimate target for this message.
- `accrued_cost`: Upon origination, the `accrued_cost` of a message is set to 0.0. When the message is relayed, the relaying node increments this field by one. Thus, `accrued_cost` represents the estimated number of hops required to return a message to `originator_id`.
- `remaining_value`: Upon origination, this field is initialized to the estimated number of hops to `target_id`. Whenever the message is relayed, this field is decremented by one. The `remaining_value` field represents the "time to live" of the message: if it ever reaches zero, the message is dropped.

C. Cost Table

Each node maintains a *cost table*, analogous to the routing table of other algorithms[1]. The cost table plays two important roles in GRAd. First, the cost table can answer the question *"Is this message a copy of a previously received message?"* This is determined by comparing the `seq_#` in the message from a particular originating node against the last `seq_#` recorded in the cost table for that originator. Second, it can answer the question *"What is the estimated cost of sending a message to target node X?"* This cost estimate is formed by recording the `accrued_cost` fields for each `origninator_id` in received messages.

[1] The term "cost table" is chosen over "routing table" to emphasize the fact that GRAd does not prescribe a specific route to a target node, but rather it maintains an estimated cost to the target.

1) Cost Table Format

Each entry in the table holds state information about a remote node, as shown in Fig. 1.

TABLE 2 Cost Table Entry

target_id	seq_#	est_cost	expiration

- `target_id`: The id of a remote node to which this cost entry refers.
- `seq_#`: The highest sequence # received so far in a message from `target_id`. When compared against the `seq_#` of a newly arrived message, this field discriminates between a new message and a copy of a previously received message.
- `est_cost`: The most recent and best estimated cost (number of hops) for delivering a message to `target_id`.
- `expiration`: When a cost entry is updated, this field is set to the current time plus `cost_entry_timeout`. If the current time ever exceeds `expiration`, the cost entry is purged from the table.

2) Cost Table Maintenance

When a message is received at a node, the `originator_id` of the message is compared against the `target_id` of each entry in the cost table.

If no matching entry is found, a new cost entry is created, for which `target_id` is copied from the message's `originator_id`, `seq_#` is copied from the `seq_#` field, and `est_cost` is copied from the `accrued_cost` field. The message is marked as "fresh."

If a `target_id` is found that matches the `originator_id` of the incoming message, and if `seq_#` in that entry is lower than the `seq_#` of the incoming message, the message is marked fresh and the cost entry fields are updated from the corresponding fields in the message.

Otherwise, the message is marked as "stale"—it is a copy of a message previously received. However, if the messages offers a lower cost estimate in its `accrued_cost` field than the recorded cost in the `est_cost` field, the lower cost is recorded. This has the effect that if a copy of a previously received message subsequently arrives by means of a shorter path, the shorter path is recorded.

D. Message Origination and Relaying

When a node wishes to send a message to a destination for which the cost to the target is known, it transmits a message with the `msg_type` field set to `M_DATA`, specifying the destination in the `target_id` field and the cost to that destination in the `remaining_value` field.

Of the neighboring nodes that receive the message, only those that can relay the message at a lower cost, as indicated by their cost tables, will do so. Before a neighboring node relays a message, it debits the `remaining_value` field by one. As this process repeats, the message "rolls downhill," following an ever decreasing gradient from the originator to the target.

At the same time, the message carries the originator of the message in the `origination_id` field and the accumulated relay cost since origination in the `accrued_cost` field. Upon origination, the `accrued_cost` is set to 0. Each node that receives the message increments the `accrued_cost` field of the message and then updates its cost table entry for the originating node based on this information. If and when the message is relayed, it is resent using the incremented `accrued_cost`. By this process, any node that receives a message can update its cost estimate for returning a message to the originating node, whether or not the node is actively involved in relaying the message.

E. Reply Request

When a node wishes to send a message to another node for which there is no entry in the cost table, it initiates a "reply request" process. To do so, the originating node transmits a message whose `msg_type` field is set to `M_REQUEST`, specifying the destination in the `target_id` field and initializing the `remaining_value` field to `default_request_cost`.

Relaying of the message proceeds much in the same manner as for a `M_DATA` message, but with one important exception: any node that receives an `M_REQUEST` message will *always* relay the message the first copy of the message it receives, unless the `remaining_value` field has reached zero. As with an `M_DATA` message, the node will increment the `accrued_cost` and decrement the `remaining_value` fields of the message before relaying the message.

If a node receives a copy of a previously received message, it will update its cost table entry for the originator of the message if the copy represents a lower cost to the originator, but the node will not relay the copy.

Two important things happen in the reply request process. First, if the destination node is present anywhere in the network (within a radius of `default_request_cost`), it will receive the `M_REQUEST` message and initiate a reply. Second, each node that receives the `M_REQUEST` message establishes a cost estimate for returning a message to the originator. Consequently, when the destination node responds to the originating node's request, it can use the more efficient `M_DATA` message to deliver the reply.

F. Call and Response

GRAd is uses *on demand* routing: none of the nodes have any *a priori* knowledge of one another. Until a node turns on its transmitter, its presence in the network is not known to other nodes. The general rule for such networks is "if you wish to be spoken to, you must first speak."

The following two figures illustrate the Reply Request process in an ad hoc network, in which Node A initiates a request to Node B, and Node B subsequently responds. It is assumed that initially none of the nodes in the network have any knowledge about nodes A or B.

Fig. 1 shows the state of the network after the propagation of a Reply Request message from Node A to Node B. The dashed circle around Node A shows the range of an individual transmitter. Node A starts by transmitting a Reply Request message with an accrued cost of 0 and a target_id set to the ID of Node B. The two neighbors to A each increment the accrued_cost field of the message, record the fact that they are each one hop away from A, and relay the message. This process continues until all the nodes in the network have received and relayed the Reply Request message.

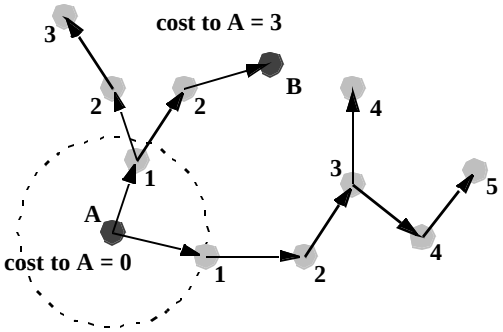


FIGURE 1: REPLY REQUEST FROM NODE A TO NODE B

By the time A's Reply Request message has arrived at node B, all of the intervening nodes in the network have established a cost estimate for returning a message to A, as shown by the numbers next to each node in Fig. 1.

When Node B receives the Reply Request message from Node A, it responds by originating an "ordinary" message with `msg_type` set to `M_DATA`, accrued_cost set to 0, and `remaining_cost` set to 3, the known cost required to reply to Node A.

Referring to Fig. 2, when B transmits this message, the neighboring nodes C and D lie within range and receive the transmission. The cost table of C indicates that A is two hops away, and since the message has an advertised `remaining_cost` of three, node C should relay the message after decrementing its `remaining_cost`. Node D, on the other hand, is four hops away from node A, and since it is unable to relay the message at a cost lower than the `remaining_cost` advertised by the message, it drops the message.

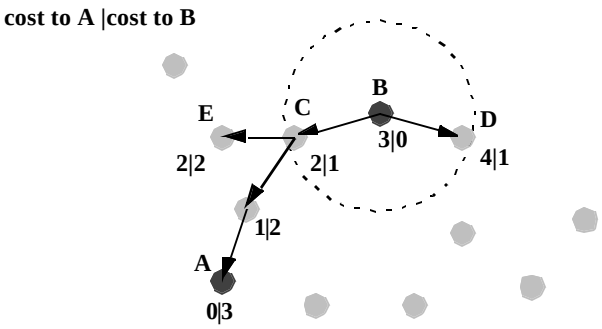


FIGURE 2: NODE B REPLIES USING THE REVERSE PATH

As the message is relayed towards A, intermediate nodes also create entries for returning a message to node B. Fig. 2 shows the state of the cost tables after the reply has been received at Node A. Next to each node that participated in the reply, the estimated cost for sending a message to node A is shown to the left of the vertical bar, the cost to node B is shown to the right.

In this example, nodes D and E received the message from B, but did not actively participate in relaying it. Nonetheless, by virtue of "overhearing" the message, these nodes have established an estimated cost for sending a message to Node B should the need ever arise.

G. Route Repair

If any nodes in the network are mobile, the topology of the network can change dynamically, rendering the individual nodes' cost estimates inaccurate.

If the path between originator and target becomes shorter, GRAd will automatically compensate for the change by "skipping over" one or more intervening nodes, and the revised cost estimates will be reflected in the participating nodes' cost tables after a single call and response pair of messages.

In the event that the path become longer, or intervening nodes change their positions, there is the possibility that the originator's cost estimate no longer has sufficient "potential" to reach the target destination.

GRAd uses end-to-end acknowledgments. If an acknowledgment is not received within a fixed amount of time, the originator can re-send the message, but this time using a higher estimated cost to the destination in the `remaining_cost` field of the message. This has the effect that more intermediate nodes will participate in relaying

the message towards its destination. As before, by the time the message reaches its destination, all of the intermediate nodes will have fresh cost estimates for returning a message to the originator, so the destination node's acknowledgement will be able to follow an updated gradient back to the originator.

If the first attempt to send a message with an increased estimated cost fails to reach the destination, the originator can repeat the process, incrementing the initial estimated cost each time.

If, after several attempts, the message fails to reach the destination, the originator can issue a new Reply Request message to create fresh cost estimates from scratch.

H. Implicit Acknowledgment

To reduce the number of redundant messages transmitted, GRAd uses a variant of passive acknowledgment called *implicit acknowledgment*. A message to be relayed is stored in a MAC-level buffer while it awaits transmission. If a node overhears a neighbor relay a copy of that same message, but at a lower `remaining_cost`, then the node can assume that the neighbor has succeeded in delivering the message closer to the destination than this node can, therefore it can delete the message from the MAC queue and cancel its transmission.

When the ultimate target node receives a message, it re-transmits the message with a `remaining_cost` set to zero. This has the effect of notifying any neighbors still waiting to relay the message that the target has received the message and that they may abandon their efforts.

III. Simulation and Results

Performance of GRAd was simulated using *Jasper*[11], an event driven network simulator. The main objectives of the simulation were to characterize the performance of GRAd as a function of transmitted packets and the amount of mobility among the nodes in the network.

A. Simulation Environment

Jasper provides detailed models for components of a multi-hop, mobile, wireless ad hoc network.

The radio modelled by Jasper emulates an FM or spread-spectrum radio, such as would be used in a wireless local area network, operating in an urban or dense office environment. In the absence of other transmissions, a transmitter/receiver pair has a nominal range of 250 meters. Transmit power falls off as the cube of the distance, and a receiver can acquire lock on a transmitter if the signal to interference ratio exceeds 10db. Once locked, a receiver can hold lock as long as the signal to interference ratio exceeds 6db. During reception of a packet, if the signal to interference ratio drops below 6db, the packet is marked as corrupted. The bit rate of the transmitter is 2Mb/sec.

GRAd's MAC layer uses a technique of carrier sense with exponential back off: when a node wishes to transmit a packet, it first waits for a random interval between T_b and $2T_b$ seconds. At the end of that time, if the carrier sense detects that the local airwaves are in use, it doubles the value of T_b (up to an upper limit) and waits again. If the airwaves are free, it halves the value of T_b (down to a lower limit) and transmits the packet. If the MAC transmit buffer becomes empty, T_b is reset to its minimum value.

Mobility and traffic models were chosen to emulate those described in [1]. Fifty nodes in a 1500m×300m arena travel according to the *random waypoint* algorithm: each node travels towards randomly chosen locations within the arena at random speeds (evenly distributed between 0 and 20m/sec.). After reaching its destination, the node pauses for a fixed amount of time before setting out for its next randomly chosen location. The *pause time* is varied from 0 seconds for continuous motion to 900 seconds in which case the nodes are stationary for the duration of the simulation.

Traffic is generated by *constant bit rate* (CBR) sources, randomly chosen among the 50 nodes. Each CBR source targets a randomly chosen destination among the remaining 49 nodes. A CBR source generates four 64 byte packets per second. The load on the network is controlled by the changing number of CBR sources. In the tests, 10, 20, 30 and 40 CBR sources were used to generate traffic.

Messages from the CBR source sit in a queue until a route is discovered. To prevent indefinite buffering, messages are dropped if they remain in the queue for over 30 seconds.

Entries in cost tables are set to time out if not updated within four seconds.

In the simulation, a target sends a 32 byte acknowledgment to the CBR source once every two seconds. This acknowledgement message has the dual effect of notifying the CBR source that messages are reaching the target, but more importantly, it refreshes the path from the CBR to the target.

Each test was run for 900 simulated seconds and the results were averaged over ten consecutive runs in order to account for different network topologies.

B. Evaluation and Discussion

As in [2], three key performance metrics are evaluated: (i) *Packet Delivery Fraction*—the ratio of data packets successfully delivered to those originated; (ii) the *Average Latency*—the measure of the total end-to-end delay

in delivering a packet to a destination; (iii) *Normalized Routing Load*—the ratio of the total number of packets transmitted by any node to the number of packets successfully delivered to the destination[2].

Fig. 3 shows the Packet Delivery Fraction as a function of pause time and for different numbers of CBR sources. As can be seen from the graph, GRAd is nearly impervious to variable mobility—the percentage of good packets delivered remains essentially constant as the pause time changes.

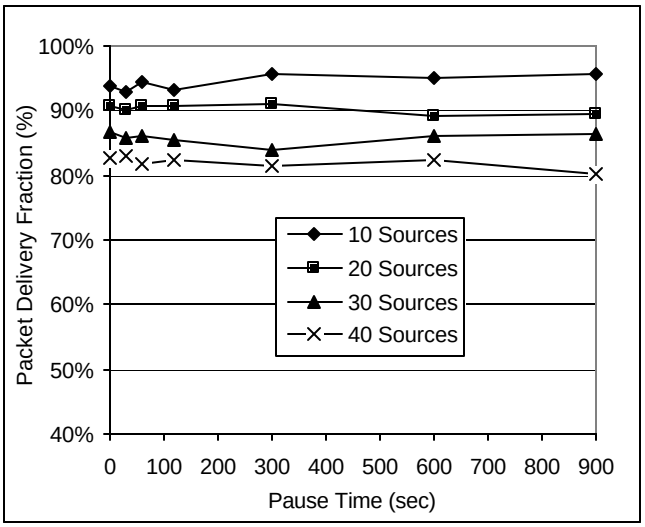


FIGURE 3: PACKET DELIVERY FRACTION

GRAd is robust in the face of changing topology because it enlists multiple neighboring nodes to relay messages from one place to another. If one node moves out of place, other nodes are often available to relay the packet without resorting to rebuilding the route.

Fig. 4 shows the average end-to-end latency for successfully delivered packets. In GRAd, latency remains under 7 milliseconds, even under conditions of high mobility and load. By contrast, [2] reports end-to-end delays of more than 100 milliseconds, and as high as one second for heavily loaded networks. It must be pointed out that is not an exact comparison: in [2], packet size was 512 bytes and the radio is simulated using a different path loss model.

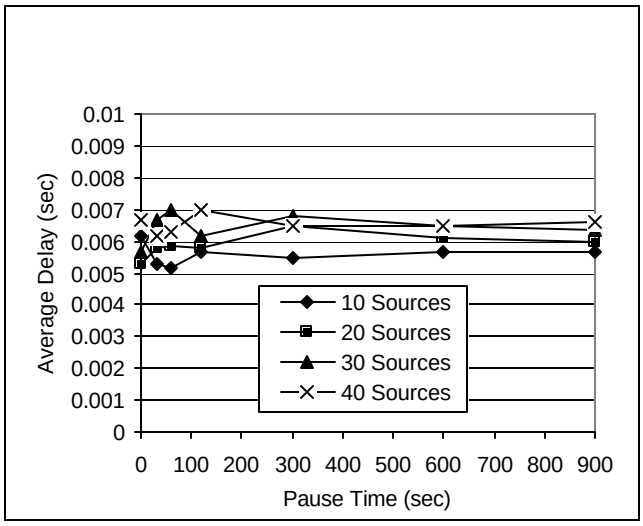


FIGURE 4: AVERAGE DELAY

In any practical implementation, GRAd is still likely to show small latencies since it avoids the RTS/CTS link to link handshake used in other protocols.

However, there is a price to pay for the "fast and loose" routing approach used by GRAd. Fig. 5 shows the routing load for various pause times and CBR sources.

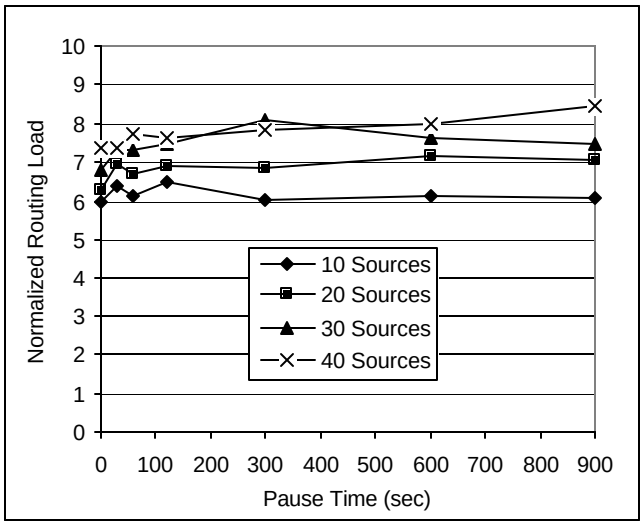


FIGURE 5: ROUTING LOAD

As Fig. 5 shows, for every one data packet received at the ultimate destination, between six and eight packets have been transmitted. Some of these "extra" packets are inevitable, for example, if a path requires two hops from source to destination, this will be recorded as a routing load of two. A destination node always sends an implicit acknowledgment notification, as described in Section II(H), which contributes to the load. Relatively few of the overhead packets are Reply Request messages, even in scenarios with high mobility[3]. The majority of the over-

[2] Note that in [2], the Normalized Routing Node counts the ratio of network control packets to all delivered packets. In the tests described here, *all* transmitted packets—data and control—are included in the numerator.

head packets are due to multiple neighbors attempting to relay the same packet. A consequence of this overhead is that GRAd networks exhibit more congestion than other network algorithms for the same offered load.

C. Effects of MAC layer

In any ad hoc network, there is no centralized control to control access to the airwaves, so nodes depend upon the MAC mechanism to cooperatively share the airwaves. GRAd, in particular, taxes the MAC layer since multiple neighboring nodes will attempt to relay a message soon after receiving it. It was therefore suspected that performance of GRAd would be sensitive to the choice of MAC layer.

The 802.11 MAC layer [4] is considerably more "fair" than GRAd's simple carrier sense and exponential back off approach described in section III(A). In the 802.11 approach, the MAC layer implements a countdown timer which is initialized to a random duration proportional to an exponential back off constant. The timer counts down only when the local airwaves are clear. When the timer expires, the MAC layer transmits the packet. This approach more evenly distributes air time to neighboring nodes.

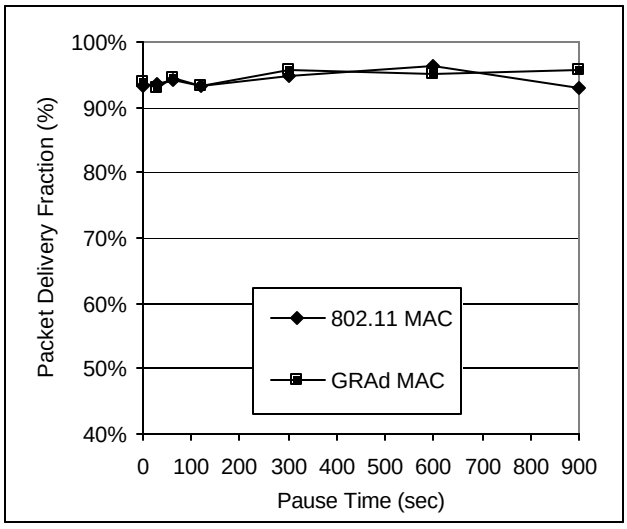
The tests were run for 10 CBR sources using the 802.11 MAC layer and compared against the same tests using the GRAd MAC layer—the results are shown in Fig. 6. It is interesting to note that there was little effect on the overall performance of GRAd using two substantially different MAC layers.

D. Disabling Route Repair

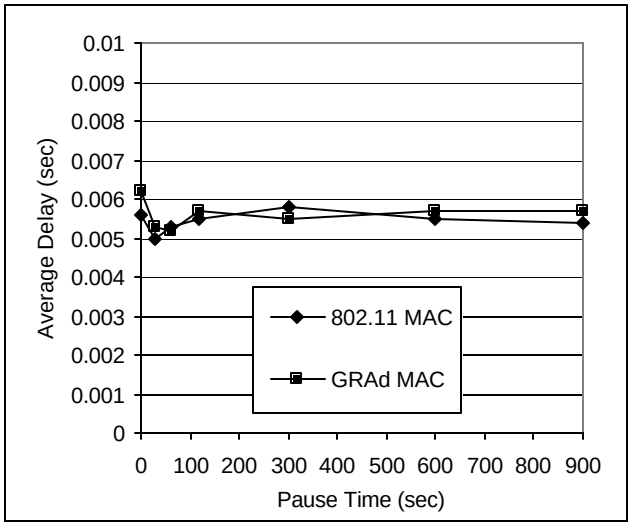
Section II(G) describes GRAd's mechanism for Route Repair: if the receiver fails to receive a packet from a sender within the expected period of time, it sends a reply to the sender with an increased `remaining_cost`.

To gain insights to the effectiveness of the route repair mechanism, the tests were run with route repairs disabled: if the network topology changed so an originator no longer reached its destination (more accurately, if an originator stopped receiving packets from its destination), the entries in the cost table would time out and the originator would start a new Reply Request process.

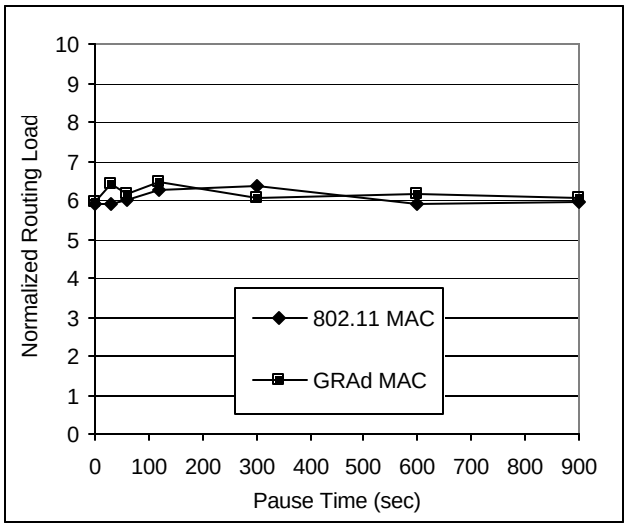
Fig. 7 shows the effects of disabling the Route Repair mechanism. The packet delivery fraction drops almost insignificantly, but somewhat surprisingly, the average latency and routing load are both improved.



(a) Packet Delivery Fraction (10 Sources)



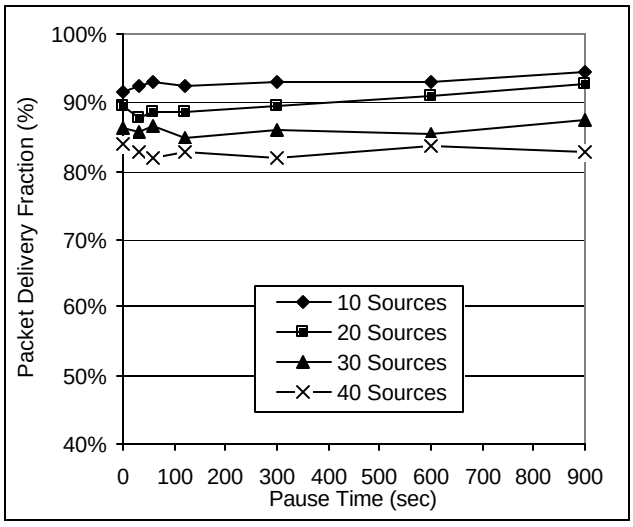
(b) Average Delay (10 Sources)



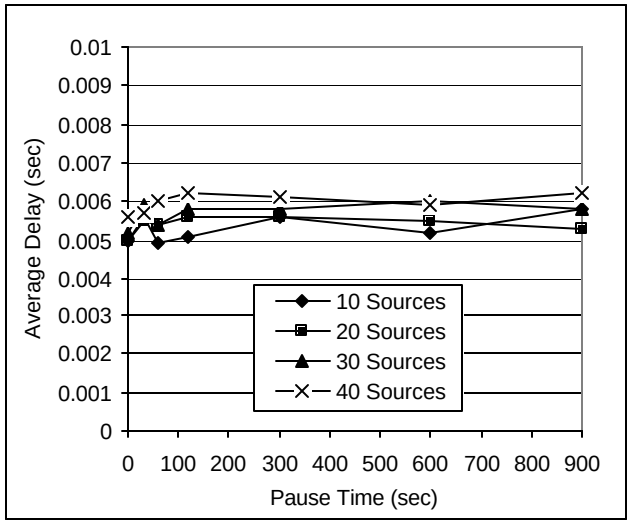
(c) Normalized Routing Load (10 Sources)

FIGURE 6: EFFECT OF DIFFERENT MAC LAYERS

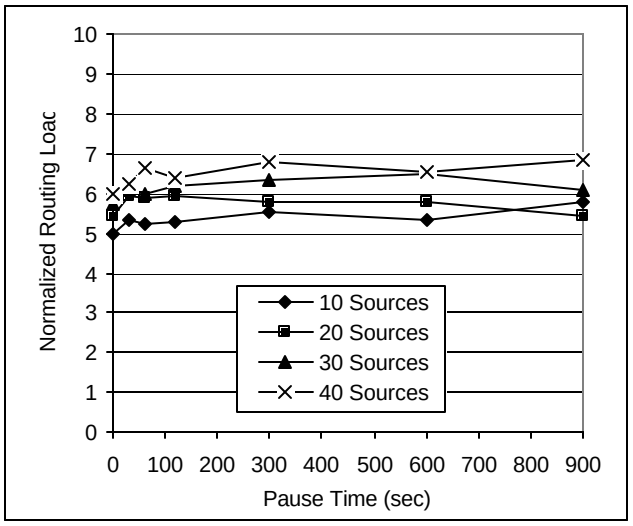
[3] In a typical test with 10 sources and 0 second pause time, only 2.5% of all messages transmitted were `M_REQUEST` messages.



(a) Packet Delivery Fraction



(b) Average Delay



(c) Normalized Routing Load

FIGURE 7: DISABLING ROUTE REPAIR

IV. CURRENT STATUS AND FUTURE WORK

The initial results of GRAd are encouraging, but there are a number unanswered questions whose answers may give insights to the operation of GRAd and suggest areas for improvement.

A. Better Quantitative Comparison

The *ns-2* simulator [3] has been extended as described in [1] and used extensively to test various ad hoc wireless routing protocols in [1][2][6]. Although Jasper has been written to test GRAd using similar test conditions, dependable comparisons can only be made if GRAd is tested using ns-2. Work is underway to implement GRAd in ns-2.

B. Hardware Implementation

Despite attempts to simulate the physical characteristics of the environment, "there's nothing like the real thing," and plans are underway to build a network of wireless nodes as a test bed in order to characterize the performance of GRAd in real-world conditions.

C. More intelligent routing

Throughout this paper, the term "cost" has been used to mean "number of hops," but metrics other than the number of hops are possible. For example, a node can charge a higher cost for relaying a message if it notices that its local airwaves are becoming congested, or if its local topology is changing rapidly. The higher relay cost will cause messages to flow around the node if there are other nodes that can relay at a lower cost.

To generalize, in a multi-hop wireless network, the only real choice a node can make is whether or not to relay a message that it has received, and if so, when to relay it. As suggested in [8], it may be useful to structure the problem of routing as a set of software agents residing in the nodes and in the messages; the agents decide what should be relayed and when. The network can then be viewed as a series of *activation* and *inhibition* functions, the former causing a message to be transmitted, the latter preventing it.

D. Preferred Neighbors

GRAd and AODV share many traits, including on-demand route discovery and updating reverse path information as a message is relayed from one node to next. GRAd permits any neighbor to participate in relaying a message, AODV insists upon a particular neighbor. A compromise between these two approaches shows some promise: A relaying node advertises a cost for relaying a message (a la GRAd) and suggests a preferred neighbor

to do the relaying (a la AODV). If that neighbor is not observed to relay the message within a certain amount of time, then non-preferred neighbors may attempt to relay the message. This approach could reduce some of the routing overhead observed in GRAd while still maintaining robustness in dynamically changing networks.

E. Functional Addressing

The broadcast nature of GRAd's Reply Request encourages *functional addressing*, in which a node initiates an `M_REQUEST` message containing a predicate rather than a specifying a fixed target ID. The predicate is a piece of software that embodies a query such as "Are you a color printer?," "Are you a gateway to a wired network?," or "Are you an ARP server?" Each receiving node evaluates the predicate and sends a reply to the requestor if the predicate evaluates to be true. If the requestor receives multiple replies, it can choose the reply that offers the lowest `accrued_cost` (i.e. is topologically closest) or that best satisfies some other application specific criteria.

F. Per-Session Addressing

In GRAd, routes are created on demand, entries in cost tables are short lived and persist only for the duration of a dialog between two nodes. The identities of the intermediate nodes are not required for passing messages. This opens the possibility of *per-session addressing*, in which an originating and replying nodes choose network IDs at random to be used for the duration of a session.

The space of IDs can be made large enough so the chance of two nodes choosing the same ID is insignificant.

Per-session addressing offers two advantages. The first is security: by changing its advertised address for each session, a node gains some measure of anonymity and protection against malicious eavesdroppers.

Second, manufacturing costs are reduced since network IDs don't need to be assigned and individually burned in at the time of manufacturing.

V. Conclusion

GRAd offers a new approach to ad hoc, on-demand routing. Rather than sending unicast packets, it exploits local broadcasting to contact multiple neighboring nodes. Messages descend a cost gradient from originator to destination without needing to identify individual intermediate nodes. Cost functions are updated opportunistically as messages are passed from one node to the next.

Through simulation, the performance of GRAd has been tested and characterized under a variety of load and mobility conditions. The results of the tests show that GRAd exhibits very low end-to-end packet delays and offers good immunity to rapidly changing topologies.

Acknowledgments

I would like to thank the members of my doctoral committee, Michael J. Hawley, Andrew B. Lippman and William J. Kaiser, for continually challenging my assumptions and keeping my attention focused on the "real problems." I have enjoyed discussions with Bob Metcalfe, Scott Corson, and Andrew Mendel, and appreciate the chance to work out my ideas with good critics. The MIT Media Laboratory and the Motorola Fellows Program have provided very tangible and much appreciated support. Charlotte Burgess provided excellent "just in time" editing support.

Finally, I'd like to give a special note of thanks to Hari Balakrishnan for his interest in my work and for encouraging me to develop it more fully.

References

[1] J. Broch, D. A. Maltz, D. B. Johnson, Y-C. Hu, J. Jetcheva. "A performance comparison of multi-hop wireless ad hoc networking protocols," in *Proceedings of the 4th International Conference on Mobile Computing and Networking (ACM MOBICOM '98)*, pp. 85–97, October 1998.

[2] S. Das, C. Perkins, E. Royer. "Performance comparisons of two on-demand routing protocols for ad hoc networks," *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Tel Aviv, Israel, March 2000, pp. 3–12.

[3] K. Fall, K. Varadhan (Eds.) *ns* notes and documentation, available from http://www.isi.edu/~salehi/ns_doc/.

[4] IEEE Standards Department. "Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," ISO/IEC 8802-11, First Edition, August 1999.

[5] J. Macker, S. Corson. "Mobile Ad-hoc Networks (manet) Charter." http://www.ietf.org/html.charters/manet-charter.html, February 2000

[6] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek. "Routing protocols for mobile ad hoc networks—a comparitive performance analysis," *Proceedings of the 5th International Conference on Mobile Computing and Networking (ACM MOBICOM '99)*, pp. 195-206, August 1999.

[7] D. B. Johnson, D. A. Maltz. "Dynamic source routing in ad hoc networks," in *Mobile Computing*, T. Imielinski and H. Korth, Eds., Kulwer, 1996, pp. 152-81.

[8] K. H. Kramer, N. Minar, P. Maes. “Tutorial: Mobile Software Agents for Dynamic Routing,” *Mobile Computing and Communications Review (ACM SIGMOBILE)*, vol. 3, no. 2, 1999, pp. 12–16.

[9] S. Kumar. “Sensor Information Techology (SenseIT) Program”, described in http://www.darpa.mil/ito/research/sensit/, DARPA Information Technology Office, April 2000.

[10] C. Perkins and E. Royer. “Ad-hoc on-demand distance vector routing,” *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, February 1999.

[11] R. Poor. “Jasper: a Java-basd dynamic simulator for ad hoc wireless networks,” unpublished.

[12] R. Ruth. “Global Mobile Information Systems (GloMo) Program,” in http://www.darpa.mil/ato/programs/glomo_mission.htm, DARPA Advanced Technology Office, March 2000