Load balanced routing in mobile ad hoc networks

Vikrant Saigala\(^a\), Ajit K. Nayak\(^b\),*, Sateesh K. Pradhan\(^c\), R. Mall\(^a\)

\(^a\)Department of Computer Science and Engineering, IIT, Kharagpur, West Bengal, India

\(^b\)Silicon School of Information Technology, BBSR, Silicon Hills, Patia, Bhubaneswar, Orissa 751 031, India

\(^c\)Department of Computer Science and Application, Utkal University, Vani Vihar, Bhubaneswar, Orissa, India

Received 20 October 2002; revised 15 August 2003; accepted 5 September 2003

Abstract

We present a new protocol called Load Aware Routing in Ad hoc (LARA) networks protocol for efficient data transmission in mobile ad hoc networks. We also define a new metric for routing called traffic density to represent the degree of contention at the medium access control layer. During the route setup, this metric is used to select the route with the minimum traffic load. We have carried out extensive simulation studies to evaluate the performance of our proposed protocol vis-à-vis the existing protocols. Performance results show that our LARA protocol outperforms existing routing protocols in terms of end-to-end delay, throughput, packet delivery fraction, and battery power consumption at the nodes.

© 2003 Elsevier B.V. All rights reserved.

Keywords: Ad hoc networks; Routing; TCP; Throughput

1. Introduction

With the ever-increasing demand for connectivity, wireless communication is gaining rapid acceptance. The use of portable laptops and hand-held devices in every day life is increasing rapidly. Most of the portable communication devices have the support of a fixed base station or access point in the last-hop-wireless model. However, such a fixed base station support may not be available in many settings. Situations like natural disaster and military settings are noteworthy examples in this regard. Such situations require ad hoc networking.

Routing protocols are vital for the proper functioning of ad hoc networks. The routing issues in infrastructure-based networks are very different from routing in infrastructure-less networks like ad hoc networks. Each intermediate host between source and target node acts as a router in an ad hoc network and the topology of the network changes frequently. Therefore distribution of up-to-date information about the nodes can saturate the network. Also, late arrival of information can drive the network into instability. Besides this, another problem is that link failure due to mobility is usually very high. This affects transport layer protocols like TCP, which are primarily designed for wired networks. A lot of research has been reported [1–14] addressing these issues.

The available routing protocols can roughly be classified into three groups: table-driven, source initiated on-demand, and hybrid routing protocols. However, none of these addresses load balancing among the routes. In practice, some routes may get congested, while other routes remain unutilized. This results in poor performance of mobile ad-hoc network. Keeping this in mind, we have tried to develop a protocol, which will balance the load distribution among various routes. Our proposed protocol is also an on-demand protocol and has many features similar to that of DSR [1].

This paper is organized as follows: In Section 2 we discuss the protocols that are related to our work. Section 3 presents our Load Aware Routing Ad hoc (LARA) protocol. We analyze the LARA protocol vis-à-vis Dynamic Source Routing (DSR) protocol and Dynamic Load Aware Routing Protocol (DLAR) [5]. In Section 4 we present our simulation model. In Section 5 the implementation of our simulation experiments has been discussed. In Section 6 we present the results obtained from the simulation studies. In Section 7 we have compared our protocol with related work. Section 8 concludes this paper.
2. Related protocols

The table-driven protocols [13] require periodic exchange of routing information, irrespective of the change in network topology. This incurs significant routing overhead and unnecessary power drain at the nodes. This shortcoming of table-driven protocols often outweighs their advantages such as lower route discovery latency compared to on-demand protocols [13]. In on-demand protocols, only routes to the nodes to which a source needs to communicate are maintained by the source node. In bandwidth-limited ad hoc environments, on-demand routing protocols are better suited because of their lower routing overhead. Two well-known on-demand routing protocols are DSR protocol [1,2] and Ad hoc On-demand Distance Vector (AODV) protocol [3,4]. The hybrid routing protocols share the ideas of table driven as well as on-demand protocols. The Zone routing protocol [14] is an example of a hybrid protocol.

Most of the on-demand protocols [12] use the shortest path as their route selection metric. This scheme of route selection leads to congestion of some of the nodes in the network. It is generally believed that the performance of ad hoc networks would improve when the mobility of nodes is reduced. This appears reasonable when considering performance metrics such as packet delivery fraction and routing overhead. This may not be the case, however, when we consider packet delay or TCP throughput as the performance metrics. It was shown in Ref. [6] that the packet delay for both AODV and DSR increases as the nodal mobility is reduced. The inclination of ad hoc routing protocols to use the shortest path routes can be attributed to these results. The few innermost nodes become the backbone for most of the traffic, leading to congestion at the medium access control (MAC) layer in these nodes. This may in turn lead to high packet delays, since some nodes may carry excessive loads. This problem is further aggravated by the use of a route cache in some of the protocols. This may result in a high probability of packet drops due to congestion severely affecting the TCP performance. The heavily loaded nodes are also likely to incur high power consumption. This is clearly an undesirable situation, as it reduces battery life.

Most of the existing routing protocols do not consider the load conditions at the nodes during the setup phase; hence they are unable to take advantage of the less loaded nodes in the network topology. Routing with load balancing in wired networks has been exploited in several schemes [7]. In ad hoc networks, Associatively Based Routing (ABR) [8] considers the load as the primary performance metric. ABR, however, uses the routing load as a secondary metric. The DLAR protocol [5], on the other hand uses the number of packets buffered in the interface as the primary route selection criteria. But as we explain in Section 3.5, the interface queue length alone does not give a true picture of the actual load in 802.11 MAC based protocols. Our proposed routing protocol, LARA networks defines a new metric called traffic density, to represent the degree of contention at the MAC level. This metric is used for making the routing decisions.

3. Our proposed protocol, LARA

LARA protocol requires that each node maintain a record of the latest traffic queue estimations at each of its neighbors in a table called the neighborhood table. This table is used to keep the load information of local neighbors at each node. This information is collected through two types of broadcasts. The first type of broadcast occurs when a node attempts to discover route to a destination node. This type of broadcast is called route request. The second type of broadcasting is the hello packet broadcasting. In the event that a node has not sent any messages to any of its neighbors within a predefined timeout period, called the hello interval, it broadcasts a hello message to its neighbors. A hello packet contains the sender node’s identity and its traffic queue status. Neighbors that receive this packet update the corresponding neighbor’s load information in their neighborhood tables. If a node does not receive a data or a hello message from some of its neighbors for a predefined time, it assumes that these nodes have moved out of the radio range of this node and it changes its neighborhood table accordingly. Receiving a message from a new node is also an indication of the change of neighbor information and is handled appropriately.

Traffic queue. The traffic queue of a node is defined as the average value of the interface queue length measured over a period of time. For the node \(i\), it is defined as the average of \(N\) samples over a given sample interval

\[
q_i = \frac{\sum_{k=1}^{N} q_i(k)}{N}
\]

where \(q_i(k)\) is the \(k\)th sample of the queue length. \(q_i\) is the average of these \(N\) samples. The greater the value of \(N\), the better is the estimation of the traffic. Typically, it can be one sample/ms. The number of samples \((N)\) can typically be 50 to get a better result.

The interface queue length is averaged over a period of time to eliminate transience effects and to get a more accurate estimate of the traffic at a node over a period of time. It can be seen from Fig. 1 that the instantaneous queue length may vary rapidly with time, because of the bursty and random nature of traffic in ad hoc networks.

Traffic density. The traffic density of a node \(i\) is the sum of traffic queue \(q_i\) of node \(i\) plus the traffic queues of all its neighbors, formally

\[
Q(i) = \sum_{j \in N(i)} q_j
\]

where \(N(i)\) is the neighborhood of node \(i\) and \(q_j\) is the size of the traffic queue at node \(j\). \(Q(i)\) is the sum of traffic
queues of all the neighbors of node \( i \) plus that of node \( i \) itself.

**Hop cost.** This factor captures the transmission and propagation delay along a hop.

**Traffic cost.** The traffic cost of a route is defined as the sum of the traffic densities at each of the nodes and the hop costs on that particular route. The traffic cost of some route \( r \) is given by the following expression

\[
C(r) = \sum_{i \in r} Q(i) + \sum_{i,j \in r, i \text{ adj} j} h_{i,j}
\]

where \( i, j \) are nodes on route \( r \). \( h_{i,j} \) is the hop cost along \( i \) and \( j \). \( i \text{ adj} j \) refers to the fact that node \( j \) is adjacent to node \( i \).

### 3.1. Route discovery

LARA is an on-demand protocol and the route discovery process is similar to that of DSR. When a source has a packet to transmit to a destination for which there is no entry in the routing table, a route request process is initiated. In the route request process, the source broadcasts route request packets are broadcasted. A route request packet contains a sequence number, a source id and a destination id. Sequence number is the route record, containing the sequence of nodes taken by the route request packet; as it is propagated through the ad hoc network during the route discovery. An intermediate node listening to the request, broadcasts the request further, after appending its own traffic density to the packet. This node does not entertain any further requests with the same sequence number, source id. This process continues until the request packet reaches the destination. After receiving the first request, the destination waits for a fixed time-interval (typically 50 ms) for more route request packets (carrying the source to destination routes) to arrive. When the preset timer expires after receiving the first route request packet, the destination node selects the best route from among the candidate routes and sends a route reply to the source. The methodology for selection of the best route out of the candidate routes is described in Sections 3.2 and 3.3 for TCP and non-TCP traffic, respectively. When the source node receives the route reply, it can start data transmission. If it does not receive any route reply within a route discovery period, it can restart the route discovery procedure afresh.

### 3.2. Route selection for TCP traffic

TCP throughput drops drastically as the number of hops increases \([9]\), and then stabilizes once the number of hops becomes large (Fig. 2). This behavior is due to the TCP sender’s inability to accurately determine the cause of a packet loss. The TCP sender assumes that all packet losses are caused by congestion. Thus, when a link failure occurs, the TCP sender reacts as if congestion was the cause, reducing its congestion window and in case of time out, backing-off its retransmission time out value. Therefore, for a given TCP traffic-source, our route selection method is based on the number of hops and the traffic cost of that route. Whenever a choice has to be made, we select the route with the smallest number of hops. If there is more than one route with the same minimum number of hops, we select the route with the minimum traffic cost. This route is the one that results in the maximum throughput, as it has the least contention at the MAC level.

### 3.3. Route selection for non-TCP traffic

For each of the candidate routes, the destination determines the route’s traffic cost. The route with minimum value of traffic cost is selected. The selected route represents the route in which the packets suffer the least contention at the MAC level and hence low delay and low interference.

### 3.4. Route maintenance

Each node uses a link layer detection scheme to infer the status of the link. A node not receiving passive acknowledgement, or any hello packets from its neighbors for a certain interval is also an indication of link failure. If a link failure occurs during a data transmission session, the source is informed of the failure via a route error packet. On receiving a route error packet, the source initiates a new route search. If the data source is using TCP, an explicit link failure notification helps to avoid the slow start phase. This
otherwise would have resulted in shrinking of the congestion window and exponential back off. Also, the source queues all subsequent packets for that destination until a new route is found. However, if no route is found after a few tries, then these packets are broadcast in the network.

3.5. Working of the protocol

In mobile ad hoc networks, the wireless medium is shared among a number of devices for communication. The devices communicate among themselves directly, using radio signals, within a certain transmission range only. Within this range, only one communication channel is used, covering the entire offered bandwidth. To transmit data, mobile hosts first sense the ongoing transmissions from other nodes. The node transmits only if no other node is currently transmitting. Thus, the time required to gain access to the shared medium is directly proportional to the traffic at the neighboring nodes. Unlike point-to-point networks, packet delay is caused not only from traffic load at the current node, but also by traffic load at the neighboring nodes.

Let us examine how traffic at a node can be affected by traffic at the neighboring nodes. Let us consider the example given in Fig. 3. If the node D has a packet to transmit, it has to first gain access to the medium. The time it takes to gain access to the medium is dependent on the traffic at the nodes A–C, E and F. The bigger is the interface queue at the neighboring nodes, the more is the contention for the medium. Thus, the delay suffered by a packet at a node is dependent not only on its own interface queue but also on the interface queues of all its neighbors. Also, the probability of an interface queue reaching the capacity of the buffer (overflow condition) increases when the neighbors are heavily loaded. It means that the probability of overflow is also dependent on the traffic at the neighboring nodes.

The DSR protocol does not take into account any kind of traffic information for route selection. Therefore, whenever the number of route choices increases, it exhibits sub minimal end-to-end delay and throughput degradation. The DLAR protocol [5] considers the sum of the lengths of instantaneous interface queues of the nodes on a route during route selection hence has been found to perform better than DSR. But as we have observed earlier, the instantaneous queue length doesn’t give a good measure of the traffic at a node. The LARA protocol takes the traffic densities of the nodes into consideration.

4. Simulation model

Our simulation model consisting of source, destination and intermediate routers is shown in Fig. 4. The source and the destination nodes are static throughout the simulation period. To make the simulator simple, we have implemented a simple route discovery procedure. We assume that initially the source has a certain number of routes to the destination. These routes are shown to be disjoint in Fig. 4, as any network can be converted to such a simple disjoint form. While considering the paths between a source and a target, the intermediate nodes may be repeated in more than one path. During the route discovery procedure, the source node sends route request packets on all possible routes. The destination node, depending on the routing protocol, selects an appropriate route and sends a route reply packet back to the source.

In the simulation model, we have considered the broadcasting of route discovery packets to occur during the route discovery procedure. The number of routes between the source and the destination are made to change after every pause interval and so also the number of hops on each of the routes. This is done to reflect the changes in the topology of mobile ad hoc network due to node mobility.

The intermediate nodes remain static for a pause time, at the end of which they can change their position. This change takes place probabilistically after every pause interval. The mobility pattern is first generated using a scenario file that specifies the initial topology of the intermediate nodes and their subsequent locations in the network. The pause time should reflect the degree of mobility of nodes. The more frequently the topology changes, the lesser should be the pause time.
Our simulation experiment was carried out for a duration of 220 s, with 20 and 40 intermediate nodes in the network. All the performance metrics were computed with 20 and 40 nodes in the network, respectively, under different load conditions.

4.1. Congestion model

The load in the network is varied by changing the length of the interface queue at each of the nodes in the topology. The length of each interface queue is maintained in queue files. The queue length reflects the load and therefore the queuing delay that the packets will suffer at each of the nodes. The queue files help us to study the performance of various protocols under different load conditions. Also, this scheme helps us to control the load at each of the nodes independent of the other nodes. Study of the network behavior under such conditions is necessary, because in the ad hoc environment, it is very likely that some of the nodes are heavily loaded whereas others are lightly loaded. The candidate protocols were studied under the conditions of low, medium and heavy loads.

In our simulation environment, we have assumed that each interface queue can hold up to 50 packets awaiting transmission and is managed in a drop tail fashion. Whenever the queue length exceeds this limit, the packets that arrive last are dropped.

4.2. Link failure model

The nodes in the network are connected through bidirectional links. The link failure module in the simulator as described in Section 5 independently generates the link failures. The bandwidth of each link is assumed to be 1.6 Mbps; the propagation time is assumed to be zero as the nodes are very close in the ad hoc network topology. The links are assumed to be error free. This means that barring the failures generated by the link failure module, the links function perfectly. Though this error free model does not reflect a practical wireless scenario, but it does not significantly affect the comparative study of the candidate protocols under consideration. However, it helps in making the simulator simple. Had errors been considered, it would have only resulted in either increase or decrease of the values of the performance metrics by some constant factor for all the protocols. The link failure module periodically breaks the links on the active routes by setting their states to ‘down’ and back to ‘up’ after a specified delay, depending on the mobility pattern. The faster the change in the topology, the greater is the frequency of link failure. It is also assumed for simplicity that on a multi-hop route, only a single link can fail on a particular route at any time.

4.3. Transmission model

We have simulated both TCP and non-TCP (UDP) traffic over ad hoc networks. We have chosen CBR (Constant Bit Rate) data traffic as the representative non-TCP traffic. The CBR source is assumed to transmit data at 4 bits/s. We have employed a single traffic source in each simulation. The version of TCP used is TCP Tahoe. For each run over a given scenario, the data transmission is done between the same two nodes in order to ensure uniformity.

4.4. Traffic model

For generating the traffic queues, we have assumed that the traffic at each node varies randomly. Due to high bit error rate, high frequency of link failure, and unpredictable mobility of nodes in the topology, it is difficult to arrive at any specific model for traffic in ad hoc networks. To verify that our results hold irrespective of the exact traffic pattern, we have carried out our simulation studies under commonly used traffic models such as uniform, Pareto, Poisson’s, Gaussian, etc. and have found that the comparative results are almost independent of the exact traffic pattern subject to the assumption that there is no buffer overflow at the nodes. Therefore, the results presented in Section 6 should be considered to be the indicative performance under the commonly used traffic models.

4.5. MAC protocol model

We have simulated the IEEE 802.11 MAC protocol with the help of queue files [10]. The MAC layer defines two different access methods, the distributed coordinated function (DCF) and point coordinated function. In our DCF simulation, we assume that each node in the network is always surrounded by five other nodes as shown in Fig. 5.

To make the working of the MAC protocol simple, we have made the following assumptions:

- There is no packet loss due to contention for the channel.
- Each node gets a chance to transmit a packet in a round robin fashion; provided it has packets to send.

Fig. 5. A simplified MAC protocol model.
If each of the neighbors of node $i$ has exactly one packet in its interface queue, and node $i$ has four packets, then the last packet at node $i$ will be sent only after the five packets of neighbors and three packets at node $i$ have been transmitted. That amounts to a total delay of eight packets transmission time. This is a simplified DCF that we have used in our simulation.

Based on the above assumptions, we create queue files for each of the nodes in the topology. Each queue file entry contains four values: time, the length of the interface queue, and the traffic density of node $i$, and the time to transmit an arriving packet under the given load conditions, according to our simplified DCF. These values are generated in the queue files for different load conditions in the network. These files help us to control the load at all the nodes independently and enable us to study the performance of the candidate protocols under the conditions of low, medium and high loads.

5. Implementation

We have implemented our simulator using C programming language. For our experiments, we have used fixed packets of size 200 bytes. In Fig. 4 we have shown only a single route between source and destination having two intermediate nodes to explain the model. However, we consider all the routes between the source and the destination. Also these routes may keep changing over the period of simulation. The appropriate modules of the routing protocols, which perform the routing function run at each of the nodes.

The execution of the simulator starts with the main module (Fig. 6), which invokes the appropriate traffic source (TCP or UDP) as has been specified by the user. Let us examine how the model works with the UDP source and the DSR protocol running at all the nodes. The UDP source always has data to send to the destination. Whenever the source node module receives data from the UDP source and no route to the intended destination exists, it initiates a route discovery procedure by calling the route discovery module.

The route discovery module first loads the scenario file using the scenario update module. The scenario file decides how many routes are there between the source and destination and the number of hops in each route. Then the route discovery module sends route discovery packets on all the possible routes between the source and destination. The intermediate nodes add their id to this packet and forward the packet to the next node. The route discovery packet that reaches the destination first is selected by the destination, and a route reply packet is sent back to the destination containing the source route.

Once a route to the destination is found, the source module starts sending the data to the destination via intermediate node modules. At each node, the packets are buffered until they can be transferred to the next node. The delay that a packet suffers is determined from the queue file of that node, with the help of queue update module. After the packet has been delayed for the required time in the buffer, it is ready for transmission to the next node.

Before a packet can be sent to the next node, the link failure module checks whether it is time for the link to be broken. If the time is reached, the link is made down. The data packets on this link are dropped and an ERROR packet is sent to the source node to indicate that the host is unreachable. The source node can then restart the route discovery procedure. If the time has not been reached,
the link is assumed to be up and the data packets are forwarded to the next node after delaying it corresponding to the transmission time into account. With this the packet reaches the intended destination where it is consumed by the UDP sink.

6. Simulation results

The simulation experiment was performed using 30 different scenario files and 10 different queue files for a simulation duration of 220 s for each of the protocols. All the protocols; DSR, DLAR and LARA were tested under similar load conditions and similar patterns of link failures. The protocols were tested with 20 and 40 nodes, under low, medium and high load conditions. The simulation results in terms of throughput, end-to-end

Fig. 8. Variation of max deviation factor with pause time.

Fig. 9. Throughput under low load conditions with (a) 20 nodes, (b) 40 nodes.

Fig. 10. Throughput under medium load conditions with (a) 20 nodes, (b) 40 nodes.
Fig. 11. Throughput under high load conditions with (a) 20 nodes, (b) 40 nodes.

Fig. 12. End-to-end delay under low load conditions with (a) 20 nodes, (b) 40 nodes.

Fig. 13. End-to-end delay under medium load conditions with (a) 20 nodes, (b) 40 nodes.
delay, and packet delivery fraction are reported in the following sections.

6.1. Throughput

Figs. 7–11 show the performance results for TCP data source and Figs. 12–17 show those for non-TCP data sources. Fig. 7 shows the time each protocol takes to transmit 1 MB of data across the network from the source to the destination. The LARA protocol takes the least amount of time to transmit 1 MB, followed by DLAR and DSR. LARA selects the least loaded path from among the available paths and therefore selects the one having the least number of back offs and packet drops.

6.2. Max deviation factor

We define deviation factor for a node at any instant as the ratio of the product of queue length and number of data packets arriving at the node at that instant, to the sum of product of queue length and number of data packets at all other nodes in the network. The maximum of these values is the max deviation factor.

Formally

\[
MDF = \max \left( \frac{q_i d_i}{\sum_{j \in N} q_j d_j} \right)
\]

where \(d_i\) is the number of data packets arriving at the node and \(q_i\) is the traffic queue at node \(i\).

Maximum Deviation factor reflects a protocol’s ability to balance the load among all the available nodes and hence avoiding excessive power consumption by some nodes. A smaller value of Maximum deviation factor is desirable. As shown in Fig. 8, the LARA protocol gives the better result for maximum deviation factor among all the protocols studied.

From Fig. 9(a) it can be observed that under low load conditions DSR performs poorly. This may be explained by the fact that, it chooses the first arriving route request packet and does not take into account the number of hops. On
the other hand, the performance of LARA and DLAR are comparable when the number of nodes are about 20 or lower. LARA performs better as the number of possible routes between the source and destination increases, as shown in Figs. 10 and 11, whereas DSR shows only a minor improvement. As load increases, more and more nodes in the network start getting congested in the network. DSR due to its inability to distinguish the better paths from the bad ones, shows throughput degradation as TCP frequently backs off. LARA outperforms DLAR under these conditions.

6.3. End-to-end delay

As shown in Fig. 12 at low load conditions the end-to-end delay for all the three protocols is almost the same irrespective of the number of nodes in the network. We can see that the end-to-end delay increases for all the protocols with increase in load as can be seen in Figs. 13 and 14. This can be explained by the fact that, due to increased contention at the MAC layer, the packets now have to wait in the interface for longer time before being transmitted. Here, DSR suffers the maximum delay as it often routes the packets around heavily loaded nodes. DLAR makes a better choice than DSR. LARA makes best decision among all the three protocols.

6.4. Packet delivery fraction

From Fig. 15, it can be seen that, under low load conditions, the performance of the three protocols is indistinguishable from each other as very few packets are dropped even if a protocol makes a bad decision. At smaller pause time, the delivery fraction is small because only a few packets enroute have to be dropped when the link fails. When the load in the network builds up, the interface queue at some of the loaded nodes starts to
overflow leading to packets being dropped at these nodes as shown in Figs. 16 and 17. This is more pronounced in DSR as it is not able to route packets not involving the loaded nodes. DLAR gives better results than DSR, but LARA outperforms both the protocols using its enhanced route selection technique.

7. Comparison with related work

In our simulation study, we have compared our LARA protocol with two other protocols: DSR protocol [1], which does not take into account any load condition during route setup and DLAR protocol [5], which considers the length of interface queue of all the nodes on the path during route setup.

The DSR protocol does not consider the load condition of the nodes during route setup. This leads to high contention at the MAC layer in some of the centrally located nodes, resulting in high end-to-end delay and increased probability of congestion. The DLAR protocol on the other hand makes a better route selection than DSR, and shows a remarkable improvement in performance. But a drawback of DLAR protocol is that it does not take into account the subtleties of the contention based 802.11 MAC protocol.

LARA protocol makes an enhanced route selection attempt based on traffic density and traffic cost which leads to better performance than DLAR and DSR as far as throughput, end-to-end delay and packet delivery fraction are concerned. This is so because LARA balances the load among all the nodes in the network topology and thus prevents excessive queue build up and consequent packet drops.

8. Conclusion

In this paper, we have proposed a new routing protocol called the LARA protocol. During the route discovery procedure, the destination node selects the route with the minimum traffic cost, which basically reflects the contention at the MAC level, for the non-TCP source. For TCP sources, it takes into account both the number of hops and the traffic cost of the route. This methodology of route selection helps the routing protocol to avoid congested routes. This helps to uniformly distribute the load among all the nodes in the network, leading to better overall performance.

Since the DSR protocol does not take into account any kind of load information during the route discovery procedure, some of the nodes in the network may get highly overloaded leading to fast battery exhaustion and high end-to-end delay. The DLAR protocol, which uses the interface queue length of the nodes as a measure of load, performs better than DSR. But the interface queue does not give a true picture of the load in contention-based MAC protocols such IEEE 802.11 where the transmission medium is shared among many nodes. We have studied the performance of our LARA protocol through a comprehensive simulation study. It is found to outperform both the DLAR and the DSR protocols as far as end-to-end delay, throughput, packet delivery fraction, and max deviation factor are concerned.

We have planned a refinement to the route maintenance procedure. In the existing protocol we do not consider the condition of the route, once it has been selected for data transmission. Congestion can be handled better if the intermediate nodes can send a message to the source node in case of any imminent congestion. This would help in avoiding the slow start in case of TCP source, which results in throughput degradation, and unnecessary packet drops in the case of non-TCP source. But, it needs to be carefully investigated to check whether this tradeoff is worthwhile considering the fact that, the increase in the route re-establishment delay can lead to packets being buffered at the source until a route to the destination is found.

References