ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 11, November 2013

Fault Tolerance in Mobile Ad-Hoc Network with Enhanced_AOMDV Protocol

A. Dhanalakshmi [1], D.Maheshwari [2]
[1]*Research Scholar, M.Phil Computer Science, Dr.N.G.P Arts and Science College Coimbatore, India*
[2]*Head of the Department, Department of Computer Technology, Dr.N.G.P Arts and Science College, Coimbatore, India*

Abstract— Mobile Ad-hoc network (MANET) is a collection of mobile nodes, which form a wireless network without the use of an existing infrastructure. MANET's are usually characterized by random mobility of nodes, nodes arbitrarily entering and leaving the network and variable transmission range of nodes. These characteristics make MANET links to be intermittent and the topology to be highly dynamic. In this research work proposed two concepts, one is to reduce the overhead caused unnecessary transmission of hello packets and second to preemptively discover routes in cases of a link breakage, which make a path useless. In this concept, the nodes are not participating in the active route or route discovery, reduce the unnecessary hello packet transmissions by nodes. This research work also improves the packet delivery ratio by finding routes preemptively before the last route becomes unavailable to the source. In this research work applied various metric such as packet delivery ratio, packet dropped due to no-route, average end-to-end latency, average route discovery latency, route discovery frequency, routing overhead, number of hellos messages. These metric are compared with ENHANCED_AOMDV and AOMDV. The simulation is done using NS2. The simulation result showed to increases packet delivery ratio, reduces end-to-end delay, increased route discovery latency and frequency, reduce overhead caused due to hello messages and packets dropped due to link failure or no route availability in the network.

Index Terms— **MANET, AOMDV, ENHANCED_AOMDV.**

I. Introduction

Mobile Ad-hoc network (MANET) is a collection of mobile nodes, which form a wireless network without the use of an existing infrastructure. MANET is usually characterized by random mobility of nodes, nodes arbitrarily entering and leaving the network and variable transmission range of nodes. These characteristics make MANET links to be intermittent and the topology to be highly dynamic.

In such situations routing protocols are desired to be on-demand and robust enough to cope up with the dynamic topology changes. Many on demand routing protocols have been suggested for this purpose. Ad-hoc on demand distance vector routing (AOMDV) tries to use multiple metrics to discover routes and proposes new schemes to provide route maintenance. The objective of a routing protocol is to efficiently find routes and transfer as many data-packets possible with minimum routing overhead.

II. LITERATURE REVIEW

Fault tolerance in link failure and network failure in MANET can occur due to the fully or partially failed components in the network because of malfunction otherwise natural disaster. The node can move away from the cluster link failure will occur. So, avoid this failures lot of approaches has been proposed. Melamed et..al [4] proposed Octopus a fault-tolerant and efficient position-based routing protocol is suitable for failure prone environments. In this protocol frequently update node location using flexible state and achieves low location update overhead by using novel aggregation technique. Octopus applicable for fixed node density and not raise the network size. In [6]Khazaei and Berangi increase the data transfer and fault tolerance by creating backup path between source and destination during route reply, route maintenance and local recovery. Chandrasekaran et.al [8] proposed a model Trusted Fault Tolerant (TFT) used in Location Aided Routing (LAR) protocol. LAR focused link faults, high mobility, node congestion and capacity of buffers. The Location fault occurs the destination node move away from source. The proposed model considers that node is selfish or misbehaving node. It improves the location awareness and node trust level based on recovery of the lost packet.

In [25] Rana and Ahamed proposed a new fault tolerant routing protocol extends of DSR protocol. In this protocol identify atleast two paths between sources to destination. Link failure occurs in one path immediately select another path. Shaji et.al [27] [29] proposed a novel Self-eliminating Fault-tolerant based Un-interrupted reliable Service switching mobile Protocol (SFUSP) it includes the task of clustering and self elimination to creates a reliable route in heterogeneous network and identify the link break in early. Adeluyi and Lee present [30] a Spiral Millipede-inspired Routing Algorithm (SMiRA). It uses a resource light technique and bio-inspired approach, which is decrease the routing overhead and improved fault tolerance in link failures. Mutual Eclusion (MUTEX) algorithm [36] used to tolerate link or host failure using the time out based method. In [39] investigates communication failure in grid based topology using distributed genetic algorithm also use simple retry and reroute protocol to solve the communication failure in the network. In [42] proposed a Trusted Fault

ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 11, November 2013

Tolerant (TFT) model based on Location Aided routing protocol along with user recovery features. It covers link failures during packet forwarding and location failures.

III. ENHANCED_AOMDV

Enhanced_AOMDV identifies few shortcomings in legacy AOMDV and tries a modified approach to address the issues. First, AOMDV is a single metric routing algorithm, which uses just hop count as a routing metric, it doesn't allow the path lengths to differ more than one. Second, it uses the paths in the order they were created thereby not even considering its primary hop count metric. Enhanced_AOMDV proposes the use of multiple metrics to choose paths instead of just hop count, it also propose a local update scheme to continuously strengthen the current path. .

A. USE OF MULTIPLE METRICS FOR PATH SELECTION

AOMDV introduces the concept of multiple metrics for choosing path. It proposes to use

1. Node-to end Received Signal Strength Index (RSSI)
   i) Average cumulative Received Signal Strength Index
   ii) Minimum Received Signal Strength Index
2. Node-to end Latency
3. Node-to end occupancy metric

Instead of just hop count as a single metric in order to better suit to requirement of mobile network. Node-to end RSSI metric refers to the value of RSSI on a path from the node to the destination. Enhanced_AOMDV calculates RSSI index based on a non-linearly divided scale as it is more critical to switch to a better path as the RSSI value gets closer to the noise floor. The RSSI index is calculated from the path strength parameter which in turn is calculated when a packet is received using the packet→txinfo →RxPr variable in NS-2.The calculated path strength is then divided on a non-liner scale with increased divisions closer to the noise floor (-90 dBm) [28].

The node also updates the RREQ's path strength field before rebroadcasting it with an appropriate value depending whether it is using Minimum RSSI or average cumulative criteria. LAM-AOMDV uses the calculated RSSI in two ways either as minimum RSSI on the path or as Average Cumulative RSSI. When using Minimum RSSI criteria each node updates its routing table and the RREQ with the minimum of value of RSSI observed between the RREQ:path_strength and the value with which the packet itself is received according to the rules.

Node-to end Latency metric is calculated as the time taken by the RREQ /RREP to reach a node since it's origination at the source. It is calculated by taking the difference between current time and the time stamp in the packet. For instance in case of a RREQ it will be calculated as: Latency_upto_here = *CURRENT_TIME – RREQ::*time_stamp. The latency metric is calculated at each node. Also the value of latency once calculated is also updated in the routing tables latency field.

Similarly, Node-to end occupancy metric is an indicator of how involved or busy a particular node is, this metric is particularly important in case of networks developing hotspots. In Enhanced_AOMDV it is calculated as the number of DATA packets processed by a particular node per second irrespective of the source of data packets. The following subsections will try to explain in detail the route establishment algorithm used by Enhanced_AOMDV.

B. MULTI-METRIC ROUTE DISCOVERY

Enhanced_AOMDV follows and adds to the existing logic of AOMDV for route discovery to provide information about RSSI and latency to the nodes. It uses the routing messages like RREQ and RREP to dissipate information amongst the nodes. Similar to AOMDV when the application layer at the source generates a data-packet the node look up in it routing table whether if it has a route available to the destination, if it doesn't have one then it generates and broadcasts a RREQ packet. This broadcasted RREQ packet arrives on the neighboring nodes, which would update their routing table, but before updating their routing tables they calculate the values of the metrics they are using for example if using Average cumulative RSSI it will calculate path_strength_upto_here as:

Path_strength_upto_here = RREQ:: path_strength + packet->txinfo_->RxPr .

It will update the RREQ with the calculated value if it doesn't have a valid route to the destination according to general rules of AOMDV.It will then update its routing table with cumulative_path_strength_upto_here as:

Cumulative_path_strength_upto_here = path_strength_upto_here / RREQ : : hops.

C. REVERSE ROUTE SETUP

The reverse route setup phase is explained with an example shown in Figure:1. In this figure, when source node 'src' has a data packet to be sent to the destination but no route it initiates the route discovery and broadcasts a RREQ packet which in turn received by node 1, 2 and 3. All three of them add a reverse path entry towards the source in their routing table along with respective value of RSSI and latency which is calculate according to rules explained in section C and D these nodes also update the RREQ by increasing the hop count by one and with appropriate value of RSSI and latency. Now consider node 2 hearing the broadcast from 1, now node 2 will neither update its routing table or rebroadcast RREQ as the hop count of RREQ heard from 1 is greater than what it has already advertised and has the same sequence number. Also at this point node 4 hears the broadcast from 1 and 2. But since the hop count for both the paths is the same therefore it will update the routing table. With similar rebroadcasting from node 4 the RREQ reaches the Destination.

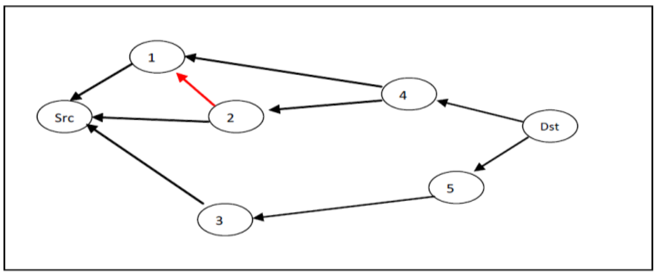
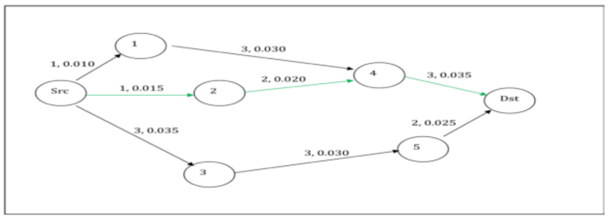


Figure: 1 Showings reverse route setup

D. FORWARD ROUTE SETUP

When Destination 'Dst' receives the first copy of RREQ it adds a reverse route entry to source 'Src' and sends a RREP. In order to avoid the route-cut-off problem the destination send out a RREP in response to each RREQ received via a unique first hop neighbor. In this example it sends out a RREP in response to both RREQ's received via node 4 and 5. Now at node 4, if RSSI would be totally neglected then it will be agree to to send out a RREP on any of the path between 1 or 2 which is often the case in common scenario. But facilated by the additional information about RSSI the RREP is forwarded on route 2 (assuming only hops + minimum RSSI criteria) .The forward route setup is also



shown in Figure:2.

Figure: 2. Showings Forward Route Setup

E. LOCAL UPDATE FOR CONTINUOUS PATH STRENGTHENING

Enhanced_AOMDV broadcasts Hello message but Enhanced_AOMDV adds to its value by using it to bring better routes into being. In AOMDV the Hello messages provide updates about node surroundings. But these updates are only partial because they only provide information about the immediate surroundings of a node i.e. if a particular node is still its neighbor or not, Enhanced_AOMDV extends the logic of Hello message by sequentially sending one route entry in them at a time. The hello messages are used to proactively upgrade the route and converge to the best path available.

Enhanced_AOMDV is inherently adaptive in nature when the source gets the first RREP it start sending out data-packets, but subsequently when the source gets more RREP's then it switches to the best available path (in terms of RSSI considering we use HOPS + RSSI as routing metric). This path is called the primary path the other paths that are discovered to the destination are called secondary routes. When RREP is unicaste back to the source form the destination then each node receiving it marks itself as forward route node (fr node). When the source gets the first RREP it starts sending data-packet to its first hop neighbor, on arrival of data-packet the nodes start sending out hello messages (which contain one route entry from the FR nodes routing table) to the one hop neighbor which are called as surrogate nodes, on receiving these hello messages the surrogate nodes themselves start sending out hello messages.

For understanding the mechanism of hello messages let's consider the beginning when none of the nodes are forward nodes or surrogate nodes. When the source has data packet to send and it doesn't have a route to the destination then it broadcast RREQ messages, the RREQ's are propagated through the network thereby also propagating the metric information in process. The destination in turn sends out RREP when the source get the first RREP it starts sending out data-packets, also with subsequent arrival on RREP the node switches to best path if there is one available. When the nodes on the path receive data-packets it marks itself as forwarding-node (FR=1) and starts sending out hello messages with the i_come from_fr flag equal to 1, with sequential route entries in them. Each Hello message contains only one route entry at a time. These hello messages are in turn heard by the one hops neighbor which mark themselves as surrogate nodes, these surrogate nodes in turn start sending out hello messages which also contains sequential route entries from the surrogate nodes itself. The forwarding nodes again in turn hear these hellos. The forwarding node if decides that it has received a better path to the destination according to the route reception algorithm, through the surrogate nodes then it starts forwarding data-packets to the source node instead of the previous next hop. On receiving the data packets the surrogate nodes now mark themselves as forwarding nodes and set the I_am_surro gate node flag as 0. The previous forwarding node time out and unmark themselves from being forwarding nodes. The process of local update through hello.

F. RULES OF LOCAL UPDATE ALGORITHM

While updating route it is important for the FR and SR nodes to maintain disjointeness and also to avoid paths merging into each other. For the same reason following rules are strictly observed:

1. If a node is neither FR nor SR then it should ignore all the hello messages. Only FR and SR are allowed to update their routing entries on hearing hellos from other FR and SR.
2. SR will ignore hellos from other SR nodes but a FR can update its routing entry on hearing hello form another FR as well.

III. UNSTABLE HELLO INTERVALS

I proposed a technique to decrease the numberofRequest messages in Enhanced-AOMDV without disturbing the Packet delivery ratio. The function of a routing algorithm is proficiently providing a path for data packets from the source to the destination. Proficiently at this point refers with smallest amount overhead. The attempt here is to decrease the overhead involvement from request messages.

Request messages act as a significant tool for sustaining local connectivity. In AOMDV simply allow a node to identify about its surrounds when a node receives request messages from its neighbors. It constantly keeps adding up and abolition of nearby node from the neighbor list based on how long it has received a request from a particular neighbor. Enhanced-AOMDV includes intelligence to the request messages by means of them to carry enhanced paths. Though, due to intrinsic mobile environment of ad-hoc network the nodes are continuously moving and it is likely for a node, which is presently take part in a lively path to no longer be connected with the path.

Also it is probable for a node to be isolated from other nodes in which the node is not essentially take part in any kind of comprehensible data communication. In such a case the invariable request messages transmitted by the node simply add towards the network overhead and also may be obstruct with the close broadcast due to the hidden node problem.

I proposed a novel scheme to decrease the number of request messages devoid of distressing the packet delivery ratio.

A. *INTRODUCTION OF KEY PACKET PERCEPTION*

The assessment of decreasing request messages occurrence else stopping request messages is depend upon the information though the node is aggressively take part in the network. The proposal of decreasing request messages in an additional perception is depend on the position of the node. Hence the request messages regularity of the node is exaggerated only when the state of a scrupulous node is not any of the following:

1. Active intermediary node in a continuing transmission
2. Node implicated the Route discovery method to set up a fresh route
3. Source for data packets
4. Target for data packets

To decide whether the node is in one of the mentioned position I obtain of the subsequent:

1. A node can be identified as it involved in route discovery process. Because, if it has received a RREQ / RREP packet in recent or past.
2. A node can be identified as a source node it has expected a data packet from the higher layers.
3. An active node, it is endlessly getting data from upstream nodes

4. Destination for data-packets if it has received a RREQ in recent or past with itself noticeable as destination or if it is endlessly receiving data packets.

Thus I make a hypothesis that the node is not take part in the network if it has not expected one of the following packets in recent past. RREQ, RREP , RERR and Data packet.

The assessment of changing interval after which the subsequent hello message will be send is depend on whether the nodes obtain one of the above revealed packets: For simplicity sake these packets hereon will be referred to as key packets.

B. *SPLITTIG UP OF ALGORITHM INTO THREE STEPS*

As mentioned previous the algorithm is alienated into three steps which are:

Step1: Decrease of hello message frequency to half.

Step2: Prevent to broadcast request message.

Step3: Restart broadcast of request messages with normal frequency.

While Step 1 and 2 are reliant on whether a node recognize one of the key packets in recent, it is also very important that the node should start rebroadcasting hello requests as rapidly as it acquires disturbed in a route discovery process or involved in a dynamic route also as an agent node, source or a destination. Before dropping the hello occurrence each node should have an redundancy period in which it stay for some time and supervise that it do not obtain any of the subsequent packets.

1. RREQ 2. RREP 3. RERR 4. Data packet

Then it can be securely understood that the node is for the time being not play a part in an active rout or route discovery. After more than a few iterations of recreation I determined the idleness time as to be 2 seconds. Somewhere idleness time refers to the period where a node does not obtain any one of the key packets.

In my original approach directly stop hello request messages after a period of idleness I experiential that the packet delivery ratio was reduced. On supplementary analysis also came to the winding up that it is best if the hello request frequency is first reduced to half instead of directly stopping hello request broadcasting.

Due to the subsequent reasons:

1. The node still be take part in the route but haven't received a data packet due to overcrowding of the upstream nodes.
2. A link breakage might occur in the upstream node in case the hello request becomes more vital as they provide a method to for neighboring update.

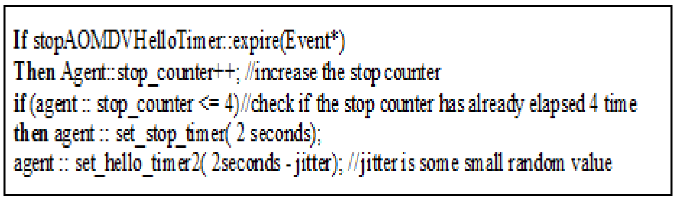
To apply the changeable hello request interval process prepared to use one of the timers called stop_hello_timer (sh_timer), it can take two values like 2 seconds or 0.5 seconds, and a counter named stop_counter used to maintain, how many times the stop request counter has been expired. The subsequent sections will explain the functioning of the algorithm all along with the counter and the timer usage.

C. *STEP 1: HELLO FREQUENCY DECREASED BY HALF*

In the establishment the sh_counter is set to zero and the sh_timer is set to expire after 2 seconds. If the node is contribute continuously in the network it means it has received one of the key packets in recent past. To ensure that the hello frequency is not affected when the node is participating in the network whenever a node receives one of the key packets it resets the value for sh_timer and AOMDV_sendrequest timer is not altered.

To understand the working for step1 considers a node 'X' which is at present participating in the network as an intermediary node forwarding data packets from source to destination. At this position, when a node is dynamically participating in the network it is undesirable to modify or to reduce the hello request frequency. To ensure that the hello frequency is unaltered when a node receives a key packet it resets the sh_timer counter to 0 and sets the sh_timer to expire after two seconds.

Due to this resetting of sh_timer to expire after two seconds it is established that the hello message frequency will be affected only after the node observes a two second inactivity period. At this point consider the node 'X' moving out from its current location to a new location. Where it is no longer participating in the network. In this position since the node 'X' is not in transmission range of any other node which is involved in data transmission therefore it cannot receive any of the key packets. Now if the node stays in such a situation for longer than 2 seconds the sh_timer will expire which in turn will reset the hello_timer to expire after 2 seconds minus some jitter. The introduction of jitter is done here to avoid the ambiguous condition where hello broadcasting is stopped due to subsequent expiry of sh_timer and its coincidence with sending hello message, because if the sh_timer code is called first by the scheduler then it will in turn reset the timer when next hello message is supposed to be transmitted.



```
If stopAOMDVHelloTimer::expire(Event*)
Then Agent::stop_counter++; //increase the stop counter
if (agent :: stop_counter <= 4)//check if the stop counter has already elapsed 4 time
then agent :: set_stop_timer( 2 seconds);
agent :: set_hello_timer2( 2seconds - jitter); //jitter is some small random value
```

Table: 1 Showing Pseudo-Code for Reduction of Hello Frequency by Half and sh_timer Expiry

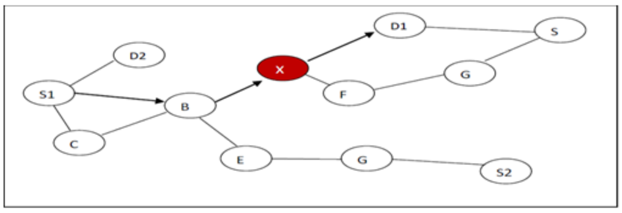


Figure:3. Showing node "X" currently participating in the network forwarding data packets from source "S1" to destination "D1"

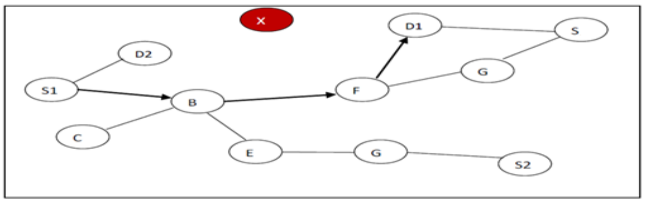
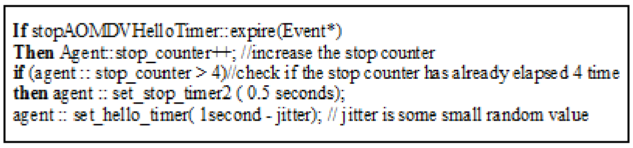


Figure:4 Showing movement of node X to its new position due to mobility

D. STEP 2: STOPPING HELLO MESSAGE FREQUENCY

In this step the node entirely stops hello message broadcasting. As mentioned earlier after each two seconds if the node not receiving any one of the concerned key packets the node increase the value of sh_counter by one. If a particular node pursues this iteration for four times which is corresponding to not receiving any of the 4 packets for eight continuous seconds, the node entirely stops the hello request message.

Ongoing the conversation on the example mentioned in the above section if node 'X' continues to be in the position then the node entirely stops broadcasting hello request packets. To do this it sets the time of next hello transmission after two seconds and sets the sh_timer to expire after 0.5 seconds accordingly if the node continues to not receive any of the key packets the sh_timer every time expires before the send_hello event is called thereby delaying it for another two seconds. The pseudo code for completely stopping hello message broadcasting after 8 seconds is shown.



```
If stopAOMDVHelloTimer::expire(Event*)
Then Agent::stop_counter++; //increase the stop counter
if (agent :: stop_counter > 4)//check if the stop counter has already elapsed 4 time
then agent :: set_stop_timer2 ( 0.5 seconds);
agent :: set_hello_timer( 1second - jitter); // jitter is some small random value
```

Table:2 Showing Pseudo-Code for Stopping Hello Message Broadcasting

E. STEP 3: RESUMING BROADCASTING OF HELLO MESSAGES

At this step consider the situation where node 'X' again decides to move to a new position as shown in Figure 5 and at the same time the S2 decided to start another route discovery for destination D2. Thus the RREQ packet is in turn heard by node 'X' .

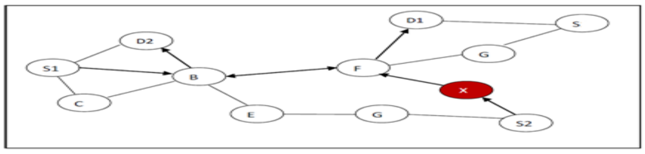
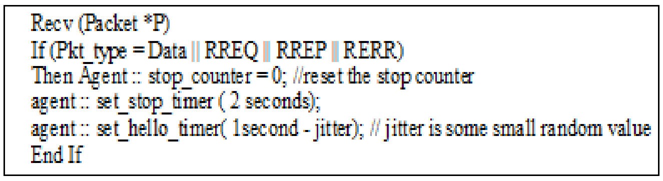


Figure:5. Showing the movement of node X to its new position

As soon as the node 'X' hears the RREQ packet and it resets the value of sh_count to 0 and sets the hello timer to expire after 2 seconds and the next send_hello event after 1 second. Thereby resuming broadcasting of hello messages as soon as it receives a key packet.
The implementation details of various timers involved and their values. The pseudo code for receiving a RREQ packet and resetting the counter and timers is shown in table 5.3.



```
Recv (Packet *P)
If (Pkt_type = Data || RREQ || RREP || RERR)
Then Agent :: stop_counter = 0; //reset the stop counter
agent :: set_stop_timer ( 2 seconds);
agent :: set_hello_timer( 1second - jitter); // jitter is some small random value
End If
```

Table:3. Showing Pseudo-Code for Resuming Transmission of Hello Messages with Normal Frequency

IV. PRE EMPTIVE MEASURE OF ROUTE DISCOVERY

The major characteristics of MANET are mobility and even with all the complicated schemes to avoid link breakage, routes are bound to break now and then. Enhanced-AOMDV also proposes such preventive measures to avoid route breakage like local update and continuous path strengthening. But it doesn't suggest any preemptive measure how to deal with route breakage. In this chapter, I proposed a pre-emptive measure to deal with link breakage problem so that the packet delivery ratio can be improved. In Enhanced-AOMDV and AOMDV whenever a link breakage occurs a RERR is generated which is transmitted to the upstream nodes, this RERR contains information about the destinations affected by this link breakage.

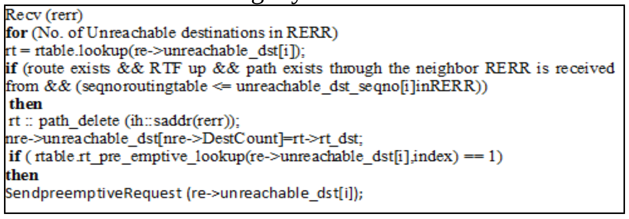
Each node on receiving such a RERR packet browses through it routing table to find destination, which will be affected by the link breakage. Once a node receives a RERR packet it browses through its routing table to find out which destination uses the downstream node as intermediate node and if no alternative roués are left for that destination then it removes the routing entry and pushes the information in the new RERR packet which in turn is send to its upstream node. This process continues until the RERR packet reaches the source, which also does the same routine and then tries to find if there is an alternative route to this destination.

If not then it starts a fresh route discovery wherein it sends out a fresh RREQ packet and until the new route is established all the packets coming to the routing layer are simply dropped. In my effort and try to reduce the number of packet drops and increase the packet delivery ratio by pre emptive discovering alternative route to the destination.

A. PRE EMPTIVE MEASURE OF ROUTE DISCOVERY SOLUTION:

As in AOMDV, Enhanced-AOMDV also discovers multiple routes towards the destination, it also adds to intelligence of selecting routes by choosing routes based on RSSI rather on just using hop count. Enhanced-AOMDV uses one route after the other until it is exhausted of all routes when it again initiates a new route discovery.

In my approach I propose to start route discovery pre-emptively when one route is left in the routing table instead of when there are no routes are left, so as to reduce the number of packets dropped due to no route towards destination. For doing the same I introduce a new field in the routing table structure which indicates the source id on a particular route. This field is updated whenever a data packet is received on the routing layer.



```
Recv (rerr)
for (No. of Unreachable destinations in RERR)
rt = rtable.lookup(re->unreachable_dst[i]);
if (route exists && RTF up && path exists through the neighbor RERR is received
from && (seqnoroutingtable <= unreachable_dst_seqno[i]inRERR))
 then
 rt :: path_delete (ih::saddr(rerr));
nre->unreachable_dst[nre->DestCount]=rt->rt_dst;
 if ( rtable.rt_pre_emptive_lookup(re->unreachable_dst[i],index) == 1)
then
SendpreemptiveRequest (re->unreachable_dst[i]);
```

Table:4 Showing Pseudo Code of Handling Route Error

Whenever a RERR packet is received at the node it first calculates the routes, which uses the source of RERR as next hop for a destination. It then looks up in the routing table to verify whether it itself is a source for this destination after creating and sending a RERR packet to its upstream node and deleting the particular path being affected. Once it has successfully identified that it is a source for a destination, which is being affected by this link breakage it then looks up the routing table to find out how many alternative paths are left for this particular destination.

It is worth mentioning here that a node has one routing entry for each destination but each routing entry may have multiple path entries. Now if calculated the number of paths left for a particular destination after deleting the affected path is just 1 it goes into pre-emptive route discovery mode and sends out a RREQ packet for new route discovery.

B. ROUTE COUPLING PROBLEM

This is a peculiar problem associated with pre-emptive route discovery. In order to analyze this problem consider the situation.

In this case consider node ‘SN’ is the source of data packets for node ‘DN’ which has two paths currently in its routing table for destination ‘DN’. Currently source ‘SN’ is transmitting data packet on path SN-N1-N2-DN. Now if suppose that node N2 decides to move to its new position. Due to movement of node N2 the link between N1 and N2 will break which will be sensed by N1 immediately N1 will generate a RERR packet and send it towards its upstream node.

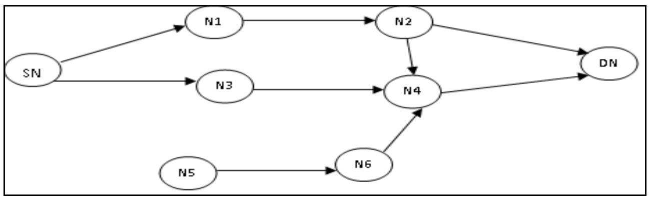


Figure:6. Showing two paths SN’s routing

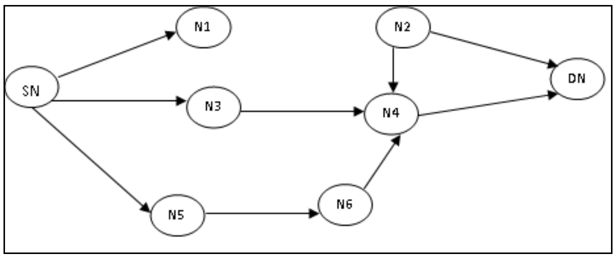


Figure:7. Showing N2 in its new position

Source ‘SN’ on receiving this RERR packet will remove this path from its routing entry and eventually will realize that there is only one path left for destination ‘DN’ in its routing table. This situation will now trigger the pre-emptive route discovery and hence node ‘SN’ will now broadcast a RREQ. For simplicity sake let consider the case where only node N5 will hear this RREQ node N5 since not having any routing entry for the destination will add a reverse route entry towards the source and then will rebroadcast this RREQ. At this point the rebroadcasted RREQ will be heard by node N4. One receiving this broadcast RREQ N4 will check its routing table for a valid route entry towards the destination and it will find one as well, so it in turn will send back an RREP back to N5 hence establishing a route SN-N5-N4-DN. This replying of RREP by N4 leads to a situation will is called route coupling where the paths merge with each other thereby violating the necessary node/link disjoint requirement of AOMDV and Enhanced-AOMDV.

Source ‘SN’ on receiving this RERR packet will remove this path from its routing entry and eventually will realize that there is only one path left for destination ‘DN’ in its routing table. This situation will now trigger the pre-emptive route discovery and hence node ‘SN’ will now broadcast a RREQ. For simplicity sake let consider the case where only node N5 will hear this RREQ node N5 since not having any routing entry for the destination will add a reverse route entry towards the source and then will rebroadcast this RREQ. At this point the rebroadcasted RREQ will be heard by node N4. One receiving this broadcast RREQ N4 will check its routing table for a valid route entry towards the destination and it will find one as well, so it in turn will send back an RREP back to N5 hence establishing a route SN-N5-N4-DN. This replying of RREP by N4 leads to a situation will is called route coupling where the paths merge with each other thereby violating the necessary node/link disjoint requirement of AOMDV and Enhanced-AOMDV.

C. SOLUTION TO ROUTE COUPLING PROBLEM

To deal and avoid such a situation I introduce and extra field in the RREQ packet called pre_emptive_rreq field which take one of two value 0 or 1. The default value is 0 which indicates that the RREQ is non-premptive type. This field is set to 1 when a node decides to send out a pre-emptive

RREQ. Thus in my proposed solution when node N4 receives a RREQ it first checks to see if the pre_emptive_rreq field is set to 1,if this filed is set to 1 then it simply drops this RREQ packet even though it has a valid route entry for the destination.

```
preemprt = rtable.lookup (rq->rq_dst);
if((preemprt&&(RREQ::src==preemprt->src_of_data_packets)&&
                        (RREQ:: re_emptive_type==1))
then Packet :: free(p);
Else
Recv (RREQ)
```

Table:5. Showing Pseudo Code for Receiving a Pre-Emptive Type RREQ

The RREQ eventually reaches the destination and the destination in turn sends out a RREP which is received by the source thereby establishing the route pre-emptively.

V. SIMULATION ENVIRONMENT

I uses network simulator NS-2.34 an object oriented and discrete event simulator to evaluate and compare routing protocols. “Wireless channel” is used as the channel type along with “two Way ground” radio propagation model.

Parameters	Value
Network size	1000m X1000 m
Number of nodes	30
Transmission range	250 m
Propagation model	Two Ray Ground
Traffic type	CBR (constant bit rate)
Packet size	512 bytes
Simulation time	1000 seconds
MAC protocol	802.11
Link bandwidth	2Mb/s
InterfaceQueue (IFQ)	Droptail/PriQueue length=100 packets
Mobility Model	Random Waypoint Model

Table :6 Simulation Environment

Parameter varied	**Constant parameter**
Node speed (1- 40 m/sec)	Number of nodes =30 Packet rate =1 pkt/sec
Packet rate (0.25 – 8 packets /sec)	Number of nodes =30 Node speed = 15 m/sec
Number of nodes (5- 30 conns)	Node speed = 15 m/sec Paket rate = 1 pkt/sec

Table:7 Simulation Parameters Used

A. *PACKET DELIVERY RATIO WITH VARYING NODE SPEED:*

The enhanced version of AOMDV shows about 2-5% advantage in terms of packet delivery ratio as shown in Figure 6.1, which is perhaps the most crucial measure of a routing protocol. Our enhanced_AOMDV version shows a consistent behavior with increasing node speed as it tries to find paths pre_emptively before AOMDV is exhausted of all paths in its routing table. It shows a little more advantage at higher average node speeds of about 30-35 m/s than lower node speeds of 15- 20 m/s as more route breakages are expected to take place at higher node speeds.

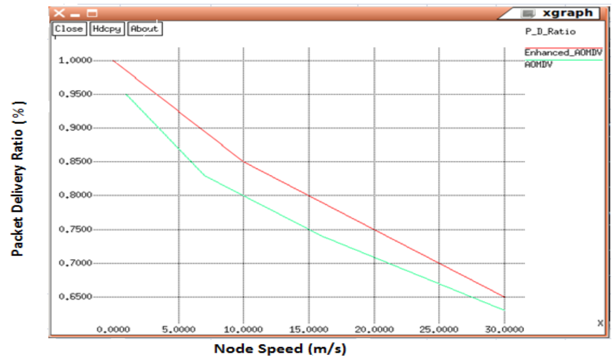


Figure:8 Packet Delivery Ratio with average varying Node Speed

B. *PACKETS DROPPED DUE TO NO-ROUTE WITH VARYING NODE SPEED*

This is perhaps a better suited performance index for a routing protocols coz it provides a better index of a routing protocol performance in terms of finding routes. Simply understood it is a measure of the number of packets dropped at the routing layer due to non-availability of routes to the destination. As clearly seen in Figure: .6.2, our proposed Enhanced-AOMDV has clearly more advantage than AOMDV in terms of packets dropped at the routing layer also the protocol shows more resilience to packet drops at higher speeds.

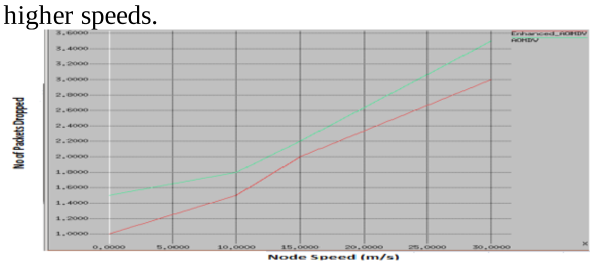
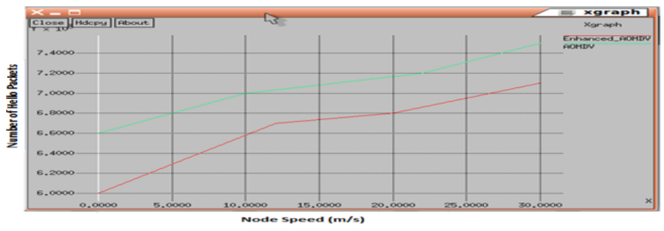


Figure:9 Packet dropped due to no route with varying average node speed.

B. *ROUTING OVERHEAD WITH VARYING NODE SPEED*

I calculate routing overhead in terms of the number of hello messages transmitted and in terms of normalized routing overhead. Normalized routing overhead can be understood as the number of routing packets transmitted for each successfully received Data packet at the destination. As shown in Figure: 6.3 (A) the total number of hello messages transmitted for Enhanced_AOMDV is much lesser than AOMDV especially in lower node speed conditions as due to lesser node movements there is lesser chances isolated nodes to become intermediate nodes to other ongoing transmissions.



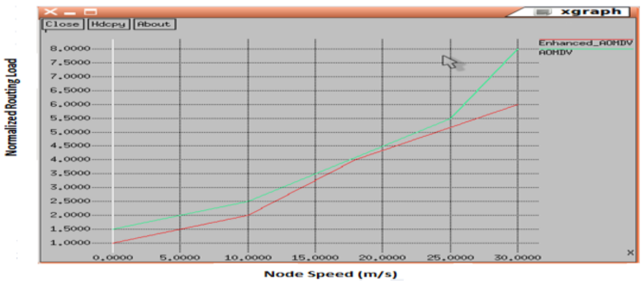


Figure:10 Overhead with Varying average node speed

Also the contribution of stop-hello mechanism is clearly visible in the normalized routing overhead calculation, moreover due to increase in the packet delivery ratio the normalized routing overhead is further helped, as the number of successful packets delivered to the destination is more than AOMDV. The normalized routing overhead is also contributed by the number of RREQ's that are transmitted by the node, as I don't allow nodes which have an active route for the same source destination pair and the nodes which have an active route to the source to retransmit RREQ's thereby helping the routing overhead to reduce further.

C. *PACKET DELIVERY RATIO AND PACKET DROPPED DUE TO NO-ROUTES*

Enhanced_AOMDV shows a little improvement in terms of packet delivery ratio when the numbers of connections are varied. Although the improvement is more when the numbers of connections are more indicating route breakage due to congestion. In Figure 6.4 (b) shows that the difference in number of packets dropped due to no route at higher number of connections is higher, this is because when there is route breakage due to congestion pre-emptive route tries to avoid those nodes which are already serving as intermediate nodes for the same source destination pair.

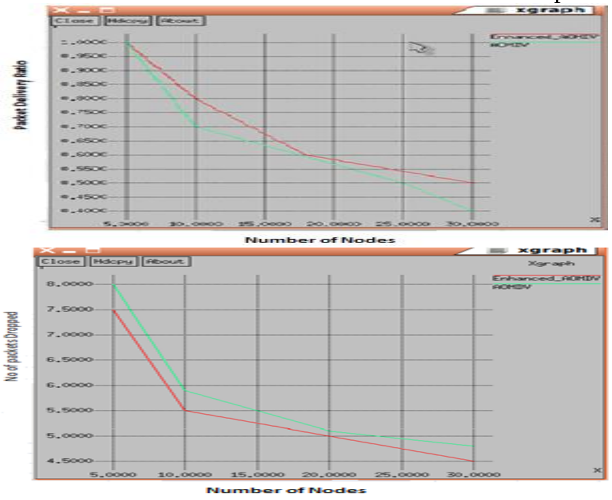


Figure:11 Packet Delivery and Drops with varying number of Connections

D. *ROUTING OVERHEAD WITH VARYING NUMBER OF CONNECTIONS*

Figure:12 suggest that the routing overhead for Enhanced_AOMDV is lower than AOMDV both in terms of number of hello messages broadcasted and normalized hello messages. The number of hello messages are greatly reduced at lower number of connections as lesser number of nodes are involved in data transfer by virtue of the stop hello algorithm these nodes don't broadcast hello messages the difference slightly narrows down at higher number of connections.

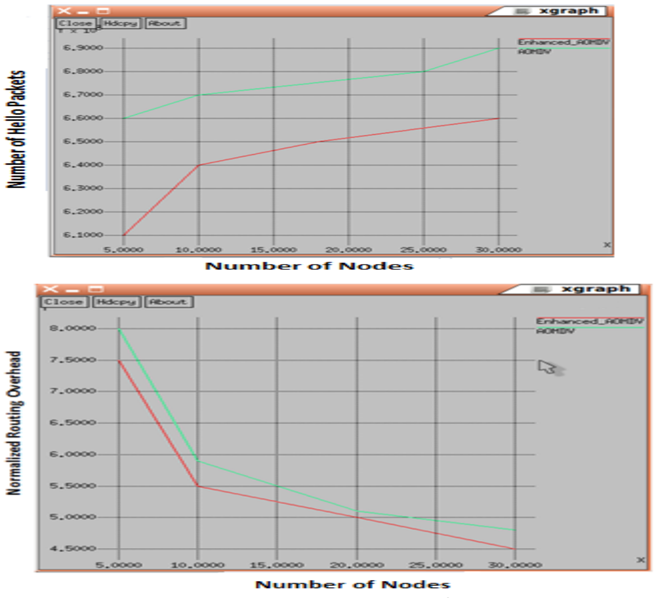


Figure:12 Overhead with Number of connections

E. *PACKET DELIVERY RATIO AND PACKETS DROPPED DUE TO NO-ROUTE*

Figure:13 shows the performance of both routing schemes in terms of varying packet rates. Both protocols perform similar when the packet rates are very low while they start showing difference from 4 Pkt / sec. At 4 Pkt / sec the 1000 lesser packets are dropped by Enhanced_AOMDV while the difference increases to about 3500 at 8 packets per second. I see the tread rising first and then falling down from 2 packets/sec the route timeout values can be accountable for the increasing trend when the packet rate s very small the routes timeout very easily and hence the packet delivery rate is smaller and similarly more than 2 packets per second congestion increases and more number of packets are dropped at the MAC layer.

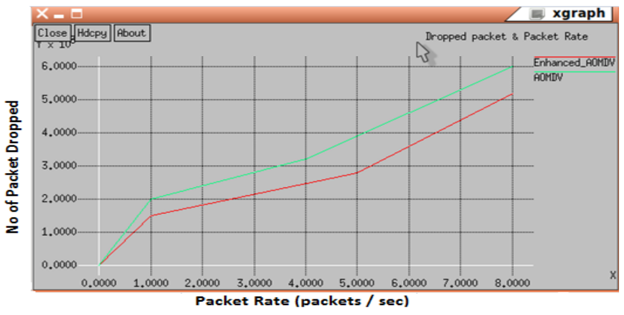
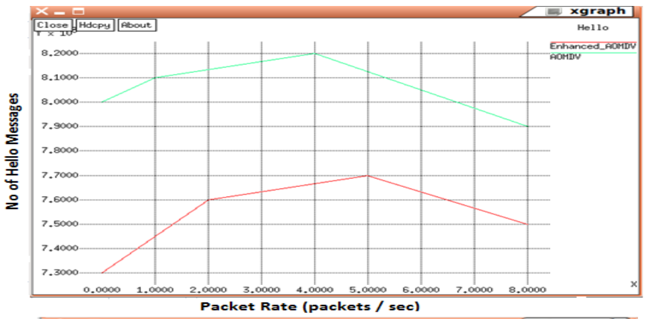


Figure:13 Packet delivery and no-route packet drops with varying packet rate.

F. *LATENCY: OVERALL RESULTS*

In this section I discuss more about latency metric measure for both routing protocols under consideration. For measuring latency I take into consideration two types of latency values firstly End-to-End data packet latency, which is the time delay experienced by the data packets to reach the routing layer at the destination from the instant they reach source's routing layer. Secondly, Route-discovery latency, which is the time take to discover a route measured as the time difference between the sending of RREQ and receiving of a RREP.



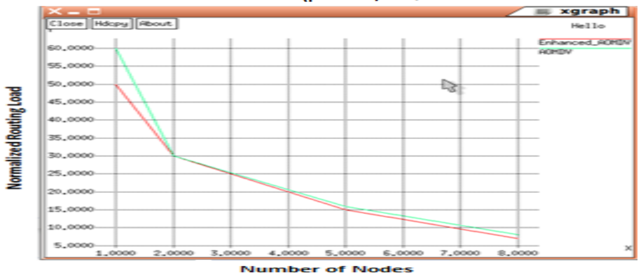
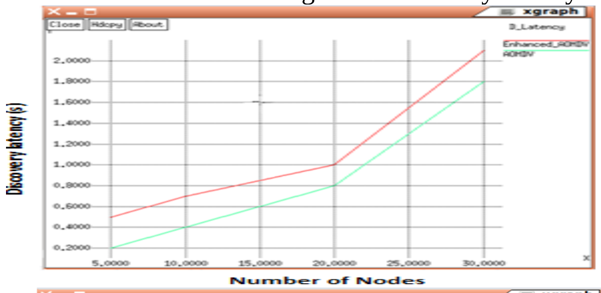


Figure:14 Packet delivery and no-route packet drops with varying packet rate

Figure:15 shows the comparison of both routing schemes in terms of route discovery latency. Enhanced_AOMDV shows higher route discovery latency than AOMDV. A mojor reason for this increased route discovery latency is that I don't allow the nodes that are currently acting as intermediate nodes for an ongoing transmission between the same source destination pair. Also I don't allow the nodes, which have a current route to the source with routing flag as up or simply understood nodes, which have an active route to the source to retransmit the RREQ's so as not to disturb the current transmissions.

Due to the same reason the RREQ's that are transmitted preemptively generally take a longer to the destination, which accounts for the increased delay as even those RREQ's which are transmitted preemptively as also considered while calculating Route discovery latency.



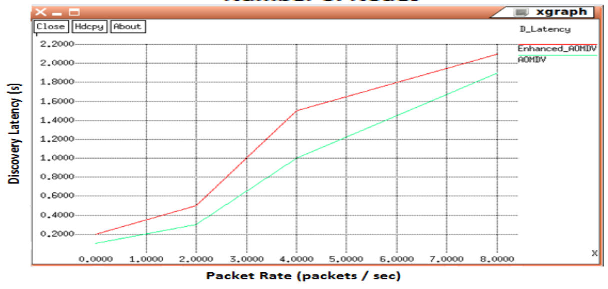


Figure:15. Route discovery latency: Overall results

G. Route Discovery Frequency

In all Figures:16 I can see that Enhanced_AOMDV has higher route discovery frequency than AOMDV, this is due to the fact that Enhanced_AOMDV always tries to maintain at least 2 paths in its routing table for each active destination. As whenever it receives a RERR and it realizes that it has only one route left in its routing table then it initiates another route discovery while AOMDV waits until the last path is broken as well and then it initiates a route discovery.

AOMDV has higher route discovery frequency than AOMDV, this is due to the fact that Enhanced_AOMDV always tries to maintain at least 2 paths in its routing table for each active destination. As whenever it receives a RERR and it realizes that it has only one route left in its routing table then it initiates another route discovery while AOMDV waits until the last path is broken as well and then it initiates a route discovery.

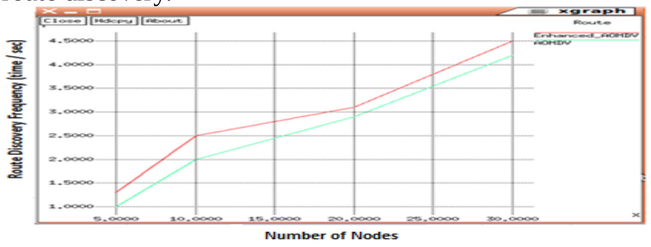


Figure:16. Route discovery frequency

VI. CONCLUSION

A MANET is characterized by existence of network without any centralized control where there nodes can move randomly and may have varying capacity in terms of transmission range, battery life, processing power etc. Nodes in such a network communicate with each other wirelessly, and due to the random nature of movement the links between them are unpredictable and often break due to same reason. A good routing protocol should pay attention to multipath capabilities, should be adaptable with the changing conditions and should have a low overhead.

In this thesis I proposed two routing mechanism which try to reduce the unnecessary overhead and increase the throughput. I introduce the concept of reducing and stopping hello message broadcasting based on key packet and inactivity period concept. The hello messages are stopped from being broadcasting as to reduce the contribution of hello messages in total routing overhead. I suggest a method to reduce the number of packets being dropped in routing layer due to no route availability by using preemptive routing.

The comparative analysis is done in terms of performance metrics of Packet Loss Rate, End-to-End delay, Route Discovery delay, Route Discovery Frequency, Routing Overhead and packets dropped due to no route availability.

REFERENCES

[1] Larry C. Llewellyn, Kenneth M. Hopkinson, Scott R. Graham, "Distributed Fault-Tolerant Quality of Wireless Networks", IEEE Transactions On Mobile Computing, Volume.10, Number.2, February 2011.

[2] S. Singh, N.Rajpal and A.K.Sharma, " K-fault tolerant in Mobile Ad Hoc Network under cost constraint", 3rd International Conference on Electronics Computer Technology (ICECT) , Volume -6 ,Page(s) 368 - 372 , April 2011.

[3] Jipeng Zhou and Chao Xia, "A Location-Based Fault-Tolerant Routing Algorithm for Mobile Ad Hoc Networks ", WRI International Conference on Communications and Mobile Computing, Volume 2 ,Page(s) 92 – 96,Jan 2009.
[4] R. Melamed, I. Keidar and Y. Bare, "Octopus: a fault-tolerant and efficient ad-hoc routing protocol ", 24th IEEE Symposium on Reliable Distributed Systems, Page(s) 39 - 49 ,OCT 2005.
[5] O. Riganelli, R. Grosu, S.R. Das, C.R. Ramakrishna, and S.A. Smolka., " Power Optimization in Fault-Tolerant Mobile Ad Hoc Networks ",11th IEEE Conference on High Assurance Systems Engineering Symposium, Page(s): 362 - 370 ,Dec 2008.
[6] M. Khazaei and R. Berangi, "A multi-path routing protocol with fault tolerance in mobile ad hoc networks " , 14th International CSI Computer Conference, Page(s): 77 – 82, Oct. 2009.
[7] Rabindra Kumar Shial, K. Hemant Ku Reddy and K. L. Narayana, "An Optimal RPC Based Approach to Increase Fault in Wireless Ad-Hoc Network", Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 84, pp 372-382, 2012.
[8] S. Chandrasekaran, S. Udhayakumar, U. Mohan Bharathy,and D. Jitendra Kumar Jain, " Trusted Fault Tolerant Model of MANET with Data Recovery ",4th IEEE International Conference on Intelligent Networks and Intelligent Systems(ICINIS), Page(s):21-24, 2011.
[9] R.E.N. Moraes, C.C. Ribeiro and C. Duhamel, "Optimal solutions for fault-tolerant topology control in wireless ad hoc networks " IEEE Transactions on Wireless Communications, Volume:8 , Issue: 12 ,Page(s): 5970 - 5981 ,December 2009.
[10] Yang Qin and Kong Ling Pang, " A Fault-Tolerance Cluster Head Based Routing Protocol for Ad Hoc Networks", IEEE Vehicular Technology Conference, Page(s): 2472 – 2476, May 2008.
[11] Sun.Ruozi ,Wang.Yue, Yuan.Jian, Shan.Xiuming and Ren.Yong, "Topology control algorithm using fault-tolerant 1-spanner for wireless ad hoc networks Tsinghua Science and Technology", Volume: 17, Issue: 2, Page(s): 186 -193 , April 2012.
[12] Larry C. Llewellyn, Kenneth M. Hopkinson and Scott R. Graham, " Distributed Fault-Tolerant Quality of Wireless Networks" IEEE Transactions On Mobile Computing, VoL. 10, No. 2, February 2011.
[13] Bernd Thallner, Heinrich Moser and Ulrich Schmid ,"Topology control for fault-tolerant communication in wireless ad hoc networks ", Wireless Networks, Volume 16, Issue 2, pp 387-404 , February 2010.
[14] B.J Oommen and S. Misra., "A Fault-Tolerant Routing Algorithm for Mobile Ad Hoc Networks Using a Stochastic Learning-Based Weak Estimation Procedure", IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Page(s): 31 - 37 ,June 2006. [15] Sudip Misra, P. Venkata Krishna, Akhil Bhiwal, Amardeep Singh Chawla, Bernd E. Wolfinger and Changhoon Lee , "A learning automata-based fault-tolerant routing algorithm for mobile ad hoc networks", The Journal of Supercomputing , Volume 62, Issue 1, Page(s) 4-23, October 2012.
[16] B.John Oommen and Sudip Misra ,"Fault-tolerant routing in adversarial mobile ad hoc networks: an efficient route estimation scheme for non-stationary environments", Telecommunication Systems,Volume 44, Page(s) 159-169, June 2010.
[17] R.K. Sahu, R.Saha.R and N.S. Chaudhari, "Fault Tolerant Reliable Multipath Routing Protocol for Ad Hoc Network ", Fourth International Conference on Computational Intelligence and Communication Networks , Page(s) 117-121, Nov 2012.
[18] Giuseppe Anastasi, Alberto Bartoli and Flaminia L. Luccio ," Fault-Tolerant Support for Reliable Multicast in Mobile Wireless Systems: Design and Evaluation", Wireless Networks , Volume 10, Issue 3, pp 259-269, May 2004.
[19] Md.Ehtesamul Haque and Ashikur Rahman , "Fault Tolerant Interference-Aware Topology Control for Ad Hoc Wireless Networks", Ad-hoc, Mobile, and Wireless Networks ,Lecture Notes in Computer Science Volume 68, pp 100-116,2011.
[20] L. Demoracski," Fault-tolerant beacon vector routing for mobile ad hoc networks ", 19th IEEE International Parallel and Distributed Processing Symposium, April 2005.
[21] Sheng Xu , S.Papavassiliou and L. Zakrevski, " Fault-tolerant cluster-based routing approach in wireless mobile ad hoc networks", IEEE VTS 54th Conference on Vehicular Technology, Volume: 4 , Page(s): 2613 – 2617, 2001.
[22] Yuan Xue and K. Nahrstedt, "Fault tolerant routing in mobile ad hoc networks", IEEE Conference on Wireless Communications and Networking, 2003, Volume: 2 ,Page(s): 1174 – 1179, March 2003.
[23] Ching-Hua Chuan and Sy-Yen Kuo, "Cache management of dynamic source routing for fault tolerance in mobile ad hoc networks", International Symposium on Dependable Computing, Page(s): 199 - 205 ,2001.
[24] F. Corradini, R. Gagliardi, M. Papalini, A. Polzonetti and O. Riganelli, "On the fault tolerance of mobile ad hoc networks", 30th International Conference on Information Technology Interfaces, Page(s): 827 – 832, June 2008.
[25] Rana E. Ahmed, "A Fault-Tolerant Routing Protocol for Mobile Ad Hoc Networks", Journal of Advances in Information Technology, VOL. 2, no.2, May 2011.
[26] M. Nazeeruddin, G. Parr and B. Scotney, "Fault - Tolerant Dynamic Host Auto-configuration Protocol for Heterogeneous MANET", 14th IST Mobile & Wireless Communication, 2005.
[27] R.S. Shaji, Dr.R.S. Rajesh and B. Ramakrishnan, "A Fault-tolerant Scheme for Routing Path Re-establishment for reliable communication in Heterogeneous Networks", International Journal of Scientific & Engineering Research, Volume 1, Issue 3, ISSN 2229-5518 , December-2010.
[28] Swati Saxena and Dr. Madhavi Sinha, "Statistical study of performance metrics of Adaptive Fault Tolerant Replication Routing Protocol for MANET", International Journal on Computer Science and Engineering (IJCSE), Vol. 4 No. 11 Nov 2012.
[29] R.S. Shaji, Dr.R.S. Rajesh and B. Ramakrishnan, "SFUSP: A Fault-tolerant Routing Scheme for path

establishment among Mobile Devices in Pervasive Spaces", Journal of Computing, Volume-2, Issue 11, ISSN 2151 – 9617, Nov.2010.

[30] Adeluyi, O. Jeong-A Lee, "SMiRA: A bio-inspired fault tolerant routing algorithm for MANETs", International Conference on ICT convergence (ICTC), Page(s): 78 – 84, Oct.2012.

All Rights Reserved © 2013 IJARCET