Indian Journal of Science and Technology, *Vol 9(45), DOI: 10.17485/ijst/2016/v9i45/96658, December 2016*

ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

LUNAR: Working and Performance Evaluation in MANETs

Shitalkumar A Jain[1*] and Vijay T Raisinghani[2]

School of Engineering, MPSTME, NMIMS University, Mumbai – 400056, India; sajain@comp.maepune.ac.in, rvijay@ieee.org

Abstract

Background/Objectives: Load eqUilibrium Neighbor Aware Routing (LUNAR)[1] is proposed to address broadcast storm problem. In this paper, we discuss algorithm, flow-chart, working example along with simulation results of LUNAR in detail. **Methods/Statistical Analysis:** To address the problem, existing reactive routing protocols use one of the node metrics such as size of routing table, available queue space, neighbor count, available battery life, etc. However, measurement of single parameter at an intermediate node may not be the true measure of route stability or lifetime. LUNAR combines the advantages of neighbor coverage knowledge and load balancing techniques to implement decision making system at every intermediate node. **Findings:** We evaluate the performance of LUNAR with respect to performance metrics like Normalized Routing Overhead, End-to-End Delay and Packet Loss Rate in different network scenarios. LUNAR minimizes the routing overhead of the network by 31-54% compared to AODV and NCPR due to the reduction in routing packets required for route discovery. LUNAR reduces end-to- end delay by 15-32% and packet loss rate by 9-35% compared to AODV and NCPR. **Applications/Improvements:** Simulation result shows the reduction in rebroadcasting of routing packets.

Keywords: Broadcasting Storm, End-End Delay (EED), MANET, Normalized Routing Overhead (NRO), Packet Loss Rate (PLR), Route Request Storm

1. Introduction

Mobile Ad Hoc Networks (MANETs) consist of a collection of mobile nodes which can move in any direction with variable speed. Communication among the mobile nodes in a MANET takes place in a multi-hop manner. MANET poses multiple challenges during the design of routing protocol as each mobile node has to act as a host as well as a router; it is prone to failure due to limited energy; due to frequent channel contention and congestion in the network; it works with limited resources like bandwidth, memory and processing power. Design of efficient routing protocol is required to overcome the challenges of MANETs. A number of protocols have been proposed in literature like Ad hoc On Demand Distance Vector (AODV)[2], Dynamic Source Routing (DSR)[3], Probabilistic Counter-Based Route Discovery for Mobile Ad Hoc Networks (PCBRD)[4], Load-balancing in MANET shortest-path routing protocols (LBR)[5], Congestion Adaptive Routing in Mobile Ad Hoc Networks (CRP)[6], Neighbor Coverage-Based Probabilistic Rebroadcast (NCPR)[7,12,31] to address the above routing challenges of MANETS. Broadcasting is used in most of the reactive routing protocols like AODV[2], LBR[5], CRP[6], NCPR[7] to discover the route in the network[8-10]. Uncontrolled RREQ broadcasting could result in a Broadcast Storm problem or Route Request Storm problem[8]. To address the broadcast storm problem various routing protocols have been proposed in literature[4-7]. Existing MANET routing protocol design focuses on node parameters like queue length, routing table size and energy available[4-7]. However, adaptations in the routing protocol based on these parameters are not sufficient to improve the performance of the network in challenging conditions such as highly dense networks with high mobility. Thus, we proposed Load equilibrium Neighbor Coverage Routing (LUNAR)[1] protocol in which we combine the advantages of neighbor coverage knowledge and one of the load balancing

* *Author for correspondence*

techniques. In LUNAR, due to proper decision making system at the intermediate nodes, rebroadcasting decision of RREQ packets are taken appropriately with fewer calculations. Our simulation results show that LUNAR significantly decreases the retransmission of RREQ packets and thus reduce the overall routing overhead of the network.

2. Load eqUilibrium Neighbor Aware Routing (LUNAR)

We discuss here Load eqUilibrium Neighbor Aware Routing (LUNAR) through algorithm, flow-chart and working example. Our analysis of existing protocols shows that most of the reactive routing protocols make use of a single Node State Measure (NSM) (energy level, available queue length, neighbor information, routing table size or speed of the node, etc) during route discovery process. However, measuring a single NSM may not correctly reflect the status of the node and the network with respect to route stability. Thus, use of single NSM, during route discovery, could lead to sub-optimal routes and thus repeated route discovery overheads. Our analysis of existing protocols shows that LBR[5] does not address the broadcast storm problem. NCPR[7] does not address load balancing problems. Based on the above observations, we proposed routing protocol Load eqUilibrium Neighbor Aware Routing (LUNAR)[1], which incorporates and enhances the useful features of AODV, LBR and NCPR. LUNAR is a new mechanism which combines the advantages of load balancing[5] and uncovered neighbors (UCN) set knowledge[7] Basic purpose of this mechanism is to improve the network performance by minimizing routing overhead and end-to-end delay. The outline of the route discovery using LUNAR is as follows:

(a) Source node initiates route discovery for a destination node by generating RREQ packet with a new sequence number, if no route is available. (b) Each node identifies its neighbor nodes using exchange of Hello packets to update the Neighbor Set Table (NST) (similar to the concept in AODV[2]). (c) Initially, the source node computes its own Active Path Count (APC) using a single NSM i.e. size of routing table and creates the Uncovered Neighbor (UCN) Set using NST. It updates the RREQ packet with Cumulative Active Path Count (CAPC) and Neighbor Set (NS) information. CAPC value is same as that of APC for the source node. The source node broadcasts the modified RREQ packet to its one hop neighbors. (d) Upon receiving RREQ packets from neighbor nodes, each node using a single NSM computes its own APC. Cumulative Active Path Count (CAPC), for each path, is computed by a node using its own APC and CAPC information received in RREQ packets from its neighbor nodes (with same sequence number). Node accepts the RREQ packet from its neighbor nodes till the acceptance timer (Acceptance_timer) expires.

(e) Each node adjusts the UCN, using its own NST and neighbor information received through RREQ packets. If its UCN set is null then all the received RREQ packets with the same sequence number will be discarded else the RREQ packet having the least value of CAPC is rebroadcasted. (g) At the destination node, on receipt of RREQ packets through different paths, the destination finally selects the path with the least value of CAPC and sends a Route Reply (RREP) packet through the reverse path up to source node.

2.1 Working of LUNAR

Whenever a source node has data for any destination node, it checks its routing table to see whether a route is available for the destination. If the source node does not find a route then it initiates the route discovery process by creating a Route Request (RREQ) packet with a new sequence number. The source node inserts its own Active Path Count (APC) information along with its neighbour information in the RREQ packet. Active Path Count (APC) information is the number of active routes that are supported by the node at the time of RREQ packet creation. APC is obtained from the routing table size. The route discovery process of LUNAR comprises three algorithms. The algorithms are presented below along with the flow charts. The algorithms (section 2.4) explain the activities carried out by a source node, intermediate nodes and a destination node for route discovery. Notations used in the algorithms are as follows:

s - Source node
d - Destination node
$\mathbf{n_i, n_j, n_k}$ - Intermediate nodes
RREQ - Route Request Packet
RREP - Route Reply Packet
UCN($\mathbf{n_i}$) - Uncovered Neighbor Set of a node n_i
APC ($\mathbf{n_i}$) - Active Path Count (routing table size of a node n_i)

<table>
<tr><td colspan="8">Byte 0</td><td colspan="8">Byte 1</td><td colspan="8">Byte 2</td><td colspan="8">Byte 3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td colspan="8">Type</td><td>J</td><td>R</td><td>G</td><td>D</td><td>U</td><td colspan="11">Reserved</td><td colspan="8">Hop Count</td></tr>
<tr><td colspan="32">RREQ ID</td></tr>
<tr><td colspan="32">Destination IP Address</td></tr>
<tr><td colspan="32">Destination Sequence Number</td></tr>
<tr><td colspan="32">Originator IP Address</td></tr>
<tr><td colspan="32">Originator Sequence Number</td></tr>
<tr><td colspan="16">Number of Neighbor N (NS)</td><td colspan="16">CAPC</td></tr>
<tr><td colspan="32">Neighbor 1 IP Address</td></tr>
<tr><td colspan="32">Neighbor 2 IP Address</td></tr>
<tr><td colspan="32">----------</td></tr>
<tr><td colspan="32">----------</td></tr>
<tr><td colspan="32">Neighbor N IP Address</td></tr>
</table>

Figure 1. Route Request (RREQ) Message Format.

CAPC (n_i) - Cumulative Active Path Count computed at node n_i

NS(n_i) - Neighbor Set Table of n_i

RTS- Routing Table size

OSN – set of old sequence numbers of RREQ packets accepted and processed at a node

Acceptance_timer - Acceptance timer is uniformly set for all the nodes. Acceptance-timer starts when the first RREQ packet with a new sequence number arrives from a neighbor node. Unlike AODV, before Acceptance_timer expires, RREQ packets with the same sequence number (duplicate RREQ packets) are accepted which come from different routes (neighbor nodes). In LUNAR, every node keeps track of the RREQ sequence number and only after the Acceptance_timer expires the sequence number becomes old.

2.2 LUNAR RREQ Packet Format

We have modified RREQ packet of AODV by adding the information about CAPC and neighbor information. Other packet formats (Hello, RREP, RERR, etc) are same as that of AODV[1]. Figure 1 shows the RREQ packet format of LUNAR along with details of the field. We have modified the RREQ packet format of AODV and retained all the fields as is and following additional fields have been added - Number of Neighbor N, CAPC, Neighbor i IP Address: IP address of neighbor node i (where i = 1, 2,…, N). We have made a provision of 2 bytes each for the fields Number of Neighbor N and CAPC. Further, Neighbor i IP Address takes 4 bytes for each neighbor i. Thus, the minimum size of the RREQ packet is 32 bytes and the maximum size depends on the number of neighbor nodes. As compared to AODV, more stable routes would be discovered by LUNAR which compensates for the increase in initial routing overhead.

2.3 LUNAR Route Discovery Process: Call Flow

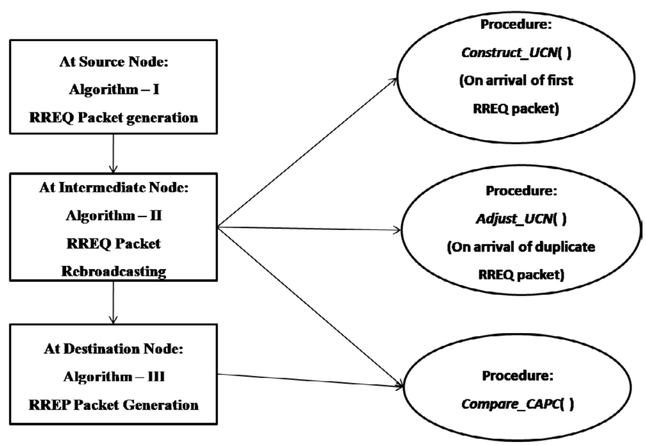

Figure 2. LUNAR Route Discovery Process: Call Flow (Schematic – each node has a copy of all procedures).

2.4 Algorithm for Route Discovery Process of LUNAR

Input: Source node (s) has data packet for destination node (d)
Output: RREQ packet with new sequence number, CAPC and NS(s) information
At Source node s: In this algorithm, source node generates RREQ packet with a new sequence number, inserts CAPC and NS(s) information and broadcasts RREQ to one hop neighbor nodes.

Begin
Initialize: APC(s) - routing table size of s, NS(s) – Neighbor set table of s, RREQ.seq_num – Sequence number of RREQ packet

1. Generate: RREQ. seq_num;
   //Routing table size of s is used to determine APC
2. Read: APC(s) and NS(s);
   //CAPC is same as APC for source node
3. Compute: CAPC(s) = APC(s);
4. Insert CAPC(s) and NS(s) into RREQ packet;
5. Broadcast RREQ packet to one hop neighbors;

End;

Figure 3. Algorithm for Route Request Generation at source node s.

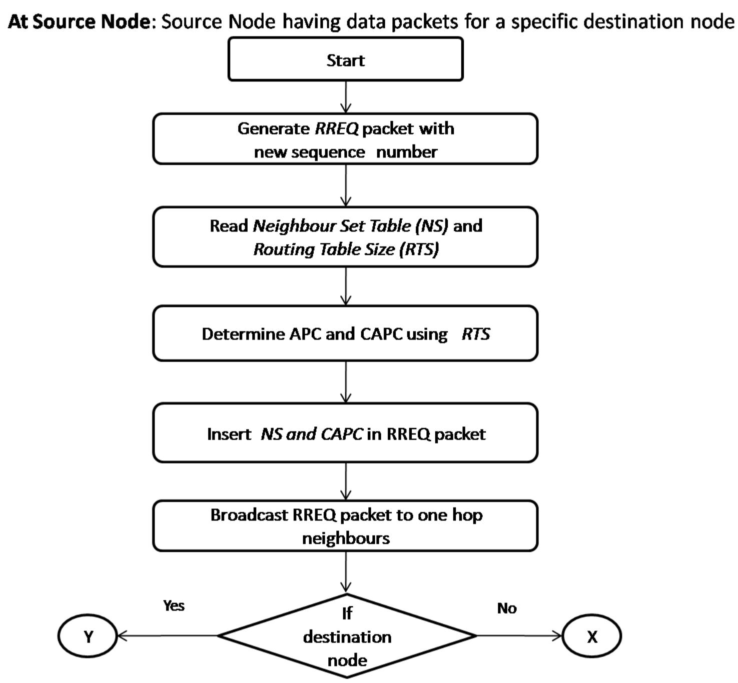

Figure 4. Flow-chart for Route Request Generation at `source node s.

Algorithm II and Flow chart: Route Request Packet Rebroadcasting

Input: RREQ packet with CAPC(n_j) and NS(n_j) information
Output: RREQ packet with updated CAPC(n_i) and NS(n_i) information
At intermediate node n_i with previous node n_j: In this algorithm, intermediate node (n_i) receives RREQ packets from one hop neighbor nodes (n_j). It computes CAPC for every received RREQ packet if UCN is not null. RREQ packet having least value of CAPC is rebroadcasted to one hop neighbors.

Begin
Initialize: APC(n_i) – Active Path Count of n_i, NS(n_i) – Neighbor Set Table n_i; N – number of neighbor nodes; j = 1,2,...,N; CA – set of CAPC values computed for each previous node n_j

1. j = 1;
   // n_j represents the previous node which forwards the RREQ packet to current node n_i;
2. Receive RREQ packet from node n_j;
   /* If RREQ packet with new sequence number is received from neighbor node n_j then accept the RREQ and start Acceptance_timer */
3. **If** (RREQ.seq_num does not exist in OSN)
   Then Accept RREQ packet; temp_SN = RREQ.seq_num; Start Acceptance_timer;
   Else Drop RREQ packet; Go to End;
4. **WHILE** (Acceptance_timer != Expired andand j <= N)
   // Routing table size of n_i is used to determine APC
   a. Read: APC(n_i) and NS(n_i);
   /* On acceptance of first RREQ packet with new sequence number, procedure to construct the UCN of a node n_i is called */
   b. **If** (j= 1) **Then** Call procedure: UCN(n_i) = Construct_UCN(NS(n_i), NS(n_j););
   Go to d;
   /*On acceptance of duplicate RREQ packet with same sequence number, procedure to adjust the UCN of a node n_i is called*/
   c. **If** (j > 1) **Then** Call procedure: UCN(n_i) = Adjust_UCN(UCN(n_i), NS(n_j));
   /* If all the neighbor nodes of node n_i has received the RREQ packet then node n_i discards all received RREQ packets having same sequence number*/
   d. **If** (UCN(n_i) == Null) **Then** Discard all RREQ packets; Go to End;
   /* Compute Cumulative Active Path Count for each node n_j using APC(n_i) and CAPC value received in RREQ packet from node n_j */
   e. Compute CAPC(n_j) for node n_j as:
   CAPC (n_j) = [APC (n_i) + CAPC (n_j)] / 2
   // Insert CAPC values computed for each node n_j in set CA;
   f. CA = CA ∪ CAPC(n_j)
   g. j++;
   /* Node n_i accept and process duplicate RREQ packet, having same sequence number, received from neighbor node n_j */
   h. Receive RREQ packet from neighbor node n_j;
   If (RREQ.seq_num = temp_SN)
   Then Accept RREQ packet;
   Else Drop RREQ packet;
5. End **WHILE Loop**;
   /* Sequence number of RREQ packet becomes old, once Acceptance_timer expires. No RREQ packet with same sequence number is accepted by the intermediate node n_i */
6. Insert RREQ.seq_num in OSN;
   // procedure to compare for least cumulative active path count
7. Call Procedure: CAPC(n_i) = Compare_CAPC (CA);
8. Insert CAPC(n_i) and NS(n_i) into RREQ packet;
9. Rebroadcast RREQ packet to one hop neighbors;

End;

Figure 5. Algorithm for route request rebroadcasting at intermediate node n_i.

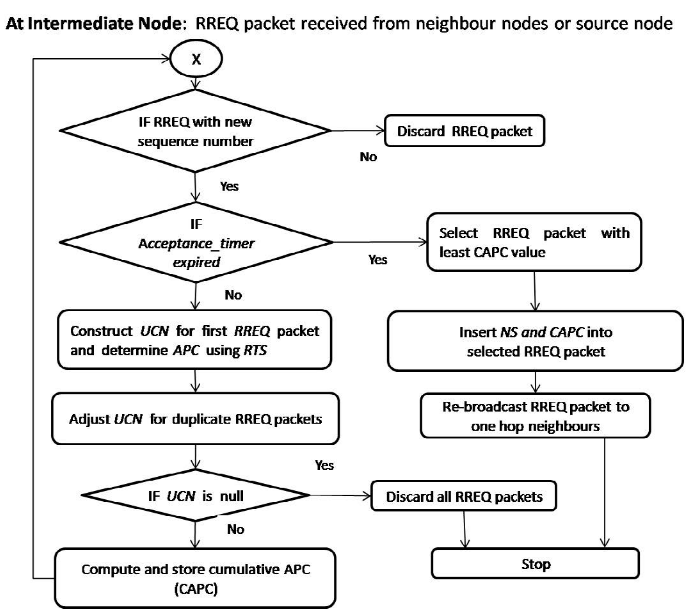

Figure 6. Flow-chart for route request rebroadcasting at intermediate node n_i.

Explanation of CAPC Computations

If UCN set is not NULL then each intermediate node n_i calculates Cumulative Active Path Count ($CAPC(n_j)$) as the arithmetic average of CAPC ($CAPC(n_j)$) received with RREQ packet from neighbor node n_j and its own APC ($APC(n_i)$).

CAPC computation: $CAPC\ (n_j) = [APC(n_i) + CAPC(n_j)]\ /\ 2$

Intermediate node computes CAPC value for every RREQ packet received from different paths till Acceptance_timer expires. After Acceptance_timer expires, intermediate nodes compare the CPAC value of all accepted RREQ packet having same sequence number. It rebroadcasts the RREQ packet to one hop neighbor which has the least CAPC value. In our protocol all the intermediate nodes actively participate in route discovery process and take the decision of rebroadcasting of RREQ packets.

Algorithm III and Flow chart: Route Reply Packet Generation

Input: Multiple RREQ packets from different routes

Output: RREP packet with reverse route information

At destination node d with previous node n_j: In this algorithm, destination node (d) receives RREQ packets from one hop neighbor nodes (n_j). It computes CAPC for every received RREQ packet. It creates a reverse path for the RREQ packet having the least value of CAPC. It then sends a RREP packet to source node through the reverse path, to form a stable route.

```
Begin
Initialize: APC(d) – Active Path Count of d, NS(d) – Neighbor set table of d; N – number of neighbor
nodes; j = 1,2,...N;CA – set of CAPC values computed for each previous node nj
1.  j = 1;
    // nj represents the previous node which forwards the RREQ packet to destination node d;
2.  Receive RREQ packet from node nj;
    /* IfRREQ packet with new sequence number is received from neighbor node nj then accept the
    RREQ and start Acceptance_timer*/
3.  If (RREQ.seq_num does not exist in OSN)
    Then Accept RREQ packet; temp_SN = RREQ.seq_num; Start Acceptance_timer;
    Else Drop RREQ packet; Go to End;
4.  WHILE (Acceptance_timer != Expired andand j < = N)
    a.  Read: APC(d);

    /* Compute Cumulative Active Path Count for each node nj using APC(d) andCAPC value
    received in RREQ packet from node nj */
    b.  Compute CAPC at dfor node nj as:
              CAPC (nj) = [APC (d) + CAPC (nj)]/2
    // Insert computed CAPC values for each node nj in the set;
    c.  CA = CA ∪ CAPC(nj)
    d.  j++;
    /* Destination node d accept and process duplicate RREQ packet, having same sequence number,
    received from neighbor node nj */
    e.  Receive RREQ packet from node nj;
              If (RREQ.seq_num = temp_SN)
              Then    Accept RREQ packet;
              Else    Drop RREQ packet;
5.  End WHILE Loop;
6.  Insert RREQ.seq_num in OSN;
7.  Call Procedure:CAPC(d) = Compare_CAPC (CA);
    // procedure to compare for least cumulative active path count
17. Select the RREQ packet having least cumulative active path count;
18. Generate route reply packet: RREP(RREQ.seq_num);
19. Send RREP to source node s through reverse path;
End;
```

Figure 7. Algorithm for route reply generation at destination node d.

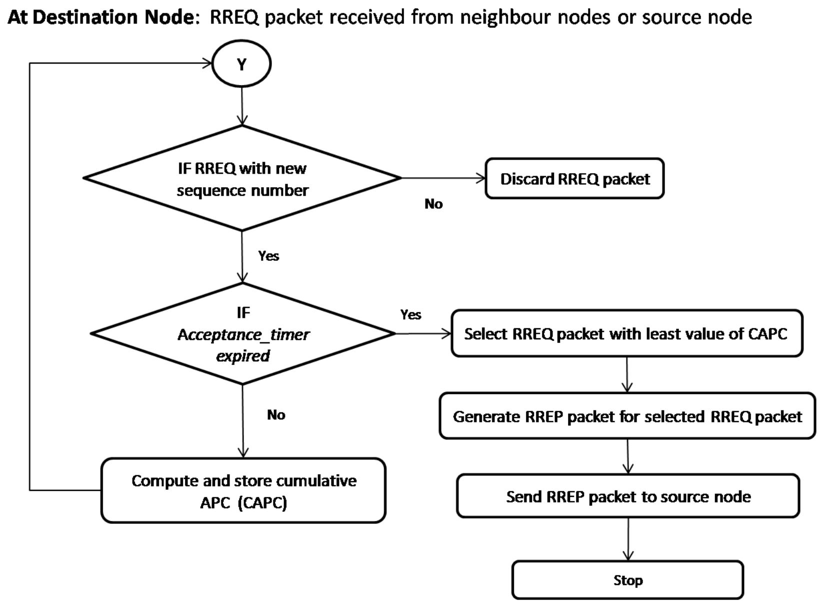

Figure 8. Flow-chart for route reply generation at destination node d.

Below we explain the procedures, which are called in the algorithms, for comparing for CAPC values, for constructing the UCN set and for adjusting the UCN set.

Procedure Compare_CAPC (CA)

Input: Set of CAPC values

Output: Return CAPC having least value

In this procedure, all CAPC values computed for the neighbor nodes n_j are compared and it returns CAPC having least value.

Begin

Initialize: n_j – previous node, N – number of neighbor nodes, j = 1,2, …., N, CA – set of CAPC values for each previous node n_j, CAPC(n_j)- Cumulative Active Path Count of a node n_j; temp_CA – temporary CAPC value

1. For each value of set CA
2. **FOR** j = 1 to N
   If (CAPC(n_j) > CAPC(n_{j+1}))
   Then temp_CA = CAPC(n_j); break;
   Else temp_CA = CAPC(n_{j+1}); break;
3. End **If;**
4. j++;
5. End **FOR Loop;**
6. Return temp_CA ;

End;

Figure 9. Procedure to compare CAPC values.

Explanation of UCN Computation

UCN construction phase

Each intermediate node (n_i) computes its Uncovered Neighbor set ($UCN(n_i)$) from the Neighbor Set (NS) information received in first RREQ packet from the source ($NS(s)$) or its previous node ($NS(n_j)$) and its own neighbor set ($NS(n_i)$).

Initial UCN computation:

$UCN(n_i) = NS(n_i) – [NS(n_i) \cap NS(s)] – \{s\}$ OR

$UCN(n_i) = NS(n_i) – [NS(n_i) \cap NS(n_j)] – \{n_j\}$

Procedure Construct_UCN (NS (n_i), NS (n_j))

UCN adjustment phase

Each intermediate node (n_i) adjusts its Uncovered Neighbor set ($UCN(n_i)$) from the Neighbor Set (NS) information received in duplicate RREQ packet from its previous node ($NS(n_k)$) and its own neighbor set ($NS(n_i)$). UCN adjustment: $UCN(n_i) = UCN(n_i) - [UCN(n_i) \cap NS(n_k)]$

If the UCN set is NULL then it simply drops the RREQ packet, since NULL means that every neighbour has already received the same RREQ packet from the source node or the previous node.

Procedure Adjust_UCN (UCN (n_i), NS (n_j));

At the destination node when multiple RREQ packets of same source are received from different routes, it compares the CAPC values. Destination node selects the reverse path based on the least value of CAPC from these multiple RREQ packets. It creates the Route Reply (RREP) packet and sends it along the reverse path to the source node. Every intermediate node, along this reverse path records, the path information in its routing table. After receiving the RREP packet from the destination node, the source node starts sending data packets to the destination node. In summary, LUNAR should reduce the routing overhead and delays because (i) only UCN and CAPC are computed at intermediate nodes lying on routes from source to destination (ii) load balancing is achieved using CAPC, and (iii) If RREQ packet is rebroadcasted by a node then overhead of a network is reduced by avoiding periodic Hello packet broadcasting.

```
Begin
Initialize: ni – intermediate node, nj – neighbor node, NS(ni) – neighbor set information of a node ni,
NS(nj) – neighbor set information of a node nj, UCN(ni) – Uncovered Neighbor Set for node ni
    a.  Compute: UCN(ni) = NS(ni) – [NS(ni) n NS(nj)] – {nj}
    b.  Return UCN(ni);
End;
```

Figure 10. Procedure to construct UCN set.

```
Begin
1.  If (i==1)
    Then Compute: UCN (ni) = UCN(ni) - [UCN(ni) n NS(nk)];
2.  Return UCN (ni);
End;
```

Figure 11. Procedure to adjust the UCN set.

3. Working Example of LUNAR

Consider a network scenario as shown in Figure 12 where S represents a source node which initiates the process of route discovery, D represents a destination node which selects the route based on least value of CAPC, Node N1 to node N11 represent intermediate nodes through which a route can be formed from the source to the destination

We assume that some communication already exists in the network. Table 3-1 gives the information about the number of active paths currently supported by the nodes. This information is used for computing APC and CAPC. Table 1 also gives the details of neighbor node information for each node. This information is used for computing or adjusting the UCN set.

At time t1, S generates the RREQ packet with a new sequence number and inserts the information of CAPC and NS in the RREQ. Next, it broadcasts the RREQ packet to neighbor nodes. Figure 13 shows the RREQ broadcasting from source node S. Label on the arrow includes the identity of nodes along the path through which RREQ packet is received at a node.

Table 1. APC and Neighbor node information of all network nodes

Node	APC	NS
S	3	{N1,N2, N3}
N1	4	{S, N2, N4, N5}
N2	5	{S,N1,N3,N4, N5, N6}
N3	3	{S, N2, N5, N6}
N4	4	{N1, N2, N5, N7, N8}
N5	3	{N1,N2, N3, N4, N6, N7, N8, N9}
N6	2	{N2, N3, N5, N8, N9}
N7	3	{N4, N5, N8, N10}
N8	6	{N4, N5, N6, N7, N9, D}
N9	3	{N5, N6, N8, N11}
N10	5	{N7, N8,D}
N11	3	{N11, N8,D}
D	2	{N8, N10, N11}

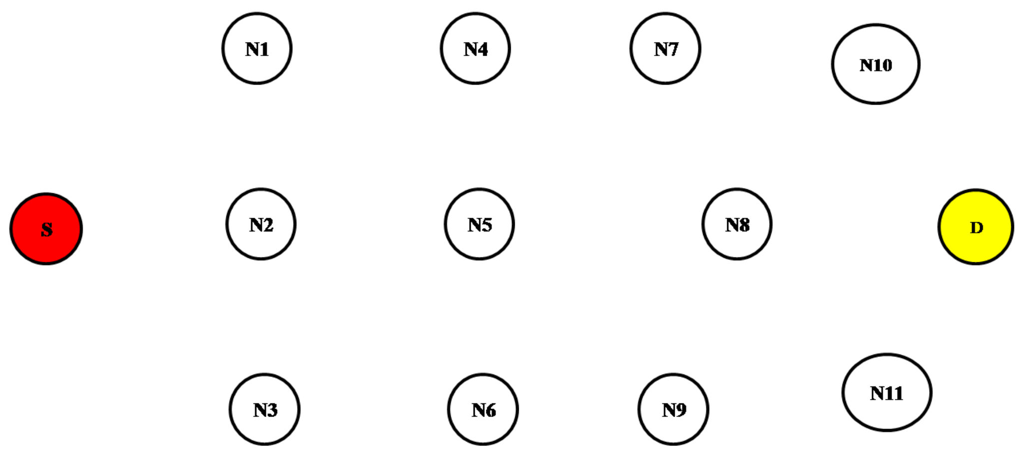

Figure 12. Initial network topology of LUNAR working example.

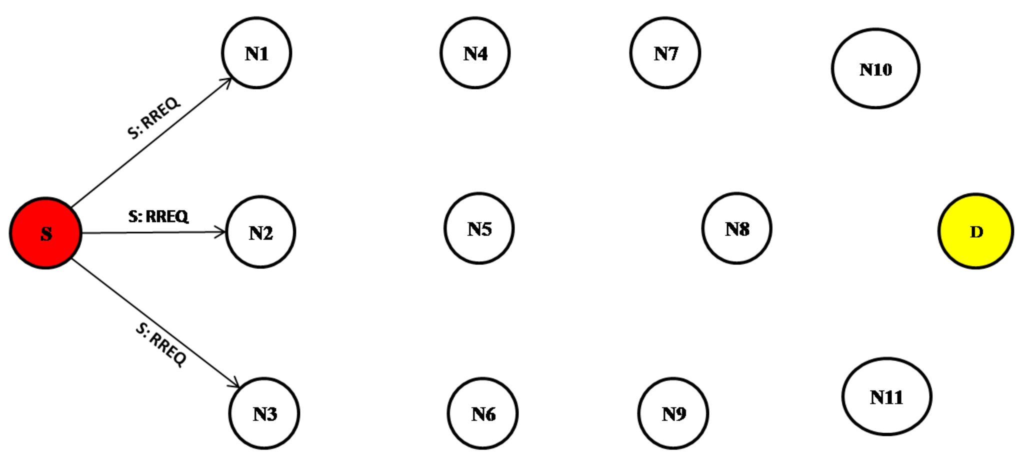

Figure 13. RREQ broadcasting by source node S at time t1.

Table 2. Computation of CAPC at nodes N1, N2 and N3 at time t1

Node	APC	Action at node OR CAPC computation at time t1
S	3	RREQ: S to N1,N2,N3
N1	4	S: (3 + 4) / 2 = 3.5
N2	5	S: (3 + 5) / 2 = 4
N3	3	S: (3 + 3) / 2 = 3

At time t2, intermediate nodes N1, N2 and N3 compute the CAPC for the RREQ packet received from S. Table 2 shows the computation of CAPC at node N1, N2 and N3. They also compute their UCN sets and if UCN is not NULL then till Acceptance_timer expires, intermediate nodes wait for duplicate RREQ packets. In this example, till Acceptance_timer expires, only one copy of RREQ is received at intermediate nodes N1, N2 and N3. Figure 14 shows rebroadcasting of RREQ packets from intermediate nodes N1, N2 and N3. The connecting arrows are only shown for the nodes which receive and accept the RREQ packets. Table 3 shows the computation of CAPC at nodes N4, N5 and N6.

At time t3, nodes N4, N5 and N6 rebroadcast the RREQ packet with the updated fields of CAPC and UCN. At each node, after Acceptance_timer expires and if the UCN set is not null then the updated CAPC of RREQ packets is compared and the RREQ packet with least value of CAPC is selected for rebroadcasting. Figure 15 shows the rebroadcasting of the RREQ having the least value of CAPC from intermediate nodes N4, N5 and N6. Table 4 shows the computation of CAPC at node N7 and N9.

Figure 16 shows that at time t4, only nodes N7 and N9 rebroadcast the RREQ packet having least value of CAPC to all neighbor nodes. Whereas, as acceptance timer of node N8 has not expired it still waits for duplicate RREQ packet. Node N8 also receives the RREQ packet from node N7 and N9. Table 5 shows the computation of CAPC at node N8, N10 and N11.

Table 3. Computation of CAPC at nodes N4, N5 and N6 at time t2

Node	APC	Action at node OR CAPC computation at time t1	Action at node OR CAPC computation at time t2
S	3	RREQ: S to N1,N2,N3	---
N1	4	S: (3 + 4) / 2 = 3.5	RREQ: N1 to N4,N5
N2	5	S: (3 + 5) / 2 = 4	RREQ: N2 to N4,N5,N6
N3	3	S: (3 + 3) / 2 = 3	RREQ: N3 to N5,N6
N4	4	---	N1: (3.5 + 4) / 2 = 3.75
			N2: (4 + 4) / 2 = 4
N5	3	---	N1: (3.5 + 3) / 2 = 3.25
			N2: (4 + 3) / 2 = 3.5
			N3: (3 + 3) / 2 = 3
N6	2	---	N2: (4 + 2) / 2 = 3
			N3: (3 + 2) / 2 = 2.5

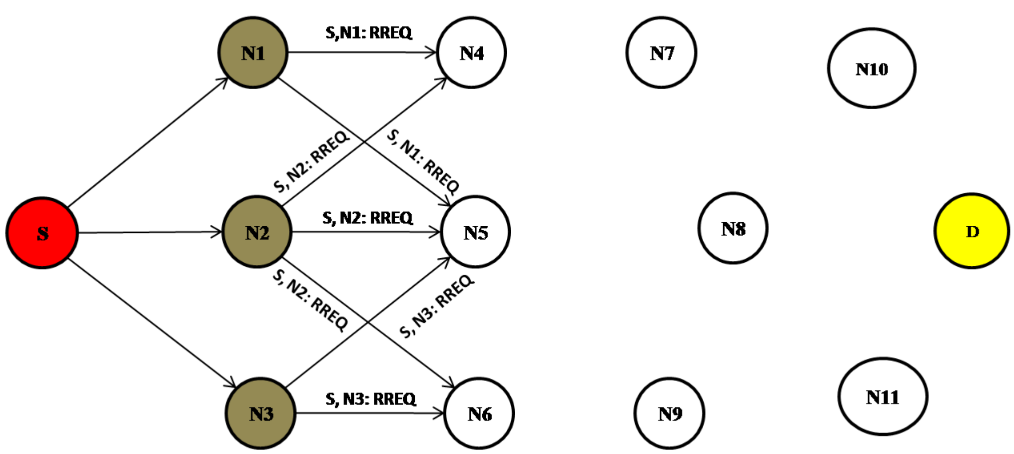

Figure 14. RREQ rebroadcasting by intermediate nodes N1, N2 and N3 at time t2.

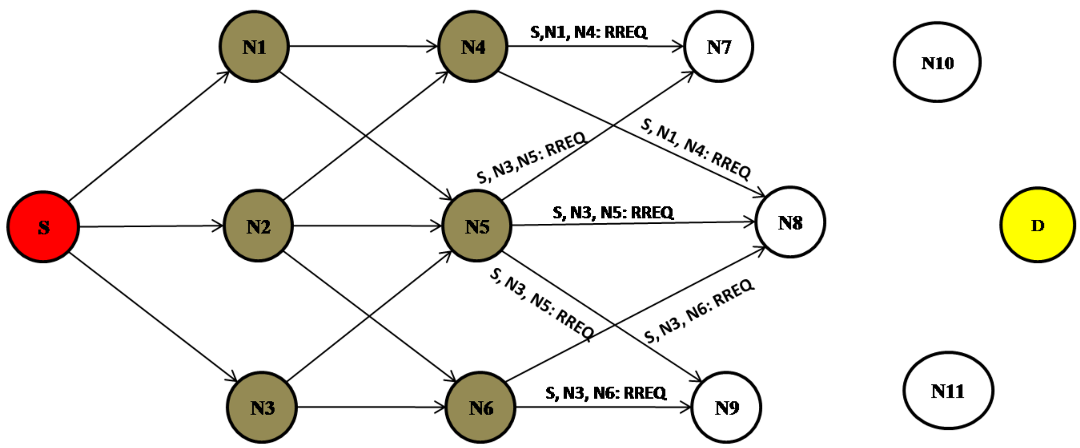

Figure 15. RREQ rebroadcasting by intermediate nodes N4, N5 and N6 at time t3.

Table 4. Computation of CAPC at nodes N7 and N9 at time t3

Node	APC	Action at node OR CAPC computation at time t1	Action at node OR CAPC computation at time t2	Action at node OR CAPC computation at time t3
S	3	---	---	---
N1	4	S: (3 + 4) / 2 = 3.5	---	---`
N2	5	S: (3 + 5) / 2 = 4	---	---
N3	3	S: (3 + 3) / 2 = 3	---	---
N4	4	---	N1: (3.5 + 4) / 2 = 3.75	---
			N2: (4 + 4) / 2 = 4	
N5	3	---	N1: (3.5 + 3) / 2 = 3.25	---
			N2: (4 + 3) / 2 = 3.5	
			N3: (3 + 3) / 2 = 3	
N6	2	---	N2: (4 + 2) / 2 = 3	---
			N3: (3 + 2) / 2 = 2.5	
N7	3	---	---	N4: (3.75 + 3) / 2 = 3.37
				N5: (3 + 3) / 2 = 3
N8	6	---	---	Acceptance_timer of N8 has not expired hence it will wait for more duplicate RREQ packet from neighbor nodes
N9	3	---	---	N5: (3 + 3) / 2 = 3
				N6: (2.5 + 3) / 2 = 2.75

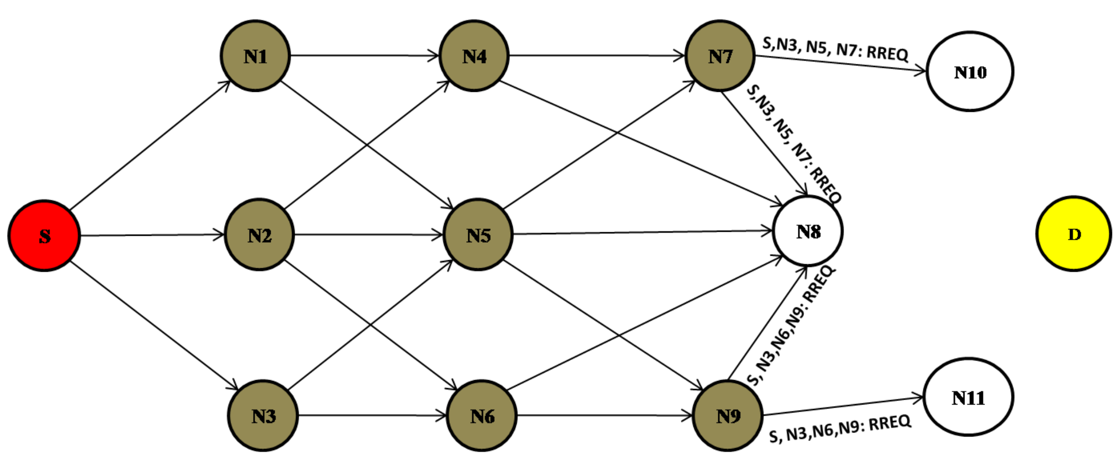

Figure 16. RREQ rebroadcasting by intermediate nodes N7 andN9 at time t4.

Table 5. Computation of CAPC at nodes N8, N10 and N11 at time t4.

Node	APC	Action at node OR CAPC computation at time t1	Action at node OR CAPC computation at time t2	Action at node OR CAPC computation at time t3	Action at node OR CAPC computation at time t4
S	3	---	---	---	---
N1	4	S: (3 + 4) / 2 = 3.5	---	---	---
N2	5	S: (3 + 5) / 2 = 4	---	---	---
N3	3	S: (3 + 3) / 2 = 3	---	---	---
N4	4	---	N1: (3.5 + 4) / 2 = 3.75	---	---
			N2: (4 + 4) / 2 = 4		
N5	3	---	N1: (3.5 + 3) / 2 = 3.25	---	---
			N2: (4 + 3) / 2 = 3.5		
			N3: (3 + 3) / 2 = 3		
N6	2	---	N2: (4 + 2) / 2 = 3	---	---
			N3: (3 + 2) / 2 = 2.5		
N7	3	---	---	N4: (3.75 + 3) / 2 = 3.37	---
				N5: (3 + 3) / 2 = 3	
N8	6	---	---	---	N4: (3.75 + 6) / 2 = 4.87
					N5: (3 + 6) / 2 = 4.5
					N6: (2.5 + 6) / 2 = 4.25
					N7: (3 + 6) / 2 = 4.5
					N9: (2.75 + 6)/2 = 4.37
N9	3	---	---	N5: (3 + 3) / 2 = 3	---
				N6: (2.5 + 3) / 2 = 2.75	
N10	5	---	---	---	N7: (3 + 5) / 2 = 4
N11	3	---	---	---	N9: (2.75 + 3)/2 = 2.87

Figure 17 shows that at time t5, nodes N8, N10 and N11 rebroadcast the RREQ packet to their neighbor nodes. At destination node, CAPC values are computed for every RREQ packet received from different routes. It selects the RREQ packet with the least CAPC value, generates the Route Reply (RREP) packet for selected RREQ packet and sends it to source node through the reverse path. Figure 18 shows RREP packet transmission from destination to source node through the reverse path.

In Table 6, we have summarized the computation of

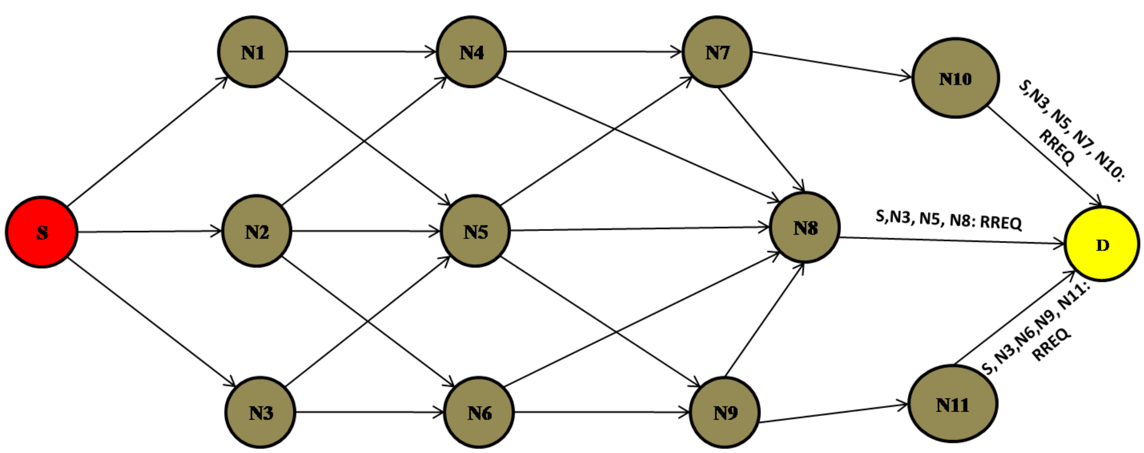

Figure 17. RREQ rebroadcasting by intermediate nodes N8, N10 and N11 at time t5.

Table 6. Computation of CAPC at D to select RREQ packet with least value of CAPC at time t5.

Node	APC	Action at node OR CAPC computation at time t1	Action at node OR CAPC computation at time t2	Action at node OR CAPC computation at time t3	Action at node OR CAPC computation at time t4	Action at node OR CAPC computation at time t5
S	3	---	---	---	---	---
N1	4	S: (3 + 4) / 2 = 3.5	---	---	---	---
N2	5	S: (3 + 5) / 2 = 4	---	---	---	---
N3	3	S: (3 + 3) / 2 = 3	---	---	---	---
N4	4	---	N1: (3.5 + 4) / 2 = 3.75 N2: (4 + 4) / 2 = 4	---	---	---
N5	3	---	N1: (3.5 + 3) / 2 = 3.25 N2: (4 + 3) / 2 = 3.5 N3: (3 + 3) / 2 = 3	---	---	---
N6	2	---	N2: (4 + 2) / 2 = 3 N3: (3 + 2) / 2 = 2.5	---	---	---
N7	3	---	---	N4: (3.75 + 3) / 2 = 3.37 N5: (3 + 3) / 2 = 3	---	---
N8	6	---	---	---	N4: (3.75 + 6) / 2 = 4.87 N5: (3 + 6) / 2 = 4.5 N6: (2.5 + 6) / 2 = 4.25 N7: (3 + 6) / 2 = 4.5 N9: (2.75 + 6)/2 = 4.37	---
N9	3	---	---	N5: (3 + 3) / 2 = 3 N6: (2.5 + 3) / 2 = 2.75	---	---
N10	5	---	---	---	N7: (3 + 5) / 2 = 4	---
N11	3	---	---	---	N9: (2.75 + 3)/2 = 2.87	---
D	2	---	---	---	---	N8: (4.25 + 2) / 2 = 3.12 N10: (4 + 2) / 2 = 3 N11: (2.87 + 2) / 2 = 2.43

CAPC and selection of RREQ packet based on the least value of CAPC. We can see that the destination node selects the RREQ packet which has arrived through the path S – N3 – N6 – N9 – N11 and has the least CPAC value of 2.43. It generates the RREP packet and sends it back to the source node through the reverse path D - N11 - N9 – N6 – N3 – S. After receiving the RREP packet at source node, actual data packets are transmitted through the selected route (S – N3 – N6 – N9 – N11- D). Table 6 shows the computation of CAPC at node D. The above working example shows that our routing protocol LUNAR is able to achieve the load balancing in the network using the Active Path Count APC and Cumulative Active Path Count CAPC computation during route discovery from source to destination. Although the concept of Uncovered Neighbor (UCN) set for minimizing the route request broadcasting is not shown in the working example, it helps to reduce the route request storm in the network.

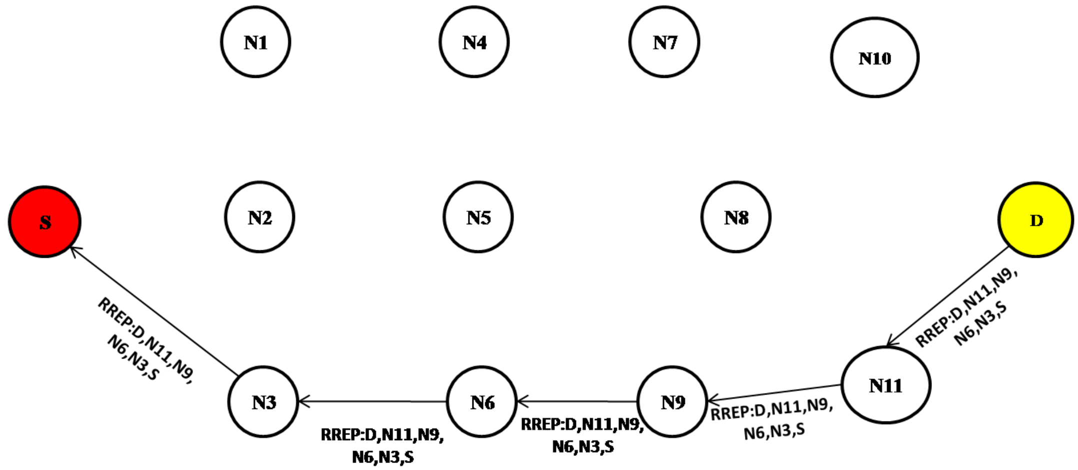

Figure 18. RREP through reverse path at time t6.

4. Performance Evaluation using Network Simulator

We used ns-2.35[6] for our simulations. In this section we describe the simulation setup. We compared LUNAR with the existing routing protocols AODV[2] and NCPR[7]. To evaluate the routing protocols following simulation parameters have been considered: number of connections, node density (number of nodes per unit area), interface queue length (buffer at a node), simulation area, transmission range of a node, node mobility (speed). Using various combinations of these control parameters, we evaluated the routing protocols using multiple performance metrics. The performance metrics considered for evaluation of the routing protocols are Normalized Routing Overhead (NRO), End-to-End Delay (EED) and Packet Loss Rate (PLR), as described below. We conducted two sets of simulations i) static scenario i.e. all nodes with 0 m/s mobility and ii) random mobility i.e. each node has a random speed between 0 to 10 m/s. In both the sets of simulation the performance of the protocols are analyzed by varying node density: 10 to 300 nodes, interface queue length: 20 to 100, number of connections: 3 to 20, simulation area: 500 * 500 m^2 to 1500 * 1500 m^2 and Transmission range: 100m to 500m. Simulation setup details are given in Table 2.

Table 7. Simulation Setup

Characteristics	Parameter	Value
MAC and Physical Characteristics	Simulator	NS 2.35
	MAC Type	802.11 g
	Signal Propagation Model	Two Way Ground
	Channel Type	Wireless Channel
	Antenna Model	Omni
	Simulation Time	100 sec
Network and Traffic Characteristics	Routing protocols	AODV, NCPR and LUNAR
	Traffic Type	TCP/FTP
	Data Payload	512 bytes/packet
	Maximum packet rate per source	4 packet /sec
	Network Topologies Used	Static OR Mobile
	Interface Queue and Type	Droptail / PriQueue
Performance Parameters Detail	Interface Queue Length	20, 50 and 100 packet
	Node Density = Number of nodes / simulation area	10, 20, 50, 100, 200 and 300 nodes
	Node Mobility	0, 5 and 10m/s
	Number of source to destination pairs	3, 5, 10, 20
	Simulation Area	500m X 500m , 1000m X 1000m, 1500m X 1500m
	Transmission range	100m, 250m, 500m

5. Simulation Results and Observations

We have simulated the protocols using all the combinations of the parameter listed in Table 2. In all we conducted a total of 2000+ experiments. The results shown are an average of 10 runs for each experiment. In this paper we report the results of a representative set of experiments for both static as well as mobile topology. We have grouped the results by the performance metrics. We discuss simulation results in five categories by varying one control parameter at a time and keeping all other control parameter values constant a) varying number of nodes (node density), b) varying number of connections, c) varying node mobility, d) varying queue length, and e) varying transmission range.

5.1 Performance with Variation in Node Density

5.1.1 Normalized Routing Overhead

Figure 19 shows the Normalized Routing Overhead (NRO) with different network density, for the control parameters as stated in the figure caption. We observe and compare NRO values of different routing protocols for both sparse and dense network. We have considered minimum 10 nodes for a sparse network and 300 nodes for a dense network. Node density is an important control parameter which influences the connectivity in an ad hoc network. The number of routing packets in LUNAR are restricted by computation of UCN set and Hello packets broadcasting is also controlled by using RREQ packets as a substitute (No Hello packet transmission by a node if a node has already forwarded RREQ packet). Hence irrespective of increase in number of nodes in the network, the LUNAR protocol is able to minimize the control packet traffic. The NRO of LUNAR is lesser than AODV and NCPR by 43% and 31% respectively on an average. These results indicate that LUNAR protocol is efficient as compared to the other three protocols.

5.1.2 End-to-End Delay

Figure 20 shows the End-to-End Delay (EED) with different network density, for the control parameters as stated in the figure caption. We observe and compare EED values of different routing protocol for both sparse and dense network. EED of the network depends on the number of connections and available queue size at the interface of the network nodes. As number of connections increase in the network (increase in data traffic), data packets face the problem of queuing delay. In case of AODV and NCPR, route discovery process selects shortest path (using minimum number of hops) in the network. In contrast, LUNAR selects the intermediate nodes having maximum available routing table size during the construction in the network. This is ensured in LUNAR by APC and CAPC computation at every intermediate node. The EED of LUNAR is lesser by 32% and 24% compared to AODV and NCPR respectively on an average.

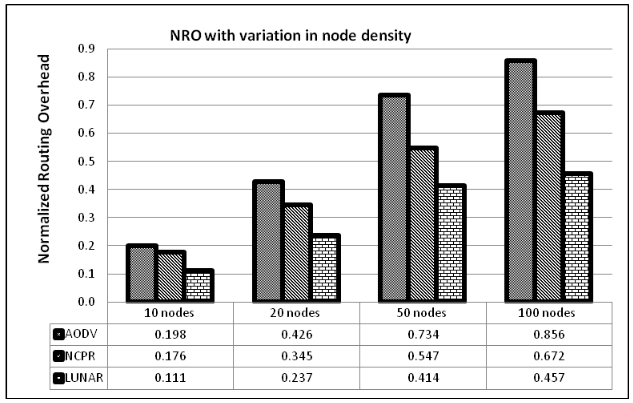

Figure 19. NRO with variation in node density [Node mobility = 0 m/s, simulation area = 1000 * 1000m^2, transmission range = 250m, simulation time = 100sec, 10 connections, queue = 100 packets, initial energy = 10 Joules].

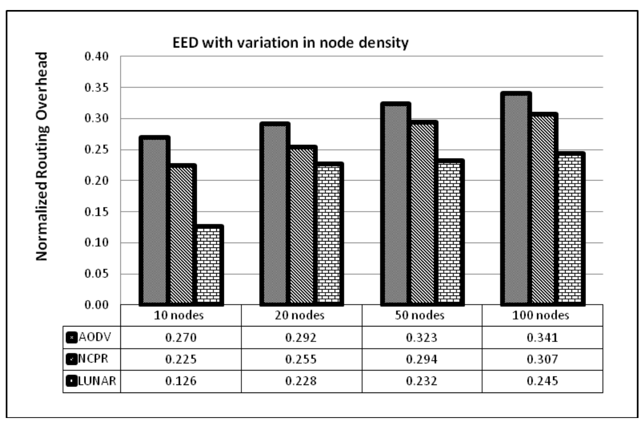

Figure 20. EED with variation in node density [Node mobility = 0 m/s, simulation area = 1000 * 1000m^2, transmission range = 250m, simulation time = 100sec, queue = 20 packets, 10 connections, initial energy = 10 Joules].

5.1.3 Packet Loss Rate

Figure 21 shows the Packet Loss Rate (PLR) with different network densities, for the control parameters as stated in the figure caption. We compared the PLR values of different routing protocols for both sparse and dense networks.

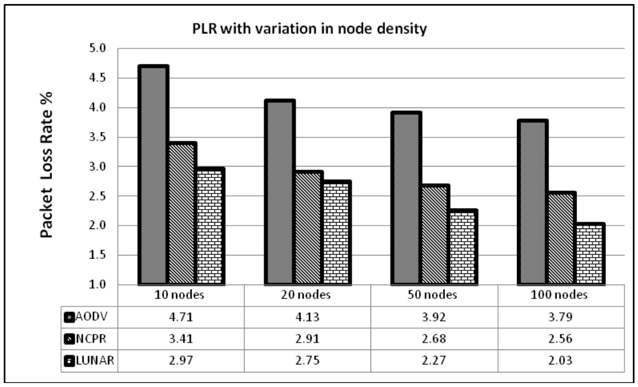

Figure 21. PLR with variation in node density [Node mobility = 0m/s Simulation area = 1000 * 1000m^2, transmission range = 250m, simulation time = 100sec, queue = 20 packets, 10 connections, initial energy = 10 Joules].

We observe that when the queue size is 20 packets and number of connections are fixed at 10 then as the numbers of nodes increase in the network the PLR decreases. This happens because as the node density increases a number of alternate nodes (and hence the node buffers) would be available while forming the route between any source-destination pair. PLR of LUNAR is lower by 39% and 13%, on an average, as compared to AODV and NCPR respectively.

5.2 Performance with Variation in Node Mobility (speed)

5.2.1 Normalized Routing Overhead

Figure 22 shows the Normalized Routing Overhead (NRO) for different node mobility in the network, for the control parameters as stated in the figure caption. We observe and compare NRO values of different routing protocols for low as well as high node mobility. Mobile nature of the network nodes frequently changes the network topology. Change in topology leads to frequent route breakages. Due to which route discovery is invoked frequently in highly mobile networks in case of AODV and NCPR. The NRO of LUNAR is lesser than AODV and NCPR by 54% and 40% respectively, on an average.

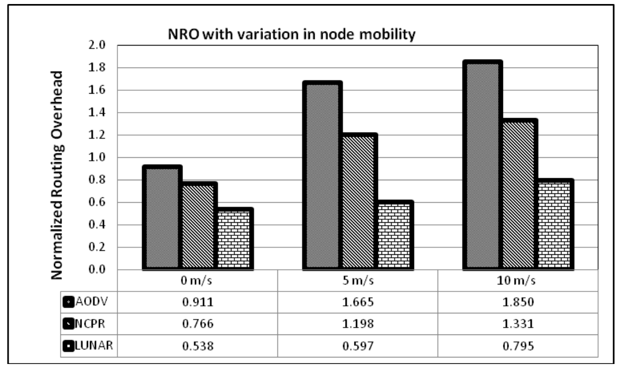

Figure 22. NRO with variation in node mobility (speed) [Simulation area = 1000 * 1000m^2, transmission range = 250m, simulation time = 100sec, queue = 100 packets, 200 nodes, 10 connections, initial energy = 10 Joules].

5.2.2 End-to-End Delay

Figure 23 shows the End to End Delay (EED) with variation in node mobility of the network, for the control parameters as stated in the figure caption. We observe and compare EED values of different routing protocol for both low as well high speeds of network nodes. Mobile nature of the network node changes the network topology continuously. Change in topology leads to frequent route breakages. This leads to the retransmission of some of the data packets which are unable to reach to the destination node because of route failure and delay increases. The LUNAR protocol reduces the EED by computing the UCN set which gives the information about neighbor nodes. The EED of LUNAR is lesser around 21% and 15% compared to AODV and NCPR respectively, on an average.

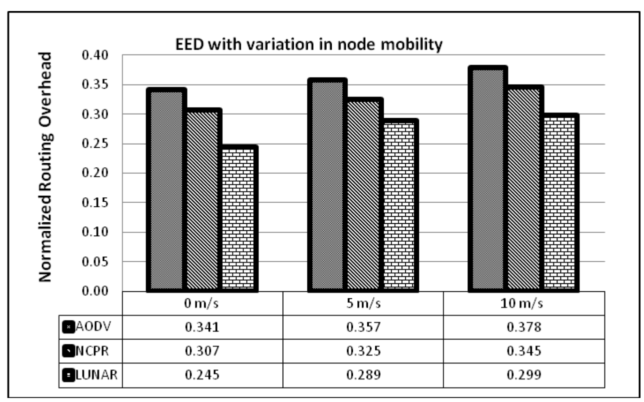

Figure 23. EED with variation in node mobility (speed) [Simulation area = 1000 * 1000m^2, transmission range = 250m, simulation time = 100sec, queue = 20 packets, 10 connections, 100 nodes, node initial energy = 10 Joules].

5.2.3 Packet Loss Rate(PLR)

Figure 24 shows the Packet Loss Rate (PLR) with variation in node mobility of the network with the queue size fixed at 100 packets and numbers of nodes are fixed at 100. We compare the PLR values of different routing protocols for both low as well high speed of network nodes. The PLR of the LUNAR protocol is lower by 36% and 9 %, on an average, as compared to AODV and NCPR respectively.

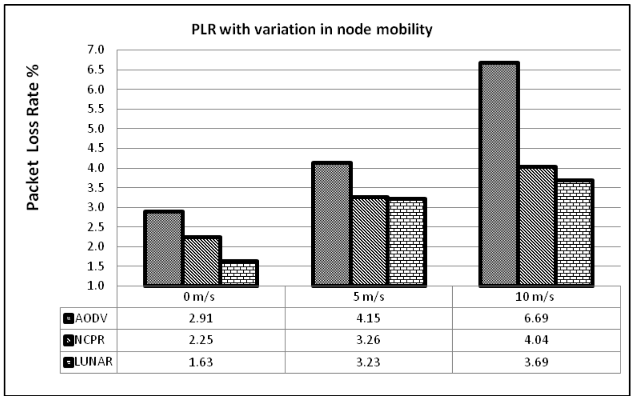

Figure 24. PLR with variation in node mobility (speed) [Simulation area = 1000 * 1000m^2, transmission range = 250m, simulation time = 100sec, queue = 100 packets, 10 connections, 100 nodes, initial energy = 10 Joules].

6. Conclusion

In this paper, we discussed working of Load eqUilibrium Neighbor Aware Routing (LUNAR) protocol through algorithm, flow-chart and working example. LUNAR protocol is proposed to solve the broadcast storm problem and to reduce the routing overhead in MANETs. LUNAR dynamically calculates the Cumulative Active Path Count (CPAC) at every intermediate node to decide whether to rebroadcast the route request packet in the network. Neighbor coverage information is used to compute the uncovered neighbor set (UCN) which further reduces the redundant broadcasts. Our simulation results confirm that LUNAR generates lesser rebroadcast traffic compared to AODV and NCPR. The NRO for LUNAR is lesser by 31-54% as compared to protocols simulated. Further, the PLR is decreased by 9-35% and the EED is decreased by 15-32%. Further investigations are needed to determine the cause of minor improvements in PLR and EED and to determine the performance of LUNAR as the number of active connections increase.

7. References

1. Jain SA, Raisinghani VT. Load eqUilibrium Neighbor Aware Routing (LUNAR). IEEE INDICON-14, (1-6), 2014.
2. Perkins C, Belding-Royer E, and Das S. Ad Hoc On-Demand Distance Vector (AODV) Routing, IETF RFC 3561, 2003.
3. Johnson D, Hu Y, and Maltz D. Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR) for IPv4. IETF RFC 4728. 2007. 15. p. 153–181.
4. Mohammed A, Ould-Khaoua M, Mackenzie LM, Perkins C, and Abdulai JD. Probabilistic Counter-Based Route Discovery for Mobile Ad Hoc Networks, Int'l Conf. Wireless Comm. and Mobile Computing: Connecting the World Wirelessly (IWCMC '09). 2009. p. 1335–1339.
5. Souihli Oussama, Frikha Mounir, Hamouda Mahmoud Ben. Load-balancing in MANET shortest-path routing protocols. Elseveir ScienceDirect. Ad Hoc Networks. 2009. p. 431–442
6. Tran Duc A, Raghavendra Harish. Congestion Adaptive Routing in Mobile Ad Hoc Networks. IEEE Transactions on Parallel And Distributed Systems. 2006. 17.
7. Zhang Xin Ming, Wang En Bo, Xia Jing Jing, Sung Dan Keun. A Neighbor Coverage-Based Probabilistic Rebroadcast for Reducing Routing Overhead in Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing. 2013. 12(3).
8. Ni SY, Tseng YC, Chen YS, Sheu JP. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In: Proc. ACM/IEEE MobiCom. 1999. p. 151–162
9. Toh Chai Keong, Le Anh-Ngoc, Cho You-Ze. Load Balanced Routing Protocols for Ad Hoc Mobile Wireless Networks. IEEE Communication Letter, 2009.
10. Williams B, Camp T. Comparison of Broadcasting Techniques for Mobile Ad Hoc Networks. In: Proc. ACM MobiHoc, 2002. p. 194–205
11. Hui Xu, Xianren Wu, Hamid R. Sadjadpour, JJ. Garcia-Luna-Aceves. A Unified Analysis of Routing Protocols in MANETs. In: IEEE Transactions On Communications. 58(3), March 2010.
12. Kim J, Zhang Q, Agrawal DP. Probabilistic Broadcasting Based on Coverage Area and Neighbor Confirmation in Mobile Ad Hoc Networks. In: Proc. IEEE GlobeCom, 2004.
13. Abdulai JD, Ould-Khaoua M, Mackenzie LM. Improving Probabilistic Route Discovery in Mobile Ad Hoc Networks. In: Proc. IEEE Conf. Local Computer Networks. 2007. p. 739–746.
14. Haas Z, Halpern JY, Li L. Gossip-Based Ad Hoc Routing. In: Proc. IEEE INFOCOM, 2002. 21. p. 1707–1716.
15. Peng W., Lu X. On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks. In: Proc. ACM MobiHoc, 2000. p. 129–130.
16. Abdulai JD, Ould-Khaoua M, Mackenzie LM, Mohammed A. Neighbour Coverage: A Dynamic Probabilistic Route

Discovery for Mobile Ad Hoc Networks. In: Proc. International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS ’08), 2008. p. 165–172.
17. Song JH, Wong V, Leung V. Load-Aware Ondemand Routing (LAOR) Protocol for Mobile Ad Hoc Networks. In: Proc. 57th IEEE VTC-Spring, 2003. p. 1753–57.
18. Johansson P. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad Hoc Networks. In: Proc. ACM/IEEE MobiCom. 1999.
19. Hassanein H, Zhou A. Routing with Load Balancing in Wireless Ad hoc Networks. In: Proc. 4th ACM MSWiM ‘01, 2001. p. 89–96.
20. Toh CK. Associativity-Based Routing for Ad Hoc Mobile Networks. Wireless Pers. Commun., 1997. 4(2). p. 103–139.
21. Altalhi AH, Richard G. Load-Balanced Routing through Virtual Paths: Highly Adaptive and Efficient Routing Scheme for Ad Hoc Wireless Networks. 23rd IPCCC, 2004.
22. Senthil Kumaran T, Sankaranarayanan V. Early congestion detection and adaptive routing in MANET. Elseveir ScienceDirect: Egyptian Informatics Journal. 2011. p. 165–175.
23. Ahmadi Mitra, Mohammad Shojafar, Khademzadeh Ahmad, Badie Kambiz, Tavoli Reza. A Hybrid Algorithm for Preserving Energy and Delay Routing in Mobile Ad-Hoc Networks. Springer: Wireless Pers Communication, 2015. p. 2485–2505.
24. Smail Omar, Cousin Bernard, Mekki Rachida, Mekkakia Zoulikha. A multipath energy-conserving routing protocol for wireless ad hoc networks lifetime improvement, Springer: EURASIP Journal on Wireless Communications and Networking. 2014.
25. Wenjing Yang , Xinyu Yang , Shusen Yang , Dongxu Yang. A greedy-based stable multi-path routing protocol in mobile ad hoc networks. Elseveir ScienceDirect: Ad Hoc Networks. 2011. p. 662–674.
26. Reddeppa Reddy Y, Raghavan SV. SMORT: Scalable multipath on-demand routing for mobile ad hoc networks. Elseveir ScienceDirect: Ad Hoc Networks. 2007. p. 162–188.
27. Marina Mahesh K, Das Samir R. Ad hoc on-demand multipath distance vector routing. WILEY Publications: Wireless Communications And Mobile Computing. 2006. p. 969–988.
28. Cadger Fraser, Curran Kevin, Santos Jose, Moffett Sandra. Location and mobility aware routing for multimedia streaming in disaster telemedicine. Elseveir ScienceDirect: Ad Hoc Netwroks. 2016. p. 332–348.
29. Khamayseh Yaser, Darwish Omar M, Wedian Sana A. MA-AODV: Mobility Aware Routing Protocols for Mobile Ad hoc Networks. IEEE Fourth International Conference on Systems and Networks Communications, 2009.
30. Fraser Cadger, Kevin Curran, Jose Santos, Sandra Moffett, “Location and mobility aware routing for multimedia streaming in disaster telemedicine. Elseveir ScienceDirect, Ad Hoc Netwroks. 2016. 332–348.
31. Yaser Khamayseh, Omar M. Darwish, Sana A. Wedian. MA-AODV: Mobility Aware Routing Protocols for Mobile Ad hoc Networks. IEEE Fourth International Conference on Systems and Networks Communications. 2009.