# AODV Enhancement Based on the Minimization of Route-Request Packets

Puja Rani and G.P. Biswas

Department of Computer Science & Engineering
Indian School of Mines (ISM), Dhanbad – 826004, Jharkhand, India
{puja20.rani,gpbiswas}@gmail.com

**Abstract.** A MANET is a self-configuring network of mobile devices connected by wireless links. Reactive Routing protocols for MANETs flood RREQ control packets to discover and establish route between the source-destination pairs. This situation becomes worse and degrades the network performance whenever several connections need to be established simultaneously. This paper proposes a mechanism to minimize the route-establishment overhead in AODV by controlling the flooding of RREQ packets. In brief, each intermediate node counts the number of RREQ packets flooded by it and if the count exceeds a threshold, stops further flooding. On the other hand, an intermediate node, after getting a RREP packet within reverse-route-lifetime reduces the counter value; otherwise the reverse link in its cache is removed along with the reduction of the counter value. The proposed scheme has been simulated using ns-2 and compared with the original AODV and it has been found that it enhances network performance.

**Keywords:** MANETs, Proactive and Reactive or On-demand routing protocol, AODV protocol, M-AODV, Network simulator.

## 1 Introduction

Mobile Ad hoc Networks (MANET) [3, 6] form a class of dynamic multi-hop network consisting of a set of mobile nodes that intercommunicate on shared wireless channels. Each node in MANET can operate as a host as well as a Router. Design of an efficient routing protocol for MANETs has been proven to be a very challenging task. Routing protocols [10] proposed for ad hoc networks can be classified into two groups: Proactive and Reactive and a short description of them is addressed now.

The Proactive routing was the first attempt for designing routing protocols for MANETs. For instance, the Proactive protocols such as DSDV (Destination-Sequenced Distance Vector) [1] and GSR (Global State Routing) [12] were based on the traditional distance vector and link state algorithms. These protocols periodically maintain and distribute route information to all nodes within the network. The disadvantages of these strategies were their lack of exceedingly large overhead produced due to blind flooding. Blind flooding result in broadcast storm Problem [5] and is thus not efficient. The On-demand or Reactive routing protocols [10], on the other hand, are more efficient for routing in large ad hoc networks because they only maintain that route that is currently needed, initiating a path discovery process whenever a route is needed for message transfer, such as the Lightweight Mobile Routing (LMR) protocol [15], Ad-hoc On Demand Distance Vector Routing (AODV) [2,5], Temporally-Ordered Routing Algorithms (TORA) [14], and Dynamic Source Routing Protocol (DSR) [4] and ABR [13]. In AODV [2,5], the routing table at the nodes caches the next hop router information for a destination and use it as long as the next hop router remains active (originates or relays at least one packet for that destination within a specified time-out period). The DSR [4] is a source-based routing that identifies all the intermediate nodes to be included in the packet header.

When a number of connections need to be established simultaneously, the performance of the On-demand routing protocols is decreased because very few connections get established, due to enormous increase of RREQ/RREP packets. In this paper, we have proposed a technique to enhance the performance of AODV (any other Reactive protocol may be enhanced) by reducing the flooding of the control packets. In this case, each node maintains and updates a counter with respect to a threshold value such that it increases the counter for each flooding of new RREQ packet and stops the same when the counting reaches the *RREQ_Threshold* value, and it decreases the counter when a RREP packet is received or the reverse route entry is deleted from the routing table when the RREP is not received during reverse-route lifetime. The *RREQ_Threshold* value is selected by simulation such that the proposed modification of AODV ensures the establishment of the routes. The proposed scheme allows each node to forward only a limited number of RREQ packets, before getting any RREP packet or within a specified time-out period, thus avoid the network congestion during route discovery phase.

Rest of the paper is organized as follows. The proposed scheme and its implementation are given in section 2. The comparison of the proposed scheme with AODV is presented in section 3. Finally, the conclusion of the paper is given in section 4.

## 2 Proposed Modification of AODV for Minimization of RREQ/RREP Packets

In this section, a scheme has been proposed for minor modifications of AODV such that a large reduction of the RREQ overhead is possible during route discovery process of AODV if several routes are needed to be established simultaneously. For this, each node in the proposed scheme is allowed to broadcast only a limited number of RREQ packets, and for which each node maintains a counter and a predefined threshold, called *RREQ_Threshold*, beyond which no RREQ packet will be transmitted by a node of the network, before getting any RREP packet or within a specified timeout period. The detail implementation of the proposed modifications is addressed now.

## 2.1 Route Discovery in Proposed M-AODV

At the beginning the counter of each node is initialized to zero and it is updated as follows:

1) **Counter Increment**: If the counter value does not exceed *RREQ-threshold*, a node broadcasts the received RREQ packet to its neighboring nodes and increments the counter value. It also sets a timer for the reverse link expiry time, called REV_ROUTE_LIFE and a reverse-link in its routing table to a node from which a RREQ packet is received. Otherwise, instead of broadcasting, it starts dropping the received RREQ packets.
2) **Counter Decrement**: A node decrements the counter value whenever it receives RREP packet within REV_ROUTE_LIFE corresponding to a RREQ packet forwarded earlier. On the other hand, if it does not receive RREP within REV_ROUTE_LIFE, in addition to the decrement of the counter, the reverse-link entry is deleted from its routing table.

The algorithmic steps of the proposed modification are given below:

Step-1: Whenever a source node, say *S*, wants to communicate with a destination node, say *D*, it broadcasts the RREQ packet to all of its neighbors, provided the route to *D* is not known to *S* previously.

Step-2: Upon receiving the RREQ packet, each node checks whether it had received this RREQ packet before. If the RREQ packet is received before, the node discards packet and remains silent, else it executes the following operations:
(a) The node establishes a reverse link to the node from which the RREQ is received.
(b) It checks whether the node is the *D* itself or it has a fresh enough route to the *D*. If anyone of the above conditions is true, the node responds by unicasting a RREP back to the source node *S* using the reverse path established, otherwise it compares counter value, say C-Value, with the RREQ-threshold and executes any of the following cases:

Case-1: If *C-Value* >= *RREQ_Threshold*, the node rejects the RREQ packet, i.e., the RREQ packet is not broadcasted.
Case 2: If *C-Value* < *RREQ_Threshold*, the node increments its counter value, *C-Value* by 1 and broadcasts the RREQ packets to all of its subsequent neighbors.

Step-3: Upon receiving the RREP packet within REV_ROUTE_LIFE, each node checks whether it has received this RREP before or not and if so, it discards the RREP packet and remains silent, else it continues with functions given below.
(a) It establishes a forward link to the node from which the RREP is received.

(b) It checks whether it is the *S* itself or not and if so, the node starts sending the data packets using the route just discovered, else it decrements *C-Value* and forwards the RREP towards *S* using the reverse link.

Step-4: If a RREP is not received within REV_ROUTE_LIFE, a node deletes the reverse route from its routing table and decrements the *C-Value*.

## 2.2 Implementation of the Proposed M-AODV

As stated earlier, our proposed scheme makes some modifications in the route discovery process of AODV that reduce connection establishment overhead when a number of routes need to be established simultaneously. Each node in the proposed scheme keeps following information:

- A **routing table:** Each entry (*rt*) of the routing table has following fields: [Destination address (*dst*), Next hop address (*next_hop*), Destination sequence number (*rt_seqno*), Hop count (*hops*), Route type (*rt_type*)], where

$$rt\_type = \begin{cases} 1 & \text{for reverse link} \\ 0 & \text{for forward link} \end{cases}$$

  The *rt_type* is a newly added field in routing table and it is used for decrementing the counter value at a node which forwards the RREQ packet, but does not receive any RREP packet within REV_ROUTE_LIFE.
- A **Counter** (*C-Value)*: The *C-Value* counts the number of RREQ packets that are forwarded by a node to its neighboring nodes. *C-value* is initialized to 0 for every node.
- A **RREQ Threshold (RREQ-Threshold*)***: The RREQ-Threshold is the maximum number of RREQ packets that can be broadcasted by a node, before getting any corresponding RREP or within a specified timeout period.
- **A broadcast id (*bid)*:-** it is initialized to 0 for every node.
- **A sequence number (*seqno*):-** it is initialized to 0 for every node.

The pseudo code for the proposed M-AODV (modified AODV) is given in Annexure 1.

## 2.3 Comparasion of RREQ/RREP Packets Injected in AODV and Proposed M-AODV

Since each node in M-AODV is allowed to broadcast a limited number of RREQ packets, so the total number of RREQ/RREP control packets injected into the network is much less than the RREQ/RREP packets injected in AODV. An estimation of the number of control packets inserted in both AODV and M-AODV for establishing fixed number of connections and their comparison are given below.

According to [9], the average number of RREQ packets, say $N_{RREQ1}$, needed to be injected into the network to discover a single route in a network of diameter *E (h)* is

$$N_{RREQ1} = E(d) + E(d)^2 + E(d)^3 + \dots\dots\dots\dots + E(d)^{E(h)}$$

where *E (d)* is the average degree of a node.

Thus the total number of RREQ control packets injected into the network to set up *C (t)* routes is

$$N_{TOTAL\text{-}RREQ1} = C(t) \times N_{RREQ}$$
$$N_{TOTAL\text{-}RREQ1} = C(t) \times [E(d) + E(d)^2 + E(d)^3 + \dots\dots\dots\dots + E(d)^{E(h)}]$$

Although both RREQ and RREP packets create control overhead in the AODV, the major-part of this overhead is due to RREQ packets, because the destination node for each complete path, unicasts a single RREP towards the source node *S* in the route discovery phase of the protocol. Thus if *x* number of alternate routes per source-destination pair are identified and all are replied, then the total control overhead in AODV 'contrl-ovhd' is

$$\text{Contrl-ovhd-AODV} = N_{TOTAL\text{-}RREQ1} + C(t) \times x \tag{1}$$

Now the estimation of control packet injected in the network using proposed M-AODV is given below:

The number of RREQ packets broadcasted by the $i^{th}$ node, for $1 \le i \le n$, where *n* is number of nodes present in the network is:

$$N_{RREQ2} = (RREQ\_Threshold + x_i) \times E(d)$$

where $x_i$ is the additional RREQ packets over RREQ-Threshold broadcasted, due to receiving of RREP packets.

Total number of RREQ packets injected into the network is

$$N_{TOTAL-RREQ2} = \sum_{i=1}^{n} (RREQ\_Threshold + x_i) \times E(d)$$
$$= E(d) \times (n \times RREQ\_Threshold + \sum_{i=1}^{n} x_i)$$

Since $x_i \le$ RREQ_Threshold, then we have,

$$N_{TOTAL-RREQ2} \le n \times E(d) \times RREQ\_Threshold + n \times E(d) \times RREQ\_Threshold$$
$$\le 2n \times E(d) \times RREQ\_Threshold$$

If the control overhead due to RREP is included, then the total control overhead in M-AODV is

$$\text{Contrl-ovhd-M-AODV} \le N_{TOTAL\text{-}RREQ2} + C(t) \times x$$

Now the total reduction in the control overhead of the AODV and M-AODV can be calculated as

$N_{RED} = N_{TOTAL\text{-}RREQ1} - N_{TOTAL\text{-}RREQ2}$

$=(C\ (t) \times [E\ (d) + E\ (d)^{2} + \ldots\ldots.. + E\ (d)^{E\ (h)})] - (2n \times E\ (d) \times \text{RREQ\_Threshold})$

$=C\ (t) \times E\ (d) \times [1+ E\ (d) +\ldots\ldots... + E\ (d)^{E\ (h)-1}] - 2\text{n} \times E\ (d) \times \text{RREQ\_Threshold}$

$$= \frac{C(t) \times E(d) \times [E(d)^{E(h)} - 1]}{E(d) - 1} - 2n \times E(d) \times RREQ\_Threshold$$

$$\cong C(t) \times E(d)^{E(h)} - 2n \times E(d) \times RREQ\_Threshold$$

$$\cong E(d)[C(t) \times E(d)^{E(h)-1} - 2n \times RREQ\_Threshold]$$

Since $E(d)^{E(h)-1} \geq RREQ\_Threshold$, then $N_{RED} > 0$ if $C\ (t) > 2n$, where $n$ is the number of nodes in the network. Thus, the RREQ/RREQ control packets in AODV can be reduced to a large extent, if the number of connections to be set up is more than $2n$.

## 3 Simulation and Performance Analysis

Both the AODV and the proposed M-AODV has been simulated using NS-2 network simulator [7, 8] and their performance is reported in this section. The simulation results are measured in terms of two well known comparison metrics, called NRL (normalized routing load) that gives how many control packets are needed to deliver one data packet to the destination, and Percentage of PDF (packet delivery fraction). PDF is defined as the ratio of the total data packets delivered to the destination, to the total data packets sent by the source. The NRL and PDF have been obtained by varying the number of network nodes, number of connections to be established and the RREQ_Threshold used. The following simulation parameters have been considered in our simulation experiments:

| | | |
|---|---|---|
| Network topology size | : | 500 m × 500 m |
| Number of nodes | : | 50, 75 and 100 |
| Simulation time | : | 100 sec |
| Packet rate | : | 4 Packets/sec |
| Packet size | : | 512 bytes |
| RREQ-Threshold | : | 45, 90 and 150 |

The figure 1 and figure 2 show the NRL values required for a network of 50 nodes for different number of connections established in the AODV and the proposed

M-AODV respectively, where three threshold values for RREQ-Threshold = 45, 90 and 150 have been used in M-AODV. It can be seen that the NRL values required in AODV increase very rapidly with the increase of number of connections, whereas the NRL values in M-AODV do not increase with the increase of number of connections, i.e., a large number of connections can be established with almost a constant NRL value. Also the differences of NRL values in AODV and M-AODV are significant.

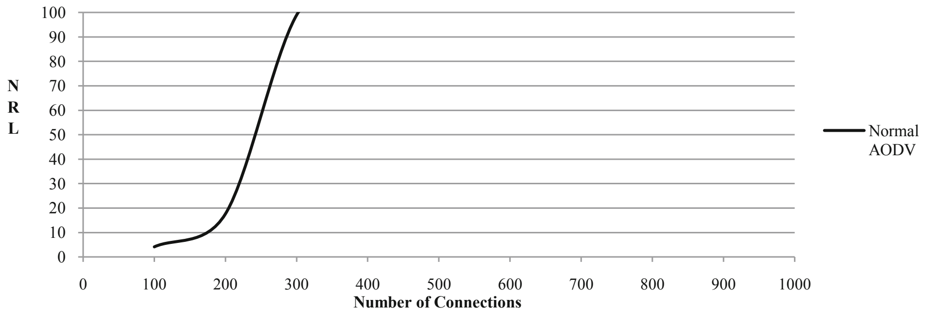

**Fig. 1.** NRL versus number of connections in AODV (Network with 50 nodes)

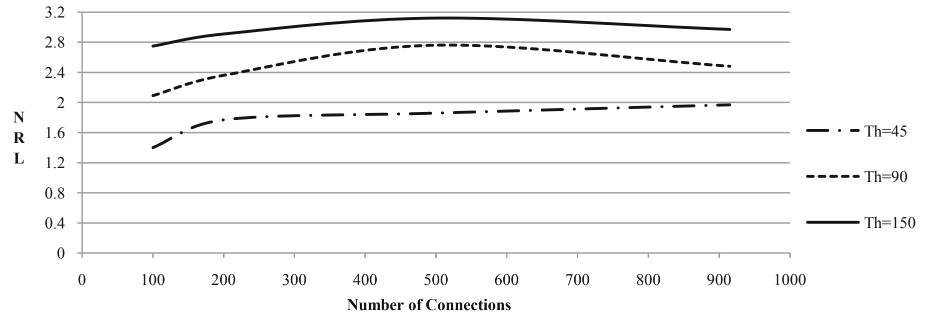

**Fig. 2.** NRL versus number of connections in M-AODV (Network with 50 nodes)

The PDF values versus number of connections in a network of 50 nodes are given in figure 3, which shows better PDF in M-AODV than the AODV. Also it shows that PDF values in the proposed M-AODV can be further increased by properly adjusting the RREQ-Threshold values.

Similar experiments have been carried out with the networks having 75 and 100 nodes and their results have been shown in figures 4-9. It can be seen that the proposed M-AODV improves both NRL and PDF significantly, and thus performs far better than the original AODV.

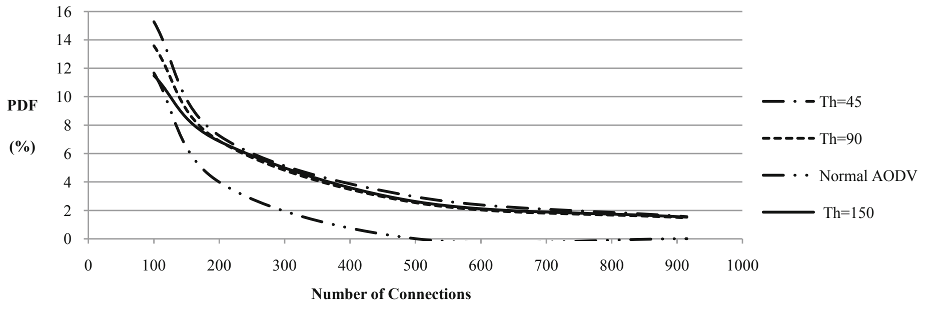

**Fig. 3.** PDF versus number of connections in AODV and M-AODV (Network with 50 nodes)

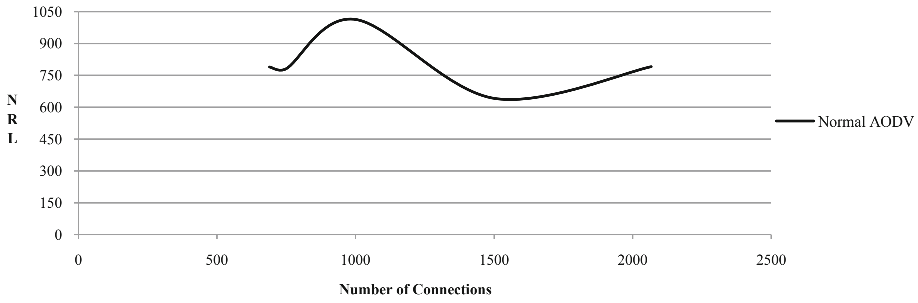

**Fig. 4.** NRL versus number of connections in AODV (Network with 75 nodes)

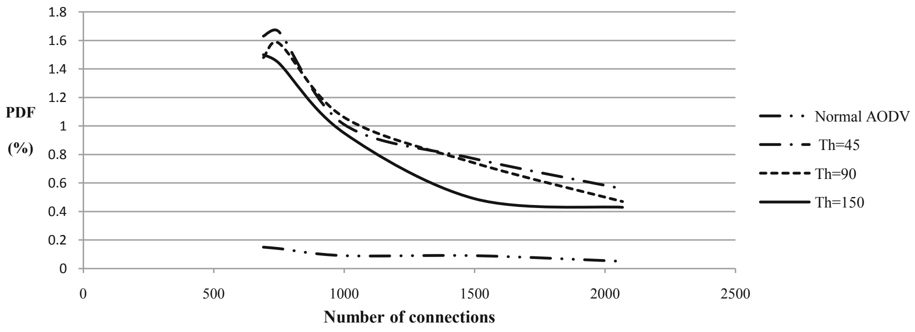

**Fig. 5.** NRL versus number of connections in M-AODV (Network with 75 nodes)

In summary, our proposed scheme M-AODV achieves more performance when compared to Normal AODV and this achievement can be maximized by critically adjusting the RREQ-Threshold value, otherwise, the network performance is decreased. Because, when we increase the threshold value, the PDF start decreasing and NRL starts increasing and as a result, the control packets injected into the network are increased. These control packets then contend with the data packets to access the shared wireless medium, and thus reduce the data delivery to the destination and hence reduce the network performance. On the other hand, if the threshold value is decreased too much, the injection of control packet into the network is reduced, but the time required to set up the pre-specified number of connections is increased, thus the network is underutilized and reduces the performance.

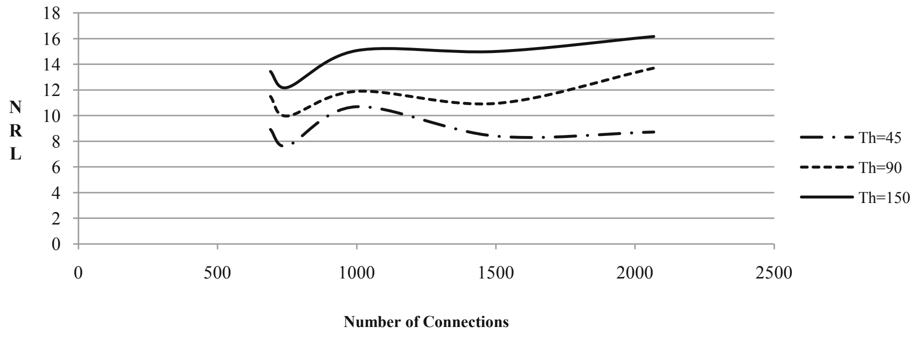

**Fig. 6.** PDF versus number of connections in AODV and M-AODV (Network with 75 nodes)

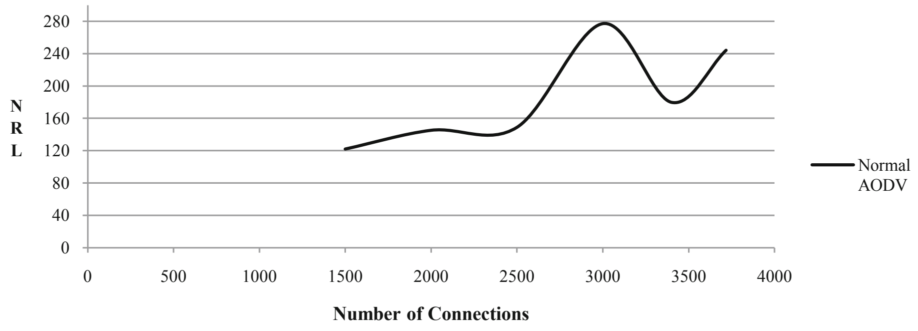

**Fig. 7.** NRL versus number of connections in AODV (Network with 100 nodes)

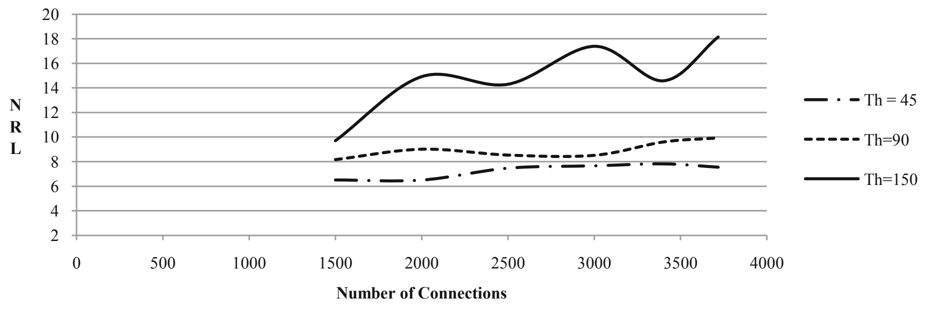

**Fig. 8.** NRL versus number of connections in M-AODV (Network with 100 nodes)

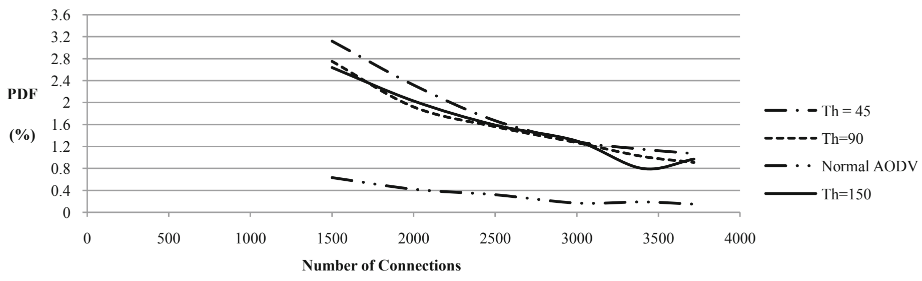

**Fig. 9.** PDF versus number of connections in AODV and M-AODV (Network with 100 nodes)

## 4 Conclusion

In this paper, we proposed a scheme for minimizing the routing overhead in the Ad hoc on demand routing protocol (AODV), when a number of connections need to be established simultaneously. In AODV, when a number of connections need to be established simultaneously, then due to the large control overhead, very few data packet gets delivered. Our proposed scheme improves the performance of AODV in such situations. We also simulated our proposal on ns-2.34 and verify that our proposal actually give better result than AODV in worse situation (when a lot of connections need to be established simultaneously).

## References

[1] Perkins, C.E., Bhagvat, P.: Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceeding of the Conference on Communications Architectures, Protocols and Applications (SIGCOMM 1994), pp. 234–244 (October 1999)

[2] Perkins, C., Belding-Royer, E., Das, S.: RFC-3561, Ad Hoc on-Demand Distance Vector Routing (July 2003), http://www.faqs.org/rfcs/rfc3561.html

[3] Corson, S., Macker, J.: RFC-2501, Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations (January 1999), http://www.ietf.org/rfc/rfc2501.txt, http://www.faqs.org/rfcs/rfc2501.html

[4] Johnson, D., Maltz, D., Hu, Y.: RFC-4728, The dynamic source routing protocol for Mobile ad hoc networks (DSR) (February 2007), http://www.faqs.org/rfcs/rfc4728.html

[5] Perkins, C.E., Royer, E.M.: Ad-Hoc On Demand Distance Vector Routing. In: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), New Orleans, LA, pp. 90–100 (February 1999)

[6] Spojmenovic, I.: Handbook of wireless network and mobile computing. Publication viley-Interscience

[7] The NS Manual (October 2, 2006), http://www.isi.edu/nsnam/ns

[8] The Network Simulator NS-2 tutorial homepage, http://www.isi.edu/nsnam/ns/tutorial/index.html

[9] Mann, R.P., Arbindi, S., Namuduri, K.: Control Traffic Analysis of On-Demand Routing Protocols in Ad-hoc wireless Networks. In: IEEE 62nd Vehicular Technology Conference, pp. 301–305
[10] Royer, E.M., Toh, C.-K.: A Review of Current Routing Protocols for Ad-hoc Wireless Mobile Networks. IEEE Personal Comn., 46–55 (April 1999)
[11] Chiang, C.C., Gerla, M.: Routing and multicast in multi hop, mobile wireless networks. In: Proceedings of IEEE ICUPC 1997, San Diego, CA, pp. 546–551 (October 1997)
[12] Chen, T.-W., Gerla, M.: Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks, pp. 171–175 (June 1998)
[13] Toh, C.-K.: Associativity based routing for ad hoc mobile networks. Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems 4(2), 103–139 (1997)
[14] Park, V., Corson, S.: Temporally-Ordered Routing Algorithms (TORA) Version 1 Functional Specification. IETF, Internet Draft: draft-ietf-manet-tora-spec-01.txt (August 1998)
[15] Corson, M.S., Ephremides, A.: A distributed routing algorithm for mobile wireless Networks. ACM/Baltzer Journal of Wireless Networks 1(1), 61–81 (1995)

## ANNEXURE 1: Pseudocode of the Proposed M-AODV

### /* Source sends data to destination */

```
Procedure send-data (S, D)      //S: Source node, D: Destination node
  {
      if ( rt == 0)           //rt is a routing table entry for D and
                                   rt=0  means no entry for D
           Send -RREQ (D);
       else
           S sends data to D using the routing table entry rt;
  }

  Procedure Send-RREQ (D)       //S broadcast the RREQ packet to find
route to D
  {
      Broadcast  (P);      //P  is  a  RREQ  packet  that  contains  the
                                    fields:hop_count,         RREQ_id,
                                    dst_address,            dst_seqno,
                                    src_address, src_seqno
  }
```

**/* An intermediate node receives a RREQ packet */**

```
  Procedure Receive-RREQ (Packet P)
  {
       Check the RREQ packet for duplication;
       if (Already_received == 1)
       {
           Print "received duplicate RREQ packet, so discard";
```

```
            Return;
        }

        Receive the packet P;
        Create a reverse link to the source node if it does not exist;

        //check whether the received node is the destination or not
        if ( P.dst_address == myaddress)
        {
            Print "destination sending route reply";
            Update destination sequence number (seqno);
            Sendreply ( P.src_address, 1,  myaddress, seqno);
            Return;
        }

      // check whether the received node has a fresh route to D or not,
                where
      //dstrt is the routing table entry of the received node that points
                to D
        if ( dstrt != 0 && dstrt.rt_seqno > P.dst_seqno )
        {
            Sendreply  (  P.src_address,  dstrt.hops  +  1,P.dst_address,
                          dstrt.rt_seqno);
            Return;
        }

        //forward the RREQ packet
        if (C-value > = RREQ-Threshold)
        {
            Print "exceed the limits of forwarding the RREQ packets";
            discard (P);            return;
        }
        C-value++;
        P.hop_count = P.hop_count + 1;
        Broadcast_RREQ (P);
    }
```

```
/* An intermediate node receives a RREP packet */
Procedure Receive-RREP (Packet P)
  {
   Create a forward link to the destination node if it does not exist;
      //check whether the received node is the original source or not

     if (myaddress == P.src_address)
     {
         Send all the buffered data packets;
         Return;
     }

     // otherwise forward the RREP
     if (revrt != 0) //revrt is the reverse link routing table entry
     {
          P.hop_count = P.hop_count + 1;
          forward the RREP packet P using routing entry revrt;
          C-value--;   return;
     }
     else                     // backward link to source does not exist.
     {
          drop the RREP packet P;  return;
     }
  }
```

**/* Counter update when a RREP packet is not received in time*/**

```
Procedure Counterupdate ()    //This procedure take care of decrementing
  {                      //C-value of a node which does not receive RREP
      // rt and rtn are two variable routing table entries
      for ( rt = routingtable.head; rt; rt = rtn)
      {
         rtn = rt.nextentry();
    if ( current_time >= rt.expiry_time )    // if rt is a stale entry
      {
      If ( rt.rt_flag == 1)            // if rt is a backward link entry
                 C-value--;
          routingtable.removeentry (rt); //Remove rt from routing table
         }
      }
  }
```