Optimization on Control Overhead in MANET

Swarnali Hazra
University of Calcutta,
India

Sanjit Setua
University of Calcutta
India

ABSTRACT
MANETs are comprised of mobile hosts communicating via their wireless interfaces and their topology is continuously changing. During communication process between mobile hosts or nodes, control message-transfers among the nodes occur. Control messages consume bandwidth, processing resources to both transmit and receive a message. Since bandwidth is at a premium, routing protocols should not send more than the optimal number of control messages they need for operation, and should be designed so that this number is relatively small. In this paper, Optimized AODV and Optimized DSR are proposed to minimize mainly the control overhead

Keywords
Control overhead, optimal path, RREQ, RREP, AODV, DSR

1. INTRODUCTION
MANET stands for Mobile Ad Hoc Network. The term mobile denotes the capability of changing from one state to another. Ad Hoc Network is a collection of wireless network platforms consisting of mobile nodes, where nodes may organize and maintain a routing infrastructure among them in a relatively arbitrary fashion. Characteristics of ad hoc networks include resource-poor devices, limited bandwidth and a continuous changing topology. Among the available resources, battery power is typically the most constraining. Hence, there are some typical design goals for ad hoc network routing protocols. These are minimal control overhead, minimal processing overhead, multi hop routing capability, dynamic topology maintenance, loop prevention. Here in this work, the main optimization is on control overhead and processing overhead. Other design goals are also maintained in the proposed protocols.

The existing routing protocol AODV and DSR supports three phases, Route Request (RREQ), Route Reply (RREP) and Packet Delivery. Destination node can receives more than one Route Request (RREQ) from different one hop neighbor nodes and for this reason, source node can receives more than one Route Reply (RREP) through different paths. So, many message-transfers occur and that could be the cause of control overhead. At the source node, there is an extra processing for selection of optimal path with smallest hop count and current sequence number, for packet delivery. That could be the cause of processing overhead. In our proposed protocols, destination node receives only one RREQ from a neighbor node which forwards the RREQ to destination node first. To stop unnecessary flooding, after receiving first RREQ, destination node sends an acknowledgment (N_{ack}) to those nodes which have not transmitted RREQ yet, to ensure that destination node has received the RREQ and to inform other non-transmitted nodes not to transmit identical RREQ and discard it. So, destination node does not receive more than one RREQ and that is why source node receives RREP only through the optimal path. N_{ack} consists of RREQ ID to identify the identical RREQ. On the other hand, as the existing protocol AODV supports "expanding ring search" method, source node and intermediate nodes can broadcast RREQ multiple number of times. In the proposed protocol Optimized AODV, if destination node is in the transmission range of source node, then source node needs to broadcast only once and no other broadcast of RREQ is needed for the source node and other nodes in the network. If destination node is not in the transmission range of source node, then source node needs to broadcast only twice and other intermediate nodes need to broadcast RREQ, only for once. Here we also have concentrated on minimal control overhead and cycle prevention. It is also implemented in the proposed protocol Optimized DSR. In the existing protocol DSR, while forwarding RREQ, every intermediate nodes need to append their IPs into RREQ. In the RREQ process, almost entire network can be flooded. A node, which is far away from source node, can have a RREQ, with many appended IPs of previously traversed nodes. It causes the requisite overhead. To minimize this overhead, in the Optimized DSR, we have modified the RREQ process. No intermediate node need to append their IP addresses to RREQ, they only need to maintain parent node information, which is the next hop to the source node from current node.

In the section2, we have presented the status of the considered domain and we have discussed our proposed work in the section3. A ground of future work is raised in the section 4, with the conclusion.

2. REVIEW WORK
An ad hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. Several routing protocols have been proposed for such environment. The protocols are mainly classified into three types, Proactive, Reactive and Hybrid. In Proactive protocol, each node in the network maintains the route to every other node in the network all the time and route creation and maintenance are based on periodic updates [6]. In Reactive protocol, routes are only discovered between nodes when they are actually needed [6]. Hybrid protocol is the combination of proactive and reactive protocol [6].

AODV [6, 1, 2] and DSR [6, 3, 4] are the reactive routing protocols. AODV is based on distance vector routing and DSR is based on source routing. AODV improvement protocol [5] with the Hello message mechanism is presented based on wireless link availability prediction. Here the control overhead is reduced

by dynamically updating the environment parameter of Hello interval. As an optimization for the current DSR, a new adaptive routes selection scheme based on the stability of nodes was proposed, considered joint hop-count and node stability. The enhanced-DSR [7] has higher throughput and less end-to-end packet delay.

3. PROPOSED WORK

Our proposed protocols are reactive in nature. Whenever communication is needed between two nodes, the route will be discovered. Otherwise no node maintains route related information.

In this section we have discussed our two proposed algorithm, with the following set of symbols:

i. S = Source node, D = Destination node
CN= Current node, RP = Reverse Parent IP,
FP = Forward Parent IP

ii. IR =Intermediate Routers or nodes between S and D.

iii. RREQ = Route Request message, RREP = Route Reply.

iv. RREQ ID.s.id, RREQ ID.dis.id = Route Request id with corresponding extension.
(.s.id)→S broadcasts first time for 1hop destination.
(.dis.id)→S broadcasts second time for multi hop destination to discover route.
RREP ID.s.id, RREP ID.dis.id = Route Reply id with corresponding extension.
(.s.id)→It is for one-hop destination node.
(.dis.id)→ It is for multi-hop destination node.

v. RT = Routing Table.

vi. RREQ_RP = RREQ parameter: <S, D, hop count, sequence number, RP, RREQ ID>.

vii. TR = Transmission Range of a node.

viii. $T_{RREQ.s}$ = [Difference between first broadcast time of RREQ from source node (for which time range is considered) and the time at which a 1hop destination node of source node will receive that broadcasted RREQ of corresponding RREQ ID] + [waiting time before processing at 1hop destination node].

ix. $T_{RREP.s}$ = [Difference between broadcast time of RREQ from 1hop destination node of source node and the time at which source node will receive that broad-casted RREP of corresponding RREP ID] + [waiting time before processing at source node].

x. $T_{RREQ.dis}$ = [Difference between broadcast time of a node (for which time range is considered) and the time at which a 1hop node of that node will receive that broad-casted route request (RREQ) of corresponding RREQ ID]+ [waiting time before processing at receiver node].

xi. m = Smallest hop from source to reach a node, for which the time range is considered.

xii. n = Largest hop from source to reach a node, for which the time range is considered.

xiii. N_{ack} = Acknowledgment to ensure that destination has received the RREQ and to inform other non-transmitted nodes not to transmit further identical RREQ and discard it.

3.1 Optimized ad-hoc on-demand distance vector (OAODV) routing

In OAODV if source node have a route to destination node, it does not need to discover any route, otherwise route discovery is needed. If destination node is 1hop node of source node, then there is no need to maintain the parent information. Otherwise route discovery process will maintain the parent information. OAODV route discovery follows RREQ process, RREP process and packet delivery. In the RREQ process, source node first broadcasts RREQ of unique ID (RREQ.s.id). If source node receives RREP of unique ID (RREP.s.id) within ($T_{RREQ.s}$+$T_{RREP.s}$), it implies that destination node is in the transmission range of source node. Otherwise source node broadcasts RREQ of unique ID (RREQ.dis.id) for the second time. When intermediate nodes decide to process the received RREQ of unique ID (RREQ.dis.id), they store Reverse Parent IP, RREQ ID and discard further identical RREQ till $2T_{RREQ.dis}$. After receiving a RREQ of unique ID (RREQ.dis.id), destination node broadcasts N_{ack} to stop unnecessary further broadcasting of identical RREQ. After receiving N_{ack}, other nodes which have not broad-casted identical RREQ yet, discard it from time instant $mT_{RREQ.dis}$ to $nT_{RREQ.dis}$. RREQ (of ID: RREQ.dis.id) parameters: < Source IP, Destination IP, Reverse Parent IP, Hop Count, Sequence no., RREQ ID > . If destination node is in the transmission range of source node, then it sends RREP of unique ID (RREP.s.id) to source node, within the time ($T_{RREQ.s}$+$T_{RREP.s}$). Otherwise it sends RREP of unique ID (RREP.dis.id). RREP of unique ID (RREP.dis.id) traverses only the optimal path from destination node to source node, using Reverse Parent information. In the RREP process, all traversed nodes store the Forward Parent IP. RREP (of ID: RREQ.dis.id) parameters : < Source IP, Destination IP, Forward Parent IP, Hop Count, Sequence no., RREP ID > . The source node delivers the packet through this optimal path, using Forward Parent information.

3.1.1 Algorithm

S first checks its route table to determine whether it already has a route to D. If such a route exists, it can use that route for packet delivery, otherwise route discovery is needed. Route discovery process consists of following processes.

RREQ Process:

1. S broadcasts the RREQ of (RREQ ID.s.id) first.

1.a. If D is in the TR of S, then S sends the RREQ directly to the D. D returns RREP of unique ID (RREP.s.id) to the S, within ($T_{RREQ.s}$+$T_{RREP.s}$). Other 1hop nodes are not D and they check the extension part of the RREQ ID, if it is (.s.id) then they discard it.

1.b. Else If S does not receive RREP of unique ID (RREP.s.id) from D within ($T_{RREQ.s}$+$T_{RREP.s}$), then it broadcasts the RREQ of unique ID (RREQ.dis.id) second time to find the D. IR receive this and check the extension part of the RREQ ID , if it is (.dis.id) then they pass it until the D is found. When IR decide to process the received RREQ, they store the RP, which is next hop to the S from CN (While forwarding RREQ, S does not need to store the RP) and also stores the RREQ ID. All nodes maintain the RREQ_RP.

All nodes which have already broad-casted RREQ once, discard identical RREQ (by checking the RREQ ID in received RREQ packet and stored RREQ ID in its routing table) till $2T_{RREQ.dis}$ after broadcasting RREQ.

2. When D receives a RREQ of unique ID (RREQ.dis.id), it does not receive any further identical RREQ.

After receiving first RREQ, D broadcasts N_{ack} to stop unnecessary further broadcasting of identical RREQ. After receiving N_{ack} other nodes, which have not broad-casted identical RREQ, discard it (by checking the RREQ ID in further received RREQ and stored RREQ ID in its RT) from time instant $mT_{RREQ.dis}$ to $nT_{RREQ.dis}$.

All IR which have already broad-casted RREQ once, discard N_{ack} of same ID (by checking the RREQ ID in received N_{ack} and stored RREQ ID in its RT).

RREP Process:

3. After receiving first RREQ, D initiates RREP.

3. a. If S is in the TR of D, then D sends the RREP of unique ID (RREP.s.id) directly to the S, within the time ($T_{RREQ.s}+T_{RREP.s}$).

3. b. Else D sends the RREP of unique ID (RREP.dis.id) to its RP, which is next hop to the S. The RP of D is now the CN. This CN send the RREP of unique ID (RREP.dis.id) to its RP which is next hop to the S. This procedure continues until the S is reached. It is done by unicast method. IR and S store the FP which is next hop to the D from the CN. (D does not need to store the FP while passing the RREP).

Packet Delivery:

4. It is done by unicast method. S sends the packet to its FP which is the next hop to the D (maintained in RT of S, while passing RREP). S deletes the FP from its RT. The FP of S is now the CN. This CN send the packet to its FP which is next hop to the D. CN deletes the FP from its RT. This procedure continues until the D is reached.

3.1.2 Example

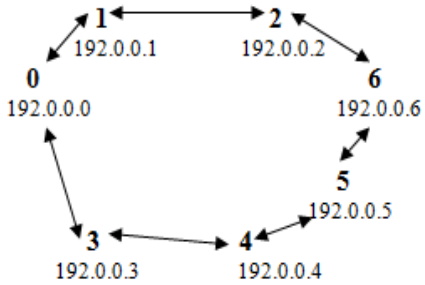

Figure-1: Example Network

In the Figure1, nodes are represented by their node id, such as 0, 1, 2 and so on. Arrows are denoting which node is in the transmission range of which node. Locations of all nodes are mentioned by their IP addresses. Node 0‘s IP address is 192.0.0.0, node 1‘s IP address is 192.0.0.1 and so on.

- Source: node0 and Destination: node1 (Figure-1)
  - Node1 and node3 are in the transmission range of node0. Since the destination node1 is in the transmission range of node0, no need to maintain reverse parent in RREQ process and forward parent in RREP process.
  - The source node0 broadcasts RREQ of unique ID (RREP.s.a) with node0’s IP address (192.0.0.0) and the IP address of destination node1 (192.0.0.1). Node1 and node3 receive it.
  - Node1 will check the extension of RREQ ID and it is (RREQ.s.a). Node1 will find itself as destination node by matching the destination IP in the RREQ packet and its own IP. Then node1 will send back RREP of unique ID (RREP.s.a) directly to the node0, within the time ($T_{RREQ.s}+T_{RREP.s}$).
  - Node3 will check the extension of RREQ and it is (RREQ.s.a). Since node3 is not the destination node and the extension of RREQ ID is (RREQ.s.a), node3 will discard that RREQ. No further broadcast is required.
  - After receiving the RREP, node0 will check the extension of RREP ID which is (RREP.s.a) and will check whether RREP of unique ID is received within the time ($T_{RREQ.s}+T_{RREP.s}$). If all condition is satisfied then node0 will send the packet directly to the node1.
- Source: node0 and Destination: node6 (Figure-1)

  RREQ Process:
  - Node6 is not in the transmission range of node0. Node1 and node3 are in the transmission range of node0.
  - The source node0 broadcasts RREQ of unique ID (RREQ.s.a), with node0’s IP address (192.0.0.0) and the IP address of destination node6 (192.0.0.6). Node1 and node3 receive it.
  - Node1 and node3 will check the extension of RREQ ID and it is is (RREQ.s.a). Then node1 and node3 will not find themselves as destination node by matching the destination IP in the RREQ packet and their won IPs. Node1 and node3 will discard the RREQ of unique ID (RREQ.s.a). That means node0 will not receive RREP of unique ID (RREP.s.id) within the time ($T_{RREQ.s}+T_{RREP.s}$)
  - After that, node0 will broadcast RREQ for the 2nd time but with different extension of RREQ ID (RREQ.dis.a). Node1 and node3 (non destination nodes) will receive it and check the extension of RREQ ID which is (RREQ.dis.a).

- Then node1 and node3 create Reverse Parent IP (for both, reverse parent IP is node0's IP, i.e. 192.0.0.0). Then node1 and node3 broadcast the RREQ of unique ID (RREQ.dis.a). Node0, node2 and node4 will receive it.
- Node0 will receive two RREQ of ID (RREQ.dis.a) from node1 and node3.We have discussed that a node which already has broadcasted a RREQ of unique ID (RREQ.dis.a) will discard the further identical RREQ till $2T_{RREQ.dis}$, after broadcasting it. So, node0 will discard it. On the other hand node2 and node 4 will receive RREQ of ID (RREQ.dis.a) from node1 and node3 respectively and create reverse parent IP (node2's reverse parent IP is. 192.0.0.1 and node4's reverse parent IP is 192.0.0.3).
- Then node2 and node4 will broadcast it. When node2 broadcast identical RREQ then node1 and node6 will receive it. Node1 discard it and node6 create the reverse parent IP i.e. 192.0.0.2. On the other hand, when node4 broadcast identical RREQ then node3 and node5 will receive it. Node3 discard it and node5 create the reverse parent IP i.e. 192.0.0.4.
- Then node6 will find itself as destination node by matching destination IP (192.0.0.6) in RREQ and its own IP (192.0.0.6). Then node6 immediately will broadcast N_{ack} to stop further broadcasting of identical RREQ. N_{ack} contains RREQ ID (RREQ.dis.a). Node2 and node5 receive N_{ack} .
- Node2 will check the RREQ ID in N_{ack} and the stored RREQ ID in the routing table. Both RREQ ID will match. That means node2 already has broadcasted RREQ of ID (RREQ.dis.a). Then node2 will discard N_{ack} .
- On the other hand, after receiving N_{ack} node5 will not find the identical RREQ ID in its routing table. That means node5 has not broadcasted RREQ of ID (RREQ.dis.a) yet. Then node5 will check whether it has received any RREQ of identical RREQ ID. In this case node5 has received a RREQ of identical RREQ ID. Node5 will discard that RREQ and broadcast the N_{ack} .
- After receiving N_{ack} from node5, the Node4 will check the RREQ ID of N_{ack} and the stored RREQ ID in the routing table. Both RREQ ID will match. That means node4 already has broadcasted RREQ of ID (RREQ.dis.a). Then node4 will discard N_{ack} .
- Intermediate nodes which have broadcasted RREQ of unique ID (RREQ.dis.a): node0, node1, node2, node3, node4. Node6 is the destination node.
- Intermediate nodes which have broadcasted N_{ack} with unique ID (RREQ.dis.a): node6 and node5.
- Intermediate nodes which have discarded N_{ack} : node2 and node4.
- Intermediate node which has not broadcasted RREQ of unique ID (RREQ.dis.a) : node5.
- RREQ parameters at node4 at a time instance: < Source IP: 192.0.0.0, Destination IP: 192.0.0.6, RP: 192.0.0.3, Hop Count: 2, Sequence no., RREQ ID: RRE.dis.a >.

RREP Process:

- Node6 has received only one RREQ from node2. So, node6 will find node2 as reverse parent, in its routing table. Then node6 will send RREP of unique ID (RREP.dis.a) to its reverse parent node2.
- After receiving this RREP, node2 will create forward parent (node6's IP: 192.0.0.6). Node2 will find node1 as reverse parent, in its routing table. Then node2 will send RREP of unique ID (RREP.dis.a) to its reverse parent node1.
- After receiving this RREP, node1 will create forward parent (node2's IP: 192.0.0.2). Node1 will find node0 as reverse parent, in its routing table. Then node1 will send RREP of unique ID (RREP.dis.a) to its reverse parent node0 (source node).
- Traversed nodes: node6, node2, node1, node0.
- RREP parameters at node1 at a time instance: < Source IP: 192.0.0.0, Destination IP: 192.0.0.6, FP: 192.0.0.2, Hop Count: 2, Sequence no., RREP ID: RREP.dis.a >.

Packet Delivery:

- Node0 will receive only one RREP from node1, through the optimal path. Node0 will find node1 as forward parent, in its routing table and send packet to its forward parent node1.
- Node1 will find node2 as forward parent, in its routing table and send packet to its forward parent node2.
- Then Node2 will find node6 as forward parent, in its routing table and send packet to its forward parent node6 (destination node).
- Packet Delivery path: node0→ node1→ node2→ node6.

3.2 Optimized dynamic source (ODSR) routing

In ODSR if source node have a route to destination node, it does not need to discover any route, otherwise route discovery is needed. If destination node is 1hop node of source node, it does not need to maintain the parent information in RREQ process and no need to maintain source routing in RREO. Otherwise route discovery will follow the parent information and source routing. ODSR route discovery follows RREQ process, RREP process and packet delivery. In the RREQ process, source node first broadcasts RREQ of unique ID (RREQ.s.id). If source node receives RREP of unique ID (RREP.s.id) within ($T_{RREQ.s}$+$T_{RREP.s}$), it implies destination node is in the transmission range of source node. Otherwise source node broadcasts RREQ of unique ID (RREQ.dis.id). When intermediate nodes and destination node receive a RREQ of unique ID (RREQ.dis.id), they store Reverse Parent IP, RREQ ID and discard identical RREQ till $2T_{RREQ.dis}$. After receiving a RREQ of unique ID (RREQ.dis.id), destination node broadcasts N_{ack} to stop further broadcasting of identical RREQ. After receiving N_{ack} , other nodes which have not broad-casted identical RREQ yet, discard it from time instant $mT_{RREQ.dis}$ to $nT_{RREQ.dis}$. RREQ (of ID: RREQ.dis.id) parameters: < Source IP, Destination IP, Reverse Parent IP, Hop Count, Sequence no., RREQ ID > . If destination node is in the transmission range of source node, then it sends RREP of unique ID (RREP.s.id) to source node, within the time ($T_{RREQ.s}$+$T_{RREP.s}$). Otherwise it sends RREP of unique ID (RREP.dis.id). RREP of unique ID (RREP.dis.id) traverses the optimal path from destination node to source node, using parent information. Destination node places the Source's IP address, as well as its own IP address, into RREP of unique ID (RREP.dis.id) and then sends it to source node via optimal path, using parent information. When the intermediate nodes receive RREP of unique ID (RREP.dis.id), they append their own IP addresses to it. RREP (of ID: RREP.dis.id) parameters: <Destination IP, Source route entry, Hop Count, Sequence no., RREQ ID>**.** If destination node is in the transmission range of source node, then source node sends the packet directly to the destination node. Source node sends the packet only through the optimal path from source node to destination node, using source node route information.

3.2.1 Algorithm

S first checks its route table to determine whether it already has a route to D. If such a route exists, it can use that route for packet delivery, otherwise route discovery is needed. Route discovery process consists of following processes.

RREQ Process:

1. S broadcasts the RREQ of (RREQ ID.s.id) first.

1.a. If D is in the TR of S, then S sends the RREQ directly to the D.D returns RREP of unique ID (RREP.s.id) to the S, within ($T_{RREQ.s}$+$T_{RREP.s}$). Other 1hop nodes are not D and they check the extension part of the RREQ ID, if it is (.s.id) then they discard it.

1.b. Else If S does not receive RREP of unique ID (RREP.s.id) from D within ($T_{RREQ.s}$+$T_{RREP.s}$), then it broadcasts the RREQ of unique ID (RREQ.dis.id) to find the D. IR receive this and check the extension part of the RREQ ID , if it is (.dis.id) then they pass it until the D is found. IR and D store the RP, which is next hop to the S from CN (While forwarding RREQ, S does not need to store the RP) and also stores the RREQ ID. All nodes maintain the RREQ_RP.

All nodes which have already broad-casted RREQ once, discard identical RREQ (by checking the RREQ ID in received RREQ packet and stored RREQ ID in its routing table) till $2T_{RREQ.dis}$ after broadcasting RREQ.

2. When D receives a RREQ of unique ID (RREQ.dis.id), it discards identical RREQ till $2T_{RREQ.dis}$.

After receiving first RREQ, D broadcasts N_{ack} to stop unnecessary further broadcasting of identical RREQ. After receiving N_{ack} other nodes, which have not broad-casted identical RREQ, discard it (by checking the RREQ ID in further received RREQ and stored RREQ ID in its routing table) from time instant $Mt_{RREQ.dis}$ to $Nt_{RREQ.dis}$.

All IR which have already broad-casted RREQ once, discard N_{ack} of same ID (by checking the RREQ ID in received N_{ack} and stored RREQ ID in its routing table).

RREP Process:

3. When D receives first RREQ, it initiates RREP.

3. a. If S is in the TR of D, then D sends the RREP of unique ID (RREP.s.id) directly to the S, within the time ($T_{RREQ.s}$+$T_{RREP.s}$).

3. b. Else D places the IP address of S, as well as its own IP address into the RREP of unique ID (RREP.dis.id) and then D sends it to its RP which is next hop to the S. The RP of D is now the CN. This CN places its own IP into the RREP of unique ID (RREP.dis.id) and then it will send the RREP to its RP. This procedure continues until the S is reached. In RREP process, no need to maintain any parent information.

Packet Delivery:

4. It is done by unicast method. S sends the packet to its next hop node to the D (mentioned in the RREP packet header). S deletes the RP from its RT. The next hop node to the D of S is now the CN. This CN send the packet to its next hop node to the D, which is next hop to the D. CN deletes the parent information from its RT. This procedure continues until the D is reached.

3.2.2 Example

Here we are using the figure-1 to explain the example.

- Source: node0 and Destination: node1 (Figure-1)
  - RREQ, RREP and Packet Delivery processes are same as OAODV, which is already described in the section3.1.2.
- Source: node0 and Destination: node6 (Figure-1)

  RREQ Process:
  - Unlike DSR, the ODSR will not follow the source routing in the RREQ process.

    The RREQ process is same as the OAODV, which is already described in the section3.1.2.

RREP Process:

- Node6 has received only one RREQ from node2. So, node6 will find node2 as reverse parent, in its routing table. Then node6 will place the IP address of node0 (192.0.0.0), and append its own IP address (192.0.0.6) into the RREP of unique ID (RREP.dis.a) and then node6 will send it to its reverse parent node2, which is next hop to the node0.
- Then node2 will find node1 as reverse parent, in its routing table. Node2 will append its own IP (192.0.0.2) into the RREP of unique ID (RREP.dis.a) and then it will send the RREP to its reverse parent node1.
- Then node1 will find node0 as reverse parent, in its routing table. Node1 will append its own IP (192.0.0.1) into the RREP of unique ID (RREP.dis.a) and then it will send the RREP to its reverse parent node0 (source node).
- RREP parameters at node1 at a time instance: < Source IP: 192.0.0.0, Source Routing: 192.0.0.1 <->192.0.0.2 <->192.0.0.6, Hop Count: 2, Sequence no. , RREP ID: RREP.dis.a >
- Traverse Nodes: node 6, node 2, node1, node0.

Packet Delivery:

- Node0 will receive only one RREP from node1, through the optimal path. Now node0 can learn the entire path (optimal path) to destination. Node0 will find node1 as next node in the RREP packet header and send packet to node1.
- Node1 will find node2 as as next node, in the RREP packet header and send packet to node2.
- Node2 will find node6 as next node, in the RREP packet header and send packet to node6 (destination node).

4. CONCLUSION

Optimized AODV and Optimized DSR protocols for mobile ad hoc network have been presented. Considering the integrity of the protocol we have tried to present the every detailed step of the proposed algorithms with an example network. The transmission overhead is reduced by stopping the unnecessary broadcasting of Route Request packets and unnecessary traversing of Route Reply packets. In the next step of our work, we will implement our protocol and conduct a performance analysis on optimized control message over the other measures of control messages and also on the number of broadcastings. More research into novel mechanisms for optimized routing in such networks is necessary.

5. REFERENCES

[1] C. Perkins. AODV Routing Implementation for Scalable Wireless Ad-Hoc Network Simulation (SWANS). http://jist.ece.cornell.edu/docs/040421-swans-aodv.pdf

[2] C. Perkins, E. Belding-Royer, S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. 2003. http://www.ietf.org/rfc/rfc3561.txt

[3] D. Johnson, Y. Hu, D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. 2007. http://www.ietf.org/rfc/rfc4728.txt

[4] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. COMPUTER SCIENCE. Mobile Computing. The International Series in Engineering and Computer Science, 1996, Volume 353, 153-181

[5] Liu Chao and Hu Aiqun. Reducing the Message Overhead of AODV by Using Link Availability Prediction. Lecture Notes in Computer Science, 2007, Volume 4864/2007, 113-122,

[6] Stefano Basagni, Marko Conti, Silvia Giordano and Ivan Stojmenovic. Mobile Ad Hoc Networking (chapter: 9). A JOHN WILEY & SONS, INC., PUBLICATION.

[7] Xia ofeng Zhong,Youzheng Wang, Shunliang Mei, and Jing Wang . A Stable Adaptive Optimization for DSR Protocol. Lecture Notes in Computer Science, 2003, Volume 2524 /2003, 179-199