# Multipath Dynamic MANET On-demand Protocol with QoS Implementation

Vipul Maheshwari
Research Scholar
Bharati Vidyapeeth College of Engineering & Research, Pune, India

Shrikant S. Jadhav
Assistant Professor
Bharati Vidyapeeth College of Engineering & Research, Pune, India

## ABSTRACT

Wireless sensor networks (*WSNs*) consist of densely deployed sensor nodes, which have limited computational capabilities, power supply, and communication bandwidth. To ensure reliability in delivering the sensing data through a large field of sensors remains a research challenge. The multipath routing technique is one of the best solution to this which can enforce network robustness in case of node failures. DYMO_MQ [1] is a novel multipath approach which is a multipath extension to Dynamic MANET On-demand (*DYMO*). In this paper the author are making an effort to implement the DYMO_MQ protocol *NS2*. Results gives the confirmation of successful implementation.

## General Terms

Wireless Sensor Network, MANET, DYMO.

## Keywords

Multipath Routing, QoS, NS2 simulator, DYMO_MQ.

## 1. INTRODUCTION

Wireless sensor network (*WSN*) is a wireless network which consists of spatially distributed nodes with optionally attached sensors. These sensors can be used for monitoring physical and environmental conditions such as temperature, sound or vibration at different locations. *WSNs* are very useful in variety of application in industrial environments. They can be used to monitor areas which are difficult to reach even by humans in a cost-effective way. The main challenge in the field is achieving suitable performance with minimal processing and bandwidth usage for maximum energy efficiency, as the only power source is the included battery.

A routing protocol is needed to achieve the connectivity throughout the network. It should allow for decentralized network configuration and for reconfiguration in the event of a defective path [2]. In the past single path routing protocols have been heavily discussed and examined.

A more recent research topic for *WSN* is multipath routing protocols. Multipath routing allows more than one paths to be maintained between a source and a destination. These multiple paths play very important role to deliver data reliably. These alternative paths can be used in case of failure of current path or simultaneously for load balancing by using several paths parallely [3]. Second case is out of the scope of our project work. These paths can also be classified in three principal categories such as: link disjoint paths, not disjoint paths and node disjoint paths. Project only considers the third category because the independence and resilience that the nodes disjoint paths provide.

In this paper the author are implementing the protocol *DYMO_MQ* (*Multipath Dynamic MANET On-demand with Quality of Service*) described in Section 3. It adapts and extends Dynamic MANET On-demand (DYMO) routing protocol for *WSN*. *DYMO* Routing is an on-demand unicast hop-by-hop protocol for *MANETs*. It is being developed as a simplified combination of previous reactive routing protocols [4], [5] and uses distance vector routing like *AODV* to maintain loop-free routes. *DYMO_MQ* is a multipath on-demand protocol with *QoS* that establishes multiple loop-free node disjoint paths between a source and a destination. Protocol can establish multiple paths in one route discovery using single query flood and uses them to backup routes in case of link failure.

The remainder of the paper is organized as follows. In section 2, describes some more multipath routing protocols. Section 3 explains *DYMO_MQ* protocol in detail. Section 4 gives implementation work and Section 5 elaborate on results achieved. Finally section 6 concludes this paper.

## 2. RELATED WORK

In this section various on- demand multipath routing protocols especially from the viewpoint of route discovery strategy are briefly described.

*AODV-BR* [6]: All neighboring nodes of the primary route maintain a backup route to the destination. In case a link in the primary path fails, data packets are sent to a neighbor which redirect packet and sends it to the destination. It is actually not a multipath protocol since only single path per destination is maintained.

*AODVM* [7]: It is a multipath extension of AODV. But *Ye et al* found the number of paths to be very low and propose using special reliable nodes (in terms of being capable of combating fading, more secure and equipped with better batteries) to provide a reliable routing framework.

*AODV Multipath* [8]: is an extension of the *AODV* protocol designed to find multiple node-disjoint paths. In these intermediate nodes forwards Route Request (*RREQ)* packets towards the destination. Duplicate *RREQ* for the same source-destination pair is recorded in the *RREQ* table instead of simply discarding them. The destination tries to maximize the number of calculated multiple paths by replying appropriately to all route requests (RM). Routing Reply (*RREP)* packets are forwarded to the source via the inverse route traversed by the *RREQ*. Node disjointness in paths is ensured by deleting the corresponding entry of the transmitting node from their *RREQ* table.

The *DYMO* routing protocol enables reactive, multi-hop routing between source and sink nodes. It is a newly proposed protocol currently defined in an *IETF* Internet-Draft [9]. *DYMO* is a successor of *AODV* routing protocol. It operates similarly to *AODV*. However, it adopts some techniques found

in source routing protocols (such as *DSR*), under the path accumulation method described in the protocol. In this case, each node adds itself in the routing message it forwards, in order for nodes receiving these packets to construct paths towards these nodes accumulated, before this is needed.

*DYMO_MQ* is an on-demand multipath routing protocol with *QoS* developed as an extension to *DYMO* that enables the source and the destination nodes to maintain more than one path towards each other. It is detailed in following section.

# 3. DYMO_MQ OVERVIEW

DYMO_MQ is an 'On-demand Multipath Routing Protocol with QoS', developed as an extension to DYMO. This enables the source and the destination nodes to maintain multiple path towards each other.
DYMO_MQ algorithm has following two main phases:

*a) Multiple Route Discovery Phase*
*b) Route Maintenance Phase*

In DYMO_MQ routing messages from source and destination are used to establish path between source and destination. Since our main objective is to provide fault-tolerance and reduce frequency of query floods, paths which are loop-free and node-disjoint are only selected. To ensure QoS, routing message of type RREQ is enriched with two new fields as follows:

| *IP header* | *UDP header* | *Messa-ge Header* | *Messa-ge TLV* | *Addr header Target node* | *Addr TLV Target node* |
|---|---|---|---|---|---|
| *Addr header Origi-nal node* | *Addr TLV Origi-nal node* | *Addr header Addi-tional node* | *Addr TLV Addi-tional node* | *T_ delay* | *T_error* |

Figure1. Extended RREQ message structure format

Routing messages are of two types RREQ and RREP. They carry similar information but are handled differently by each node. Both contain all fields shown in above diagram except *T_delay* and *T_error* which are only included in RREQ, not present in RREP.

*T_delay*: Maximum delay for a transmission from source node to destination node

*T_error*: Refer to error rate of the path

*A. Multipath Route Discovery*

When a source node wants to send data to the destination node and no route to this destination is available in its routing table, in this case, the source creates and broadcasts a RREQ packet. The initial value of the T_delay field represents the end-to-end delay requirement for a QoS path fixed by the target application.

Routing table is modified so that multiple paths to destination node can be maintained.

1) *Intermediate node task:* Intermediate node processes RREQ packet in following way:

- It subtracts from the T_delay indicated in the RREQ the time required by this node to process the RREQ.
- Calculate the T_error as follows:

$$T(i,j) = \frac{\alpha * D(i,j)}{V_j} \qquad (1)$$

Where, T (i, j): the error rate between the node i and the node j α: a factor, D (i, j): the distance between i and j (a constant distance), $V_j$: unused space in the queue of the node j (calculated dynamically when a RREQ packet is received), such as the value of the *T_error* is the multiplication of all the error rates of nodes include on the actual path. The purpose of the error rate calculation is to keep only paths with a minimal congestion.

If *T_delay* is not positive and *T_error* > threshold, the node drops the RREQ packet.

2) *Destination node task and selecting node disjoints paths*: From a fault tolerance perspective, in case of route failure, node disjointness of available multiple paths are highly desirable. This node disjointness is ensured by the destination node in centralized manner using RREQ and RREP packets.

The destination node selects node disjoint paths in following way: When the destination node receives first RREQ packet, it records the list of all node IDs for the entire route path in its cache and sends a RREP packet. If it is a RREP, the destination includes next hop in the forward route entry. For the duplicate Routing Message (RM) packets received by the destination and which are not all node disjoint paths, the destination compares the accumulated path of the received RM to all the existing node disjoint paths in its cache. Paths not containing any common node recorded in the destination's cache memory. Otherwise the RM packet is dropped. RREP packets which arrive to the source node after the node disjoint selection, source records them in the route table as an alternative, can be used when a broken link is detected.

# 4. IMPLEMENTATION

The authors have implemented DYMO_MQ in popular simulator called NS-2. Details about NS-2 and implementation are explained in following sub-section.

## 4.1 NS-2

Following are the features of NS-2 used for implementing the project work.

1. NS-2 is an open source discrete event simulator.
2. NS-2 is a clean slate design, aiming to be an easier to use, more readily extensible platform.
3. Interface of NS-2 is 'Tool Command Language (TCL)' while back end is in C++.

NS-2 is released under the terms of the GNU General Public License (GPL) and is freely available for download on internet. NS-2 is modular in nature. It contains all commonly used networking algorithms in it. It can be extended to even implement newer algorithms. Module for new algorithm has to be developed in C, C++ and then it can be patched in NS-2.

## 4.2 DYMOUM (NS-2) Package

DYMOUM is an implementation of the DYMO (Dynamic Manet On-demand) routing protocol for

both linux kernels and the NS-2 (Network Simulator). The code is also released under the terms of the GNU General Public License (GPL) and it freely available for download on internet.

DYMOUM is developed in C, C++. Major modules are explained with their functionality in following Table 1.

| Module | Functionality |
|---|---|
| dymo_generic.c | Implements general DYMO protocol |
| pending_rreq.c | Handles the pending RREQ requests in queue of nodes. |
| dymo_re.h | It is header file in which structure of RREQ packet is defined |
| dymo_re.c | Implements the processing methods for different RM packets |
| dymo_socket.c | Handles the socket creation part in UDP packets |
| blacklist.c | Stores the broken path information |
| dymo_hello.c | Sends 'Hello Packet' to other nodes |
| dymo_rerr.c | Handles Route Error (RERR) packet |
| dymo_timeout.c | Implement timer for 'Hello packet' |
| d_queue.c | Buffer management |
| k_route.c | Handles route information change |
| Icmp_socket.c | Internet Control Message Protocol |
| d_netlink.c | Network Management like bandwidth, delay etc. |
| r_table.c | Routing table |
| dymo_nb.c | Handles neighbor information |
| main.c | Main function |

Table1: DYMOUM Modules Description

## 4.3 DYMO_MQ

DYMO_MQ as explained in above sections is nothing but enriched DYMO protocol with multipath route handling capacity as well as facility to ensure QoS included. It is done by extending RREQ packet by including *T_delay* and *T_error* fields as explained in Section 3. This is implemented in NS-2 by incorporating these in DYMOUM patch as follows:

1) *Modification in Dymo_re.h*: dymo_re.h is header file. It defines the RREQ packet structure. Here, add following two lines in struct i.e. *T_delay* and *T_error* fields are included.

```
double t_delay;
double t_error;
```

Also function at line no. 101 is modified as below:

```
RE *re_create_rreq(struct in_addr target_addr,
        u_int32_t target_seqnum,
        struct in_addr re_node_addr,
        u_int32_t re_node_seqnum,
        u_int8_t prefix, u_int8_t g,
         u_int8_t ttl, u_int8_t thopcnt);
```

2) *Modification in dymo_re.c*: dymo_re.c is implement the processing methods for different RM packets like RREQ and RREP. As explained in Section 3 in DYMO_MQ, RREQ and RREP are handled differently as compared to DYMO. Initial values of T_delay and T_error can be set at line no. 62. Also function at line no. 387 is modified as below:

```
void NS_CLASS re_forward_rreq_path_acc(RE
*rreq, int blindex)
{
   int i;
   struct in_addr bcast_addr;
   rreq->t_delay=rreq->t_delay-1000.0;
   //DYMO_MQ
   rreq->t_error=5*200/(1000-searchposition());
   if(rreq->t_delay>=0||rreq->t_error<0.5)
   {
      dlog(LOG_DEBUG, 0, __FUNCTION__,
       "forwarding RREQ to find %s",
        ip2str(rreq->target_addr));

   bcast_addr.s_addr = DYMO_BROADCAST;
   i=0;
  while (i < DYMO_MAX_NR_INTERFACES)
   {
       if (DEV_NR(i).enabled)
       {
            rreq->re_blocks[blindex].re_node_addr =
            (u_int32_t) DEV_NR(i).ipaddr.s_addr;
            dymo_socket_queue((DYMO_element *)
            rreq);
            dymo_socket_send(bcast_addr,
             &DEV_NR(i));
        }
        i++;
    }
   }
   else
   {
      printf("\n RREQ packet dropped for QOS \n");
      free(rreq);
    }
}
```

3) *Modification in pending_rreq.c*: It handles the pending RREQ requests in queue of nodes. *T_delay* is the complete time required to reach RREQ packet from source node to destination on node. Every time RREQ packet reaches an intermediate node, total time required to process RREQ packet by that node is subtracted from *T_delay*. Following function which calculates this time is added at the end of pending_rreq.c file.

```
int NS_CLASS searchposition()
{
        dlist_head_t *pos;
        int count;
        dlist_for_each(pos, &PENDING_RREQ)
        {
                count++;
        }
        return count;
}
```

## 5. Experimentation and Results

In this section, author would like to show the result & analysis of the DYMO-MQ. In *Figure 2* shows the result of scenario having 3 nodes which are in moving state i.e. dynamic nodes. When nodes are dynamic they may go out of range, where they are not accessible from other nodes. In this case packets sent to such nodes will be dropped.

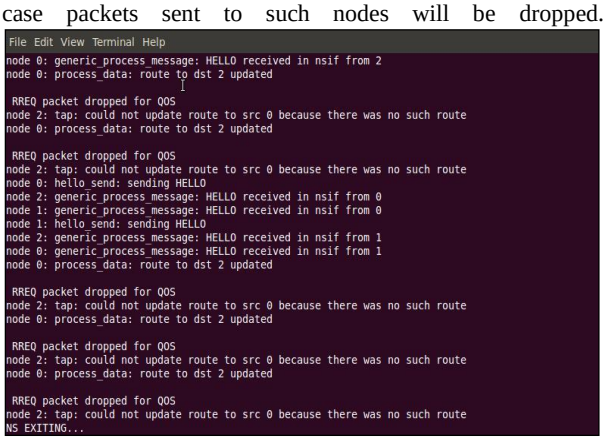


Figure2. DYMO_MQ with 3 Dynamic Nodes

Since RREQ packet is broadcasted every time it is arrived at any node, there are high chances of it getting delayed. Also every node takes some time for processing it. In above snapshot it can be clearly see that few RREQ packets are dropped since they are not satisfying the QoS criteria. This way enhanced DYMO_MQ overcomes the limitation of DYMO and DYMO.

In *Figure 3* shows the execution result for scenario which considers 2 moving nodes. In case of just 2 nodes there is no consideration of delay or error introduction.

Since in case of just 2 nodes either RREQ packet reaches the target node or it does not. Also since RREQ packet goes only to second node obviously there is no introduction of delay in it. Therefore as shown above there is no packet drop for QoS. Hello packet is sent to establish connection and as soon connection is established RREQ packet is sent.

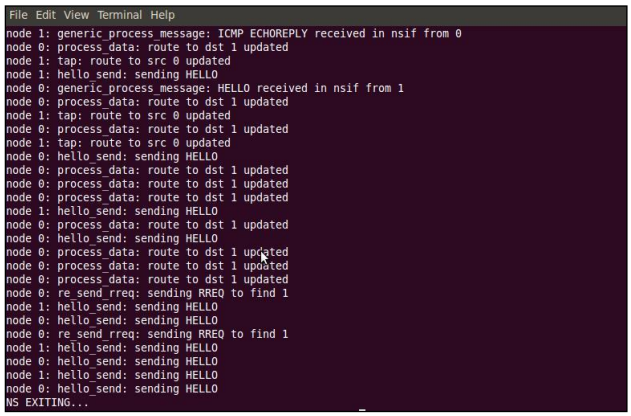


Figure3. DYMO_MQ with 2 Dynamic Nodes

In *Figure 4* shows the execution result of scenario which considers 3 static nodes. Nodes are static i.e. non-moving.

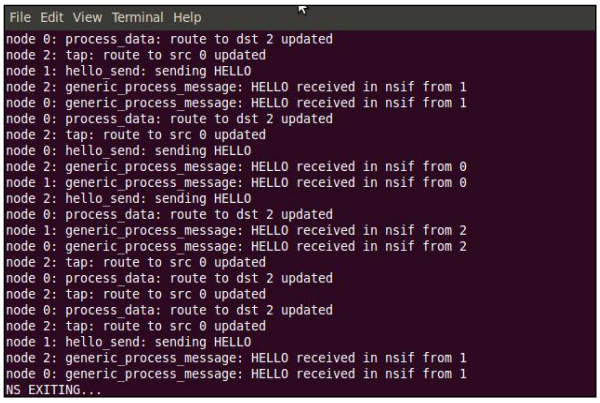


Figure4. DYMO_MQ with 2 static Nodes

Since nodes are not moving packets will always be received, once connection is established it will remain. So there is no question of packet drop even in this case.
From the above results, it can be clearly see that by including the 2 parameters as *t_delay* and *t_error* in RREQ Packet, it improves QoS. Based on the values of these parameters
RREQ Packet which are not satisfying the QoS are dropped. Hence, the Quality of Service (QoS) in DYMO protocol is ensured.

## 6. Conclusion

In this paper, authors have implemented DYMO_MQ a multipath routing protocol supporting QoS for WSN. This protocol can search multiple node disjoint paths during a single multiple route discovery with QoS constraints and using them as backup routes in case of link failure in DYMO. For implementation authors have used NS-2 which is an open source modular software. DYMOUM is a patch of NS-2 which implements DYMO protocol. Authors have modified DYMOUM code to incorporate all additional functionalities of DYMO_MQ. Results confirm the successful implementation of DYMO_MQ protocol.

## 7. REFERENCES

[1] Nawel BENDIMERAD, Bouabdellah KECHAR. IEEE 2011. Performance evaluation of QoS aware Multipath extensions For the Dynamic MANET On-demand protocol in Wireless Sensor Networks.

[2] I. Zarov, "Mesh Routing for Low-Power Mobil Ad Hoc Wireless Sensor Networks Using DYMO-low", Guided Research Report, Jacob University Bremen, Germany, May 15, 2007.

[3] N. Bendimerad, B. Kechar, Z. Bidai, H. Haffaf, "Multipath On-demand Routing Protocol with Quality of Service for Wireless Sensor Networks", International Conference on Applied Informatics, Bordj Bou Arréridj, Algeria, November 15-17, 2008.

[4] David B. Johnson, David A. Maltz, and Yih-Chun Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," July 2004, INTERNET-DRAFT draft-ietf-manet-dsr-10.txt.

[5] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," July 2003, RFC 3561.

[6] Sung-Ju Lee and Mario Gerla, "AODV-BR: Backup Routing in Ad hoc Networks," in Proceedings of the IEEE Wireless Communications and Networking

Conference (WCNC 2000), Chicago, IL, September 2000.

[7] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi, "A Framework for Reliable Routing in Mobile Ad Hoc Networks," in INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE, 2003, vol. 1, pp. 270-280.

[8] Ye , Krishnamurthy, Tripathi, " A framework for reliable routing in mobile ad hoc networks". IEEE INFOCOM (2003)

[9] I. Chakeres, C. "Perkins. Dynamic MANET On-demand (DYMO) Routing", draft-ietf-manet-dymo-17, March 8, 2009