SMORT: Scalable multipath on-demand routing for mobile ad hoc networks

L. Reddeppa Reddy *, S.V. Raghavan

Network Systems Laboratory, Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai 600036, India

Received 23 June 2004; received in revised form 17 October 2005; accepted 20 October 2005
Available online 17 November 2005

Abstract

Increasing popularity and availability of portable wireless devices, which constitute mobile ad hoc networks, calls for scalable ad hoc routing protocols. On-demand routing protocols adapt well with dynamic topologies of ad hoc networks, because of their lower control overhead and quick response to route breaks. But, as the size of the network increases, these protocols cease to perform due to large routing overhead generated while repairing route breaks. We propose a multipath on-demand routing protocol (SMORT), which reduces the routing overhead incurred in recovering from route breaks, by using secondary paths. SMORT computes fail-safe multiple paths, which provide all the intermediate nodes on the primary path with multiple routes (if exists) to destination. Exhaustive simulations using GloMoSim with large networks (2000 nodes) confirm that SMORT is scalable, and performs better even at higher mobility and traffic loads, when compared to the disjoint multipath routing protocol (DMRP) and ad hoc on-demand distance vector (AODV) routing protocol.

© 2005 Elsevier B.V. All rights reserved.

Keywords: Mobile ad hoc networks; Multipath routing; Fail-safe path; Fault-tolerance; Disjoint multipath routing protocol (DMRP); Primary path

1. Introduction

Mobile ad hoc networks are self-configuring and self-organizing wireless networks, which operate without any fixed infrastructure or wired backbone. Nodes typically communicate over multiple hops, while intermediate nodes act as routers by forwarding data. Topology of ad hoc network is highly dynamic because of mobility and limited battery power of nodes. Routing protocols should adapt to such dynamism, and continue to maintain connection between the communicating nodes in the presence of path breaks caused by mobility and/or node failures.

As the user base of wireless devices increases, routing protocols need to scale to networks with thousands of nodes. Typical examples of large ad hoc networks are technical festivals in universities and military communication networks (which involve hundreds to thousands of devices).
Maintaining routes in large networks becomes cumbersome due to longer path lengths between node pairs. Longer the paths, more the number of nodes on the path and more is the possibility of route breaks because, any single node failure disconnects the source from destination. The limitations of existing ad hoc routing protocols in supporting scalability is explained below.

**Proactive routing protocols** [1–3] are based on either link-state [4] or distance-vector [5] routing schemes. These protocols compute routes to all the nodes in the network, and maintain them in background by periodically exchanging routing updates. Nodes also exchange triggered updates (when the neighboring nodes move) to maintain the consistent view of topology. These protocols cannot scale because of their overwhelming storage and communication overhead, which exists irrespective of the route requirement. Destination-sequenced distance vector (DSDV) [1] routing protocol for ad hoc networks, is an optimized version of the popular Distributed Bellman Ford distance-vector routing algorithm [6]. Nodes broadcast routing messages whenever they detect a topological change, and they exchange periodic updates with very low frequency. But, this also leads to considerably high overhead when nodes are moderate to highly mobile. Pei et al. [2] proposed fisheye state routing (FSR) based on link-state exchanges. This protocol limits the routing overhead, by optimizing the frequency of link-state updates based on the distance between nodes. Nodes send link-state updates with high frequency to the nearest nodes and with lower frequency to those far away. Although, the amount of routing overhead incurred is lower than that of original link-state algorithm, it becomes unmanageable in large networks as movement of any node in the network triggers an update. Further, nodes often possess stale routing information to far away nodes, if they are highly mobile. This makes the route convergence difficult.

**Hierarchical routing protocols** [7–9] reduce the overhead generated by periodic updates, using clustering. Hierarchical state routing (HSR) [7] groups nodes into clusters based on their geographical proximity, and a node in the cluster is elected as cluster-head to represent that cluster. These clusters are further grouped to form higher-level clusters and so on. Clustering frees the nodes inside a cluster, from communicating routing updates to the nodes in other clusters. Only cluster-head maintains routes to the nodes outside its cluster, by exchang-
significant impact on AODV’s performance in large networks. When networks grow to thousands of nodes, number of route breaks increases due to longer path lengths and mobility. A route error packet is sent to source for every route break, and the source node initiates a new route discovery to reestablish its connection to destination. Flooding the entire network each time a route is required consumes considerable part of the limited processing power of nodes, as well as expensive network bandwidth. These characteristics of the protocol limit its ability to scale to larger networks. Scalability study of AODV done by Lee et al. [14] asserts this fact. Although, local-repair techniques proposed for AODV tries to minimize the route recovery overhead by locally repairing the route break, the network is still flooded to a significant extent if the point of route break and destination node are far apart (which is usually the case in large networks). Also, local-repair techniques involve a non-zero delay in resuming the session after the route break occurs.

In this paper, we propose a novel multipath routing protocol, SMORT, in order to minimize the route break recovery overhead. SMORT provides most of the intermediate nodes on the primary path with multiple paths to destination, along with source node. Primary path is the first path received by source node after initiating the route discovery, which is usually the shortest path. SMORT does not impose the disjoint-ness constraint on the set of multiple paths that it generates. We believe that having multiple paths to destination at intermediate nodes of the primary path, avoid overhead of additional route discovery attempts, and reduce the route error transmitted during route break recovery. Nasipuri et al. [15] have also showed, analytically as well as through simulations, that having multiple paths at all intermediate nodes on the primary path improves the performance of a multipath routing protocol, more efficiently than having multiple paths at the source node only. But, their performance study is based on theoretical analysis, where it is difficult to take into account the influence of arbitrary movement of nodes and unreliable radio transmission. SMORT is an extension to the unipath routing protocol AODV [11]. Although SMORT is based on AODV, the basic idea of multipath routing for limiting the routing recovery overhead is applicable to all on-demand routing schemes. We study the scalability of SMORT and compare it with the disjoint multipath routing protocol (DMRP), which computes only disjoint multiple paths to destination. The results are compared with the AODV protocol also because, it is important to know if the multipath protocol provides better scalability than its unipath counterpart, despite the extra overhead involved in computing and maintaining multiple paths.

Rest of the paper is organized as follows. In Section 2, we present the related work. Section 3 describes SMORT’s operation. Section 4 presents the simulation model used for performance evaluation, while Section 5 analyzes the simulation results. Finally, Section 6 concludes the paper.

2. Related work

In this section, we briefly present the research work related to multipath routing in literature. Multipath routing has been well studied in wired networks. Maxemchuck [31] proposed a routing mechanism called dispersity routing for store and forward data networks. It discusses the ways of splitting data and dispersing them over multiple paths to achieve smaller average and variance in delay. The popular link-state protocol OSPF [32] can find multiple paths of equal cost. In [33], Vutukury and Garcia-Luna-Aceves propose a routing protocol called MDVA to find multiple loop-free paths, which can be used for traffic load balancing and minimizing delay. The distance-vector routing protocol proposed in [34] can find multiple loop-free paths using the distributed algorithm for shortest path (DASM). But, all the above protocols include high routing overhead and do not suit bandwidth-constrained ad hoc networks. They were proposed for wired networks, where routing overhead is not an issue.

Recently, some multipath routing protocols [35,37,15] have been proposed for ad hoc networks also. Multipath source routing (MSR) [35,36], extends DSR’s route discovery and route maintenance phases to compute multiple node-disjoint paths. It also proposes a mechanism to distribute load over multiple paths, based on the RTT measurement. SMR [37] finds maximally disjoint multiple paths and uses a per-packet allocation scheme to distribute data packets on to multiple paths. This enables the effective utilization of network resources and avoids nodes from being congested. SMR computes only two paths to each destination. All the above protocols are based on the source routing protocol DSR [10]. They cannot scale to large networks because source routing requires every data
packet to carry full path to the destination. In large networks, size of data packets become prohibitively high due to the longer paths they carry. Also, scalability study of all the above protocols is not prominent in the literature.

Multipath protocols [38–41] based on distance-vector routing scheme have also been proposed for ad hoc networks. AODV-BR [38] calculates multiple paths without any extra control overhead. In this protocol, neighboring nodes hear the route reply transmissions by being in promiscuous mode, and store a route to the destination through the neighbor that transmitted the reply packet. The new path thus found is called a backup path. But, effectively nodes on the primary path contain only single path to the destination. It is the neighboring nodes who store backup paths. When a node on the primary path moves away due to mobility, it loses connection to its immediate downstream node on the path. Then, the node broadcasts the future data packets that it receives for that destination, assuming that any of its neighbors would have stored a backup path to the destination. The node also sends a route error packet to the source node, informing the route disconnection. This scheme has two disadvantages. Firstly, after a node detects a link break, the future data packets it receives are broadcasted for which there is no link layer acknowledgment. So, the recipient that has a backup path has to send an explicit network layer acknowledgment to inform the safe receipt of data packets, which increases the control overhead. If an acknowledgment is not sent for packets, packet drops are not detected at the sending node, which affects the throughput performance of the protocol. Secondly, this scheme works only if the nodes that moved away are within the transmission range of its immediate upstream node on the path and one of its neighbors that stored backup paths to destination.

Caching and multipath routing protocol (CHAMP) [39] uses packet cache and multipath routing to reduce the number of packet drops caused by route breaks. But, CHAMP can find only equal length multiple paths, which may not be possible when the network is moderate to highly sparse. Ad hoc on-demand multipath distance vector (AOMDV) [40] is a multipath routing protocol based on AODV [11], which can compute both node-disjoint and two segment link-disjoint paths. It uses a notion of advertising-hopcount to form an invariant that ensures loop freedom. But, AOMDV also can compute only equal length multiple paths due to the invariant it maintains, and hence suffers from the same disadvantage as CHAMP. Further, in large networks two segment link-disjoint paths also take considerable amount of time to recover the route break, as route error has to traverse multiple hops to inform the route disconnection to the node that has alternate path(s). This increases the overhead of route error transmissions also.

Ad hoc on-demand distance vector multipath (AODVM) [41] is also a multipath routing protocol based on AODV. It proposes a routing framework to provide robustness to route breaks. The protocol computes node-disjoint paths between source and destination. Through simulation results, it shows that only a few such multiple paths can be found in the network and hence they cannot provide much robustness. In order to increase robustness to route breaks, AODVM assumes the existence of a set of reliable nodes in the network and place them at the junctions of the link-disjoint multiple paths. A mechanism to identify such reliable nodes in the network would be a good addition to the protocol.

Many disjoint multipath routing techniques [16,20,23,17,22,25,26,29] have been proposed for ad hoc networks, which have focused on improving the reliability of routing using path disjointness or redundancy. Wu and Harms [16] proposed a node-disjoint multipath routing protocol for traffic load-balancing. They introduce a correlation factor for a set of multiple paths between source and destination, which measures the disjointness of paths in the set. The routing algorithm selects the set of multiple paths with minimum correlation so as to minimize the interference between transmissions in the individual paths. Saha et al. [20] proposed a maximally zone-disjoint multipath routing, which computes a set of zone-disjoint shortest paths for traffic load-balancing. The zone-disjointness of paths minimizes the congestion for the traffic sent simultaneously over the multiple paths. Disjoint multipath source routing proposed in [23], statically multiplexes the data traffic over multiple disjoint paths at all nodes on the primary path. It achieves better transport capacity [23] by doing so, when compared to the original source routing algorithm, in which packets go on a single path from source to destination. Tsirigos and Haas [17] proposed a disjoint multipath routing protocol that can be used in the presence of frequent topological changes. It uses multiple paths.

\[1\text{ Refer to Section 3.}\]
simultaneously, by splitting the information among the multitude of paths, so as to increase the probability that essential portion of the information is received at the destination, without incurring excessive delay. The stability based multipath routing algorithm proposed in [22] computes a set of stable independent (disjoint) multiple paths, which can be used for a longer time without rerouting, to recover from the path breakage.

Li and Cuthbert [25] proposed a stable node-disjoint multipath routing, which applies the path accumulation feature of DSR to AODV. But, this path accumulation feature requires the route request packet to carry the full path it has traversed. This requirement increases the size of route request packet, particularly in large networks where paths between nodes are longer. These large-sized route request packets, which are flooded across the entire network for route discovery, increase the routing overhead and thereby limit the network scalability. Disjoint multipath routing [26] proposed by Abbas and Jain tries to reduce the effect of path diminution problem in finding node-disjoint multiple paths. As this routing technique also requires the route request packets to carry the traversed path, it suffers from the same disadvantage as the previous protocol. In [27], Ducatelle et al. propose a hybrid multipath routing based on ant colony optimization framework for traffic load-balancing. Multipath fresnel zone routing [28] proposed by Liang and Midkiff take the capacity of intermediate nodes into consideration for selecting disjoint multiple paths. It evaluates the capacity and the transmitting cost of different intermediate nodes, and formulates end-to-end paths of different capacity and cost. Then the protocol forwards the traffic through these different paths, by adjusting the amount of traffic on each path based on path capacity and congestion conditions.

Papadimitratos et al. [29] proposed a reliable disjoint multipath selection approach using an efficient heuristic mechanism. Roy et al. [30] compared the two disjoint multipath techniques that use omnidirectional and directional antennas, respectively. They showed through simulations that directional antennas help in computing multiple paths efficiently, when compared to omnidirectional antennas. Fault tolerant routing proposed by Xue and Nahrstedt [24] uses a path estimation mechanism for selecting a reliable route. This algorithm relies on destination nodes sending feedback to source nodes, about the packet delivery capacity of multiple paths, which may increase the control overhead in the network. Also, there is some protocol overhead involved in implementing the path estimation mechanism. All the disjoint multipath routing techniques discussed above compute maximally disjoint multiple paths, whose availability is very less due to the disjointness constraint imposed in the path selection. Hence, disjoint multiple paths cannot provide efficient fault-tolerance towards route breaks. Also, these techniques involve considerable delay and overhead in selecting disjoint multiple paths. The variation of availability of disjoint multiple paths with network size is discussed in Section 3.

The multipath routing mechanisms proposed in [18,19,21] use path redundancy for ensuring confidentiality to data transmission over ad hoc networks, which are not closely related to fault-tolerant multipath routing techniques that we are addressing in this paper.

3. Scalable multipath on-demand routing (SMORT)

The principal objective of SMORT is to reduce the amount of routing overhead generated by an unipath on-demand routing protocol, using multipath routing. Alternate paths to destination avoid the overhead generated by the additional routing discoveries and route error transmissions, during route break recovery. Reduction in routing overhead allows the protocol to scale to larger networks.

Multiple paths between a source and a destination are of two types, namely node-disjoint and link-disjoint multiple paths. Node-disjoint paths do not have any nodes in common, except the source and destination, as shown in Fig. 1(a). Nodes labeled S and D are source and destination nodes, respectively. Node-disjoint multiple paths are used for traffic load-balancing (by dispersing data over multiple paths), and provide fault-tolerance towards route breaks. The advantage of node-disjoint multiple paths is that they fail independent of each other. Breakage of any link on one path can be corrected by resuming the data session through one of the other paths. Link-disjoint paths do not have common links, but may have nodes in common. A set of link-disjoint paths are formed by a series of node-disjoint segments. Each segment is a node-disjoint path between any two nodes. For example, Fig. 1(b) shows link-disjoint multiple paths between S and D, formed with two segments. Although, link-disjoint paths are more available than node-disjoint
paths, movement of nodes at the junctions (for example, node C in Fig. 1(b)) causes the failure of all the paths going through that node.

SMORT uses the idea of fail-safe multiple paths. A path between source and destination is said to be fail-safe to the primary path, if it bypasses at least one intermediate node on the primary path. In other words, the fail-safe path can be used to send data packets in case the bypassed node(s) on the primary path move away. For example, in Fig. 1(d), the paths S–A–H–C–E–D and S–A–B–C–L–D are fail-safe paths to the primary path S–A–B–C–E–D. The data session between S and D is unaltered even if nodes B and E move away at the same time, as the packets can be redirected through the fail-safe paths.

Fig. 1(d) shows a set of such paths, which together bypass all the nodes on the primary path and provide most of the intermediate nodes with alternate routes to the destination. Fail-safe multiple paths are different from both node-disjoint and link-disjoint multiple paths, in the sense that fail-safe paths can have both nodes and links in common. This less restrictive constraint allows the computation of more fail-safe paths than node-disjoint or link-disjoint multiple paths. We refer to the segment of the fail-safe path that bypasses the node(s) on the primary path as fail-safe segment, and alternate paths at nodes as secondary paths. For example, B–K–E in Fig. 1(d) is a fail-safe segment that bypasses the node C on the primary path, and path at node B through the node K is a secondary path.

On-demand routing scheme that computes fail-safe multiple paths reduces the route recovery time and path maintenance overhead more effectively than the node-disjoint multipath routing scheme. When node-disjoint multiple paths are used, only the source can correct the route disconnections, as alternate paths exist only at that node. In effect, route error packets have to be sent to the source node for every link break. In large networks, these error packets are likely to take considerable amount of time to reach the source node from the point of route break. Besides, the number of route errors communicated may also be high, as more number of nodes transmit these packets. Alternatively, usage of fail-safe paths has the advantage that route disconnection gets corrected at an intermediate node itself, thereby reducing the route recovery time and the number of route error transmissions. For example, in Fig. 1(a) when the node E moves away, breaking its link to D, C has to inform S about the link break by sending an error packet, which all the way has to traverse three hops to reach the source. In large networks, this number would be in the order of tens. On the other hand, in Fig. 1(d) when the same node moves away, C simply redirects the future data packets to the destination through R, without generating and communicating any error packets back to the source.

As shown in Fig. 1(c), if node-disjoint multiple paths to destination are available at all nodes on the primary path (instead of only at source), intermediate nodes on the primary path can also recover route breaks using their secondary paths. The added advantage now is that multiple paths at intermediate nodes are disjoint, whose failure is independent of each other. We call the protocol that computes disjoint multiple paths at all nodes on the primary path as disjoint multipath routing protocol (DMRP).

However, we found through simulations that such paths are scarce in the network, when compared to the number of fail-safe multiple paths. Fig. 2 shows the availability of node-disjoint and fail-safe paths found at all the nodes on the primary path, and for various networks sizes (from 100 to 2000 number of nodes). The availability is represented as the average distance (in number of hops) between multipath nodes on the primary path. Multipath nodes are the nodes that have at least

---

**Fig. 1.** Different types of multiple paths.

(a) Node–disjoint multiple paths only at source  
(b) Link–disjoint multiple paths  
(c) Node–disjoint multiple paths at all nodes on the primary path  
(d) Fail–safe multiple paths
two paths to destination. The lower the inter-distance between nodes, higher is the availability of multiple paths, as more number of nodes on the primary path possess secondary paths. The variation of the length of the primary path is also shown in Fig. 3. Twenty data sessions were generated between randomly selected source and destination pairs, and the distance between nodes on the primary path that have multiple paths to destination are counted. Each node on the primary path accepts at most two secondary paths, in order to limit the multipath computation overhead. The plot in Fig. 2 shows that fail-safe multiple paths are more available than that of node-disjoint multiple paths. Also, availability node-disjoint multiple paths drops more rapidly with increase in network size, when compared to fail-safe multiple paths. The reason for the lower number of node-disjoint paths is that each node-disjoint path requires a different set of nodes to pass through, as they should not intersect the other paths.

Fig. 2. Availability of fail-safe and node-disjoint paths.

Fig. 3. Variation of the length of the primary path.
to the same destination—at that node and at its downstream nodes. Thus, in networks with insufficient number of nodes, total number of node-disjoint multiple paths turn out to be very low. However, the number of fail-safe multiple paths are abundant due to the less restrictive constraint (multiple paths need not be disjoint) imposed in computing multiple paths.

Even if node-disjoint paths are made available at all nodes on the primary path (by deploying more number of nodes), they tend to be longer than fail-safe multiple paths. This is because, node-disjoint paths generally span more number of nodes in maintaining disjoint-ness. The plot in Fig. 4 supports this statement. It shows the average number of hops by which the secondary paths are longer than the shortest path. The plot clearly shows that, node-disjoint secondary paths are longer than fail-safe secondary paths. The limitations of longer paths have already been mentioned. The following probabilistic analysis shows that fail-safe multiple paths have higher fault-tolerance to route breaks due to their higher availability, when compared to that of node-disjoint multiple paths.

We define fault-tolerance $F$ of a set of multiple paths between source and destination for a given node failure probability, as the probability that at least one of the paths in the set are intact.

The derivation makes use of the following definitions.

For a network with $n$ number of nodes,

- $m$ number of fail-safe paths,
- $m'$ number of node-disjoint paths,
- $k$ average length of a fail-safe path (in number of nodes),
- $k'$ average length of a node-disjoint path (in number of nodes),
- $p$ probability that a node on a path is functioning,
- $P_i$ probability that a fail-safe path $i$ is intact,
- $P_i'$ probability that a node-disjoint path $i$ is intact,
- $F_{fs}$ fault-tolerance of fail-safe multiple paths,
- $F_{nd}$ fault-tolerance of node-disjoint multiple paths.

As $P_i$ is the probability that all $k$ nodes on a fail-safe path are functioning,

$$P_i = p^k.$$

Similarly,

$$P_i' = p^{k'}.$$

According to our definition of fault-tolerance,

$$F_{fs} = 1 - (1 - P_1)(1 - P_2) \cdots (1 - P_m)$$
$$= 1 - (1 - p^k)^m.$$
Similarly,
\[ F_{nd} = 1 - (1 - p^k)^m. \]

If \( F_{fs} - F_{nd} > 0 \) is proved, then the fault-tolerance of fail-safe multiple paths is greater than that of node-disjoint paths, i.e., \( F_{fs} > F_{nd} \).

As \( k' > k \) from the simulation results presented in Fig. 4,
\[ (1 - p^k)^m > (1 - p^k)^{m'} \tag{1} \]

Considering,
\[ F_{fs} - F_{nd} = (1 - p^k)^{m'} - (1 - p^k)^m \]
\[ > (1 - p^k)^m - (1 - p^k)^m \]
substituting the result from expression (1)
\[ > (1 - p^k)^m \left( 1 - \frac{(1 - p^k)^m}{(1 - p^k)^m} \right) \]
\[ > (1 - p^k)^m \left( 1 - (1 - p^k)^{m-m'} \right). \tag{2} \]

(1 \(- (1 - p^k)^{m-m'} > 0 \), as \( m > m' \) from Fig. 2 and (1 \(- p^k)^{m} > 0 \), as \( m, m' > 0 \). Substituting these two results in the expression (2), \( F_{fs} - F_{nd} > 0 \), i.e., \( F_{fs} > F_{nd} \). Hence, fail-safe multiple paths provide better fault-tolerance than node-disjoint multiple paths.

3.1. Overview of SMORT

As mentioned earlier, SMORT is a multipath extension to the well-known unipath ad hoc routing protocol AODV. SMORT has three basic phases; namely route discovery, route reply and route maintenance. When a node needs a route to some destination, it initiates route discovery process, by flooding a route-request packet into the network. Intermediate nodes receiving the route-request, send a route-reply packet back to source if they have a valid path to the destination. Otherwise, they re-broadcast the request. Finally, when the destination receives the request, it initiates route reply process by sending a route-reply packet back to the source node. Unlike AODV, SMORT allows nodes to accept multiple copies of route-request packet, in order to enable computation of multiple fail-safe paths. Also, the destination replies to multiple copies of route-request for the same reason.

Route reply reaches the source node through the reverse path recorded in a special table, during the route discovery phase. In order to avoid loops in the routes that may form due to the acceptance of multiple copies of route-request, route-reply packets carry the full path to the destination. Although, loops can be avoided in the route discovery phase itself, by carrying the list of nodes traversed in the route-request packet, SMORT does not carry full path in route-request packets as it may increase network wide collisions due to the flooding of large-sized route-request packet across the entire network. On the contrary, the number of route-reply packets communicated are limited when compared to the number of route-request packet transmissions (many copies of the route-request do not generate reply as they move away from destination), and they traverse on the actual routes between source and destination. Finally, when the source receives the first route, it starts sending data packets to the destination. Intermediate nodes may receive multiple route-reply packets as the destination replies to multiple copies of route-request, but they relay only the first reply. The reply is relayed through multiple neighbors through which nodes received the route-request packet previously. Extra replies are dropped after nodes copy secondary paths carried in them into their routing tables.

Route maintenance involves two important actions. Firstly, re-establishing the connection between source and destination, if all the routes between them fail in the middle of the session. Secondly, deleting the expired routes from the routing table.

3.2. Route discovery phase

A node initiates route discovery process, when it wants to communicate to a destination, for which it does not have a valid route. Valid route is a route to the destination, whose lifetime has not expired, i.e., the lifetime value of the route entry should be greater than the current time at the node. The source node inserts address of the destination into a route-request packet and broadcasts it. Route request packet structure of SMORT is same as that of AODV, except that SMORT does not use any destination sequence numbers.

An intermediate node receiving the route-request, replies by sending a route-reply packet if it has a route to the destination. Otherwise, it simply rebroadcasts the route-request. Although, nodes accept multiple copies of route-request, only the first copy of the route-request is re-broadcasted. Nodes store all route-request copies in a table called request-rcvd table. Route request information is
stored in the request-rcvd table, instead of establishing a reverse route to source as in AODV because, the reverse route may contain loops due to the acceptance of multiple requests. Each entry in the request-rcvd table contains address of the previous node that relayed the route-request to it (called last-hop), and the number of hops the route-request has traversed from the source node. Nodes use this information to relay route-reply packets back to source node. If none of the intermediate nodes possess a fresh route to the destination, the destination itself replies to the route-request, if it receives a copy of the route-request.

### 3.3. Route reply phase

Route replies follow the reverse paths stored in the request-rcvd table to reach the source node. The route-reply packet used by SMORT contains three extra fields, apart from some of the fields of AODV’s reply packet. These extra fields (shown in bold letters in Fig. 5) are required to eliminate routing loops, and to compute fail-safe multiple paths. The node-list field contains the list of all nodes that the reply packet has traversed so far. The reply-gen field is for storing address of the node from which that particular copy of the route-reply packet originated, and mul-reply is a Boolean variable, whose significance will be explained a little later.

Before sending route-reply, the destination node initializes the node-list and reply-gen fields to its address. The mul-reply field is set to TRUE for the first reply. For the extra replies that the destination generates, it sets the mul-reply value to FALSE. The nodes receiving the route-reply accept it, if it is the first reply for that destination, and store the route information carried in it into the routing table, along with the full path. The structure of the typical routing entry is shown in Fig. 6. Multiple routes to destination are stored in route-list of the routing entry. Each individual route route; has, nexthop; as the address of the neighbor through which the route goes to destination, hopcount; as the distance to destination (in number of hops), and fullpathi as full path to destination. precurs-list is the list of last hops through which route-reply packet is relayed to source. The purpose of lifetime, is explained in Section 3.4.

After storing the routing information, intermediate nodes relay the route-reply packet through reverse paths to the source node. They append their addresses to the node-list before relaying the reply packet. Nodes relay route-reply packets through multiple reverse paths, each one through a different lasthop, if the mul-reply field of the received route-reply is TRUE. We limit the number of such multiple replies a node can relay to MAX-REPLY, in order to avoid route-reply storm. The list of last-hops, through which replies are sent, are stored in the precurs-list of the routing entry. This list is required to send the route error packet to source node, in case the node detects a route break and no secondary paths to the destination are available at that node. Nodes send the first copy of the reply without changing the values of mul-reply and reply-gen fields. In rest of the copies, mul-reply is set to FALSE so that the nodes on the secondary path do not send multiple replies. This is because, nodes on the secondary paths need not have multiple paths. This arrangement limits the number of route-reply packet transmissions also. Further, the reply-gen field is changed to the node’s address, as this is the node originating this particular copy of the route-reply. This arrangement helps in eliminating routing loops (refer Section 3.3.1).

If mul-reply value of the route-reply received by a node is FALSE, the node is on a fail-safe segment. Hence, it relays only one copy of route-reply without changing mul-reply and reply-gen values. For example, in Fig. 7, node E receives route-reply with mul-reply value TRUE from destination D. It relays three copies of the reply packet through nodes C, H.
and M. But, only the route-reply sent through C has mul-reply value as TRUE, as C is on the primary path. Thus, only the nodes on the primary path can relay multiple replies. Replies sent through H and M contain mul-reply value as FALSE, and they do not relay multiple replies, even if they have received multiple copies of route-request. reply-gen values of route-reply packets relayed through nodes H and M are changed to E. Similarly, node D replies to extra route-request packets through J and N with mul-reply value FALSE.

When the node receives an extra route-reply, it accepts the route-reply only if the node-list of the route-reply does not contain its address, and this copy of route-reply was originated by one of the nodes on the primary path. Otherwise, the route-reply is discarded by the node. The node can check whether the route-reply was generated by one of the nodes on its primary path or not, using the reply-gen field of route-reply and the fullpath field of the primary path. Nodes do not relay the extra route-reply packets to the source node. They are dropped after the node stores the routing information carried by them into the routing table. When node C in Fig. 7 receives extra replies through J and N, it does not relay them further as it has already relayed the route-reply received from E. At anytime nodes use the shortest of the multiple paths to forward data packets to destination.

In case an intermediate node receiving the route-request has a route to the destination, it initiates the reply process by copying its full path to the destination in the node-list field of route-reply packet. reply-gen value is set to this node’s address and mul-reply is set to TRUE. The aforementioned steps involved in processing the route-reply packet are given in Algorithm 1.

Algorithm 1. Rules for processing route-reply received by an intermediate node nodei for the destination dest. LIFETIME is the lifetime of newly updated route.

Module HandleReply(reply, nodei)
begin
if (reply is the first reply for destination dest)
create (routentry_dest);
routentry_dest.destaddr = reply.destaddr;
insert (reply.nexthop, reply.hopcount, LIFETIME, reply.node-list) into routentry_dest.route-list;
counter = 0;
append (nodei.addr, reply.node-list); /* current node’s address is appended to node-list */
if (reply.mul-reply = TRUE) /* send multiple replies */
for (every lasthop of reply.srcaddr in request-rcvd table)
if (counter = 0) /* first copy of the reply is sent as it is */
relay (reply, lasthop);
counter = counter + 1;
end if
else
reply.mul-reply = FALSE;
reply.reply-gen = nodei.addr;
relay (reply, lasthop);
counter = counter + 1;
end else

Fig. 7. Fail-safe multiple paths between nodes S and D.
3.3.1. Eliminating routing loops

As SMORT allows nodes to accept multiple copies of route-request packet, and the extra route-reply packets received at intermediate nodes are not relayed to the source node, there is possibility for loops in the end-to-end route. First, we discuss how loops form on the paths, and then prove that these loops are eventually eliminated by SMORT.

In SMORT, loops can exist either on the primary path or, fail-safe segments can form loops with the primary path. Loops do not form on a fail-safe segments alone because, nodes on fail-safe segments relay only one copy of the route-reply packet. The reply is sent through the node through which they have received the first route-request packet.

Loops form on the primary path, when an intermediate node replies to route-request with a path, which goes through one or more of the nodes that have relayed the route-request packet previously. Loops formed by fail-safe segments are of two types. The first type has only one node of the primary path on the loop. The second type includes multiple consecutive nodes of the primary path on the loop. Among the multiple consecutive nodes common to loop and the primary path, we call the first node from the source node side as first loop node, and the last node as last loop node. For example, in Fig. 8, the primary path is S–A–B–C–D. Fig. 8(a) shows the first type of loop with fail-safe segments. The loop B–E–F–G–B has only one node of the primary path, i.e., B on it. The second type loop A–B–C–E–A shown in Fig. 8(b) includes three nodes (A, B and C) of the primary path. Here, node A is the first loop node and node C is the last loop node. We explain two example scenarios in which fail-safe segments form loops with the primary path, in different ways.

Fig. 9 shows an example scenario for the first type. This type of loops from when a node accept a copy of route-request, which it itself has relayed previously. In Fig. 9(a), node B re-broadcasts the route-request received from A. It also shows node K residing far away from B, which cannot hear the route-request broadcasted by B. Fig. 9(b) shows the state where K moves to the transmission range.
common to H and B and receives the route-request relayed by node H. It also shows the destination D receiving the route-request from C. Now, when K re-broadcasts the route-request, node B receives the route-request which it itself has relayed in the past, as shown in Fig. 9(c). Node B accepts this duplicate route-request, assuming that it forms a valid fail-safe path. Fig. 9(d) shows the loop formation when the node B receives the reply sent by destination D through node C. The arrows in Fig. 9(d) point to the direction in which route replies are being relayed. Node B in turn relays the reply through K and A. Reply relayed through K is again received at B, through the reverse path forming loop with B. If B transmits data packets through K (when the primary path is broken), they go in loop B–K–H–G–B until they get dropped when packet’s time-to-live limit is exceeded.

The second type of loop forms when nodes receiving extra route-reply packets do not relay them to source node. Fig. 10 depicts an example situation. In Fig. 10(a) when nodes B and E receive the route-request broadcasted by A, node G cannot receive the route-request as it is out of the transmission range of A. But, it may move into the common transmission range of nodes B and A due to mobility, by the time node B is ready to broadcast the route-request packet. As shown in Fig. 10(b), node A receives the same route-request from G, which G received when node B broadcasted it. If it so happens that destination node D receives the route-request through H first, instead of C, which is the shorter path, it sends a route-reply packet with mul-reply value TRUE through H. This is possible in case of congestion at node C. It sends a route-reply packet through C also after receiving route-request from it. So, when the node A receives the route-reply from E first, it relays the route-reply through S and G as its mul-reply value is TRUE. Fig. 10(c) depicts this action. When node B receives the extra route-reply from G it stores the alternate route to D in the routing table. Node B does not relay the route-reply further as it already relayed one from C. Node B uses route through C to forward data packets to D as it is shorter than the route through G.

Now a route break due to movement of node C causes the data packets to go in loop as shown in Fig. 10(d). After node B detects the break, it redirects data packets through the alternate path through G. So, node A receives the data packets it itself transmitted, from G. Then node A again forwards them to destination through B, unaware of the route break. Data packets go in loop until

---

**Fig. 9.** Loops formed by fail-safe segments that have single common node with the primary path.

**Fig. 10.** Loops formed by fail-safe segments that have multiple common nodes with the primary path.
their time-to-live fields reach the maximum limit, after which they get dropped. The main reason for this loop formation is that node B accepting a route-reply generated by an upstream node (node A in Fig. 10(d)) on the path and not relaying it to source. Had the route-reply been relayed to source, node A on the reverse path would have dropped the route-reply as its address already present in the node-list. Now, we answer the question, is SMORT loop-free? Answer lies below.

Lemma 1. In SMORT, primary path is loop-free.

Proof. Let us assume that loops exist on the primary path. According to the protocol description, at least one node on the primary path should have accepted a route-reply with its address already in the node-list of the reply. This is because, every node relaying the route-reply appends its node address to the node-list field, or the intermediate node that is replying to the route-request copies its full path to destination into the node-list field. But, nodes do not accept a route-reply that has their address in the node-list field. Hence contradiction. Thus, primary path of SMORT is loop-free. □

Lemma 2. In SMORT, fail-safe paths are loop-free.

Proof. The proof is divided into two parts. In the first part, we prove that loops formed by fail-safe segments that have single common node with the primary path are eliminated. Second part proves that loops formed by fail-safe segments that have multiple consecutive nodes common with the primary path are eliminated.

Let us assume that loops formed by fail-safe segments that have single common node with the primary path are not eliminated. Let us also assume that the first loop node on the loop is \( a_l \) and last loop node is \( a_l \). This means that \( a_l \) forwarded both route-request and the route-reply to \( a_l \). If \( a_l \) forwarded route-reply to \( a_l \), it means that it received a duplicate route-request relayed by \( a_l \). So, \( a_l \) forwarded at least two copies of the route-reply, one through the lasthop from which it received the first route-request and another through the lasthop through which duplicate route-request relayed by \( a_l \) reached \( a_l \). These two lasthop nodes are different because, a node does not broadcast more than one copy of route-request. Among the multiple copies of the route-reply sent by \( a_l \), the route-reply received by \( a_l \) contains the address of \( a_l \) as the reply-gen value. The reason being, when nodes send multiple copies of route-reply they send first copy without altering the reply-gen value through the lasthop through which they received the first route-request. In the rest of the copies the reply-gen value is changed to the node’s address.

The route-reply received at \( a_l \) from \( a_l \) is an extra reply as \( a_l \) already has the primary path. Now two cases are possible. In the first case, the route-reply’s reply-gen value, i.e., \( a_l \) is one of the nodes on the primary path. Then, \( a_l \) accepts this route and does not relay it as it is the extra route-reply. But, the first route-reply received by \( a_l \) that has \( a_l \) its node-list is forwarded to \( a_l \) because, \( a_l \) sent a route-request to \( a_l \). So, node \( a_l \) receives a route-reply with its address already in the node-list of the route-reply. If a loop has to be formed with the primary path, \( a_l \) should accept this reply as this is the primary path for \( a_l \). But, nodes do not accept route-reply packets if the route-reply contains their address in the node-list field. Thus contradiction. Hence, loop in this case is eliminated. In the second case the reply-gen value of the reply is not one of the nodes on the primary path of \( a_l \). Nodes discard the extra route-reply packets whose reply-gen is not one of the nodes on the primary path. Thus, \( a_l \) discards this route-reply. Hence, there is no connection between \( a_l \) and \( a_l \), i.e., between first loop node and last loop node. Hence, loop in this case is also eliminated. In conclusion, loops formed by fail-safe segments that have multiple consecutive nodes common with the primary path are eliminated. Hence, fail-safe paths in SMORT are loop-free. □

Theorem 1. SMORT is loop-free.

Proof. By Lemmas 1 and 2, SMORT is loop-free. □
3.4. Route maintenance phase

Route maintenance phase maintains the routes established during the route reply phase, for the time duration of session. The lifetime of routing entries is used for this purpose. The lifetime of route, represents the time until when the route through next-hop is valid. Nodes on the primary path refresh the lifetime of their routing table entries, each time a data packet for the corresponding destination is forwarded. The lifetime of routes at the nodes on the secondary path is initiated to a sufficiently large value. This value can be decided based on the frequency of path breaks due to mobility and probability of node failures. We call this parameter as SEC_ROUTE_LIFETIME. If a requirement for the secondary route arrives before this time, the secondary route is used for data transmission, and then its lifetime is updated as long as data transmission happens through it. Otherwise, secondary routes are deleted from routing tables once their initial lifetime expires.

The lifetime of route is updated to CURRENT-TIME + ACTIVE-ROUTE TIMEOUT, whenever a data packet is successfully forwarded through next-hop. This means that the route is valid and needed till the next ACTIVE-ROUTE TIMEOUT seconds. CURRENT-TIME is the absolute clock time of the node performing this update. If a route to the destination expires, i.e., if route’s existing-lifetime is less than CURRENT-TIME, the route is invalidated and cannot be used for sending data packets anymore. Later, when a data packet arrives for this destination, the node checks if a valid secondary path to the destination is available in the route-list of the routing entry. If a valid secondary route exists, the primary path is replaced with this path and packets are forwarded through it. If a valid secondary does not exist, a route-error packet is sent to all source nodes through the nodes in the precur-list of the destination’s route entry.

3.4.1. Processing of a route error packet

When a node receives a route-error packet, it invalidates the routes through the neighbor that sent the route-error packet, to all destinations mentioned in the dest-list. If the node does not have any such routes, it simply discards the route-error.

In case routes to any of the destinations are invalidated, the node replaces the invalidated route with a secondary route, if one exists. It removes that destination from the dest-list of the route-error packet. If the dest-list becomes empty, the node drops the route-error packet, as the routes to all the destinations are re-established with the secondary routes. If dest-list still has some destinations left, the node relays the route-error packet through the precursors of the remaining destinations in the dest-list. Finally, if the source nodes of the sessions receive the route-error packet, they initiate a fresh route discovery process to re-establish routes to disconnected destinations, if they too do not have valid secondary paths.

As most of the route disconnections are re-established at the intermediate nodes with secondary paths, the number of route errors communicated in the network drastically decreases. This fact is more evident when we discuss the simulation results (refer Section 5).

State-space diagram in Fig. 11 depicts SMORT’s action. Nodes S and D are source and destination nodes, communicating with each other. Nodes $N_1$ and $N_2$ are intermediate nodes that forward packets to destination.

![State-space diagram of SMORT](image)
4. Simulation model for performance evaluation

SMORT is implemented using the GloMoSim simulation library [42]. GloMoSim is a scalable simulation environment, specifically designed for simulating wireless networks. GloMoSim uses the parallel discrete-event simulation capability of PARSEC compiler [43]. In order to evaluate the performance of SMORT with other protocols, we have also implemented the DMRP protocol in GloMoSim. DMRP protocol computes only node-disjoint multiple paths to destination, at all nodes on the primary path (if exists). As SMORT is a non-disjoint multipath routing protocol, comparing its performance with a disjoint multipath routing protocol brings out the real strength of SMORT. SMORT’s performance is also compared with the unipath routing protocol AODV. The implementation of AODV that came with GloMoSim is taken for producing its simulation results.

Rectangular terrain areas were chosen for the simulation, in order to explicitly make the path lengths between the communicating nodes longer. Longer paths test the protocol’s performance rigorously due to their higher probability of breakage. Table 1 shows the dimensions of the terrain area for networks with various number of nodes. Ten runs of each scenario is simulated. Each run is continued for a period of 300 s of real time. Results of these ten runs were averaged to produce the final results.

4.1. Channel and radio model

Nodes communicate via radio signals with 250 m of propagation range and a raw channel capacity of 2 Mb/s. A free space propagation model [44] with a threshold cutoff is used in the experiments. This model predicts that transmission power is attenuated (signal fades) in proportion to the square of the distance between radios. In radio model, capture is assumed, whereby a radio has the ability to lock onto a sufficiently strong signal in the presence of interfering signals [44]. The parameters of the radio model is presented in Table 2.

4.2. MAC layer protocol

IEEE 802.11 DCF [45] is used as the medium access control (MAC) layer protocol in the simulations. DCF uses virtual carrier sensing to control the access to the shared medium. Nodes communicate RTS, CTS control packets to reserve the channel before transmitting the actual packet. This reservation reduces the impact of the well-known hidden terminal problem [46]. DCF uses carrier sense multiple access with collision avoidance (CSMA/CA) [45] technique to transmit the packets. An acknowledgment is sent by the recipient after it receives the packet. Large sized packets are fragmented at the source node and are sent to the destination in order to reduce the retransmission overhead in the presence of erroneous channel.

4.3. Node movement and traffic pattern

We used the random waypoint [10] mobility model to simulate the node movement in the network. According to this model, a node randomly selects a location with in the terrain area and moves towards it with a speed uniformly chosen between a predefined minimum and maximum values. The node stays in that position for a period (called pause time) and then again selects another random location to move. The high neighbor percentage variation [47] of random waypoint mobility model, in the initial stages of simulation, is one of its main limitations. This problem is avoided in our simulations, by discarding the first 1000 s of the simulation time. This ensures a random initial configuration in

<table>
<thead>
<tr>
<th>Number of nodes</th>
<th>Terrain area (m²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>(2000 × 1000)</td>
</tr>
<tr>
<td>200</td>
<td>(3000 × 1000)</td>
</tr>
<tr>
<td>400</td>
<td>(3500 × 2000)</td>
</tr>
<tr>
<td>800</td>
<td>(6000 × 3500)</td>
</tr>
<tr>
<td>1000</td>
<td>(7000 × 4000)</td>
</tr>
<tr>
<td>2000</td>
<td>(10,000 × 6000)</td>
</tr>
</tbody>
</table>

Table 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fading model</td>
<td>Free-space propagation model</td>
</tr>
<tr>
<td>Path-loss coefficient</td>
<td>2</td>
</tr>
<tr>
<td>Radio frequency (in Hz)</td>
<td>2.4e9</td>
</tr>
<tr>
<td>Radio transmission power (in dBm)</td>
<td>15.0</td>
</tr>
<tr>
<td>Receiver sensitivity of the radio (in dBm)</td>
<td>−91.0</td>
</tr>
<tr>
<td>Threshold of signal capture (in dBm)</td>
<td>−81.0</td>
</tr>
</tbody>
</table>
each simulation trial. The is one of the solutions suggested in [47], for the initialization problem of random waypoint mobility model. Thus, the total simulation time is initialized to 1300 s, and only the last 300 s involve actual data communication between nodes.

In order to establish the communication between nodes, we developed a simple traffic generator. It selects a random source and destination pair in the network and starts a constant bit-rate (CBR) session between them. Each CBR session generates four packets/sec and the size of each data packet is 512 bytes. In each run of the scenario, a different set of node pairs are selected by varying the seed value used by the random number generator in the simulator. The lifetime parameters of the protocols, ACTIVE_ROUTE_TIME and the SEC_ROUTE_LIFETIME, for the simulation, have been set to 10 and 30 s, respectively. Also, buffer size of each node is set to 100 packets, in the simulation.

5. Simulation results and analysis

5.1. Performance metrics

Following metrics were computed to evaluate scalability and performance of SMORT:

- **Throughput**: Throughput is calculated as the number of data bytes delivered to all destinations during the simulation.
- **Routing overhead**: Routing overhead is the total number of control packets transmitted by nodes while establishing and maintaining routes. Each hop-wise transmission of the control packet is counted.
- **Average packet transmission delay**: Average packet transmission delay is the average time taken by data packets to travel from source node to destination. This per-packet delay includes not only the absolute delay experienced by the packet in reaching the destination, but also the delay in resuming the session, after the route breaks have occurred.

5.2. Simulation experiments

Scalability of SMORT, DMRP, and AODV is studied by increasing number of nodes in the network from 100 and 2000. Also, performance (in terms of throughput and routing overhead) of these three protocols is evaluated with varying mobility and offered load.

5.3. Scalability study

5.3.1. Experiment

Throughput, routing overhead and average packet transmission delay are computed by increasing the number of nodes in the network from 100 to 2000 nodes. Mobility of nodes is kept constant at a minimum speed of 0 m/s, a maximum speed of 10 m/s, and a pause time of 30 s. Twenty CBR sessions are initiated between randomly selected source-destination pairs. Sessions at various source nodes start in Poisson fashion along the time line.

5.3.2. Observations

Fig. 12 shows the throughput comparison of SMORT, DMRP, and AODV. Packet delivery capacity of all these routing techniques decreases as the number of nodes in the network increases. This is due to the increasing number of route breaks as the size of the network increases. However, SMORT outperforms AODV and DMRP in packet delivery capability (for all the sizes of network) because, most of the route breaks are corrected with secondary paths at intermediate nodes. This avoids packet drops at all the upstream nodes of the intermediate node that detected the route break. Although, DMRP also uses multiple paths at intermediate nodes to redirect unfinished sessions, the number of intermediate nodes that possess secondary paths are very few in case of DMRP, when compared to SMORT. The reason for low availability of disjoint multiple paths has already been explained in Section 3. So, the packet drops, at intermediate nodes with no secondary paths to destination, are the cause of lower throughput of DMRP. On the other hand, in AODV, all upstream nodes of the broken link drop packets queued to the disconnected destinations as they do not have secondary paths. Some of the packet drops are also due to the congestion caused by high routing overhead in AODV (as we will see later).

One interesting point observed is that throughput of DMRP becomes worse than unipath routing protocol AODV, at higher network sizes. This is because, longer secondary paths in DMRP fail quickly due to their higher probability of route breakage. So, when sessions are resumed over already broken paths (source does not know whether a secondary path is broken or not, as no
notification is sent to it when secondary paths break), all the data packets that have reached the intermediate nodes are dropped, until a route break notification is sent to the source node. As AODV initiates a fresh route discovery instead of using a secondary path to resume the session, these additional packet drops are saved. SMORT has better throughput even in larger networks as fail-safe segments are much shorter when compared to disjoint multiple paths. Further, fail-safe multiple paths have high availability. Packet drops occur in case of SMORT also, due to either the non-availability of secondary paths at some intermediate nodes of the primary path, or the secondary paths were also broken by the time the primary path fails.

Routing overhead is computed as the total number of control packets communicated in the network, for establishing and maintaining routes. These control packets comprise route-request, route-reply, and route-error packets. Fig. 13 shows the variation of routing overhead of all the three routing techniques. The value increases for all these protocols with network size because, the number of nodes communicating these packets and number of route computations increase as the network size increases. Number of route computations increase with network size because of increase in number of route breaks.

Despite the lower number of route-reply packets per route computation in AODV, it has higher routing overhead than SMORT and DMRP, at all network sizes. This is because, AODV involves additional route computations and route-error packet transmissions for recovering route breaks. Route error packets initiated by the upstream nodes of broken links, travel all the way to the source node every time (due to the non-existence of secondary paths at intermediate nodes as in SMORT or DMRP). More number of nodes participate in forwarding route-error packets as the network size increases due to the increase in path length between nodes. Hence, AODV has enormously higher route-error overhead when compared to either SMORT or DMRP, as shown by the plot in Fig. 14. SMORT and DMRP have lower route-error overhead because, intermediate nodes that have secondary paths to destination drop the route-error packet, as broken sessions can be resumed through these secondary paths. DMRP has slightly lower route-error overhead that of SMORT, as route-error packets are sent over multiple paths to source in SMORT. Whereas, in DMRP route-error packets traverse along a single path to source.

In AODV, source nodes initiate a fresh route discovery each time they receive route-error packet, in order to re-establish its connection to the destination. Every route discovery attempt causes network wide spread of route-request packets, and each route-request reaching the destination generates corresponding route-reply. Whereas, in DMRP and SMORT only a limited number of route breaks cause fresh route discoveries, as most of the route

![Fig. 12. Variation of throughput with network size.](image-url)
breaks are corrected using secondary paths at intermediate nodes. Hence both route-request overhead and route-reply overhead of SMORT and DMRP are lower than that of AODV, as shown in Figs. 15 and 16, respectively. Reply overhead of SMORT and DMRP is almost same at all the network sizes. Intuitively, route-reply overhead of DMRP should have been less than that of SMORT, as the availability of disjoint multiple paths is lower than fail-safe paths, and so is the number of route-reply packets for establishing each individual path. But, higher number of route computations in the case of DMRP will in turn result in higher route replies, canceling the effect of its fewer route replies per route computation. At higher network sizes, route-reply overhead of DMRP should have grown over that of SMORT, due to the drastic decrease in availability of disjoint multiple paths as shown in Fig. 2.
But, higher packet drops at these network sizes reduce the overall routing activity in the network, and hence route-reply overhead too.

DMRP has higher routing overhead than SMORT due to the lower availability of disjoint paths when compared to fail-safe paths. Also, it has lower routing overhead than AODV due to the usage of secondary paths for correcting some of the route breaks. Actually, the difference of control packet overhead between DMRP and SMORT should be increasing with network size, as the availability of disjoint multiple paths decreases more rapidly with the increase in network size. But, it is not so due to the same reason as in the case of route-reply overhead, i.e., reduced routing activity caused by large packet drops over broken secondary paths. In order to show the effect of lower availability of disjoint multiple paths, we computed the

![Fig. 15. Variation of route-request overhead with network size.](image1)

![Fig. 16. Variation of route-reply overhead with network size.](image2)
normalized routing overhead. It is the average number of all packets (data, route-request, route-reply, and route-error) transmitted for communicating a single data packet to destination. Fig. 17 shows this comparison for the three protocols. This statistic shows, how the drastic fall in the availability of disjoint multiple paths with network size in DMRP, and non-existence of secondary paths in the case of AODV, increases the overhead incurred to deliver every single data packet to destination.

High availability of secondary paths at intermediate nodes in the case of SMORT reduces its routing overhead by 50% (on an average), when compared to AODV. This reduction in overhead allows SMORT to scale to double the number of nodes that AODV supports. For example, in Fig. 13, the average number of routing packet transmissions in the 1000 node network is approximately 1,400,000. But, SMORT reaches to this amount of overhead only at 2000 nodes. Thus, comparatively SMORT supports double the scalability of AODV.

Average route calculation time of SMORT and AODV is shown in Fig. 18. It is the average time taken by source nodes, to receive the first route to the destination after they have initiated the route discovery process. The value increases for these three protocols with number of nodes in the network because, average length of paths between the communicating nodes increases. AODV’s route calculation time is higher than that of SMORT, as the co-channel interference created by AODV’s large routing overhead delays routing packets that are traversing towards source or destination. DMRP has slightly lower routing overhead at higher network sizes compared to SMORT because of reduced routing activity in DMRP, at these network sizes.

Fig. 19 shows the comparison of average packet transmission delay experienced by data packets for AODV, DMRP, and SMORT. This metric reflects the delay involved in resuming the sessions after route breaks have occurred. The delay is higher for DMRP when compared to SMORT and AODV because, DMRP takes longer time to resume the session when compared to AODV or SMORT, at higher network sizes. In large networks, the broken secondary paths of DMRP delay the initiation of fresh route discovery. Also, packets go on longer paths in case of DMRP. SMORT has the lowest delay value at all network sizes, as it finishes the session with lowest number of route computations when compared to other two protocols.

5.3.3. Inference
SMORT increases throughput by a maximum of 60%, when compared to AODV. An average of 50% reduction in routing overhead of SMORT, enabled it to scale to double the number of nodes that AODV supports. SMORT achieved an average of 45% improvement in throughput and 40% reduction in routing overhead when compared to DMRP.
5.4. Effect of varying mobility

5.4.1. Experiment

Mobility is kept constant in the previous experiment in order to limit the number of variables. Here, we study the behavior of SMORT, DMRP, and AODV by varying mobility. Mobility is increased by changing the maximum speed of the node from 0 to 50 m/s. The minimum speed and pause time are kept constant at 0 m/s and 30 s. A 300-node network with terrain area (3000 × 1500) is used for this experiment. Throughput and routing overhead of these three protocols are measured and compared.

5.4.2. Observations

The comparative results of throughput and control overhead are shown in Figs. 20 and 21,
respectively. An interesting fact observed is that, even when the nodes are stationary (at a maximum speed of 0 m/s), SMORT and DMRP show a slight improvement over AODV, both in terms of throughput and routing overhead. Actually when the nodes are stationary, both protocols should show the same throughput performance, as there are no route breaks due to mobility. The reason for this behavior is attributed to the congestion experienced by nodes, when all the 20 sessions are active. Congestion increases packet collisions and nodes attempt to retransmit the collided packets. This again increases the number of collisions. Finally, nodes drop the packets after trying retransmission for maximum number of times, and falsely conclude that the recipient node moved away, even

![Fig. 20. Variation of throughput with node speed.](image)

![Fig. 21. Variation of overall routing overhead with node speed.](image)
if it has not. Then, the node sends a route-error packet to source. The source node initiates a fresh route discovery to find a new path to destination, even though the original path is intact. Although, the same situation occurs in SMORT and DMRP, the node that falsely assumed the route break uses a secondary path to forward future packets, avoiding extra route discovery attempts and error packet transmissions.

As mobility increases, the protocols behave as expected. Routing overhead and number of packet drops of these protocols increase with mobility, because of large number of route breaks at higher speeds. In SMORT, some of the secondary paths also break at higher speeds and make source nodes to repair route disconnections through a fresh route discovery. But, SMORT and DMRP achieves improvement over AODV due to the usage of secondary paths to repair most of the route breaks. DMRP has lower performance than SMORT because of the same reasons explained before. Drastic increase of routing overhead in AODV at higher speeds show the need for methods to repair the routes breaks with minimal routing overhead.

5.4.3. Inference

Although, SMORT’s routing overhead and packet drops slightly increase with mobility, it outperforms AODV and DMRP due to the usage of highly available fail-safe multiple paths to repair route breaks.

5.5. Effect of increasing offered load

5.5.1. Experiment

The number of active sessions in the network affect protocol’s performance because it decides the amount of overhead in the network. More the number of sessions, more the number of route discovery attempts and hence more is the routing packet transmissions. We evaluated the performance of SMORT, DMRP, and AODV by increasing the offered load from 5 to 30 sessions. A 300 node network with terrain area (3000 × 1500) is used for this experiment. Mobility of the nodes is kept constant at a minimum speed of 0 m/s, a maximum speed of 10 m/s and a pause time of 30 s.

5.5.2. Observations

Figs. 22 and 23 show throughput and routing overhead variation with offered load, respectively. It is observed that AODV shows slightly better performance both in terms of throughput and routing overhead than DMRP at low offered loads (5 and 10 sessions). This is due to the lower congestion experienced by the nodes communicating data in AODV, than in DMRP. In DMRP, nodes on the primary path receive extra route-reply packets so as to enable computation of multiple paths. This slightly increases congestion at those nodes and causes some packet drops. At higher loads, number of false route breaks increases due to congestion created by more number of active sessions. False

![Fig. 22. Variation of throughput with network load.](image-url)
route breaks occur as nodes falsely assume that a route break has occurred, when there are lots of packet drops due to collisions created by congestion, though the route between source and destination is intact. So, AODV’s overhead increases as it initiates a fresh route discovery for every route break. SMORT and DMRP outperforms AODV by using secondary paths to repair route breaks.

5.5.3. Inference

Number of route breaks in the network increase with the offered load. At higher loads DMRP and SMORT shows better performance than AODV because, they minimize the routing overhead incurred in repairing route breaks using secondary paths.

6. Conclusions and future work

In this work, our objective was to propose a multipath extension (SMORT) to an unipath on-demand routing protocol, in order to improve its scalability. Intuitively, finding multiple paths in a single route discovery reduces the routing overhead incurred in maintaining the connection between source and destination nodes. Secondary paths can be used to transmit data packets, in case the primary path fails due to node mobility or battery failure, which avoids extra overhead generated by a fresh route discovery. These multiple paths are more advantageous in larger networks, where the number of route breaks are high.

We found through simulations that total number of node-disjoint multiple paths at all nodes on the primary path are scarce, even in large networks. We modified AODV [11] protocol to compute a new class of multiple paths called fail-safe multiple paths, which are more abundant and hence provides better fault-tolerance to route breaks. We proved the loop freedom of the protocol. Performance evaluation of our protocol using simulations showed that an average of 50% reduction in routing overhead, doubled the scalability of SMORT when compared with AODV. Also, average throughput of SMORT is improved by a maximum of 60% when compared to AODV. SMORT’s performance is also compared with the disjoint multipath routing protocol, which uses only node-disjoint secondary paths for correcting route breaks. SMORT over-performs DMRP also due to usage of fail-safe multiple paths, which are more available than node-disjoint multiple paths. SMORT showed, on an average, 45% improvement in throughput and 40% reduction in routing overhead, when compared to DMRP. SMORT showed better performance than DMRP and AODV, even in highly dynamic networks (nodes moving with maximum speed of 50 m/s) and heavily loaded networks (30 concurrent sessions).

As part of the future work, we want to adopt information theoretic approach to find relatively
stable nodes in the network, expecting to show that overall routing activity involved in computing multiple paths is minimized.

Acknowledgements

The authors would like to thank Satya Sai Prakash, Chakravarthi, and Vineel for their valuable suggestions and comments.

References


L. Reddeppa Reddy received his B.Tech. degree in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad, India in 2002. He received his M.S. (by Research) degree in Computer Science and Engineering from Indian Institute of Technology Madras, India in 2005. He is currently working as a Software Engineer in Samsung ISO, India. His research interests are multipath routing and quality-of-service in mobile ad hoc networks.

S.V. Raghavan is a Professor at Indian Institute of Technology Madras, India. He received his M.E. degree in Computer Science and Automation from Indian Institute of Science, India in 1976. He served Indian industry in various capacities from 1976 to 1981. He joined in the Computer Science and Engineering Department of Indian Institute of Technology Madras and received his Ph.D. degree from the same institute in 1985.

He has made significant contributions to the creation of the Education and Research NETwork (ERNET) in India. He was one of the founders of ERNET and the Principal Investigator of the Programme at IIT Madras. His areas of research, where he has close to 100 publications in reputed international journals and conferences, include Performance Evaluation, Information Security, High Performance Computing and Networks, Protocol Software Engineering, Multimedia and Network Design.