Journal of King Saud University – Computer and Information Sciences (2013) **25**, 25–34

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Dynamic congestion detection and control routing in ad hoc networks

T. Senthilkumaran [a,*], V. Sankaranarayanan [b]

[a] *BSA Crescent Engineering College, Vandalur, Chennai, Tamil Nadu, India*
[b] *B.S. Abdur Rahman University, Vandalur, Chennai, Tamil Nadu, India*

Received 13 January 2012; revised 1 May 2012; accepted 19 May 2012
Available online 29 May 2012

KEYWORDS

Congestion;
DCDR;
CFS;
EDOCR;
EDCSCAODV;
EDAODV;
AODV

Abstract In mobile ad hoc networks (MANETs), congestion can occur in any intermediate node, often due to limitation in resources, when data packets are being transmitted from the source to the destination. Congestion will lead to high packet loss, long delay and waste of resource utilization time. The primary objective of congestion control is to best utilize the available network resources and keep the load below the capacity. The congestion control techniques to deal with TCP have been found inadequate to handle congestion in ad hoc networks, because ad hoc networks involve special challenges like high mobility of nodes and frequent changes of topology. This paper proposes a method for dynamic congestion detection and control routing (DCDR) in ad hoc networks based on the estimations of the average queue length at the node level. Using the average queue length, a node detects the present congestion level and sends a warning message to its neighbors. The neighbors then attempt to locate a congestion-free alternative path to the destination. This dynamic congestion estimate mechanism supporting congestion control in ad hoc networks ensures reliable communication within the MANET. According to our simulation results, the DCDR showed better performance than the EDOCR, EDCSCAODV, EDAODV and AODV routing protocols.

1. Introduction

In recent times, a number of techniques and applications have been used widely for transmitting information through heterogeneous wireless networks. In general, a wireless network may belong to one of the two types: infrastructure network and ad hoc network. Ad hoc networks are usually defined as an autonomous system of nodes connected by wireless links for communicating in a multi-hop fashion. The ad hoc network offers several advantages, including low cost, simple network maintenance, and convenient service coverage (Murthy and Manoj, 2007; Johnson and Maltz, 1996; Ramanathan and Redi, 2002).

Congestion may occur in a network if the load on the network (the number of packets being sent through the network) is greater than the capacity of the network (the number of packets the network can handle). Thus, network congestion can cause to severely increase the delay and packet loss and

[*] Corresponding author.
E-mail addresses: senthilkumaran@bsauniv.ac.in (T. Senthilkumaran), sankarammu@bsauniv.ac.in (V. Sankaranarayanan).

Peer review under responsibility of Kind Saud University.

reduce network throughput. Congestion control refers to techniques and mechanisms that can either prevent or remove congestion (Lochert et al., 2007; Tran and Raghavendra, 2006).

The main objective of congestion control is to minimize the delay and buffer overflow caused by network congestion and hence enable the network to perform better. In wired networks, congestion control is implemented at the transport layer and is often designed separately from the functions of other layers (Lochert et al., 2007; Tran and Raghavendra, 2006; Yingqun and Giannakis, 2008). However, such congestion control techniques do not apply directly to ad hoc networks, which involve special challenges like limited wireless bandwidth, power constraints, and route failures due to node mobility and limited buffer size. This means that as more number of packets is being sent through a network, network congestion should result in high packet loss rate, re-routing instability, loss of energy and bandwidth, and retransmission of lost packets. However, delays and packet losses need not necessarily be caused by network congestion, but these can be misinterpreted as congestion losses (Lochert et al., 2007; Perkins et al., 2003; Yen et al., 2010). The routing protocols for MANETs fall into three categories based on the routing information update mechanism (Murthy and Manoj, 2007; Johnson and Maltz, 1996; Chen and Heinzelman, 2007), namely proactive, reactive (or on-demand) and hybrid. Proactive routing protocols include DSDV and OLSR (Murthy and Manoj, 2007; Johnson and Maltz, 1996; Chen and Heinzelman, 2007; Perkins et al., 1994). On-demand routing protocols are, for example, AODV (Murthy and Manoj, 2007; Johnson and Maltz, 1996; Perkins et al., 2003) and DSR (Murthy and Manoj, 2007; Johnson and Maltz, 1996; Broch et al., 1999). Hybrid routing protocols (Murthy and Manoj, 2007; Johnson and Maltz, 1996; Chen and Heinzelman, 2007) combined the features of proactive and on-demand routing protocols.

There is another dimension to classify routing protocols: congestion-control routing versus congestion non-control routing. We note that the existing routing protocols are largely of congestion non-control type (Lochert et al., 2007; Tran and Raghavendra, 2006). When a new route is established, it will remain so until the mobility or failure results in a disconnection. When congestion occurs during packet transfers between the source and the destination, the existing routing protocols do not seem to handle it effectively. Congestion becomes more easily visible in large-scale transmission of traffic-intensive data, such as multimedia, and the impact of packet loss on service quality is of prime concern (Lochert et al., 2007; Tran and Raghavendra, 2006). Unlike well-established networks such as the Internet, a dynamic network like a MANET is expensive in terms of time and overhead, and it is possible only to remove a congestion but not prevent it altogether (Lochert et al., 2007; Tran and Raghavendra, 2006). To eliminate network congestions, many researchers have suggested the use of active queue management (AQM) strategies. The main idea here is to provide a buffer in the network in order to manage or eliminate the problems arising out of possible congestions (Athuraliya et al., 2001). Many AQM techniques, such as the adaptive virtual queue, random early detection (RED), random exponential marking, PI controller (Athuraliya et al., 2001), and the blue and stochastic blue schemes (Feng et al., 2001) have been reported. Among these, the one recommended by the IETF for the next-generation Internet routers is RED (Braden et al., 1998). This is because RED could predict congestion by monitoring the average queue size. This paper aims to propose a technique that anticipates congestion at the MAC layer due to buffer overflow and then adapts the traffic in the network layer by finding a non-congested path. This technique has several benefits including avoiding or reducing packet loss, reduction of delay, and improving the overall network performance. In our previous work, EDAODV (early congestion detection and control routing) techniques (Senthilkumaran and Sankaranarayanan, 2010) have been proposed to predict congestion and find a non-congested alternative path bi-directionally. A technique for self-curing the congestion was proposed in Senthilkumaran and Sankaranarayanan (2011a) and is called EDCSCAODV (early congestion detection and self-cure routing). In EDOCR (early congestion detection and optimal control routing), the network is divided into sparse and dense regions by using average neighbors to find a non-congested alternative path with the help of dense nodes (Senthilkumaran and Sankaranarayanan, 2011b). Similarly, EDAPR (early congestion detection and adaptive routing) techniques (Senthilkumaran and Sankaranarayanan, 2011c) have been proposed for preventing congestion by using the NHN (non-congested two-hop neighbors).

Our proposed DCDR uses a new algorithm for detecting congestion dynamically. It uses a non-congested path discovery mechanism to prevent network congestion, and hence packet loss and end-to-end delay are reduced and throughput improved.

The remainder of the paper is organized as follows. The concept of dynamic congestion detection routing is presented in Section 2. Our NS2 simulation results (Section 3) confirm that our proposed protocol has a higher packet delivery rate, consumed less control packets, and reduced end-to-end delay. Section 4 presents the conclusion of our work.

1.1. Related work

1.1.1. Early congestion detection technique

Let us use expressions (1) and (2) to set the minimum and maximum threshold values for the queue length:

$$Minth = 25\%\ buffer\_size \tag{1}$$

$$Maxth = 3 * Minth \tag{2}$$

To predict the congestion well in advance, we compute the average queue size as

$$Avgquenew = (1 - w_q) * Avgqueold + Inst\_que * w_q \tag{3}$$

where w_q, queue weight, is a constant ($w_q = 0.002$ from RED queue experimental results (Floyd, 1997), and *Inst_que* is instantaneous queue size:

$$Queue\_status = Inst\_que - Avgquenew \tag{4}$$

If *Queue_status* < minimum threshold, the incoming traffic is low and the queue is in safe zone. If *Queue_status* > minimum threshold and *Inst_que* < maximum threshold, the incoming traffic is normal and the queue is likely to be in congested zone. If *Inst_que* > maximum threshold, the incoming traffic is heavy and the queue is in congested zone.

In this early congestion detection algorithm, the calculation of average queue length involves the previous average queue length and the instantaneous queue length modified by a weight parameter, w_q. Since w_q is a constant parameter, a short-term increase in queue size resulting from bursty traffic or transient congestion need not result in a significant increase in the average queue size. As a result, the average queue (*Avg-quenew*) changes much slower than the instantaneous queue (*Inst_que*).

1.1.1.1. Bi-directional path discovery. A simplified example is shown in Fig. 1. A route $S \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow D$ is initially identified from source S to destination D. It is called the primary route from S to D. Every packet takes this primary route. Suppose node 3 detects that congestion is likely to occur it sends a warning to its neighborhood – its predecessor node and the successor node. In response to this situation, nodes 2 and 4 then attempt to identify a bidirectional alternative route bypassing node 3, as shown in Fig. 1(a). Finally, node 2 finds an alternative path destined for D, as shown in Fig. 1(b). Now the traffic coming to node 2 will use the route $2 \rightarrow 6 \rightarrow 4$. If no alternative path was possible to be found, the traffic should flow only through the primary route, $S \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow D$ (Senthilkumaran and Sankaranarayanan, 2010).

1.1.1.2. Self-cure routing. Considering the example shown in Fig. 2(a), both nodes 2 and 3 are likely to be in congested zone and will send the congestion status packet (CSP) to its neighbors, as shown in Fig. 2(b). Here, node 4 is a common neighbor for nodes 2 and 3, and it applies a self-cure routing scheme in that, after the first redirection, node 2 is bypassed. Similarly, node 5 is a common neighbor for node 3 and node D. After the second redirection, node 3 is bypassed as shown in Fig. 2(b). Finally, the self-cure routing successfully cures the congestion and identifies a non-congested alternative path $S \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow D$ as shown in Fig. 2(b) (Senthilkumaran and Sankaranarayanan, 2011).

1.1.1.3. Optimal control routing. As shown in Fig. 3, node 5 detects a possible congestion and sends a warning to its neighboring nodes 3, 4, 6 and 8, which in turn update other neighbors in the routing table. In response to this, node 3 initiates an optimal route discovery mechanism to reach the destination. At this point, the neighbors of node 3 (node 4) belong to dense region and (node 7) belongs to medium sparse region. Finally, node 7 identifies the route to destination as shown in Fig. 3. The traffic coming to node 3 will be routed through: $S \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 9 \rightarrow D$. However, if no alternative path was found, it continues to use the primary route: $S \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 9 \rightarrow D$. It is to be noted that the new path is a non-congested path but not necessarily the shortest path (Senthilkumaran and Sankaranarayanan, 2011a).

2. Dynamic congestion detection and control routing

DCDR is a uni-cast routing protocol for MANET. It reduces network congestion by ways of reducing the unnecessary flooding of packets and finding a congestion-free path between the source and the destination. In this paper, we present the complete design and an in-depth evaluation of the DCDR protocol. When a source host has to transmit a data packet to a destination, the DCDR protocol first constructs a congestion-free set (CFS) to connect both one-hop and two-hop neighbors. Then the source initiates the route discovery procedure using the CFS to identify a congestion-free path to the destination.

In case the DCDR protocol is unable to construct a CFS due to the network being already congested, it cannot initiate the route discovery process. However, once a new route has been established, the transmission of data packets will continue. Thus, the main objective of DCDR is to find a

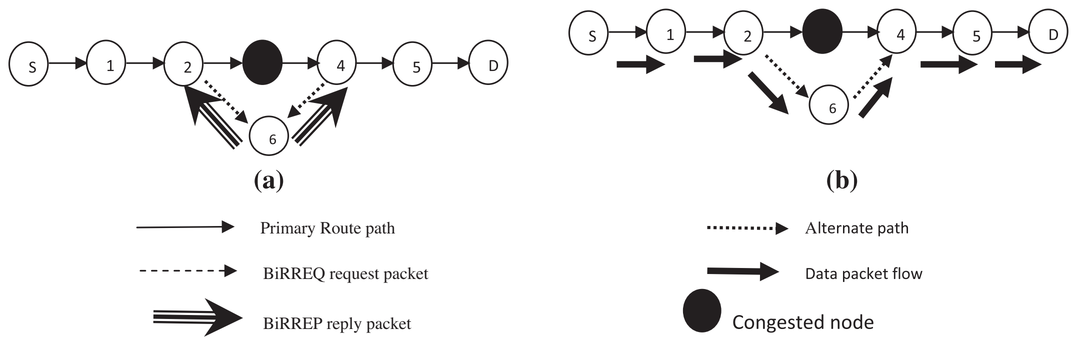


Figure 1 Finding alternate path by using Bidirectionally route discovery.

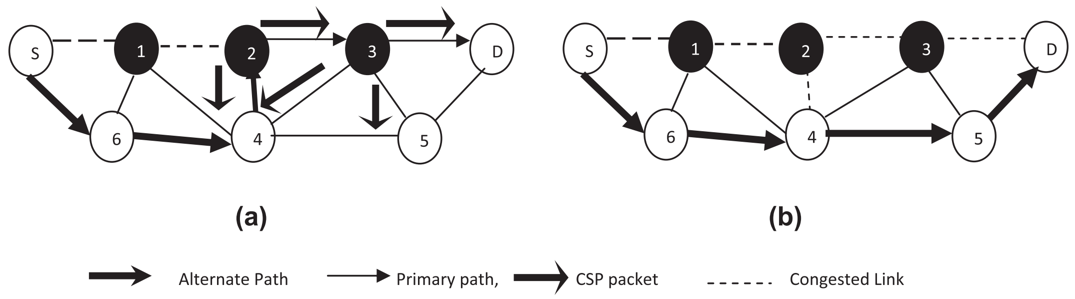


Figure 2 Example of successive local route redirection operations.

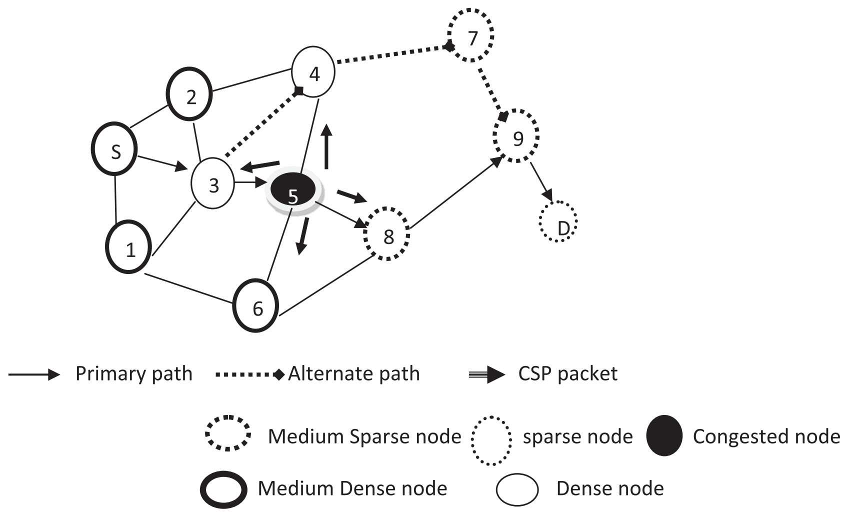


Figure 3 Optimal route discovery technique for an alternate path.

congestion-free route between the source and the destination. In doing so, it reduces the overhead and flooding of packets. The DCDR protocol consists of the following components:

1. Dynamic congestion estimation technique.
2. CFS construction.
3. Congestion-free route.
4. Congestion-free path discovery.

2.1. Dynamic congestion estimation technique

Congestion in a network signifies that a node at any interval became congested and started to lose packets. Several metrics are available to monitor the congestion status at node level. For instance, it could be based on the average queue length and the percentage of packets discarded for lack of buffer space. Every second, a node checks the occupancy of its link layer queue using the dynamic congestion estimation technique so as to detect congestion well in advance. The dynamic congestion (DC) estimation technique is a queue management algorithm that makes use of a direct measurement of the congestion status.

The DC algorithm uses three parameters, namely *Minth*, *Maxth* and w_q to standardize its performance. *Minth* and *Maxth* are queue thresholds to present the current status of the queue, and w_q is the queue weight parameter to compute the average queue size from the instantaneous queue length. The performance of DC depends on these thresholds. If the thresholds are small, then link utilization will be very low. If the thresholds are set too high, then congestion might occur even before the nodes are notified. To overcome this problem, we propose an effective threshold selection strategy.

Let us use expressions (1) and (2) to set the minimum and maximum threshold values for the queue length, which are dependent on the preferred average queue size. We initially set *Maxth* = 2**Minth*, because the DC would function most effectively when *Maxth*–*Minth* is larger than the typical increase in the calculated average queue size in one round-trip time, and a useful rule (from RED queue) is to set *Maxth* to at least twice of *Minth* (Floyd, 1997). We then changed *Maxth* dynamically based on the traffic condition. For this reason, we chose to fix the minimum threshold of 35%.

$$Minth = 35\%\ Queue\_size \tag{1}$$

$$Maxth = 2 * Minth \tag{2}$$

The objective of average queue length is to incorporate all the traffic fluctuations, and it follows the long-term changes of *Inst_que*, reflecting persistent congestion in the networks. Expression (3) is used to find the average queue length.

$$Avgque = (1 - w_q) * Avgque + Inst\_que * w_q \tag{3}$$

The weight parameter, w_q, regulates network congestion and acts as a time constant of the low-pass filter. The average queue length is desired to track recurrent network congestion that happens over a long period and, at the same time, filter short time congestion. This condition imposes a setback on the selection of w_q. If w_q is too small, the average queue length could not grasp the long-range congestion, which might result in ineffective congestion detection. If w_q is too large, the average queue length follows the instantaneous queue length, which also degrades the performance of the congestion estimation technique. Therefore, the value of w_q should be related to the flow of traffic in the queue. The proposed DC algorithm would concentrate on assigning w_q values dynamically according to the traffic flow. Initially, w_q is set to 0.002 (Floyd, 1997; Floyd and Jacobson, 1993). We used expression (4) to set w_q values dynamically, where N is the number of active flows and P is the packet rate (number of packets per second).

$$wqnew = wqold * N * P \tag{4}$$

If the average queue length is less than *Minth* and instantaneous queue is less than *warn_line* (*warn_line* = *Queue_size*/2), then the node is in zone I (safe zone). If the average queue length is greater than *Minth* and less than *Maxth*, then the node is likely to be in congestion and an alternative path discovery mechanism is initiated. In the mean time, if the instantaneous queue size is greater than *Maxth* due to heavy incoming traffic, the status of alternative path discovery becomes false.

$$Queue\_utilization = (Maxth + Minth)/2 \qquad (5)$$

In this situation, our algorithm introduces the *Queue_utilization* parameter, which will help to change the *Maxth* values dynamically until the alternative path discovery becomes true. We used expression (5) to get *Queue_utilization* value (*Minth* = 35% *Queue_size*; *Maxth* = 70% *Queue_size*; and *Queue_utilization* = 87.5% *Queue_size*), which consists of three ranges. It varies from 85% to 90% queue size with 2.5% difference. Finally, if the average queue length is greater than *Maxth*, then node's congestion status becomes Zone-III (congested zone). The algorithm for dynamic congestion estimation is shown in Algorithm I.

Algorithm I: Dynamic congestion estimation

```
//initialization
Avgnew = 0
Avgold = 0;
Inst_que = 0
Minth = 0.35 * queue_size
Maxth = 2*Minth
Queue_util[] = {0.85,0.875,0.9}
Wq = 0.002;
Warn_line = queue_size / 2
//For each arriving packet in the queue
   Inst_que++
//Calculate average queue size
   If the queue is not empty then
   Avgnew = (1-$w_q$) Avgold+ Inst_que * $w_q$
   If (Avgnew < Minth && Inst_que < Warn_line) then
   Begin
   Queue_status = "Safe";
   Else if (Avgnew > Minth && Avgnew < Maxth) then
   Begin
   Queue_status = "Likely to be congested";
// Initiate Alternate Route Discovery Process
   If (Inst_que > Maxth && alter_path = FALSE) then
   Maxth = Queue_util[i++]*buff_size;
   Else
   Queue_status = "Congested";
   Avgold = Avg;
   Wq = Wq*N*P
   End
   End
   For each departing packet in the queue
   Inst_que - -
```

Variable parameters: *N*: number of active flows; *P*: packet rate (packets/s); *Avgold*: previous average queue; *Avgnew*: new average queue; *Inst_que*: instantaneous queue.

Fixed parameters: *Wq*: queue weight; *Queue_util*: maximum queue utilization; *Minth*: minimum threshold value; *Maxth*: maximum threshold value.

2.2. CFS construction

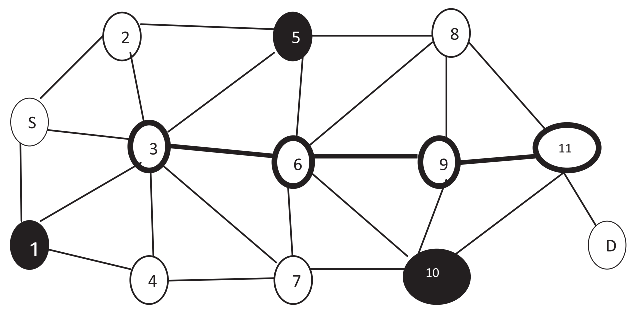
Each mobile host selects its CFS (Yen et al., 2010) from among its non-congested one-hop neighbors in such a way that it covers all two-hop nodes. The CFS of source host *S*, denoted by CFS(*S*), is then an arbitrary subset of the non-congested one-hop neighborhood of *S* that satisfies the following condition: every node in the strict two-hop neighborhood of *S* must have a link toward CFS(*S*), and it should not fall in the congested zone. The CFS setup is an initialization procedure in which each mobile host calculates its congestion status every second by using the dynamic congestion estimation technique. Every mobile host broadcasts its congestion status by using a CSP to its one-hop neighbors on the network. The CSP interval time is 1 s, and the maximum interval time is 1.25*CSP interval. Now, each mobile host learns about its one-hop non-congested neighbor nodes and records the information into its non-congested one-hop list. After that, each mobile host exchanges its one-hop non-congested neighbor information so as to learn about its two-hop non-congested neighbor nodes. At this point, each mobile host constructs its CFS by selecting a subset of its one-hop non-congested neighbor nodes so that the mobile host in the subset can forward its broadcast traffic to the two-hop neighbor nodes and minimize the flooding traffic. Each mobile host updates the information in its routing table. The format of each entry in the routing table is <*src_addr*, *dst_addr*, *hop_cnt*, *CFS_Node*, *CFSSET*, *congest_status*>, where *src_addr* is the source address, *dst_addr* is the destination address, *hop_cnt* is the hop count, *CFS_Node* is the non-congested node address, *CFSSET* is the list of non-congested neighbors, and *congest_status* is neighbor's congestion status. Fig. 4(a) shows the non-congested neighbor information, and Fig. 4(b) shows the process of CFS selection.

2.3. Congestion-free route discovery

In order to send a data packet to a destination, the source host generates the route request (RREQ) packet for communication using the CFS nodes. The source host first checks its two-hop list. If the destination host is in its two-hop list, then the datagram is transmitted by following the path in routing table. If not, the source host broadcasts the RREQ to the CFS on the network. When the CFS receives this RREQ packet, it also checks its two-hop list. If the destination host is in its two-hop list, then the CFS forwards the RREQ directly to the destination host. The destination responds to the first received RREQ and sends back an RREP packet. The RREP will travel back in the same path and add a new entry in its routing table. This path now becomes the primary route between the source and the destination. In case the destination host is not in its two-hop list, then it modifies the sequence number and hop count and rebroadcasts this RREQ to the network. The process is repeated until the destination host is found. Finally, the source finds a non-congested path to the destination. After a successful route discovery, the data packet is transmitted to the destination. A major advantage of the DCDR is that it automatically finds the non-congested path and hence reduces the overload and the flooding of the packets.

Fig. 5 shows the route discovery after CFS selection. The source host *S* has a non-congested one-hop list consisting of mobile hosts {2, 3, 4} and a non-congested two-hop list consisting of mobile hosts {4, 6, 7}; the source chooses node 3 as a CFS and adds it to the CFS list.

The first mobile host *S* checks its two-hop list to see whether it included the destination host *D*. If destination host *D* is not in this list, the sources host *S* forwards the RREQ packet to the next CFS node 3. Then, node 3 would check the two-hop list. If the destination is not inside it, the CFS node 3 forwards the RREQ to the next CFS node 6. The CFS host 6 would check the two-hop list. If the destination is not inside it, the CFS node 6 forwards RREQ to the next CFS node 9. Finally, node 9 finds that the destination node



(a)

Node	One hop non-congested	One hop congested	Two hops non congested	Two hops congested
S	2,3	1	4,6,7	5
2	S,3	5	4,6,7	1
3	S,2,4,6,7	1,5	8,9	10
4	3,7	1	S,2,6	5,10
6	3,7,8,9	5,10	S,2,4,11	1
7	3,4,6	10	S,2,8,9	1,5
8	6,9,11	5	3,7,D	10
9	6,8,11	10	3,7,D	5
11	8,9,D	10	6	5,10
D	11	-	8,9	10

(b)

● Congested node ○ CFS node

Figure 4 CFS (congestion-free set) selection.

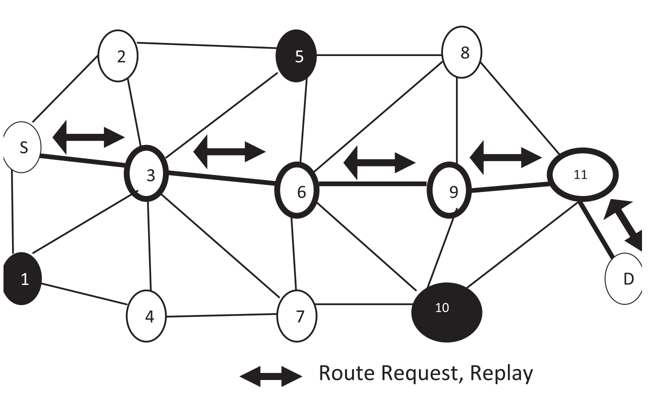


Figure 5 Route discovery process through CFS.

D is in the two-hop list, and so it forwards this packet through the CFS node 11 to the destination node *D*. The destination node *D* receives the RREQ packet and then returns it to the source. The RREP follows the reverse path of the RREQ to the source host. A route $S \rightarrow 3 \rightarrow 6 \rightarrow 9 \rightarrow 11 \rightarrow D$ is found from source *S* to destination *D*. This is a non-congested path. After a successful route discovery, the datagram is transmitted to the destination. This route is called the primary route from *S* to *D*. Every packet follows this primary route. The route discovery algorithm is given in Algorithm II.

Algorithm II: Route discovery process

```
When the source wants to find a route to a destination
Begin
  Construct CFS set for all mobile hosts
/* CFS – congestion free CFS –set nodes of the network */
  For each node pair (S, D) i. where i = 1 to (N-1) /*D = 2, 3,
4...N*/
  Hops = 0; Route_i = Null;
/* Src: source node; Dst: destination node; Route: output path set
generated for node pair (S, D), set to be Null */
  If (Dst is in two hop list of S_i) Then
  Path generated for pair (S_i, D_i)
  Set Route_i = TRUE
  Hops = 2
  Else
  CFS = S_i;
  Call Procedure PATH (input:CFS, D_i; output: Route_i)
End
Procedure PATH (input:CFS, D_i; output: Route_i)
Begin
  If (Dst is in CFS) Then
  Path generated for pair (S_i, D_i)
  Set Route_i = TRUE
  Increment Hops by 1
  Return
Else If (CFS-SET is not in Route_i) and (CFS-SET's two-hop list
does not contain D_i) Then
/* Hops: number of hops */
Begin
  Increment Hops by 1
  Add CFS-SET to Route_i
  For each neighboring node Neib of node CFS-SET Do
/* Neib: the neighbor CFS-set node of CFS-set */
  PATH (Neib,D_i, Route_i)
End
End
```

A traditional routing protocol in MANET has two phases: (i) route discovery and (ii) route maintenance. In the route discovery phase, the source node broadcasts the RREQ packet to its one-hop neighbor; then the neighbor rebroadcasts the RREQ packet to the destination and finally it finds the route to the destination. For route maintenance, every node updates its one-hop neighbors periodically by using the hello message depending on the routing protocol. Our algorithm initially constructs a CFS and then it initiates the route discovery process through CFS. The CFS construction requires some overhead, but in the route discovery process, this overhead is reduced. The updating of CFS by using CSP is necessary to maintain the congestion status of each and every neighbor. If updating is not possible, the total network becomes congested. So updating also requires some overhead. As far as the overall performance (NS2.34 simulation) of the DCDR is concerned, the end-to-end delay got reduced (around 20–28% than in AODV), packet delivery ratio increased by 28%, and control overhead got reduced by 23% because the algorithm always chose the non-congested path. Therefore, the impact of time delay in CFS construction is nil, because it occurs only once while regular packet forwarding is in progress. The details of overall results are presented in Section 3.

2.4. Congestion-free alternative path discovery

The primary path of a CFS node predicts its congestion status and periodically broadcasts a CSP to its neighboring nodes. The CSP contains node's congestion status and a set of parameters (*Src*, *Dst*) each for a destination appearing in the routing table. After receiving the CSP from a neighboring CFS node, the ancestor CFS node will identify a new CFS node from its neighbor list, and then construct a new CFS from the current node to the destination and exchange it with its neighbors.

When the CFS node receives a new CFS, it first compares the (*Src*, *Dst*) pair information in its routing table. If the entries do not match, it adds the new entry (CFS) in its routing table; otherwise, it updates itself in its routing table. Finally, the predecessor CFS node calls for a route discovery process to find a new route to the destination. The alternative path finding algorithm does not incur any significant overhead, because for every CFS, only one extra broadcast message is necessary to inform one of the neighboring nodes to update its routing table, and also the alternative path discovery process incurs no extra cost. The algorithm for alternative path discovery is given in Algorithm III.

Algorithm III: Receive CSP at predecessor CFS node

```
  Input packet p = (cong_status,src_addr, dst_addr) to all the
valid entries
/* Src: source node; Dst: destination Node; Cong_status – neighbor
congestion status*/
  Begin
  Construct new CFS set from current CFS node to destination
  Call route discovery process
/* find a new route from current CFS node to destination */
  Update new CFS set and add to all two hop neighbors CFS
node's routing table
  Set Route = True
  End
```

Fig. 6(a) illustrates how the CFS node 9 detects the congestion, sends a warning to its neighboring nodes 6 and 11, and updates the non-congested neighbor list in the routing table. In response, the predecessor CFS node 6 chooses a new CFS node 8 (a common neighbor) from its non-congested neighbor list, which then finds the route to the destination as shown in Fig. 6(b). The traffic coming to 6 will be routed through: $S \to 3 \to 6 \to 8 \to 11 \to D$. It is possible that if no CFS nodes are found, it continues using the primary route $S \to 3 \to 6 \to 9 \to 11 \to D$. The new path is a non-congested path but not necessarily the shortest path.

3. Performance study

A comparison of DCDRs' performance with that of EDOCR, EDCSCAODV, EDAODV and AODV in MANET was done

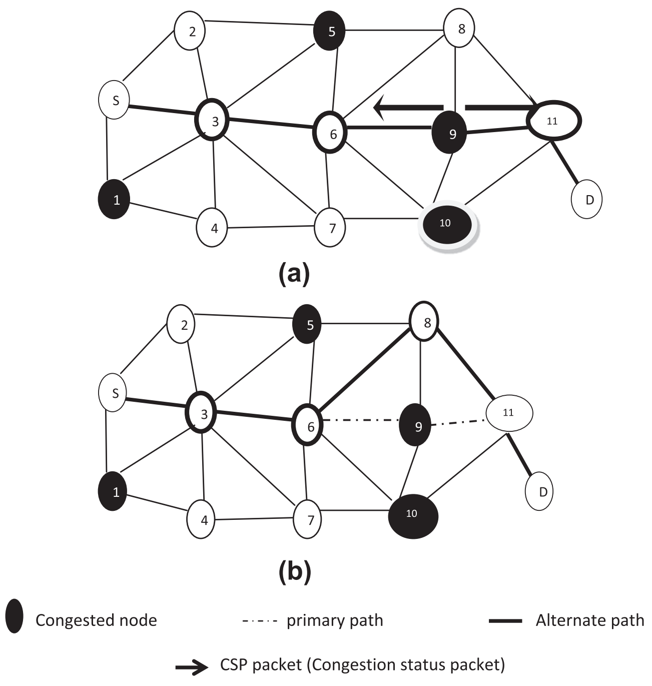


Figure 6 Alternative path finding process by using CFS.

using the Network Simulator (NS2.34) (NS2, 2000). The observations are presented below.

3.1. Performance metrics

We considered the following important metrics in this evaluation:

1. Packet delivery ratio (PDR): The ratio between the number of packets received by the destination and the number of packets sent by the source.
2. End-to-end delay: The delay a packet suffers from leaving the sender to arriving at the receiver.
3. Routing overhead: The total number of control packets transmitted during the simulation time. For packets sent over multiple hops, each transmission over one hop is counted as one transmission.

3.2. Simulation configuration

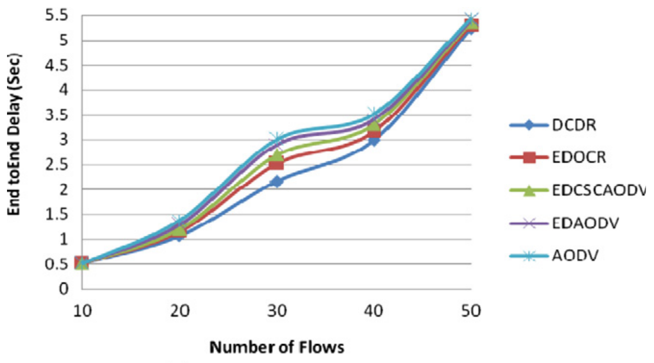
The network consists of 100 nodes in a 1400×1400 m terrain size. The radio range is 250 m with bandwidth 2 Mbps. The MAC layer is based on IEEE 802.11 distributed coordination function. The channel propagation model we used was the 2-ray ground reflection model. An interface queue at the MAC layer could hold 50 packets before they were sent out to the physical link. Link breakage was detected as feedback from the MAC layer. A routing buffer at the network layer could store up to 64 data packets. This buffer keeps in waiting the data packets for which the route discovery had started but no reply had arrived yet. The routing protocols we used are DCDR, EDOCR, EDCSCAODV, EDAODV and AODV. The data flow used constant bit rate (CBR), which varies from 4 packets to 40 packets, and the flow varies from 10 to 50. The maximum speed of the node is 10 m/s and the simulation time is 900 s.

3.3. Varying the number of connections

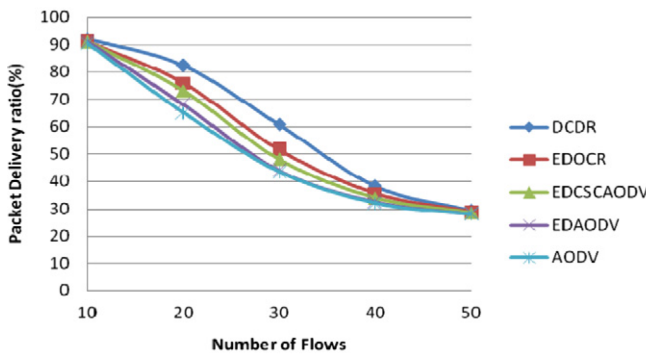
In our simulation experiments, the number of connections (source and destination) was varied from 10 to 50, CBR rate was 8 packets per second, maximum node speed was 10 m/s and pause time was 30 s.

Fig. 7(a–c) shows the end-to-end delay, packet delivery ratio and routing overhead, respectively, for DCDR, EDOCR, EDCSCAODV, EDAODV and AODV.

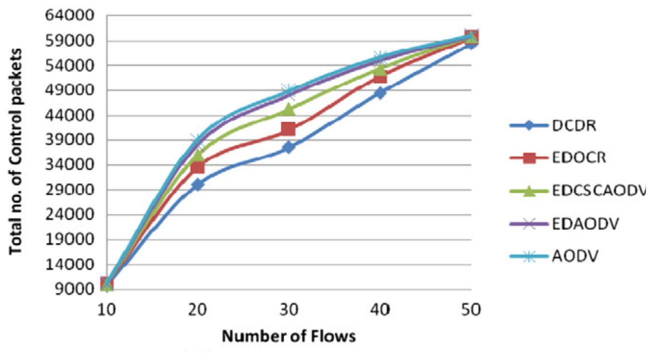
From the figure, the delays incurred by the five protocols (DCDR, EDOCR, EDCSCAODV, EDAODV and AODV) are similar when the number of flows is 10. This is because at a low load the network is in a safe zone. As a result, the end-to-end delay is low while transmitting data packets to their destinations. In the case of a high load (between 20 and 30 flows), network congestion becomes likely; the DCDR demonstrates around 14% reduction in delay over the EDOCR, 25% reduction over the EDCSCAODV, 26% reduction over the EDAODV and 28% reduction over the AODV. The reasons are as follows: when the number of flows increases, the network carries more traffic. The AODV incurred congestion due to increasing traffic but the DCDR seemed unaffected by increasing traffic because it resolved congestion by adaptively distributing the traffic into alternative paths. It has a two-hop CFS, from which it can easily choose a non-congested alternative node and establish it as a route to the destination immediately. This is due to the fact that the number of forwarding CFS nodes is minimal, which leads to reduced network congestion. For the DCDR at high loads (between 30 and 40 flows), the delay is reduced by around 15%. When the number of flows is between 40 and 50, the network became congested, and the difference between the five protocols in terms of end-to-end delay may not seem significant.



(a) End to End delay



(b) Packet delivery ratio



(c) Routing overhead

Figure 7 Performance when the number of connections (source and destination) changes.

Fig. 7(b) shows the achieved packet delivery ratio for the five protocols, which is similar when the load is below 20 flows. This is because when the number of flows is less, the number of nodes initiating the route discovery operation is also less. When the numbers of flows increases from 20 to 50, as a result, more RREQ packets are generated and transmitted; this leads to a high consumption of the node's queue, causing network congestion. This, in fact, leads to a fewer number of data packets being delivered at the destinations, thereby degrading network's performance. But it is clear from Fig. 7(b) that initially the DCDR constructed a two-hop CFS, which knows all the non-congested neighbors, both one-hop and two-hop neighbors, so that it takes lesser number of control packets than the AODV to find an alternative path. At a load of 20–30 flows, the packet delivery ratio of DCDR is increased from 7% to 15% when compared against the EDOCR. Whereas when compared with the EDCSCAODV, it increases from 11% to 21%, 17% to 27% over the EDAODV and it increases 20% to 28% over the AODV. The difference in the achieved packet delivery ratio is due to a reduction of the number of nodes involved in the broadcasting of RREQ packets in congested networks, leading to a reduction of the node's queue occupancy. As a result, more communication bandwidth is available for data transmission. When the number of flows increased from 40 to 50, the network falls into the congested zone and the delivery rate is about 29%. The difference between the five protocols in terms of packet delivery ratio may not seem significant.

With regard to the routing overhead, Fig. 7(c) shows that when the offered load is low (e.g. 10 flows), the DCDR did not give better performance than the EDOCR, EDCSCAODV, EDAODV and AODV. This is because the network became a safe zone at low offered load. When the offered load is increased from 20 to 30 flows, the AODV incurred a heavy routing overhead and consumed the heaviest control packets to find a new path, whereas the DCDR required the least number of control packets, around 8.5% as much as overhead of EDOCR, 17% over EDCSCAODV, 21% over EDAODV and 23% over AODV. The DCDR seemed unaffected by the increasing traffic because it could resolve congestion by using the CFS, which implicitly can distribute the packets over the alternative paths. This was the reason for the routing overhead of the DCDR being less than that of the AODV. When the number of flows increased from 30 to 40, the traffic was heavier, but the routing overhead of the DCDR was no more than 6% of that of the EDOCR, 9% of that of the EDCSCAODV, 11% of that of the EDAODV and 13% over AODV. When the number of flows increased from 40 to 50 and the network incurred the heaviest traffic, the routing overhead of the DCDR may not seem better than that of the EDOCR, EDCSCAODV, EDAODV and AODV.

3.4. Varying the CBR load

In our simulation experiments, the number of connections (for different sources and destinations) was kept at 20. The CBR rate varied from 4 packets per second to 40 packets per second.

Fig. 8(a–c) shows the end-to-end delay, packet delivery ratio and routing overhead for the DCDR, EDOCR, EDCSCAODV, EDAODV and AODV.

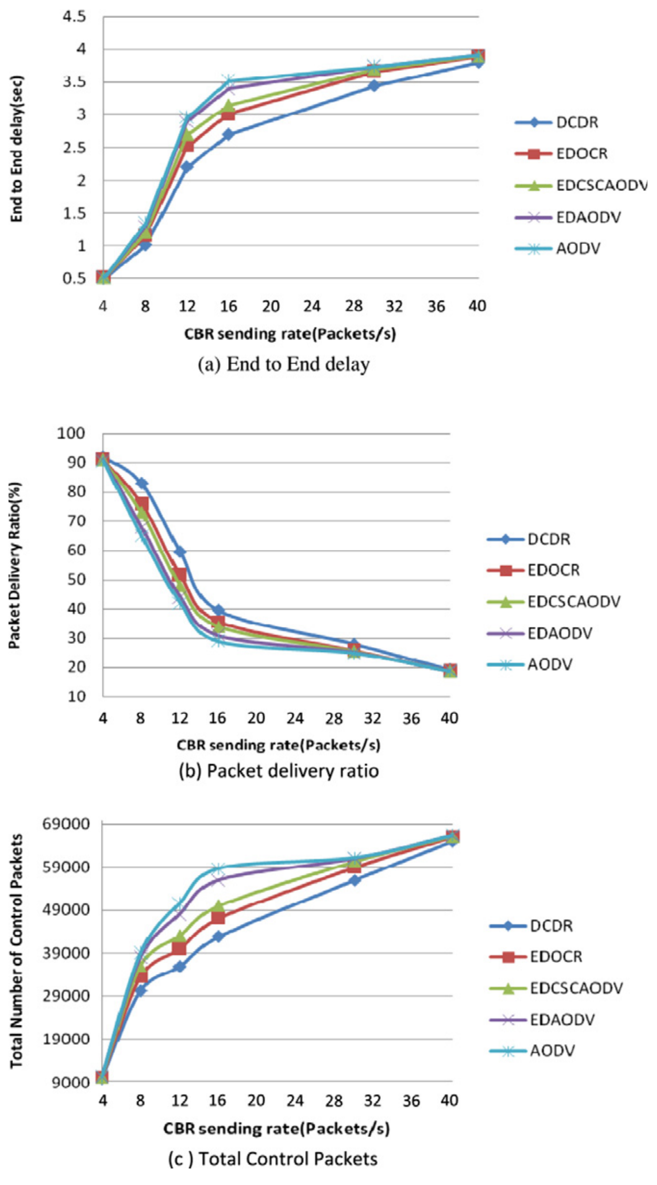


Figure 8 Performance when the CBR load changes.

When the packet rate is low (less than 8 packets per second), the delay incurred by both protocols increased almost linearly under increased load. When the packet rate is high (more than 8 packets per second), and the network is likely to be congested, the DCDR uses the CFS and all the two-hop non-congested nodes so as to find an alternative path at a minimum cost. The DCDR demonstrates a reduction in delay over the EDOCR, EDCSCAODV, EDAODV and AODV. This is because when the number of forwarding nodes gets reduced, broadcast and network congestion will arise. At a high packet rate (8–16 packets per second), the delay is reduced from 11% to 13% over EDOCR, from 14% to 19% over EDCSCAODV, 20.5% to 24% over EDAODV and from 23% to 26% over AODV. The DCDR, EDOCR, EDCSCAODV, EDAODV and the AODV converge to a similar performance when the rate was too high (30 to 40 packets per second) because the network incurred the heaviest traffic.

From Fig. 8(b), when the packet rate was small (less than 8 packet per second), the DCDR and the AODV delivered similar loads of packets because the network traffic was not yet heavy. But, when the packet rate was high (8 to 16 packets per second), the network becomes congested, when the DCDR uses a CFS to find an alternative path immediately. The DCDR shows an improvement of 9.6% to 13% packet delivery ratio than the EDOCR, from 13%to 19% than the EDCSCAODV, from 21.5% to26% than EDAODV and from 26.5% to 29.4% over AODV. When the packet rate was the highest (30–40 packets per second), no protocol could be considered the best as the delivery rate of about 20% was considered too low to be acceptable. However, our purposal was to show that the DCDR still performed better than the EDOCR, EDCSCAODV, EDAODV and ADOV in such networks.

Fig. 8(c) shows the routing overhead of the DCDR, EDOCR, EDCSCAODV, EDAODV and the AODV. When the traffic load was small (4–8 packet/s), the routing overhead of the DCDR, EDOCR, EDCSCAODV, EDAODV and the AODV was similar. More impressively, when the traffic was heavier (8–16 packets/s), the routing overhead of the DCDR was reduced from 9% to 11% than the routing overhead of the EDOCR, from 14% to16% than the EDCSCAODV, from 20% to 24% than the EDAODV and from 22% to 26% over AODV. The reason is as follows: when the traffic was heavier, the DCDR found a congestion status perfectly, so that it can easier find an alternative path than all four protocols. Therefore, a less number of route request packets were consumed by the EDOCR, EDCSCAODV, EDAODV and the AODV, but when more packets were generated into the network (30 or 40 packets/s), all five protocols incurred the heaviest routing overhead in more stressful network, and the reduction of the routing overhead by the DCDR was only 2.5% over all four protocols.

4. Conclusion

Congestion control techniques have been mainly designed for multimedia applications in MANETs. Network characteristics like congestion and route failure need to be detected and remedied with a reliable mechanism. To solve the congestion problem, we have proposed a novel dynamic congestion estimation technique that could analyze the traffic fluctuation and categorize the congestion status perfectly. After estimating the congestion status at the node level along a path, the DCDR controls the congestion by using an alternative path. The DCDR algorithm shows considerable performance over the EDOCR, EDAODV, EDCSCAODV and AODV. Our NS-2-based simulation confirms that the DCDR mechanism outperforms the EDOCR, EDAODV, EDCSCAODV and AODV in terms of reduction of delay and routing overhead and improvement in packet delivery ratio. The DCDR, however, has few limitations, which are as follows: (i) if the incoming traffic is the heaviest, the DCDR could minimize the packet loss caused by network congestion, but it still suffers from packet loss. (ii) This study did not consider any wireless losses. The limitations of our proposed algorithm may serve as directions for new research. By identifying and performing appropriate actions for router failure and channel error-induced packet losses, the performance of the DCDR could further be enhanced.

References

Athuraliya, S., Li, V., Low, S., Yin, Q., 2001. REM: active queue management. IEEE Network 15, 48–53.

Braden, B., Clark, D., Crowcroft, J., Davie, B., Eering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., Zhang, L., 1998. Recommendations on queue management and congestion avoidance in the internet. RFC 2309, IETF.

Broch, J., Johnson, D., Maltz, D., 1999. The dynamic source routing protocol for mobile ad hoc networks. IETF Internet Draft.

Chen, L., Heinzelman, W.B., 2007. A survey of routing protocols that support QoS in mobile ad hoc networks. IEEE Network 21 (6), 30–38.

Feng, W.-C., Kandlur, D.D., Saha, D., Shin, K.G., 2001. Stochastic fair blue: an algorithm for enforcing fairness, In: Proceedings of INFOCOM 2001, Anchorage, Alaska, USA, pp. 1520–1529.

Floyd, S., 1997. RED: Discussion of Setting Parameters. [Online]. Available from: <http://www.icir.org/floyd/REDparameters.txt>.

Floyd, S., Jacobson, V., 1993. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking 1 (4), 397–413.

Johnson, David B., Maltz, David A., 1996. Protocols for adaptive wireless and mobile networking. IEEE Personal Communications 3 (1), 34–42.

Lochert, C., Scheuermann, B., Mauve, M., 2007. A survey on congestion control for mobile ad-hoc networks. Wiley Wireless Communications and Mobile Computing 7 (5), 655–676.

Siva Ram Murthy, C., Manoj, B.S., 2007. Ad hoc Wireless Networks Architectures and Protocols, second ed. Pearson Education, Delhi, India.

NS2 Network Simulator. Last accessed on Feb. 3, 2000 [Online]. Available from: <http://www.isi.edu/nsnam/ns/>.

Perkins, C.E., 1994. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: Proceedings of ACM SIGCOMM 1994, NY, USA, pp. 234–344.

Perkins, C.E., Belding-Royer, E.M., Chakeres, I., 2003. Ad hoc on demand distance vector (AODV) routing. IETF Internet Draft.

Ramanathan, R., Redi, J., 2002. A brief overview of ad hoc networks: challenges and directions. IEEE Communications Magazine 40 (5), 20–22.

Senthilkumaran, T., Sankaranarayanan, V., 2010. Early detection congestion and control routing in MANET, In: Proceedings of the Seventh IEEE and IFIP International Conference on Wireless and Optical Communications Networks (WOCN 2010), Srilanka, pp. 1–5.

Senthilkumaran, T., Sankaranarayanan, V., 2011. Early detection congestion and self cure routing in MANET. In: Proceedings of Springer LNCS Computer and Information Science, vol. 142 (3), Bangalore, India, pp. 562–567.

Senthilkumaran, T., Sankaranarayanan, V., 2011b. Early congestion detection and optimal control routing in MANET. European Journal of Scientific Research 63 (1), 15–31.

Senthilkumaran, T., Sankaranarayanan, V., 2011c. Early congestion detection and adaptive routing in MANET. Egyptian Informatics Journal 12 (3), 165–175.

Tran, D.A., Raghavendra, H., 2006. Congestion adaptive routing in mobile ad hoc networks. IEEE Transactions on Parallel and Distributed Systems 17 (11), 16–28.

Yen, Yun-Sheng, Chang, Hung-Chieh, Chang, Ruay-Shiung, Chao, Han-Chieh, 2010. Routing with adaptive path and limited flooding for mobile ad hoc networks. Elsevier Transaction on Computers and Electrical Engineering 36 (2), 280–290.

Yingqun, Y., Giannakis, G.B., 2008. Cross-layer congestion and contention control for wireless ad hoc networks. IEEE Transactions on Wireless Communications 7 (1), 37–42.