Enhancing MANET Routing Efficiency by Multipoint Relaying and 2-hop Repair

Po-Jen Chuang*, Po-Hsun Yen and Ting-Yi Chu

Department of Electrical Engineering, Tamkang University, Tamsui, Taiwan 251, R.O.C.

Abstract

Building proper routing protocols for a mobile ad-hoc network (MANET) is a critical challenge. A desirable MANET routing protocol must consider not only the broadcast storm problem which comes from flooding route requests in route discovery but also the path repair problem which usually takes massive control packets to rebuild paths when damage occurs. To advance the performance of MANETs, this paper presents a new routing protocol which employs the concept of multipoint relaying and 2-hop repair (MR2R) to deal with the broadcast storm and path damage problems. The MR2R protocol is distinct from previous protocols in its ability to reach rapid damage detection and repair, decrease packet amounts and increase the repair chances. As the obtained simulation results exhibit, our new protocol can effectively discover paths and repair damaged routes at less bandwidth consumption (i.e., with reduced control packets), in contrast to other protocols.

Key Words: Mobile Ad-hoc Networks (MANETs), Broadcast Storms, On-demand Routing Protocols, Local Repair, Multipoint Relaying, Performance Evaluation and Comparison

1. Introduction

Building efficient transmission routes is of particular importance for the resource-limited MANETs. Routing protocols used to focus on route validation and transmission delay, leaving out the broadcast issue which is especially critical in reactive routing protocols. Reactive routing protocols usually send route requests by flooding. The number of broadcast packets resulting from increased hops and connections can be huge and therefore jeopardize network efficiency. The goal of this investigation is work on both the broadcast mechanism and routing protocol in order to generate stable transmission routes more efficiently for large-scale, highly mobile MANETs. Our basic idea is to incorporate the concept of multipoint relaying (MPR) [1] into a proper routing protocol. MPR allows a node to broadcast its neighbor nodes periodically and build a two-hop adjacent node list. If to broadcast a route request, it can instantly calculate the two-hop connected dominating sets (TCDS) based on the list. Upon receiving a route request, only nodes in the TCDS need to rebroadcast it, thus saving bandwidth consumption.

As an established route in a highly mobile MANET may fail quickly due to invalid information (because nodes change positions frequently), a good path repair mechanism is urgent and crucial. To repair the route, some protocols – such as AODV [2] and MPRDV [3] – choose to rebroadcast the route requests (RREQ) which is quite inefficient for a highly mobile network. Another protocol – AOMDV [4] – adopts a different multi-path approach. When sending an RREQ to build a route, it also includes the position of the first node that receives the RREQ so that a middle node can build an independent alternative path to carry on the transmission when a current path fails. Although the multipath approach of AOMDV can reduce the route rebuilding time, it can also degrade the network performance as the alternative path may fail easily when path length grows and the dropped packet (the packet which is dropped due to the current path fail-
lure) must be sent again from the source node via the alternative path. MMDV [5] then adds the MPR mechanism into AOMDV to raise the packet arrival ratio by multiple paths but ends up producing even more control packets than AODV in building the routes.

To attain more efficient transmission in a highly active MANET, we employ the local repair mechanism to outperform MPRDV in this investigation. Our major concern is to reduce the number of broadcast packets effectively and to repair the damaged paths quickly. Our repair approach works as follows. In sending an RREQ or RREP (route reply), a middle node will include the position of the last broadcasting node and record the positions of the last two nodes. After a path is built, if a node along it becomes invalid, we can repair the path by locating alternative hops from the nodes in the two-hop adjacent list and meanwhile notifying the last broadcasting node of the next two valid hop positions. Such a local repair approach is favorable in that it can ensure path validity without rebroadcasting an RREQ. As our new protocol aims to enhance MANET routing efficiency by Multi-point Relaying and 2-hop Repair, we brief it as the MR2R protocol.

Extended simulation runs are conducted to evaluate the performance of our MR2R protocol and related protocols (including AODV, AOMDV, MPRDV and MMDV) under the following performance measures of interest: the packet delivery ratio (PDR), average hop count (AHC), delay per hop (DPH) and control overhead (CO). The results exhibit that our MR2R protocol constantly outperforms other protocols in PDR and CO at very reasonable AHC and DPH.

2. Background Study

2.1 Ad-hoc On-demand Distance Vector Routing (AODV)

AODV [2] contains route request, route reply and route maintenance. It builds and repairs a path only when necessary, to reduce the extra cost of building routes in a dynamic network topology.

2.1.1 Route Request

Route discovery starts upon request. When a node needs to send a packet to a destination, it first searches its routing table for usable routing information. If there is valid information, the node will send the packet together with the next hop (indicated in the table); otherwise, it will start route request by flooding the RREQ packet. An RREQ carries the source IP, destination IP and broadcast ID. Each node maintains a broadcast ID which will increase each time when an RREQ is sent (to mark the event). After sending out an RREQ, the source will set a timer and wait for an RREP. Sending an RREQ will create a route reaching to the destination and also a reverse route for the destination to return an RREP to the source.

2.1.2 Route Reply

In sending an RREQ, if there is a valid route to the destination in the routing table of a middle node, the middle node will return an RREP to the source; otherwise, the destination needs to return the RREP. Different from an RREQ, an RREP will be sent in unicast, i.e., the middle nodes will build a forward route to the destination. Upon receiving the RREP, the source will transmit packets by the built forward route to the destination.

2.1.3 Route Maintenance

Route maintenance includes maintaining the information in the routing table and managing the route breaking problem. Maintaining the routing table information can be easy: If a path is not used or updated in a period time, simply remove it from the table. When a route breaks, AODV has two measures. One is to return an RERR (route error) to the source and clear the information of the broken route along the way. Receiving such an RERR, the source will move on to build a new route to the destination. The other is to make use of the local repair concept: If the broken point is near the destination, the middle node (at the broken point) will broadcast an RREQ to search for an alternative route. When the destination finally gets the RREQ, it will send back an RREP and complete the route repair job, to save route reconstruction time.

2.2 Multi-path Routing Protocols

A multi-path routing protocol is designed to balance the overhead of each route so as to maintain the quality of service. It uses the concept of fault tolerance to handle the route information change/ destruction problem which results from node mobility in a MANET. The main idea is to build multiple paths between a source-destination pair in route searching and replying. Thus, when a current
path fails, packets can be sent via an alternative path—
to save the cost of searching for a new route. Ad-hoc on-
demand multipath distance vector routing (AOMDV) [4] is one of the multi-path routing protocols [4–11] in-
troduced in recent years.

**AOMDV** searches a route by the sequence number in
an RREQ, like AODV, but aided by two more para-
eters, the hop counts and the ID of the first adjacent
node to receive the RREQ. It also replaces the routing
table (containing the next hop) by a path table. When a
node sends an RREQ, the first neighbor receiver will
put its ID in the RREQ and broadcast further on. When
a middle node receives the RREQ, it compares the hop
counts in the RREQ with its own hop counts to see if it
has ever received this RREQ from the source. If not and
its hop counts are bigger than the hop counts in the
RREQ: The middle node will record this path informa-
tion and transfer the RREQ to the next hop. Otherwise,
the hop counts will be replaced by the smaller hop counts.
This process will enable a middle node to record many
loop-less paths. In the end, the destination will reply an
RREP to all received RREQ and eventually to the source
node—via these different paths. A multi-path route is
thus built.

After the route is built, the path with the smallest hop
counts will be chosen as the main transmission path. If
a node detects that it runs off the transmission range of
an adjacent node, it will check if there exist any paths
with invalid next hops in the routing table: If yes, delete
such paths. When a node realizes the alternative paths
of a transmission route are all deleted, it must broadcast
an RERR along with the destination ID. Any node re-
ceiving the RERR will check if it has the specified route
items. If yes and the next hop of any path in the route
item is also the source node of an RERR, delete the path.
The same approach is employed to check if any other
route runs out of valid alternative paths: If so, send out
an RERR again.

**AOMDV** is desirable because we can choose an
alternative path from the built multiple paths (between a
source-destination pair) to carry on packet transmission
when a current routing path breaks, and as a result reduce
both the packet dropping ratio and route repair time. There
are, however, disadvantages. For instance, as its desti-
nation node needs to reply an RREP to each RREQ, a
single route search will involve higher control packet cost
than AODV. AOMDV also faces worse network congues-
tion because it asks each node to save multiple (mostly
unused) paths for alternative routing and produces lon-
ger RERR packets and higher probabilities to send the
packets than AODV.

### 2.3 Local Route Repair Mechanisms

Route repair is another way to restore a transmission
path which fails due to node mobility. AODV has an or-
iginal but simple route repair mechanism: If a node de-
tects the next hop of a transmission path breaks, it will
start route repair only when the hop counts in the routing
packet exceeds the hop counts between itself and the de-
station. The repair process is as follows. The node re-
broadcasts an RREQ to the destination: If not receiving
an RREP in a fixed period of time, it will drop the packet
and send an RERR to the source. This has apparent pro-
blems: By flooding an RREQ, all nodes in the network
(except the destination and the node that replies RREP)
will receive and transfer it—the involved bandwidth cost
gets close to that for route reconstruction. Such a route
repair approach may raise packet arrival ratios in some
circumstances but will cause bandwidth overload and de-
grade network performance in large-sized networks. For
improvement, a number of routing protocols with local
repair mechanisms are introduced in the literature [12–
17]. **PATCH** [14] is a typical example. In PATCH, when
a node encounters a broken path, the TTL of an RREQ
will be set to 2—to generate a good chance of finding the
original next 2-hop (if the next hop is not broken) and to
reduce the RREQ and RREP quantities.

### 2.4 The MANET Broadcast Issues

As mentioned, a MANET contains a large number of
mobile nodes which communicate with each other within
valid communication ranges. In such a network, packets
must be routed in multi-hops to save the limited radio
power, channel usage and energy resource. If routing pro-
tocols for a MANET adopt periodical broadcasting to
find, maintain and update transmission routes, the large
quantity of packets may produce remarkable extra cost
and degrade network performance consequently. **Spontaneous and unreliable** are two distinct features of the
MANET broadcast because (1) without previous infra-
structures or synchronous mechanisms, nodes have no
idea when they are going to receive information broad-
cast from other nodes and (2) without any acknowledg-
ing mechanisms, the sender of a packet may lose traces
of it when any receiver node gets disconnected – actively or passively.

2.4.1 The Broadcast Storm Problem
Flooding – any node receiving a message will instantly re-broadcast it – is the basic way of direct broadcasting. When flooding becomes the broadcasting mode of a MANET, it may generate huge rebroadcast cost and intensive channel competition, especially in large-sized networks. Some schemes are introduced to solve the broadcast storm problem (e.g., [18,19]). Among them, the probabilistic scheme reduces superfluous broadcasting by a set probability, the distance-based scheme uses the distance between two nodes as the threshold to decide whether to rebroadcast a received message, and the location-based scheme allows each node to control the broadcasting range by GPS.

2.4.2 Multi-Point Relaying (MPR) [1]
MPR is an efficient design among existing broadcast mechanisms. The basic idea is to select a smallest set from the 1-hop neighbors which is able to cover the 2-hop neighbors and add the set in the broadcast packet. To calculate the smallest set is an NP-complete problem, so MPR makes use of the greedy algorithm. If \( N(x) \) = the 1-hop neighbor set, \( N_2(x) \) = the 2-hop neighbor set and \( MPR(x) \) = the selected MPR set, the algorithm can be calculated as follows:
1. Start with an empty set \( MPR(x) \).
2. Select any one-hop neighbor node in \( N(x) \) which is the only neighbor of some node in \( N_2(x) \) as the multipoint relay node. Add such one-hop neighbor nodes to the multipoint relay set \( MPR(x) \) and remove nodes which are now covered by \( MPR(x) \) from \( N_2(x) \).
3. While there remain some nodes in \( N_2(x) \):
   (a) For each node in \( N(x) \) but not in \( MPR(x) \), compute the number of nodes in \( N_2(x) \) which the node can cover.
   (b) Add a node in \( N(x) \) which covers the maximum number of nodes in \( N_2(x) \) to \( MPR(x) \). Remove the nodes which are now covered by \( MPR(x) \) from \( N_2(x) \).

The MPR set can be thus obtained. Take Figure 1 as an example. In the figure, S is the source; A, C, D and E are 1-hop neighbors; B, F and G are 2-hop neighbors. S initially puts C and D into the packet. After A, C, D and E receive the packet, only C and D need to rebroadcast it, but when rebroadcasting completes, all nodes in the network receive this packet. In this case, MPR takes only 3 broadcastings to finish a job that flooding will need 6 broadcastings.

3. The Proposed MR2R Protocol
3.1 Maintaining Neighbor Tables
In our protocol, nodes will periodically broadcast the following Hello messages to each other to build the 2-hop neighbor tables.
(1) All Neighbors: including all current 1-hop neighbors (based on it, a node can build its 2-hop neighbor table).
(2) Neighbors Deleted: giving the disconnected neighbors detected by the sender.
(3) Neighbors Unchanged: indicating the current neighbor table of the sender remains unchanged (so the neighbor count is set as 0 and the neighbor IP address is null).

A Hello message also carries the sequence number of the sender so that a receiver can update its route table and save a new round of route searching if a packet is routed to the sender.

The neighbor table in our protocol uses the 1-hop tuple and 2-hop tuple to indicate the neighbor relationship. Figure 2 shows the neighbor table of node A and the Hello messages of nodes B, E, D and G. Assume each node receives at least one Hello message from a neighbor node and there is no topological change. Now, when node A receives a Hello message, it adds the source node into its 1-hop tuple and builds the 2-hop tuple with marked neighbors in the message. (In the neighbor table of node A, 1-hop tuple \(<B>\) indicates B is a neighbor of here). When broadcasting a message, S will add C and D into the packet. After A, C, D and E receive the packet, only C and D need to rebroadcast it, but when rebroadcasting completes, all nodes in the network receive this packet. In this case, MPR takes only 3 broadcastings to finish a job that flooding will need 6 broadcastings.
A and 2-hop tuple \(<B, C>\) indicates C is a neighbor 2 hops away from A but reachable by way of B.) It will not add nodes in the 1-hop tuple into the 2-hop tuple. For example, in the Hello message of B, A will record only \(<B, C>\) but not D because D is the 1-hop tuple for A.

### 3.2 Route Searching and Replying

Different from AODV, we let RREQ enclose the previous two-hop IP address and the forward node IP addresses to facilitate executing multipoint relaying. Such a design can lessen the broadcast storm problem which confronts AODV in the route searching practice and as a result bring up the performance of our protocol. When a source issues a new RREQ, it will record NULL in the previous two-hop IP address and add its ID into the originator and the previous hop IP address, calculate the nodes which need to rebroadcast using MPR and the 2-hop neighbor table, and add these nodes into the forward node IP addresses. Any node receiving the RREQ will check if this sender is in its routing table: If not, update by the information and build a reverse route. After updating route information, a node checks the ID: If having received the same RREQ or its own ID is not in the forward node IP addresses, drop the packet; if yes, return it to the route layer and start to repair the route. There are conditions for repairing a route:

1. The current route does not expire or is not under repair.
2. The next 2-hop neighbor \((N)\) must be valid and at least one of the 1-hop neighbors connecting to \(N\) must be valid.
3. The next-hop node is not the destination node.
4. The value of last_time_used must be some time (less than the packet delivery period \(\times 1.5\)) earlier than the current time.

If a damaged route fits all the conditions, start the 2-hop repair process. Otherwise, check the largest_hops_forwards: If the value is bigger than the hop counts to the destination, employ the AODV repair mechanism; if smaller, invalidate the route, generate an RERR packet which records all invalid destinations and broadcast the packet (same as AODV).

To conduct the 2-hop repair, we first choose a 1-hop neighbor with the longest expiration time and meanwhile connecting to the 2-hop neighbors, according to the next 2-hop neighbor tables. Then generate an RPRQ (repair request) that records the ID of the destination and the sequence number of the node, and send the packet to the next 2-hop node via the previously chosen 1-hop neighbor. Receiving such an RPRQ, a node will check if the destination is itself: If not, fill its ID into the previous hop IP address and broadcast it out; if yes, generate an RPRP (repair reply) packet, fill in route repair information and send it to this RPRQ originator via the next-hop nodes. However, if transmission gets interrupted or the
route repair information in an RPRQ becomes invalid, return an RPF (repair failure) – not an RPRP – to the source.

When a node receives an RPRP, it will check if the ID of the destination matches its own. If not, update route information based on the route repair information in the packet, add its ID into the previous hop IP address and broadcast the packet. If yes, the node recorded in the previous hop IP address of this RPRP becomes the new next-hop node, update this route following the route repair information in the RPRP, and generate an RTCH (route change) packet. Then, fill the failed next-hop node, the new next-hop node and the updated route information into the RTCH packet and broadcast it to notify neighbors that this route has been fixed. After neighbors receive the RPRP, the 2-hop repair operation completes.

If a node sends an RPRQ but receives no RPRP or RPF in a period of time, it will check the hop counts: If the largest_hops_forwards exceeds the hop counts between itself and the destination, repair the route as AODV; otherwise, generate an RERR and broadcast it. Figure 3 displays a complete 2-hop repair process of our protocol. As the original network shows, there is a connection between nodes S and D (Figure 3 (a)), and this connection breaks when node E moves (Figure 3 (b)). Node B detects the disconnection and starts the 2-hop repair mechanism. It searches the 2-hop neighbor table to locate the next 2-hop node G and sends an RPRQ to G by F (Figure 3 (c)). After receiving the RPRQ, G returns an RPRP to B by F (F then records the new route to D). When B receives the RPRP, it updates F as the new next 1-hop neighbor to D and broadcasts an RTCH to notify A of this route change. A then changes its next 2-hop neighbor (to D) from E to F (Figure 3 (d)). The 2-hop route repair is then completed, as Figure 3(e) shows.

3.4 The Advantages of our Protocol

3.4.1 Rapid Damage Detection and Instant Repair

Figure 3 illustrates how to recover a damaged transmission path using the 2-hop neighbor tables built by the source node.
MPR broadcast mechanism. As observed, the entire packet transmission is carried out by unicast whose ACK mechanism enables the node that sends an unsuccessful RPRQ to learn about the failed repair attempt rapidly. If transmission is carried out by broadcast, a failed repair attempt will take longer (the set broadcast cycle) to surface. Besides, if many a node detects route disconnection and simultaneously starts the repair process, the large quantity of packet broadcasting may intensively influence the network.

3.4.2 Reducing Packet Amounts
Reducing the number of packets is another distinct advantage of our protocol. As our protocol sets the destination of an RPRQ to be the next 2-hop neighbor in the route request, an RPRQ will receive only one RPRP. If we send an RPRQ like AODV-ABR [13], we may receive numerous RPRP packets (from all neighboring nodes) that hold either the requested route information or the repair failure information due to node dislocation or loss of route information. When node dislocation happens, note that we can still proceed with the repair process because our RPRP carries the route information.

3.4.3 Enhancing Repair Chances
The fact that we allow a node to update the next-2-hop node by the RTCH message will create a better chance for route repair. For instance, OPTAODV [12] can fix a damaged route fast but when a new pre-hop node – which knows nothing about the original pre-2-hop node – is disconnected from its upstream node, it is unlikely to restore the route again. OPTAODV is also incapable of repairing a reverse route built by an RPRQ because it is based on the pre-2-hop information built by RREP.

3.5 Other Discussions
As mentioned, the core of our routing protocol is the 2-hop neighbor tables built by the MPR broadcast mechanism, and building such 2-hop neighbor tables indeed takes quite an amount of control packets. Despite the fact, our protocol remains feasible because for the MPR broadcast mechanism, it reduces a lot more control packets than it consumes.

In multiple-hop repairing, we must consider the information-synchronous problem. Below, we take the 2-hop repair in our protocol as an example. As the information of a 2-hop neighbor will be received in the broadcast time of 2 Hello messages, it may result in futile route repair when an un-updated invalid 1-hop node is involved to fix the damage. If such an information-synchronous problem happens to a 2-hop repair mechanism, it can also happen to 3-hop (or more) repair schemes – with more serious consequences. Raising the broadcast frequency of Hello messages can be a relief choice, but the involved cost can be significant – especially for more-hop designs which have cost quite heavily in building the neighbor tables.

When building the 2-hop neighbor tables, our protocol lets a node judge by itself if its connection to another node is good or not, which may produce asynchronous states between nodes. However, our protocol maintains only the single-way routes (like AODV) and will hence consider or handle a 2-way route as two independent links. That is, when a connection breaks, we will respectively repair each single-way route. Despite that the inter-node asynchronous state problem casts no significant influence on the performance of our protocol, we still include it in our future study and will continue to explore the feasibility of repairing two-way routes.

4. Simulation Results
Simulation runs are carried out using network Simulator-2 version 2.33 [20] to evaluate the performance of AODV, AOMDV, MPRDV, MMDV and our MR2R. The involved parameters are listed in Table 1, and the performance measures of interest include the packet de-
livery ratio (PDR), average hop count (AHC), delay per hop (DPH), and control overhead (CO).

4.1 The Packet Delivery Ratios (PDR)

\[ \text{PDR} = \frac{\text{the total CBR packets received by all destinations}}{\text{the total CBR packets delivered from all sources}} \]

(i.e., the ratio that a packet is successfully sent from the source to the destination). Figures 4 and 5 depict PDR of each protocol under different CBR session counts and maximum mobility. Figure 4 shows that when CBR session counts increase, PDR decreases for protocols not incorporated with MPR because the rapidly increased RREQ amount congests the network. Among MPRDV, MMDV and our protocol, we produce the highest PDR as our 2-hop repair mechanism can repair all current routes. In Figure 5, the protocols yield nearly the same PDR at node mobility = 0. When node mobility increases, the advantage of employing MPR becomes obvious: Significantly reduced control packet amounts and route searching time – along with shorter new routes – lead to higher PDR. Our protocol yields the best performance due to its ability to repair most damaged routes at small cost. In fact, our 2-hop neighbor tables can fix route damages even when all original nodes of a route run out of the communication range of the to-be-repaired node.

4.2 The Average Hop Counts (AHC)

\[ \text{AHC} = \frac{\text{total hop counts travelled by all packets that reach destinations successfully}}{\text{the number of these packets}}. \]

Figures 6 and 7 give AHC for protocols under different session counts and mobility. At session count = 100, protocols using MPR take fewer AHC than those not using it because original routes built by the MPR broadcast are usually shorter than those built by flooding. When the session count increases, AHC decreases for AODV and AOMDV as packet congestion has led to packet dropping in longer routes. By contrast, protocols using the MPR broadcast are less likely to cause congestion or drop packets – AHC hence decreases in a moderate way. In Figure 7, protocols using the MPR broadcast yield quite close results and so are protocols without MPR. When
mobility grows, \textit{AOMDV}, \textit{MMDV} and \textbf{our protocol} depict higher AHC due to higher packet arrival rates. \textit{AODV} and \textit{MPRDV} keep lower AHC because they are more likely to rebuild routes which come close to the shortest paths.

4.3 The Delay Per Hop (DPH)

\( \text{DPH} = \frac{\text{the average end-to-end delay of a successfully delivered packet}}{\text{the average number of hop counts the packet has travelled.}} \)

Evaluation on packet delay usually refers to the average time for completing a successful packet transmission. This is in fact an improper evaluation measure for routing protocols with good route repair ability and high PDR – because a protocol able to fix routes rapidly usually takes more time to transmit those \textit{rearranged} packets that tend to have larger hop counts. To attain better evaluation, we mind also the total hop counts, i.e., we take into account both the end-to-end delay and DPH.

Figure 8 gives the end-to-end delay for protocols with varied session counts. The result shows that \textit{AODV} and \textit{AOMDV} yield shorter delay time at small session counts. But when counts grow, delay rises fast because neither of them uses the MPR broadcast to reduce RREQ amounts. \textit{AOMDV} produces longer delay at larger session counts because it needs extra packets to build multiple paths (due to obvious congestion). The same trend can be found for \textit{MMDV} and \textit{MPRDV}. Compared with \textit{MPRDV}, \textbf{our protocol} has shorter packet delay at small session counts, thanks to its efficient repair design which shortens the repairing/waiting duration. For us, packet arrival rates and packet delay both grow with session counts – a normal result from large numbers of packets and network congestion.

Note that, between 500 and 600 sessions, the delay time decreases (instead of increasing) for \textit{AODV}. Figure 9 which gives DPH under different session counts offers a reasonable explanation. The result here yields almost the same trend as what appears in Figure 8 for each protocol except \textit{AODV}. \textit{AODV} keeps rising DPH when session counts grow – because it needs to generate a large number of control packets which will lead to congestion and also packet dropping. As packets with longer transmission routes are more likely to suffer route damages and therefore need repair, they will produce longer DPH than packets with shorter paths. The value of DPH can duly reflect the packet delay caused by network congestion and route repair.

Figure 10 gives DPH at different mobility. At mobility = 0, all protocols hold close DPH and those without MPR have higher delay because they need to broadcast more packets. When mobility increases, the benefits of multipath designs start to show, especially for \textit{MMDV} which has the least DPH. \textbf{Our protocol} also performs well: When mobility grows, we maintain higher packet arrival rates with shorter DPH than \textit{MPRDV} – indicating our 2-hop repair mechanism works better than the original repair mechanism in \textit{AODV}. The end-to-end delay at different mobility is provided in Figure 11 for comparison. At mobility = 100, \textit{AOMDV} has longer end-to-end delay than \textit{AODV} but similar delay as \textit{MPRDV} and \textbf{our protocol} – revealing the end-to-end delay is not a proper delay indicator and justifying our adoption of DPH.

4.4 The Control Overhead (CO)

\( \text{CO} = \text{the total bandwidth consumed by control packets.} \)

The amount of control packets is the key indication for extra overload. To attain fair evaluation for protocols...
without MPR, we consider bandwidth consumption (instead of broadcast times) for total control packets under both broadcast and unicast transmissions. The results for different session counts are depicted in Figure 12. As we can see, protocols with MPR have much less CO than those without. When session counts increase, the difference gap grows even larger. For two protocols using the same broadcast mechanism, the one with multipath will consume more bandwidth than the one without. This is because in building main routes, a multipath protocol will also build alternative routes. Our protocol needs less CO than MPRDV because we repair most damaged routes by fewer and shorter control packets sent by unicast, while MPRDV needs to broadcast a large number of RREQ to repair or rebuild routes.

Figure 13 gives CO at varied mobility, with bandwidth = 11 Mbps. When mobility = 0, AOMDV needs more CO than AODV because even at mobility = 0, its link layer may mistake packet collision as transmission failure and decide to send RERR. Figure 14 displays CO at varied mobility, with bandwidth = 54 Mbps. Here, the overhead difference for AOMDV and AODV turns smaller (because AOMDV faces more serious congestion in Figure 13 and hence needs to send more RERR). The overhead difference for MMDV and MPRDV remains the same — because incorporated with MPR, MMDV can efficiently reduce network congestion. When mobility increases, our cost comes closer to that of MPRDV. This is understandable as increased mobility will generate more broken routes and our protocol tends to repair all emerging broken routes, thus consuming more bandwidth. For MPRDV, there are more chances to rebuild a new route — with fixed cost — when facing route damages, which makes its CO less affected by mobility change.

4.5 Further Discussions

Note that the main goal of our investigation is to lift the efficiency of MANET routing. To achieve the goal, our MR2R protocol adopts multipoint relaying and 2-hop repair in route discovery and maintenance, to reduce the number of broadcast packets effectively and repair damaged paths quickly. We also display that the above performance improvement is obtained with reduced control packets.

Besides MR2R, a number of other protocols are recently proposed to enhance the efficiency of MANET routing. Among them, [21] brings in the power and traffic load balancing consideration; [22,23] attempt to identify misbehaving and non-cooperating nodes in order to pursue trust-based MANET routing (because non-cooperation, selfishness and malicious behavior of nodes may result in the collapse of MANET routing). Other protocols, such as [24–26], adopt either application-driven or flow-based route discovery. In their practice, a routing packet controller will reside in each MANET node, and
each application can control route discovery by installing rules or policies in the controller. We believe that, if these protocols are brought to work together with our MR2R, the efficiency of MANET routing can be further enhanced.

5. Conclusions

This paper presents a desirable new routing protocol to attain efficient transmission for a highly active MANET. Based on MPR broadcast and a 2-hop route repair mechanism, our protocol can reduce the bandwidth cost due to route discovery and perform efficient local repair by the 2-hop neighbor tables. As route repair is carried out by unicast transmission, we can repair routes rapidly and efficiently – to handle the frequent recurrence of route damages in highly mobile networks. Simulation results show that when compared with other routing protocols such as AODV, AOMDV, MPRDV and MMDV, our MR2R protocol performs constantly better in packet delivery ratios and control overhead at very reasonable average hop counts and delay per hop.

References


Manuscript Received: Sep. 17, 2014
Accepted: Apr. 17, 2015