Ann. Telecommun. (2013) 68:539–551
DOI 10.1007/s12243-012-0339-2

Link failure and congestion-aware reliable data delivery mechanism for mobile ad hoc networks

Manowarul Islam · Abdur Razzaque · Mahfuzur Rahman Bosunia · Atif Alamri · Mohammad Mehedi Hassan

Received: 3 May 2012 / Accepted: 17 October 2012 / Published online: 21 November 2012

Abstract Link failures and packet drops due to congestion are two frequently occurring problems in mobile ad hoc networks, degrading the network performance significantly. In this paper, we propose a link failure and congestion-aware reliable data delivery (LCRDD) mechanism that jointly exploits *local packet buffering* and *multilevel congestion detection and control* approaches for increasing the data delivery performance. On the detection of link failure or congestive state, an LCRDD intermediate node buffers the incoming data packets at the transport layer queue and resumes transmission when the route is repaired locally. In addition, LCRDD's *multilevel congestion detection* helps it to take the most appropriate action proactively. Thus, it offers increased reliability and throughput and decreased end-to-end packet delivery delay and routing overhead compared to state-of-the-art protocols, as shown in results of performance evaluations carried out in network simulator v-2.34.

Keywords Mobile ad hoc network · Reliable data delivery · Link failure handling · Congestion control · Packet buffering

1 Introduction

Mobile ad hoc network (MANET) [1] is a non-infrastructure network having wirelessly connected mobile nodes. Because of its dynamic network topology [1], links between the nodes may be broken frequently. Another serious problem in MANETs is the packet drops due to congestion in the network. As a result, achieving reliable and timely data delivery is a challenging problem. We develop reliable data delivery methodologies in the presence of route failures and congestions in MANETs.

The nodes in MANETs are independent of each other and can move frequently, causing routes to break. For this reason, the lifetime of an established route is very short and hence the data delivery efficiency decreases. In local repair ad hoc on-demand distance vector (LRAODV_LP) [2], based on link prediction, if a node detects that the signal strength goes below a predefined threshold, it initiates a fresh route discovery rather than sending error message backward. However,

M. Islam
Department of Information and Communication Technology, Mawlana Bhashani Science and Technology University, Tangail, Bangladesh
e-mail: manowar_cse_du@yahoo.com

A. Razzaque (✉) · M. R. Bosunia
Department of Computer Science and Engineering, University of Dhaka, Dhaka 1000, Bangladesh
e-mail: m_a_razzaque@yahoo.com

M. R. Bosunia
e-mail: mahfuz_babu10@yahoo.com

A. Alamri · M. M. Hassan
Information Systems Department and Chair of Pervasive and Mobile Computing, College of Computer and Information Sciences, King Saud University, Riyadh, Kingdom of Saudi Arabia

A. Alamri
e-mail: atif@ksu.edu.sa

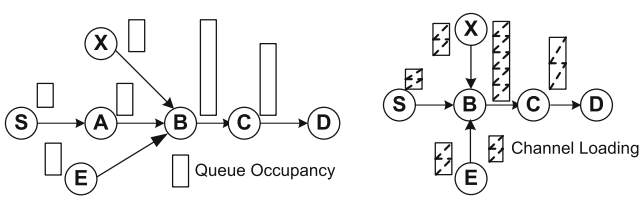
M. M. Hassan
e-mail: mmhassan@ksu.edu.sa

packets might be dropped at the intermediate nodes if the local route discovery takes longer period of time.

Another important problem in MANETs is the dropping of data packets due to congestion in the network. Congestion may occur due to link failure, queue overflow or channel or media overloading [3]. The congestion leads to packet losses, throughput degradation of networks, and wastage of time and energy for congestion recovery. Consider possible routes between source nodes S, X, and E and the destination node D in Fig. 1a, namely $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$, $X \rightarrow B \rightarrow C \rightarrow D$, and $E \rightarrow B \rightarrow C \rightarrow D$, respectively. All the sources S, X, and E send data packets via the intermediate node B. The node B receives data packets and forwards them to destination node D.

When only one route $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$ is active, B can deliver data packets properly as queue occupancy is normal. However, when all the three routes are active, queue occupancy at node B increases rapidly. It may not deliver all the received data packets if its incoming data packet rate becomes higher than the outgoing rate. As a result, packets may drop since queue overflow occurs at node B. For an active route, these dropped packets may traverse a longer path. So the network may suffer from high overhead, long delay, and high packets loss. Thus, throughput of MANETs decreases for network congestion.

Again, since there is no fixed infrastructure, all nodes in a MANET share a single transmission channel; many nodes may contend for the channel simultaneously to transmit data packets, increasing packet collisions in the network. In such a situation, the congestion collapse [4] may occur when no node will be able to transmit their data packets. Consider Fig. 1b, node B forwards data packets from three source nodes S, X, and E and thus the channel loading in the network raises to a high level. Therefore, the node will not be able to transmit data packets successfully. As a result, queue overflow occurs at the node and packets start dropping. In the literature, we find split multipath routing (SMR) [5], which uses multiple routes to split traffic and mitigate congestion; nodes in congestion-adaptive routing protocol (CRP) [6] use bypass routes to mitigate congestion, etc. However, their problem is that the multiple routes maintenance overhead affects the network performance significantly.



(a) Queue overflow (b) Channel overloading

Fig. 1 Causes of congestion in the network

The two hop routing protocol (THR) of Ela Kumar et al. [7] has much similarity with our work. Each THR node maintains a two hop routing table and transmits data packets if a route is available to the destination, and, on the failure of links, the intermediate nodes start buffering incoming data packets in a separate memory module. Thus, the THR enhances data delivery performances of the network. However, the requirement of separate memory module is not only costlier but also not implementable for all devices in the network. Unlike the aforementioned works, we address both the problems of route failures and congestion in the network simultaneously in this work, and our approach has some unique features mentioned below.

In this paper, we propose a link failure and congestion-aware reliable data delivery mechanism, named link failure and congestion-aware reliable data delivery (LCRDD), that introduces *packet buffering* concept during link failure detection and local route discovery. The nodes on an active route buffer incoming packets at their local transport layer queues, and on finding a new path, resume their transmissions. As a result, packet dropping rate of the network decreases. In addition to that, we employ a *multilevel congestion detection and control* mechanism at the source and intermediate nodes that can judiciously take the most appropriate decision for congestion control in the network proactively. The contributions of this work are summarized below:

- LCRDD provides an efficient buffering mechanism to buffer the packets when link failure occurs.
- LCRDD achieves an efficient congestion control mechanism, where nodes can detect multiple congestion levels of the network and take proper control actions to reduce the packet dropping. Thus, it avoids the congestion.
- Nodes in the network can handle both congestion and link failures more effectively, and thus it ensures end-to-end reliable and efficient data delivery.

The rest of this paper is organized as follows. We describe related works in Section 2 and network model and assumptions in Section 3. Our proposed LCRDD mechanism is presented in Section 4, and the simulation results are presented in Section 5 along with discussion. Finally, we conclude the paper in Section 6.

2 Related works

A significant research effort has been observed in recent years on handling route failures and congestion in mobile ad hoc networks. We can categorize them into four different types based on their working principles.

The first category of works use backup route(s) on the failure of primary route. In AODV with backup routing (AODV-BR) [8] and multipath AODV [9], source nodes create alternative routes to the destination, and on failure of any one of them, nodes deliver data packets using an alternative route; however, they suffer from two problems: stale route and duplicate packet transmission. In AODV-based backup routing scheme(AODV-BBS) [10], each node maintains two hop neighborhood information for finding alternative routes, but the maintenance of multiple alternative paths is difficult, costly, and time-consuming, which in turn reduces data delivery performances of the network.

The second category of routing protocols uses multiple routes to balance traffic loads on the event of congestion [4] or route failures and thus improve the network performance. For example, a distributed multipath dynamic source routing (DSR) protocol [11] improves QoS with respect to end-to-end reliability; SMR [5] uses multiple routes to split traffic and mitigate congestion; nodes in CRP [6] use bypass routes to mitigate congestion, etc., but the problem is that multiple route maintenance overhead affects the network performance significantly.

Thirdly, some research works focus on link failure of MANETs using local recovery process. Most of them try to find out a local path whenever a link failure occurs in the network. For example, in local repair AODV based on link prediction, LRAODV_LP [2], if a node detects that the signal strength goes below a predefined threshold, it initiates a fresh route discovery rather than sending error message backward. In Implicit Backup Routing-AODV (IBR-AODV) [12], a neighbor of an active route temporarily stores overheard data packets and acts like a backup node. Whenever any link failure occurs in the network, this backup node creates new route to the destination and sends data packets. Here, the problem is that many neighbor nodes have to store data packets unnecessarily. Also, more than one neighbor nodes may try to send the same data packet on the failure of primary link and this results in duplicate transmissions of data packets and reduces network efficiency.

The fourth category of works handles link failures using packet buffering or route caching. Ela Kumar et.al. proposed a THR [7] protocol, in which if a node needs to transmit data, it first checks its own routing table which contains route of two hop distance nodes. If any route is found, only then data packets are delivered; otherwise, a fresh route discovery process is initiated. If route failure occurs, the intermediate nodes start packet buffering in a separate physical memory module. However, the requirement of a separate memory module is not only costlier but also not implementable for all devices in the network.

The aforementioned protocols can handle either congestion or link failure. To cope up with both the problems simultaneously, we propose a link failure and congestion-aware reliable data delivery mechanism, namely LCRDD, in which each node is capable of buffering data packets at local transport layer. When a node detects that the network is congested or link towards the destination node is broken, it buffers the incoming packets into its own transport layer buffer. Later, on finding a new path, the node resumes transmission process from local buffer. Also, our LCRDD implements a congestion-aware reliable data delivery mechanism by using multilevel congestion detection and control mechanisms.

3 Network model and assumptions

We consider a MANET with large number of nodes communicating over multi-hop paths with each other. The nodes may be mobile phones, laptops, or PDAs, which may move frequently. The nodes use on-demand routing protocol AODV [13] or DSR [11] for establishing multi-hop routing paths. In what follows, we describe three important processes assumed in our network: link failure detection process, local route repair process, and transport layer queue (TQ) management process.

3.1 Link failure detection process

Since nodes in MANETs are self-organizing and independent from each other, they can move randomly and frequently. As a result, topology of MANETs changes very frequently, causing failure of links between neighbor nodes. Also, the link failure may occur due to highly reduced received signal strength. Link failure leads to route failure between nodes and reduces the network throughput as well. Therefore, the link failure detection is a subjective research issue in wireless networks. Link failure detection can be performed using either periodic HELLO messages [14] or link layer feedback [15, 16]. These HELLO messages are local advertisements for the continued presence of the link.

The nodes in our proposed LCRDD mechanism exchange HELLO messages periodically to ensure link connectivity. The following two parameters are associated with a HELLO message: $HELLO\_INTERVAL$ is the maximum time interval between two consecutive HELLO message transmissions and $HELLO\_LOSS_{\text{allowed}}$ is the maximum number of loss of HELLO messages that a node can tolerate before it declares the link breakage. If a node does not receive any HELLO message from its neighbor node within $HELLO\_LOSS_{\text{allowed}} \times HELLO\_INTERVAL$, then the node assumes that the link is not available for data transmission.

3.2 Local route repairing process

Link failure degrades the performance of a network significantly. Local repair of route can reduce the effects of link failure to some extent. There are a number of local route repair techniques in the literature. Query localization technique [17] keeps the history of an old path and floods the route request (RREQ) message to some restricted area during local query process. Sung-Ju Lee and Mario Gerla proposed AODV-BR (Backup Route) [18] which can improve the performance of the AODV routing protocol by providing multiple alternate routes. Expanding ring search [19] is another local repairing technique to finding local path during link failure. We will use one of these methods for finding partial path during link breakage.

3.3 Transport layer queue management process

Since a mobile node in MANETs can act both as a router and a host, in our LCRDD mechanism, all intermediate routing nodes use a separate TQ to store incoming data packets, when needed. We assume, as long as there is no link failure or the network does not become heavily congested, nodes act like a conventional router and simply forward data packets. But when the network is heavily congested or link failure occurs, nodes use their TQs for buffering incoming packets.

Figure 2 shows that the node A is acting as router in normal operation (indicated by solid line). After receiving packets, it just forwards the packets to next hop using lower three (physical, data link, and network) layers of TCP/IP model. But in case of link failure, node A uses its transport layer queue to buffer data packets (indicated by dotted line). If a new partial path is found, node A starts its transmission process from transport layer queue as well as normal delivery of packets. The dotted lines indicate performing a transmission process

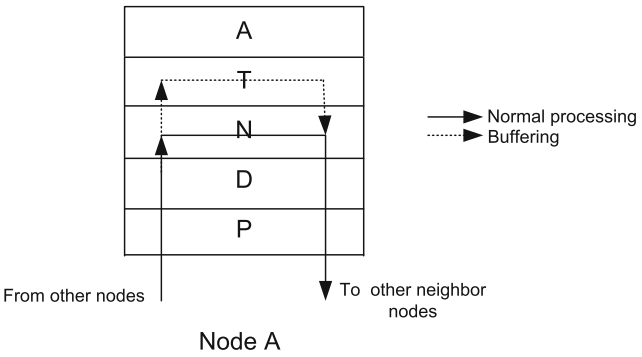


Fig. 2 A MANET node works both as a router and a host

from local TQ and solid lines indicate normal data delivery process.

Now a question arises—*Why does our LCRDD use separate queue at transport layer for buffering the packets?* Many source nodes in a MANET might deliver data packets to many other destination nodes simultaneously. An intermediate node may work as a forwarder of many such source–destination pairs, and it needs to store incoming data packets in the network layer queue for a very small period of time before forwarding the packets from the queue by examining the addresses of corresponding destination nodes. However, in case of link failure or congested state, nodes have to store data packets for relatively longer period of time since they have to wait for a partial path. For this reason, storing data packets into network layer queue during link failure or congestive states might hamper packet forwarding of other good connections and thus cause queue overflow at the node. As a result, packet dropping rate at the node will increase and overall throughput of the network will decrease. Furthermore, storing data packets at the transport layer during link failure may facilitate the intermediate node to ensure the end-to-end reliability from that node, rather than from the source. In this case, the intermediate node does not stamp any new sequence number with the stored packets and takes special care for their delivery. For the aforementioned reasons, an LCRDD intermediate node uses separate queue in transport layer to buffer incoming data packets, allowing smooth packet forwarding of other connections through the node.

In Fig. 3, three source nodes S_1, S_2, and S_3 are transferring data packets to three different destination nodes D_1, D_2, and D_3 respectively. Suppose that the links B to E and B to C are broken. If node B uses network layer queue for buffering process, B has to buffer all the data packets of source nodes S_2 and S_3 and at the same time, it has to forward the data packets

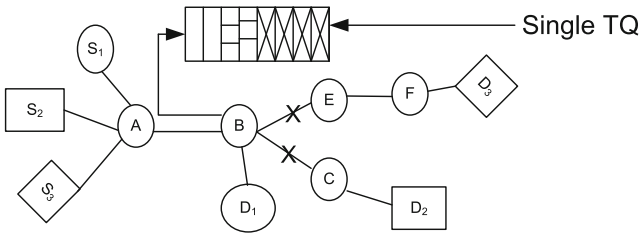


Fig. 3 The use of transport layer queues at intermediate nodes

of destination node D_1. As a result, buffering overflow will occur at network layer queue and thus network will be congested, leading to loss of data packets. To reduce packets dropping rate, in LCRDD, node B uses separate transport layer queue TQ for buffering data packets for destination nodes D_2 and D_3.

4 The LCRDD architecture

Our proposed LCRDD mechanism exploits cross-layer buffering capability of nodes, spanning through network and transport layers. In LCRDD, when an intermediate node detects a link failure, it creates a route disconnection notification (RDN) message and sends toward the source node. It then buffers incoming packets into the transport layer queue and starts a local query to find any partial path to the destination. All the intermediate nodes, on receiving RDN message, stop further delivery of the data packets and store the incoming data packets in their local transport layer queues. When the source node receives the RDN control message, it just stops transmission of data packets and waits for a new partial path to the destination.

If any partial path is found, node creates another notification message called route successful notification (RSN) and sends it back to the source. It then resumes its transmission process from transport layer queue. All the intermediate nodes resume their transmission from local buffer after receiving RSN message. When the source node receives this RSN message, it resumes the transmission process. So unlike the traditional routing protocols [1, 20, 21], the source node does not need to deliver all the data packets. The source node only retransmits the packets that are dropped during the link failure. Thus packet loss rate is minimized and overall throughput increases. But if no partial path is found, the intermediate node creates another control message called Route Unsuccessful (RUN) message and sends it back to the source node. The source node, on receiving the RUN message, starts a new route discovery process.

The LCRDD uses congestion-aware data delivery mechanism so that nodes in the network can easily identify the congestion level of the network and take appropriate action. For a heavily loaded network, nodes will send an ALERT message to previous nodes not to increase the data forwarding rate. In a congested network, nodes buffer their incoming packets to reduce packet loss rate. Thus, LCRDD controls the congestion in MANET.

To handle link failure and congestion and to provide reliable data delivery mechanism, nodes buffer their incoming packets in their local queues. Each intermediate node maintains a queue in transport layer for multiple destination nodes with the help of cross-layer interface. For example, in Fig. 3, node B delivers data packets to three destination nodes D_1, D_2, and D_3. As the links B to C and B to E are broken, node B starts to buffer the incoming data packets in its transport layer queue and it does not need to stamp any new sequence number for each packet during buffering process. Thus, the proposed transport layer queue is only for storing the packets temporarily. What follows, we discuss the LCRDD operations at different mobile nodes in detail and describe the link failure and congestion-aware reliable data delivery mechanism.

4.1 Nodal operations

In LCRDD, nodes can buffer packets in their local transport layer queues for handling congestion as well as link failure cases. Therefore, every mobile node performs some distinct functions in contrast to conventional mobile nodes. What follows, we describe the overall operations in detail.

4.1.1 LCRDD in source nodes

In most of the routing protocols, the source node starts route discovery process by creating a message called RREQ message and floods the message in the network [1]. The source node then waits for a route to the destination; on reception of RREQ message, the destination node sends back a route reply (RREP) message. Whenever the source node gets the RREP, it starts the data transmission process. If there occurs a link failure due to node mobility or any other reasons such as limited bandwidth, lacks of power, etc., and if the source node gets the route failure notification message, it stops further transmission of data packets. Then, it starts a new fresh route discovery process to deliver the data packets. Due to link failure, source node needs to retransmit a lot of data packets, especially when the packets traverse a longer path. Thus data delivery rate

Algorithm 1 LCRDD in any source node $s \in S$

```
1. Begin
2. s broadcasts RREQ message
3. s waits until a message(msg) is received
4. if (msg=RREP) then
5.       s starts delivering data packets
6. else if (msg=RDN) then
7.       s stops transmission and waits for a partial path
8. else if  (msg=RSN) then
9.       s resumes transmission process
10. else if  (msg=RUN) then
11.       s initiates a fresh route discovery process
12. else
13.       s waits for a new message
14. end if
15. End
```

Algorithm 2 LCRDD in any intermediate node $n \in N$

```
1. Begin
2. if (n detects a link failure) then
3.       n initiates local query for partial route discovery
4.       n starts buffering data packets in TQ
5.       n waits for partial route for T_t duration
6.       if (partial path is found) then
7.             n resumes transmission process.
8.       else
9.             n sends RUN message to source
10.      end if
11. else
12.      n delivers the data packets
13. end if
14. End
```

decreases and overall performance of the network also decreases.

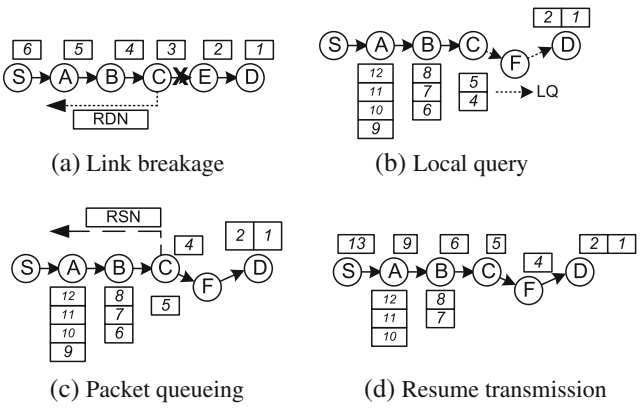
In our model, the source node stops its transmission process during link failure and waits for a local path rather conducting a fresh route discovery process for that particular destination. When a partial path is established, the source node is notified and it then resumes the transmission process. The Algorithm 1 shows the overall operation of a source node $s \in S$, where S is the set of all source nodes.

4.1.2 LCRDD in intermediate nodes

In conventional routing protocols, all the intermediate nodes act as routers [1]. Each node receives the incoming packets and forwards them to the next node. When a link is broken, the intermediate node sends a route error message to the source node and drops the packets for that destination. As a result, a lot of data packets are dropped due to link failures.

As mentioned before, in LCRDD, when an intermediate node detects a link failure, it starts buffering all the incoming packets in its local transport layer queue TQ rather dropping them. It then sends a RDN message towards the source node and all the intermediate nodes, on receiving RDN message, store incoming data packets into their TQs and wait for local path. In the meantime, the detecting node starts a local query to find a partial path using any of the existing algorithms, mentioned in Section 3, and waits for a predefined short period of time T_t. If a partial path is found within T_t, the node starts transmission process from its TQ and sends RSN message toward the source node. All the intermediate nodes resume their packet delivery process from their local queues after receiving the RSN message. However, if no partial path is found, the intermediate node sends the RUN message toward the source node. After getting RUN message, the source node starts a new fresh route discovery process. The Algorithm 2 shows the overall operations of any intermediate node, $n \in N$, where N is the set of all intermediate nodes in the route.

The following example describes the operations of the above two algorithms. In Fig. 4a, the source node S is sending data packets via route $S \rightarrow A \rightarrow B \rightarrow C \rightarrow E \rightarrow D$ to destination node D. Suppose node C detects that the link between C and E has broken. It then generates a RDN message and sends it back to the source node to inform about the link failure. Also, node C starts buffering incoming packets for node D and, at



(a) Link breakage (b) Local query

(c) Packet queueing (d) Resume transmission

Fig. 4 Link failure handling in LCRDD

the same time, it initiates a local query process to find a partial path. Figure 4b shows local query process. After receiving RDN message, all the intermediate nodes stop data delivery for the destination D and buffer the incoming data packets in their local TQs. Figure 4c shows buffering process at each intermediate nodes (A, B, and C). When the source S receives the RDN message, it stops its transmission and waits for a partial route.

After finding a partial path $C \rightarrow F \rightarrow D$, node C sends back a RSN message toward the source S and resumes its transmission process from its local TQ. The intermediate nodes A and B and the source node S resume their transmission processes on receiving the RSN message. Note here that the source node S does not need to retransmit all the data packets. Figure 4d shows the retransmission process whenever a route reconstruction process is successful. However, if node C's local query process fails to find a partial route within T_t, it sends the RUN message toward the source S. Upon reception of this RUN message, the source node S starts discovering a new route.

4.2 Congestion control in LCRDD

In MANETs, the data traffic from many source nodes may converge at a point of the network, where some nodes might be congested due to their low packet forwarding rate compared to corresponding arrival rate. As a result, their local buffers might overflow and packets may be dropped [22]. Such a congestive state also leads to the following problems in the network: (1) longer end-to-end packet delivery delay and (2) reduced network throughput.

To cope up with the aforementioned drawbacks, in this paper, we propose a congestion-aware data delivery mechanism that works as follows. At each intermediate node, we measure the congestion level and piggybacks that information toward the source node with each transmitted packet so that appropriate control actions can be taken in time. We use two bits control flag in both data packets and acknowledgement packets, referred to as *congestion notification (CN)* flag. Every node in an active route sets this flag when it forwards data packets. The value of the CN flag detects the congestion level of the network according to Table 1. From the value of the CN flag, the neighborhood nodes can easily be informed about the congestion status of the network and they can take proper actions to handle congestion. What follows, we describe how a node detects the congestion level and assigns the value of CN and how the congestion is controlled based on the value of CN flag.

Table 1 Congestion notification

Value of CN	Congestion level
00	Lightly loaded
01	Loaded
10	Heavily loaded
11	Congested

4.2.1 Detection of congestion level

Based on queue occupancy, here we use an early congestion detection technique by which a node can detect the current congestion status. We use minimum and maximum thresholds, $Q_{\min}$ and $Q_{\max}$, respectively, for queue occupancy at any node as follows:

$$Q_{\min} = l \times Q_{\text{size}}, \tag{1}$$

$$Q_{\max} = h \times Q_{\text{size}}, \tag{2}$$

where, l and h are two control parameters; in our simulation, we set $l = 0.5$ and $h = 0.9$, respectively. If the queue length of a node is less than the $Q_{\min}$, then we can say the network is lightly loaded, e.g., queue occupancy is less than 50 %; if the queue length is greater than $Q_{\min}$ but less than $Q_{\max}$, then it is operating in the safe region; and, if the queue length is greater than $Q_{\max}$, the node is considered as congested. Even though the above thresholds help to identify congestive or non-congestive states, they don't protect nodes moving from non-congestive state to congestive one. In support of implementing congestion-aware data delivery mechanism, we introduce a warning threshold parameter Q_{warn}, defined as follows:

$$Q_{\text{warn}} = w \times Q_{\text{size}}, \tag{3}$$

where, w is a weight factor, and in our simulation, we choose $w = 0.8$.

We then calculate average queue occupancy of a node every after a certain interval using exponentially weighted moving average formulae as follows:

$$Q_{\text{avg}} = (1 - \alpha) \times Q_{\text{avg}} + Q_{\text{curr}} \times \alpha, \tag{4}$$

where, α is a weight factor and Q_{curr} is the current queue size. Now, based on the value of Q_{avg}, we determine the value of CN flag as follows:

- if $Q_{\text{avg}} < Q_{\min}$, then CN = 00
- if $Q_{\text{avg}} >= Q_{\min}$ and $Q_{\text{avg}} < Q_{\text{warn}}$, then CN = 01
- if $Q_{\text{avg}} >= Q_{\text{warn}}$ and $Q_{\text{avg}} <= Q_{\max}$, then CN = 10
- if $Q_{\text{avg}} > Q_{\max}$, then CN = 11

4.2.2 Congestion control mechanism

In our proposed LCRDD, a node takes proper actions to control the congestion according to the values of CN bits. If the value of CN flag at a node is '00', it assumes the network is lightly loaded. In such case, the node performs its normal operations. It allows the other nodes of the network to transmit packets through it. So the node accepts new RREQ messages from new source and rebroadcasts to create new routes through it. When the value of CN is '01', it discards new RREQ messages, allowing no new route through it in order to avoid any future congestive states; however, in this case, the sources of the existing routes may increase their traffic rates passing through this node. If the value of the CN is '10', the network becomes heavily loaded. So, further increasing of data arrival rates from source nodes will lead the network to fall into congestive state. In this case, the node generates a new control message called ALERT message and sends back toward every source node so that they do not increase the data forwarding rates. Thus, our proposed LCRDD controls the network in ahead of time and implements a congestion-aware reliable data delivery mechanism. But if a node detects the value of CN is '11', this means the network has already fallen into congestive state and it then starts buffering data packets in TQ and stops forwarding packets afterwards. Table 2 shows the operations taken by a node for different congestive states.

4.3 Packet buffering

In this section, we describe in detail the packet buffering mechanism. As mentioned earlier, nodes in our network use separate queue in transport layer (TQ) when link failure or congestion occurs. Storing packets at separate queue not only increases the end-to-end transmission efficiency but also reduces the queuing overheads at the network layer.

We define a cross-layer interface between network layer and transport layer as shown in Fig. 5. The interface has two components: receive interface "R" and delivery interface "D". The interface "R" receives the packets from network layer queue and puts them into the transport layer queue when a link failure or congestion occurs. Similarly, all the intermediate nodes buffer the packets in their transport layer queues for that corresponding destination using their cross-layer interfaces. Whenever, a partial path is found, the network layer of the node informs transport layer through the interface. Then the interface "D" delivers the data packets to network layer and the node resumes data transmission process. Similarly, all the intermediate nodes resume their transmission processes. As a result, the source node does not need to retransmit all the data packets during a link failure, increasing the overall throughput of the network.

Table 2 Actions taken by a node to control the congestion

Congestion level	Actions
Lightly loaded	Normal operation - no change
Loaded	Stop forwarding any RREQ message
Heavily loaded	Send ALERT message to all sources
Congested	Start buffering packets

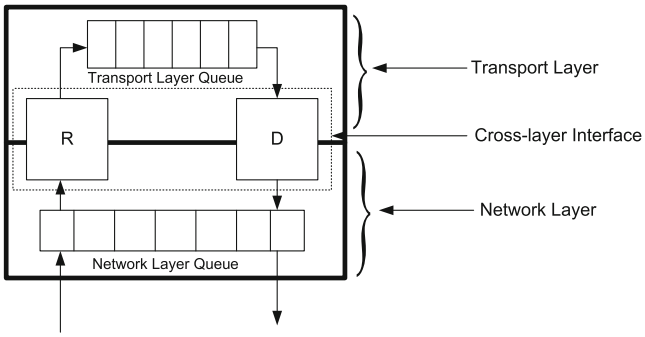


Fig. 5 Interface between network and transport layers

In what follows, we describe the aforementioned buffering mechanism with the help of an example. Consider Fig. 6a, where node B forwards data packets of three sources S_1, S_2, and S_3 to three destination nodes D_1, D_2, and D_3, respectively. Figure 6a shows the normal situation in which all the data packets are queued at network layer queue of node B.

Now, suppose the link between node B and F is broken. As soon as node B detects the link failure, it buffers all the packets for the destination node D_2 in its TQ and continues its normal operation for other destinations. Meanwhile, B starts a local query process for destination node D_2 and sends a RDN message towards the source node S_2. All the intermediate nodes buffer their packets for the destination node D_2 in their TQs on receiving RDN message. Node S_2 stops its transmission and waits for a reply from node B. Figure 6b describes this situation.

Whenever the node B finds a partial path $B \rightarrow N \rightarrow D_2$, it sends RSN control message toward source node S_2 and resumes its transmission process from transport layer queue. All the intermediate nodes also resume their transmission processes same way after getting the RSN message. Such a mechanism decreases packet dropping rate and node S_2 does not need to transfer all the packets. Figure 6c shows node B performs normal

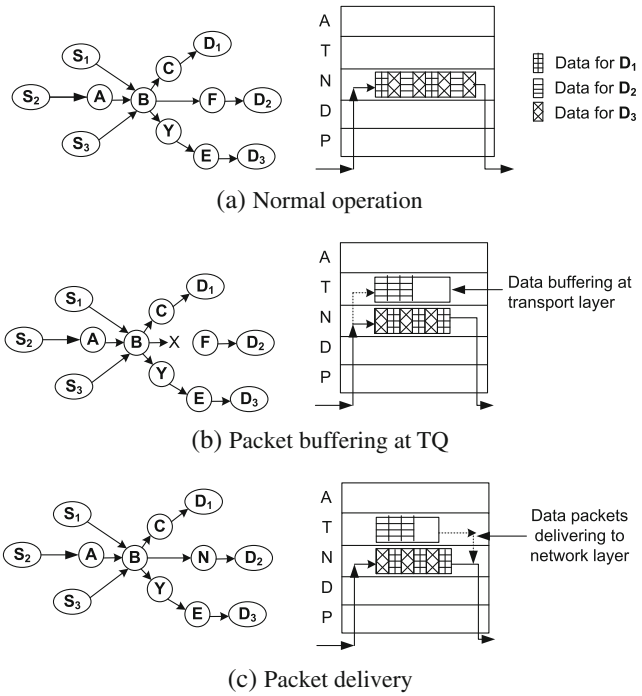


Fig. 6 Operation of the proposed LCRDD mechanism

operation from network layer as well as resumes transmission process from its TQ for destination node D_2.

5 Performance evaluation

In this section, we evaluate the performance of our proposed LCRDD mechanism in network simulator v-2.34 [23] and compare the simulation results with that of AODV-BBS [10], SMR [5], IBR-AODV [12], and THR [7]. The results of our simulation state that the LCRDD outperforms the other protocols.

5.1 Simulation environment

In our simulation, we consider a square area of size $1,000 \times 1,000$ m^2, where 100 mobile nodes are deployed randomly. The simulation time is set to 200 s. Each node has the transmission range of 100 m. The source nodes of our network generate constant bit rate data streams at the rate of 1–8 packets per s. This helps to measure performance for various traffic load at each mobile node. The size of each data packet is 512 bytes, link bandwidth is kept at 11 Mbps, and the underlying transport and MAC layer protocols are UDP and IEEE 802.11 DCF, respectively. We have used expanding ring search [19] algorithm for repairing local route failures. Table 3 summarizes the simulation parameters. For each data point in the graphs, we take the average of

Table 3 Simulation parameters

Parameter	Value
Network area	1,000 m × 1,000 m
Number of nodes	100
Deployment type	Random
Number of sources	20
Node movement model	Random waypoint
Transmission range	100 m
Transport layer protocol	UDP
MAC layer protocol	IEEE 802.11 DCF
Bandwidth	11 Mbps
Data packet size	512 bytes
Data packet generation rate	1 to 8 packets/s
Propagation model	Free space
Weight factor α	0.002
T_t	2 s

10 simulation runs that helps us in studying the steady state behavior of the protocols.

5.2 Performance metrics

We use four performance metrics to compare the results of AODV-BBS, SMR, IBR-AODV, THR, and LCRDD. Those metrics are as follows:

- *Packet delivery ratio* is measured as the ratio of the total number of received data packets by all the destination nodes to the total number of generated data packets by all the source nodes in the network. It is corresponding to the reliability of the network; the higher the value, the better is the performance.
- *Average end-to-end packet delay* is measured as the average time in milliseconds required by all the data packets to travel from the source nodes to their corresponding destination nodes.
- *Per node throughput* is measured as the average amount of data bits received per unit time by all the destination nodes in the network.
- *Normalized routing overhead* is measured as the number of control packets generated during the simulation period for each successfully delivered data packet.

5.3 Impact of varying traffic loads

In this section, we study the impact of different traffic loads on the performances of the protocols in terms of the above metrics. The data traffic loads at source nodes are varied from $1 \sim 8$ packets per s, i.e., from 0.5 to 4.0 KBps. For this experiment, we fix the mobility speed of each node at 2 m/s within the network. The graphs in Fig. 7 compare the performances of the studied protocols for various source traffic loads.

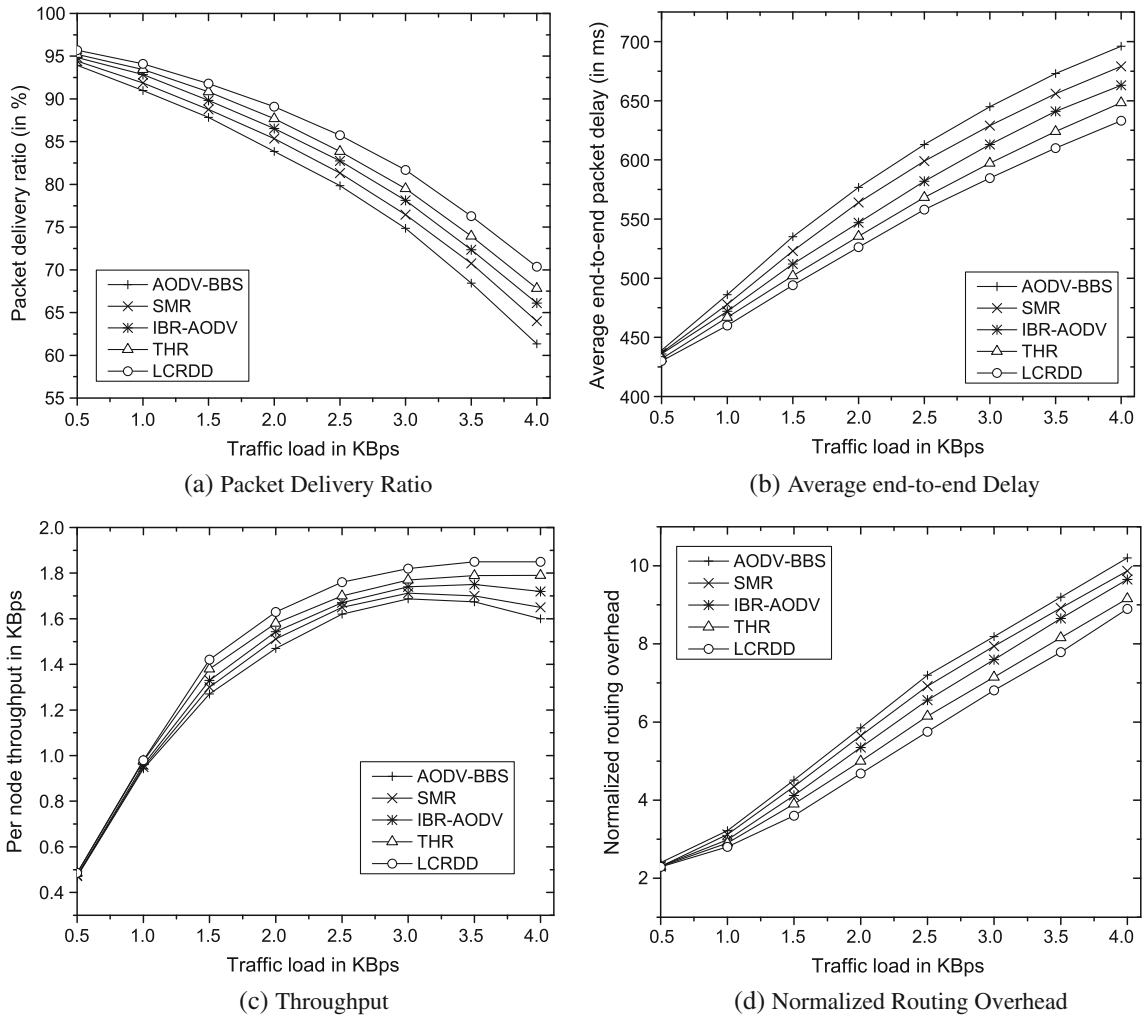


Fig. 7 Performance comparisons for different traffic loads, mobility speed = 2 m/s

The packet delivery ratio for all the protocols decreases drastically with increased source traffic loads, as shown in Fig. 7a. This is caused by increased packet drops at the intermediate nodes due to congestion, i.e., the forwarder nodes cannot deliver as many packets as they receive. The simulation results indicate that our proposed LCRDD mechanism has the higher capability to handle congestion in the network and it outperforms other protocols. Since LCRDD measures the pre-congestive and congestive states more accurately and propagates information along the routes and takes appropriate control actions in time, its congestion control mechanism becomes more efficient than others. Furthermore, the buffering mechanism of LCRDD reduces the packet drops a lot and thus increases the packet delivery ratio.

Figure 7b shows the average end-to-end packet delay performances of the protocols for various traffic loads. It states that, as expected theoretically, the packet delivery delay increases with the traffic loads. AODV-BBS and SMR experience much higher delay than others due to high latency of maintaining multiple alternate routes. IBR-AODV has longer delay than THR and LCRDD since it cannot handle the congestion uses backup nodes only for providing local recovery from route failures. However, our proposed LCRDD mechanism handles congestion proactively by not allowing an intermediate node to carry additional traffic (1) from new connections when it detects *loaded* state and (2) from existing connections when it detects *heavily loaded* state. Our in-depth look into the trace file contents reveals that the above strategy helps LCRDD nodes to operate in safe mode most of the time and thus it decreases queuing delays of the packets at the intermediate nodes, which in turn decreases the end-to-end packet delivery delay a lot.

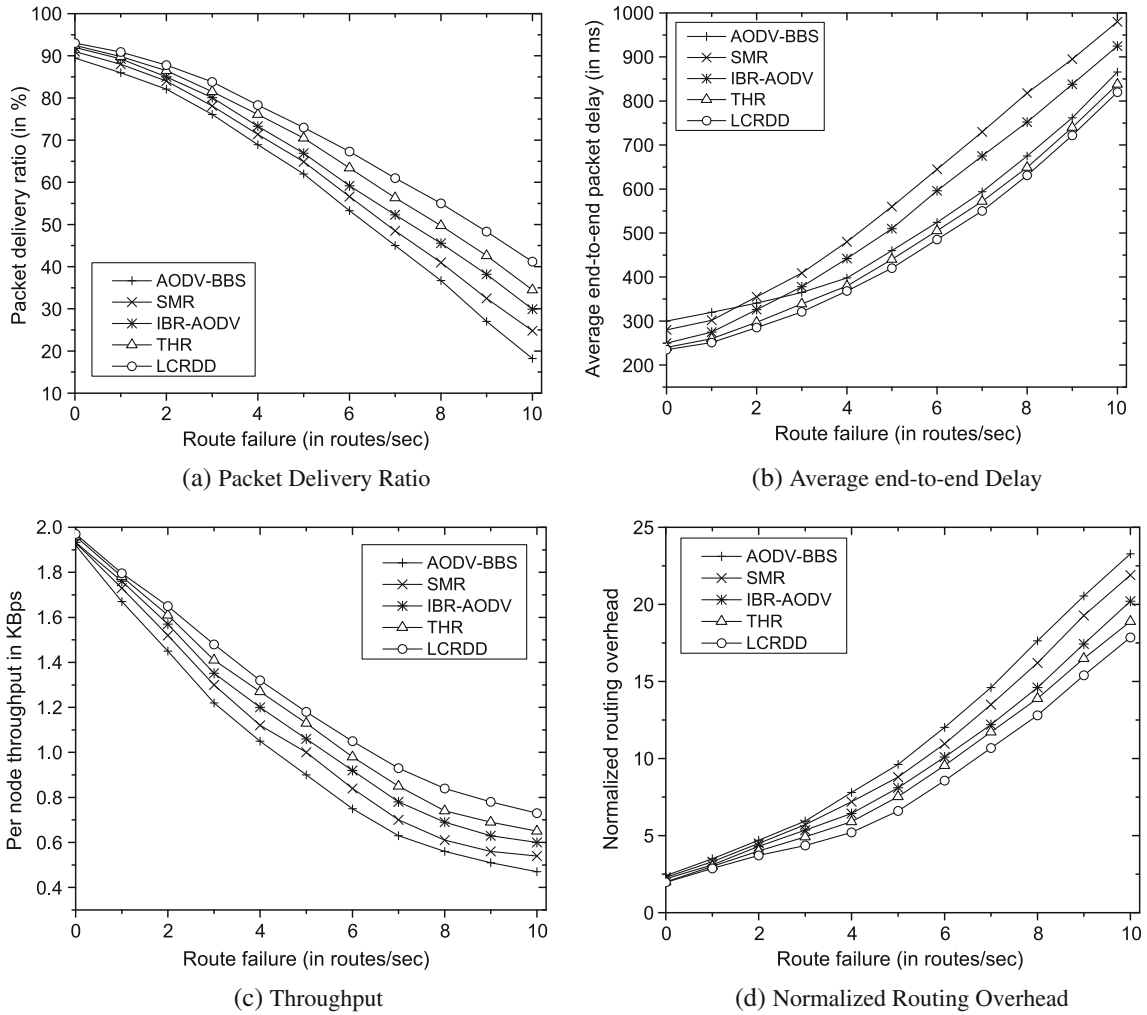
The performance comparisons for per node throughput of the studied protocols have been shown in

Fig. 7c. The throughputs of the protocols increase as the traffic load increases, but it starts decreasing at around 3.5 KBps traffic load, where the network reaches at saturation condition. The further increase of traffic load makes the network congested and thus the performance decreases. We observe that our LCRDD provides high performance than the other protocols since it ensures higher number of packets delivery at the destination within minimum end-to-end delay.

Figure 7d shows the normalized routing overhead of the studied protocols. The AODV-BBS has high latency of route discovery process for keeping multiple routes for same destination and generates large amount of control packets. During link failure, SMR needs a fresh route discovery process, producing a large number of control packets. Since IBR-AODV stores data packets at multiple neighborhood nodes and exchanges control packets on the failure of links, its overhead increases a lot. The THR can handle link failure but can't cope up with network congestion as traffic load increases and local route discovery process starts on the failure of links. On the other hand, our LCRDD neither maintains any alternative routes nor it balances loads among multiple routes; rather, it uses multilevel congestion control mechanism and local packet buffering at transport layer for the period of local route discovery to mitigate link failures and congestion in the network. As a result, it generates minimum number of control packets and thus provides with the least normalized routing overhead among the studied protocols.

5.4 Impact of varying route failure rates

In this section, we evaluate the impact of varying route failure rates on the performances of the studied protocols, keeping the packets generation rate constant at 6 packets per s (i.e., 3 KBps). We vary the route failure rates from 1 to 10 routes/s.



(a) Packet Delivery Ratio

(b) Average end-to-end Delay

(c) Throughput

(d) Normalized Routing Overhead

Fig. 8 Performance comparisons for different route failure rates, data traffic load = 3 KBps

The graphs in Fig. 8a state that the packet delivery ratio for the all protocols decrease sharply with the increasing route failure rates. This happens because the failure of routes increases the number of packet drops in all the approaches. Since our proposed LCRDD mechanism jointly exploits the multilevel congestion control and packet buffering schemes on the event of link failures, it is capable to address the route failures more effectively and saves packets from dropping and thus it can increase the packet delivery ratio compared to other protocols.

As expected theoretically, the end-to-end packet delivery delay increases with the route failure rates for all the protocols, as shown in Fig. 8b. The SMR experiences the longest end-to-end packet delivery delay. The IBR-AODV has lower end-to-end delay than SMR but higher than other protocols. The AODV-BBS uses an alternative route whenever a link failure is detected and thus it decreases the delay than SMR and IBR-AODV. The local route discovery-assisted packet buffering mechanism in LCRDD helps it to handle route failures more efficiently than others. Also, in LCRDD, The number of retransmissions required on the failure of routes decreases a lot and thus it reduces the traveling time of data packets.

Figure 8c shows the throughput performances of the studied protocols. The per node throughput decreases for all the protocols as the route failure rate increases. Because of high latency multiple paths, AODV-BBS and SMR provide lower throughput. The IBR-AODV provides better results since it initiates fast recovery. Among all the studied protocols, our proposed LCRDD has the highest throughput performance. This result is achieved due to its (1) efficient route failure handling mechanism and (2) congestion-aware data delivery mechanism.

Figure 8d shows that the normalized routing overhead increases as route failure rate rises as the later causes substantial increase in the number of control packets generated in the network. Because of packet buffering capabilities and local recovery process, our proposed LCRDD mechanism provides with lower overheads compared to other state-of-the-art protocols.

5.5 Discussion

As our proposed LCRDD mechanism learns the incipient congestive states and link conditions in ahead of time, its routing and traffic forwarding functions effectively transfer the data packets over less congestive and good links and thus it implements an efficient data delivery framework. Results reveal that the joint efforts of LCRDD's packet buffering and multilevel congestion control actions can efficiently cater for the application requirements with various network conditions, i.e., different combinations of traffic loads, route failure rates, and mobility speeds. The developed LCRDD mechanism is expected to work with good performance in a large number of emerging MANET applications, particularly in which high throughput data delivery is demanded.

The main limitation of this work is related to the lack of sufficient understanding about the dynamics of threshold values. For example, $Q_{\min}$ and $Q_{\max}$ were determined through numerous simulation experiments; the same is true for the weight factors—α and w. If we could build an analytical model for them, we could be able to dynamically select the optimal values to adapt to different situations. We leave this for our future work.

6 Conclusion

In this paper, we have addressed two very important problems: route failure and congestion in MANETs. Our proposed LCRDD mechanism introduced the concepts of local buffering at transport layer and multilevel congestion detection and proactive control actions which improve the network performance significantly. The results of our performance evaluations, carried out for various traffic loads and route failure rates, show that the proposed LCRDD mechanism outperforms a number of state-of-the-art approaches. Our mechanism is fully distributed and does not depend on network-wide information and thus it reduces the operation overhead as well.

Acknowledgements This work was supported by the Research Center of College of Computer and Information Sciences, King Saud University, Riyadh, Kingdom of Saudi Arabia. The authors are grateful for this support. Dr. Md. Abdur Razzaque is the corresponding author of this paper.

References

1. Royer EM, Toh C-K (1999) A review of current routing protocols for ad hoc mobile wireless networks. IEEE Pers Commun 6(2):46–55
2. Ravindra E, Kohir V, Mytri V (2011) A local route repair algorithm based on link failure prediction in mobile ad hoc network. World J Sci Technol 1(8):64–67
3. Wan CY, Eisenman SB (2003) CODA: congestion detection and avoidance in sensor networks. In: Proceedings of ACM SenSys. ACM, Helsinki, Finland, pp 266–279
4. Floyd S, Jacobson V (1993) Random early detection gateways for congestion avoidance. IEEE/ACM Trans Netw 1(4): 397–413

5. Lee SJ, Gerla M (2001) Split multipath routing with maximally disjoint paths in ad hoc networks. IEEE International Conference on Communications (ICC)
6. Valarmathi A, Chandrasekaran RM (2010) Congestion aware and adaptive dynamic source routing algorithm with load-balancing in MANETs. Int J Comput Appl 8(5):6–9
7. Rajotiya RN, Kumar E (2011) THR: a two-hop look ahead with packet buffering protocol for MANETs. Int J Inf Technol Knowl Manag 4:109–112
8. Toh CK (1997) Associativity-based routing protocol for mobile ad hoc networks. Wirel Pers Commun 4(2):103–109
9. Marina MK, Das SR (2001) On-demand multipath distance vector routing in ad hoc networks. In: 26th annual IEEE international conference on Local Computer Networks (LCN), 11–14 Nov 2001. IEEE (Comput. Soc.), Flordia, USA, pp 14–23
10. Huang T-C, Huang S-Y, Tang L (2010) AODV-based backup routing scheme in mobile ad hoc networks. In: Proceedings of the 2010 international conference on Communications and Mobile Computing, vol 03. CMC '10, IEEE Computer Society, Washington, DC, USA, pp 254–258
11. Leung R, Jilei RL, Poon E, Chan A-LC, Li B (2001) MP-DSR: a QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks. In: The IEEE conference on Local Computer Networks LCN, pp 132–141
12. Jeon J, Lee K, Kim C (2011) Fast route recovery scheme for mobile ad hoc networks. In: International Conference on Information Networking (ICOIN), pp 419–423
13. Perkins CE, Royer EM (1997) Ad-hoc on-demand distance vector routing. In: Proceedings of the 2nd IEEE workshop on Mobile Computing Systems and Applications, pp 90–100
14. Natsheh E, Jantan A, Khatun S, Subramaniam S (2007) Adaptive optimizing of hello messages in wireless ad-hoc networks. In: The international Arab journal of information technology
15. Johnson DB, Maltz DA, Broch J (2001) DSR: the dynamic source routing protocol for multi-hop wireless ad hoc networks. In: Perkins CE (ed) Ad hoc networking, Chapter 5. Addison-Wesley, pp 139–172
16. Valera AC, Seah WKG, Rao SV (2005) Improving protocol robustness in ad hoc networks through cooperative packet caching and shortest multipath routing. IEEE Trans Mobile Comput 4:443–457
17. Castañeda R, Das SR, Marina MK (2002) Query localization techniques for on-demand routing protocols in ad hoc networks. Wirel Netw 8(2-3):137–151
18. Lee SJ, Gerla M (2000) AODV-BR: Backup routing in ad hoc networks. In: IEEE Wireless Communications and Networking Conference (WCNC'2000), pp 1311–1316
19. Rachuri KK, Siva RMC (2010) On the scalability of expanding ring search for dense wireless sensor networks. J Parallel Distrib Comput 70:917–929
20. Ramanathan R, Redi J (2002) A brief overview of ad hoc networks: challenges and directions. IEEE Commun Mag 40:20–22
21. Tran DA, Raghavendra H (2006) Congestion adaptive routing in mobile ad hoc networks. IEEE Trans Parallel Distrib Syst 17:1294–1305
22. Lee SJ, Gerla M (2001) Dynamic load-aware routing in ad hoc networks. Helsinki, Finland, pp 3206–3210
23. The network simulator NS-2 (2011) http://www.isi.edu/nsnam/ns/. Accessed 19 Aug 2012