INFORMATICA, 2006, Vol. 17, No. 4, 481–502

Optimized on Demand Routing Protocol of Mobile Ad Hoc Network

Chinnappan JAYAKUMAR
Research Associate, Department of Computer Science and Engineering, Anna University Chennai, India-600025
e-mail: c_jayakumar2000@yahoo.com

Chenniappan CHELLAPPAN
Ramanujan Computing Center, Anna University Chennai, India-600025
e-mail: drcc@annauniv.edu

Received: February 2005

Abstract. In this paper optimization of DSR is achieved using New Link Cache structure and Source Transparent Route Maintenance Method. The new link cache effectively utilizes the memory by caching the routes in adjacent list type of data structures. It selects the shortest hop and least congested path, which in turn reduce the control packets, route request packets, route reply packets and increase the data packets forwarded by the nodes. To solve the DSR route maintenance problem during high mobility, source transparent route maintenance method is introduced in this paper. This method has two schemes namely cache validation and local route repair. These schemes reduce the packet loss, end-to-end delay and increase the throughput.

Key words: link cache, local route repair, sequence number and cache validation.

1. New Link Cache

1.1. *Introduction*

As networks are rapidly developed, mobile wireless networks have widely deployed in the computing industry. There are currently two types of mobile wireless networks, i.e., one is infrastructured network and the other is infrastructureless mobile network (Johnson *et al.*, 1998; Johnson and Maltz, 1996; Johnson *et al.*, 2003; Lou and Fang, 2002; Marina and Das, 2001; Maltz *et al.*, 1999; Perkins and Royer, 1999; Royer and Toh, 1999). The former is known as mobile networks fixed with base stations. The latter is known as an Mobile Ad hoc network (MANET). In an Ad hoc network, there are no stationary infrastructured networks such as base stations (Royer and Toh, 1999). Caching of route information is very important in any on-demand routing protocol for MANET (Hu and Johnson, 2000). Several routing protocols have the routes cached in local memory discovered by the Route Request. By caching and making effective use of this collected

network state information, the amortized cost of Route Discoveries can be reduced and the overall performance of the network can be significantly improved (Johnson and Maltz, 1996). Such caching then introduces the problem of proper strategies for managing the structure and contents of this cache as nodes in the network move in and out of wireless transmission range of one another, possibly invalidating some cached routing information. Important types of cache structure are path cache and link cache (Johnson *et al.*, 2003; Marina and Das, 2001). The path cache strategy has certain issues like increase in buffer capacity with scalability and deletion of a whole path when a link failure happens in that path (Hu and Johnson, 2000). To overcome the issue of high memory utilization, an alternative type of organization called, link cache was developed (Lou and Fang, 2002).

With the increase in number of nodes, the performance of link cache degrades and is also in need of more buffer capacity. In this research paper, a modification of link cache data structures and new path selection criterion are incorporated to achieve better results. The link cache uses only the least hop count to find the best route.

The rest of this paper is organized as follows: Section 1.2 describes the new link cache design. Section 1.3 elaborates the implementation details of new link cache. Section 1.4 gives the simulation results for Dynamic Source Routing (DSR) with new link cache compared with the basic DSR algorithm. Source transparent route maintenance design described in Section 2. Mobility impact on DSR explained in Section 2.1. Design details of cache validation and local route repair available in Section 2.2. Section 2.3 discuss about the implementation and simulation results. Section 3 has the conclusion details of new link cache, cache validation and local route repair.

1.2. *New Link Cache Design*

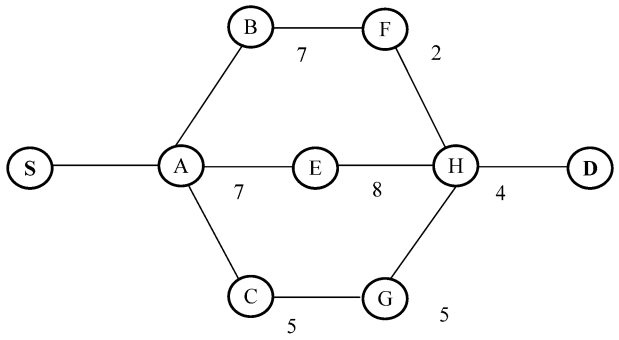
1.2.1. *Load Aware Routing*

New link cache uses the load (Lee and Gerla, 2001) as main criteria for selecting best route. Nodes attach their load information to a route request, when they receive it and broadcast the route request packet. When the destination receives a route request packet it calculates the load factor of the route based on the loads attached by each node in the route request packet by load factor-calculation algorithm described later in Section 1.2.2. The source node on receiving multiple route reply packets, choose a route with the least average load. In addition to this, as the route request propagates through the network each node stores the load information of the nodes along the path that the route request has traversed inside their link cache so that the next time if a route is needed a less loaded route may be selected. Intermediate nodes can also send route reply if they have a route to the destination and it calculates the load factor using the load information it has already stored in the link.

1.2.2. *Load Distribution Algorithm*

For selecting least loaded route, there are three algorithms are available (Lee and Gerla, 2001). Fig. 1 is used as an example network to describe each algorithm.

Algorithm 1 [Based on absolute load]



Route X: (S-A-B-F-H-D); Route Y: (S-A-E-H-D); Route Z: (S-A-C-G-H-D)

Fig. 1. A sample network.

Step 1: Add the routing load of each intermediate node for each route.
Step 2: Selects the route with the least sum of load.

$$\text{route}_i = \sum_j l_j, \quad l_j \text{ is the load of the intermediate node } j \text{ along route } i,$$

$$\text{route} = \min_i \{\text{route}_i\}.$$

Step 3: If there is a tie, the destination selects the route with the shortest hop distance.
Step 4: When there are still multiple routes that have the least load and hop distance, the path that is taken by the packet, which arrived at the destination the earliest is chosen.

In the example network (Fig. 1), route X has the sum of 20 (i.e., $7 + 7 + 2 + 4 = 20$), route Y has the sum of 19 (i.e., $7 + 8 + 4 = 19$), and route Z has the sum of 21 (i.e., $7 + 5 + 5 + 4 = 21$). Therefore, least loaded route Y is selected and used as the route.

Algorithm 2 [Based on average load]
Step 1: Add the routing load of each intermediate node for each route.
Step 2: Find the average load by using the formula average load = sum/number of nodes.
Step 3: Selects the route with the least average load.
Step 4: If there is a tie, the destination selects the route with the shortest hop distance.
Step 5: When there are still multiple routes that have the least load and hop distance, the path that is taken by the packet, which arrived at the destination the earliest is chosen.

Considering the example in Fig. 1 again, route X has the average value of 5 (i.e., $20/4 = 5$), route Y has the value of 6.67 (i.e., $19/3 = 6.67$), and route Z has the value of 5.25 (i.e., $21/4 = 5.25$). Route X is thus selected.

Algorithm 3 [Less congestion based]
Step 1: Fix a threshold load value.
Step 2: Find the number of nodes n_i in every route i, exceeding this threshold value (i.e., congested nodes).

Step 3: Select the route with the smallest value n_s of n_i number (least congested path).

$$n_s = \min\{n_i\}.$$

Step 4: If there is a tie, the destination selects the route with the shortest hop distance.

Step 5: When there are still multiple routes that have equal number of congested nodes and hop distance, the path that is taken by the packet, which arrived at the destination the earliest is chosen.

For example, if threshold is five, route X has two intermediate nodes (i.e., nodes A and B) that have the number of queued packets over the threshold, route Y has two (i.e., nodes A and E), and route Z has one (i.e., node A). Hence, route Z is selected using this algorithm. This Algorithm applies the same tie-breaking rule as in Algorithm 1.

The average load algorithm is chosen, the reason being that the other two do not consider the whole network. Algorithm 1 chooses the least loaded path, which may be an exceptional case and cannot be found always. Algorithm 3 uses only the number of nodes in the path, and hence does not consider the load factor.

The destination or an intermediate node, which is about to reply for request packet, chooses to calculate the average load of all the nodes in the path. Though the reply is immediate, if the destination happens to receive a better route a few seconds later, it intimates the source to use the latest better route.

1.2.3. *The Modified DSR Algorithm with New Link Cache*

- When a node needs a path to a destination, it checks its path cache. If the path is found, the packet is sent using that path. If not, the path cache requests the link cache to return a path.
- The link cache checks if the destination node is available in its list of nodes. If the destination is not available, it returns a false, to indicate that a route request has to be initiated. Else, it runs a shortest path algorithm (Dijikstra's) from the source to the destination. If it finds a route it returns the route and a true. Else, it returns a false.
- Whenever the link cache returns a false, the DSR initiates a route request for the destination.

The addition of new nodes and links to the link cache is done whenever a node updates its path cache with the information viz. Route Reply packet and overheard data/ control packet. The RERR packets, which are responsible for the route maintenance in the network, help the Link Cache to delete unwanted links. There is no timeout associated with the links and hence RERR is the only way of removing obsolete links. The deletion procedure also ensures that, whenever a node stored in the Link Cache is completely without any links, it is removed from the Link Cache.

1.2.4. *Structure of Link Cache*

The MANET graph shown in Fig. 2 represents the nodes and the links between them (the links is assumed to be bi-directional in new link cache). Now this topology would

be represented in the new link cache as in Fig. 3. Adjacent list is used to represent entire graph in each node.

1.2.5. *Functions of the New Link Cache*

The insertion and deletion of link in new link cache is explained as follows: A new link is inserted into the link cache whenever a new route is found by a node from the route request packet or by promiscuously overhearing packets by breaking up the route into a set of links. Then the cache is checked to verify if the node is an entirely new node or a node whose links are known already. If not found already the node is inserted and the corresponding link is also inserted both ways that is to insert A-B link we check if both A and B exist and then insert A-B and B-A to maintain the assumption that all links are bi-directional. When a node encounters a route error packet, it deletes the corresponding link in its link cache. If all the links to a given node are exhausted then the node is removed from the link cache so as to restrict the size of the cache.

The most important function of the link cache is to find the least loaded path to the destination if such path exists. Dijikstra's Shortest Path Algorithm is used to determine whether or not a route exists and if it does it returns the route else returns False. Whenever a node forwards a route reply (that is it is not the destination) it attaches its load information to the packet. Whenever a route request packet arrives at a node, it updates the load

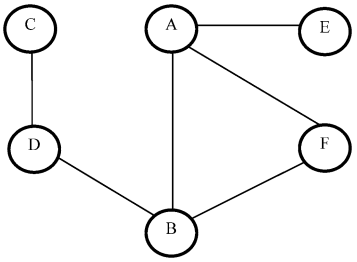
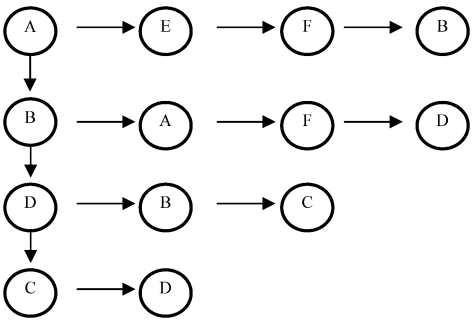


Fig. 2. A Graph representation of a MANET.



A ->{E, F, B), B ->{A, F, D), C ->{D), D->{B, C}, E ->{A}, F ->{A, B}

Fig. 3. A Snapshot representation of new link cache.

information read from the packet in its link cache. Finally, the destination calculates the load factor of all the routes it has received and attaches the information in the route reply packet.

1.3. *Implementation*

The implementation carried out on GlomoSim (Glomosim Manual). This simulator has a Path Cache. Path Cache is used as a buffer and implemented new Link Cache as the underlying cache structure and also stored the load information within the new Link Cache.

The Configuration parameters that need to be set for running the simulator are:

- **Number of nodes:** We used different numbers of nodes (10, 15, 20, 25) for simulations.
- **Node placement strategy:** Uniform node placement strategy is adopted.
- **Mobility:** Random-Waypoint mobility was for simulations. For random waypoint, nodes randomly select a destination from the physical terrain and uniformly chosen between Mobility-Min-Speed and Mobility-Wp-Max-Speed (m/s). After it reaches its destination, the node stays there for Mobility-Wp-Pause time period.
- **Statistics:** The statistics to be collected are the network layer statistics. So the Network-layer-statistics is set to Yes.
- **Application Configuration:** The configuration file (app.conf) identifies the application that will be run on each node at the specified time. Random sets of applications were made to run on each of the nodes at various points of time. Also the load was increased from a minimum with a few nodes participating to a large number of nodes participating.

1.4. *Simulation Results*

New link cache is tested under different conditions and the results are presented below. The performance factors considered are (Chin *et al.*, 2002):

- number of Control Packets sent,
- number of Route Requests sent,
- number of Route Replies sent,
- number of Packets Routed for an adjacent Node.

1.4.1. *Control Packet Overhead Minimization*

Control packets are a matter of serious concern as it increases the bandwidth overhead and consumes high battery power. New Link Cache algorithm has reduced the number of control packets in the network as can be seen from the graphs in this section. This algorithm performs better with more number of nodes.

Difference in performance is too feeble to notice for less number of nodes (less than 10). The average numbers of control packets (refer Fig. 4) that have been relayed by proposed DSR (PDSR) for 15 nodes are reduced to 37.4% when compared to the Basic DSR (BDSR). The minimum number of control packets relayed by node 15 for PDSR

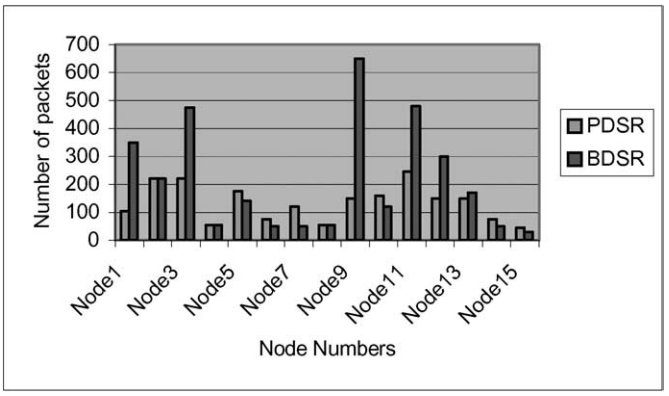


Fig. 4. Control packet overhead minimization comparisons with 15 nodes.

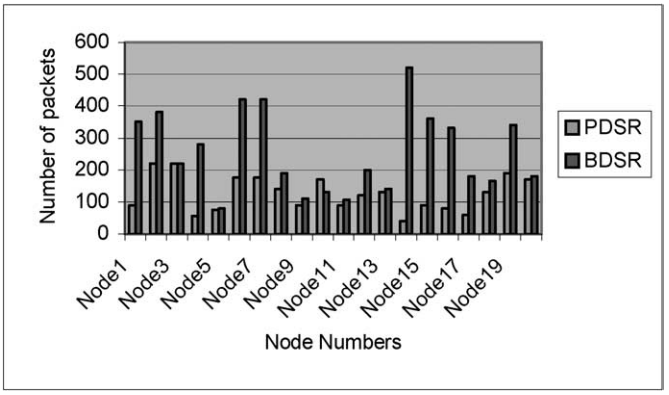


Fig. 5. Control packet overhead minimization comparisons with 20 nodes.

and BDSR are 45 packets and 30 packets respectively. The worst case is 245 packets relayed by node 12 for PDSR and 650 packets by node 9 for BDSR.

The average numbers of control packets (refer Fig. 5) that have been relayed by PDSR for 20 nodes are reduced to 50.78% when compared to the BDSR. The minimum number of control packets relayed by node 14 for PDSR is 40 packets and by node 5 are 80 packets in BDSR. The worst case is 220 packets relayed by node 2 and node 3 in PDSR and 520 packets by node14 in BDSR. The average numbers of control packets (refer Fig. 6) that have been relayed by PDSR for 25 nodes are reduced to 54% when compared to the BDSR.

During route discovery, instead of sending the control packets over the network to find out a route, the propose DSR is find the route from the link cache. So there is no need to send out route discovery packets every time. This way the control packets are reduced.

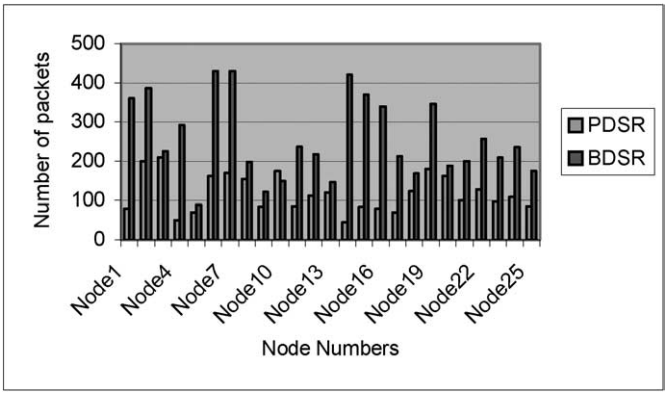


Fig. 6. Control packet overhead minimization comparisons with 25 nodes.

Table 1

Comparison of Control packet minimization by Proposed DSR and Basic DSR

Protocol Name	Average number of control packets with			Minimum number of control packets relayed (worst case) with			Maximum number of control packets relayed (best case) with		
	Node size 15	Node size 20	Node size 25	Node size 15	Node size 20	Node size 25	Node size 15	Node size 20	Node size 25
Basic DSR (BDSR)	213	255	256.32	30	80	90	650	520	430
Proposed DSR (PDSR)	133.3	125.5	117.76	45	40	70	245	220	210
% of reduction of control packets	37.4	50.78	54	33.33	50	22.22	62.03	57.69	51.16

1.4.2. *Route Requests Transmitted*

As the number of nodes increase to 15, the average numbers of route request packets that have been relayed by PDSR are reduced to 7.66% when compared to the BDSR (Fig. 7).

The average numbers of route request packets that have been relayed by PDSR for 20 nodes are reduced to 67.89% when compared to the BDSR (Fig. 8). It is noticed that the reduction of route request packets in the PDSR when compare with the BDSR. The reason being during route discovery, instead of sending the control packets over the network to find out a route, our algorithm can find the route from the link cache. So there is no need to send out route discovery packets every time. This way the route request packets are reduced.

The average numbers of route request packets that have been relayed by PDSR for 25 nodes are reduced to 72.07% when compared to the BDSR (Fig. 8).

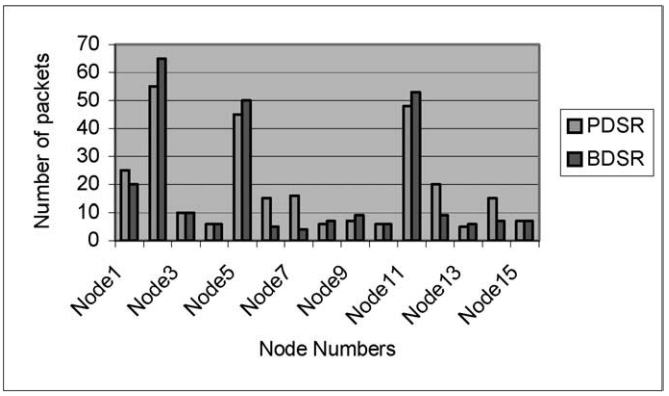


Fig. 7. Comparison of Route Request Packets with 15 nodes.

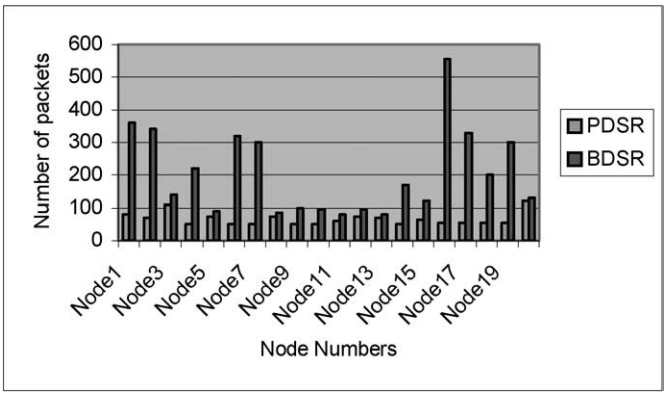


Fig. 8. Comparison of Route Request Packets with 20 nodes.

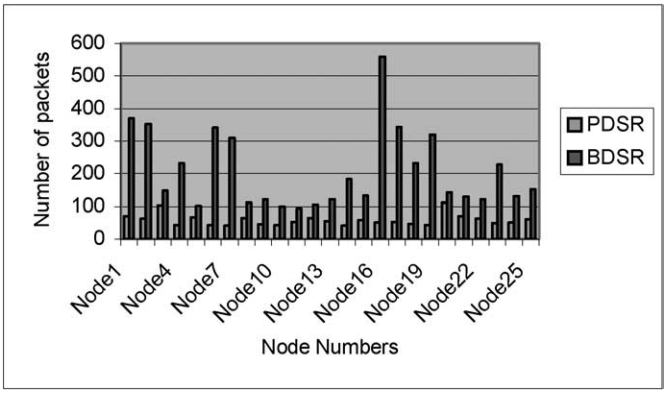


Fig. 9. Comparison of Route Request Packets with 25 nodes.

Table 2
Comparison of Route Requests minimization by Proposed DSR and Basic DSR

Protocol Name	Average number of route requests packets with			Minimum number of route requests packets relayed (worst case) with			Maximum number of route requests packets relayed (best case) with		
	Node size 15	Node size 20	Node size 25	Node size 15	Node size 20	Node size 25	Node size 15	Node size 20	Node size 25
Basic DSR (BDSR)	17.6	205.5	206.96	4	80	93	65	555	560
Proposed DSR (PDSR)	19.06	66	57.8	6	55	41	55	120	111
% of reduction of route requests	7.66	67.89	72.07	33.33	31.25	55.91	15.38	78.37	80.17

1.4.3. *Route Replies Transmitted*

Route Reply packets is one of the main source of information in DSR. In the PDSR, the Reply Packets are low, because of the drop in the number of Request packets. At the same time in some nodes Reply Packets are higher than their BDSR counterparts, because they are the intermediate nodes in the route that discovers a route from its link cache and replies to the source rather than propagating the Request.

The average numbers of route reply packets (for 15 nodes) that have been relayed by PDSR are increasing to 2% when compared to the BDSR (Fig. 10). The average numbers of route reply packets (for 20 nodes) that have been relayed by PDSR are reducing to 13.61% when compared to the BDSR (Fig. 11). The average numbers of route reply packets (for 25 nodes) that have been relayed by PDSR are reducing to 23.97% when compared to the BDSR (Fig. 12).

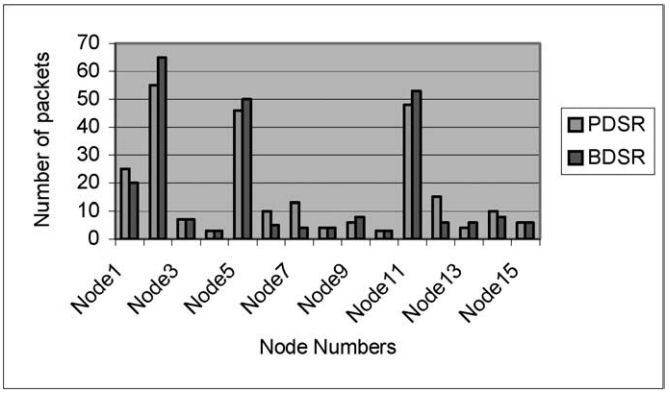


Fig. 10. Comparison of Route Reply Packets with 15 nodes.

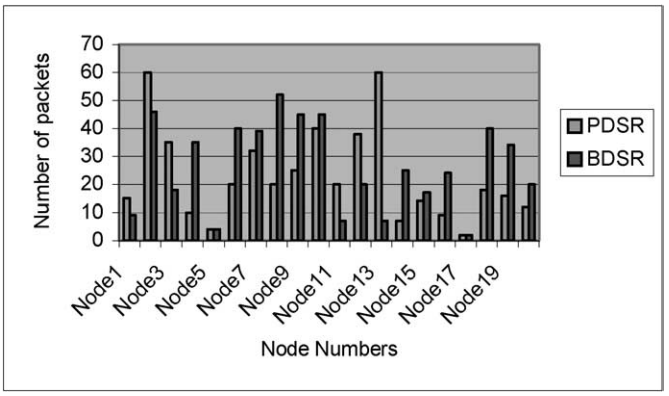


Fig. 11. Comparison of Route Reply Packets with 20 nodes.

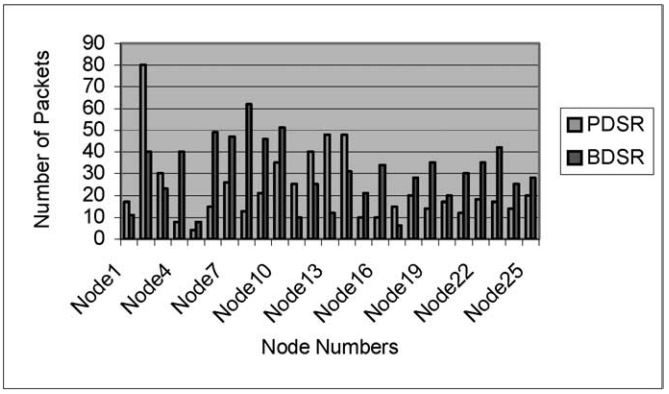


Fig. 12. Comparison of Route Reply Packets with 25 nodes.

Table 3
Comparison of Route Replies minimization by Proposed DSR and Basic DSR

Protocol Name	Average number of route reply packets with			Minimum number of route reply packets relayed (worst case) with			Maximum number of route reply packets relayed (best case) with		
	Node size 15	Node size 20	Node size 25	Node size 15	Node size 20	Node size 25	Node size 15	Node size 20	Node size 25
Basic DSR (BDSR)	16.53	26.45	30.36	3	2	6	65	52	62
Proposed DSR (PDSR)	17	22.85	23.08	3	2	4	55	60	80
% of reduction of route reply packets	2.7	13.61	23.97	0	0	33.33	15.38	13.33	22.5

1.4.4. *Number of Packets Routed by a Particular Node for Another Node*

More intermediate nodes are requested to route packets and hence participate in the routing, even if they offer a greater hop count path. The performance of the protocol is measured by the numbers of data packets that have been delivered through out network in terms of average number of packets. As the number of nodes increase to 15, the average numbers of forwarded packets that have been relayed by PDSR are increasing to 24.31% when compared to the BDSR (Fig. 13). The average numbers of forwarded packets (for 20 nodes) that have been relayed by PDSR are increasing to 15.31% when compared to the BDSR (Fig. 14). The average numbers of forwarded packets (for 25 nodes) that have been relayed by PDSR are increasing to 14.79% when compared to the BDSR (Fig. 15). Forwarded packets increase/decrease depending on the load attached to the node. If a node is loaded too much then it will handle lesser number of packets and vice versa.

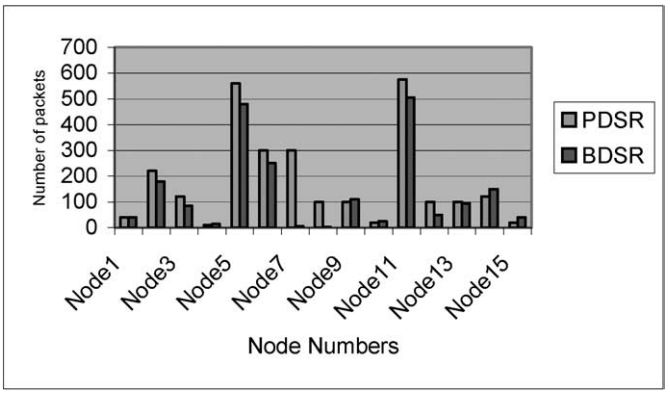


Fig. 13. Comparison of Packets Forwarded with 15 nodes.

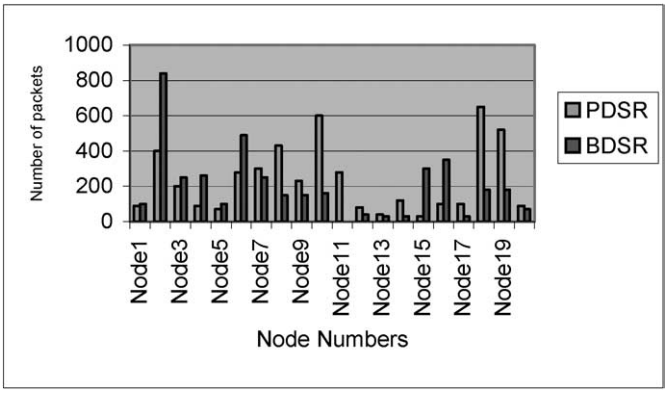


Fig. 14. Comparison of Packets Forwarded with 20 nodes.

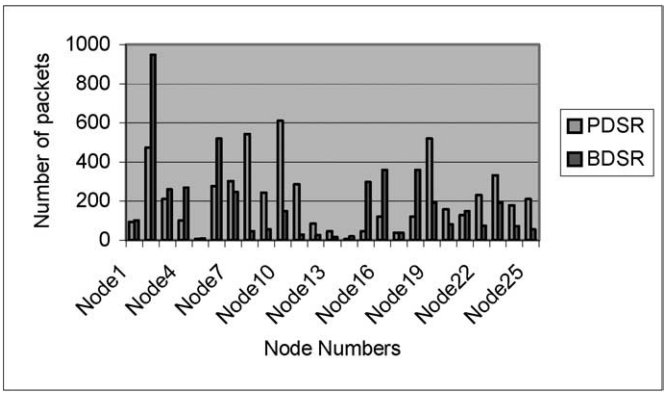


Fig. 15. Comparison of Packets Forwarded with 25 nodes.

Table 4
Comparison of Forwarded packets by Proposed DSR and Basic DSR

Protocol Name	Average number of forwarded packets with		
	Node size 15	Node size 20	Node size 25
BDSR	135	199	182.68
PDSR	178.66	235	214.4
% increase in packets forward	24.43	15.31	14.79

2. Source Transparent Route Maintenance Method

Goal of the source transparent route maintenance method is to optimize DSR (Johnson *et al.*, 2003) to gracefully handle mobile host mobility through cache validation and local route repair schemes. This design is based on observation that the source host plays a key role in route maintenance, and this strategy is the source of much performance degradation in high mobility (Hou and Tipper, 2003; Su *et al.*, 2001). The basic idea of method is have the intermediate hosts to be responsible for validating the cached route and repairing broken link that they encounter. This design seeks to solve invalidate cache entry and broken link as soon as possible to reduce transit packet loss and end-to-end delay.

2.1. *Mobility Impact on DSR*

Mobility (Bai *et al.*, 2004; Wang *et al.*, 2001) introduces two major challenges to DSR, link breakage and frequent route cache expiration. DSR's delivery ratio is lower and the routing overhead is much higher when mobile hosts' pause time is closer to zero (Leung *et al.*, 2001). In this Section, cause of performance degradation of DSR is identified.

During high mobility scenarios, the cache entries are often become invalid. This invalidation causes a big problem for source hosts that use the cached route. A mobile host with DSR uses an invalid cached route either from its own invalid cache entry or from

an RREP from an intermediate host that has an expired cached route. In either case, the source node has no way to determine the validity of the cached route. It will assume that the network has not changed since the cached route is first added to the cache. In a high mobility scenario, this is often not true. A source host that uses a broken route will receive a RRERR message from the forwarding host, and the source host must reconstruct the route by initiating the route discovery sequence. Thus, invalid cache entry causes noticeable packet losses, higher end-to-end delay and extra route overhead.

Major issues of source aware route maintenance method are explained as follows.

First, issue is that this method introduces significant communication overhead. Instead of repairing routes at the broken hop, the source host reconstructs the route from scratch. For example, if the routing path is ten hops between the source and receiving hosts, and let the intermediate host at the nine hops be moved away. With the original DSR, all of the packets that are in-transit to the receiving host are dropped and new route has to be reconstructed at the source host by re-flooding the network with RREQ.

Second issue is that, reconstructing the route from the source instead of at the broken hop significantly increases end-to-end delay because data packets are buffered at the source host while the route maintenance is in progress. The buffered packets are effectively delayed for one RTT plus the route reconstruction time starting at the source host. This route maintenance strategy significantly degrades the performance of TCP and delay-sensitive real-time applications, such as streaming video and audio applications.

Third problem is that, as the example illustrates, the route reconstruction phase can result transient packet losses. So, the overall throughput of the network is noticeably decreased. These three issues demands for a cost efficient optimization method for route maintenance.

2.2. *Design*

2.2.1. *Cache Validation*

Cache validation is a scheme used to ensure that the cached route used by source host has not been expired. This scheme applies to both the source host and intermediate hosts. To incorporate this scheme into DSR, DSR's packet header extends to have three fields, probe request (PReq), probe reply (PRep), and probe sequence number (PSeqno).

For the source host, when it needs to send a message to its corresponding receiving host and it finds that it has a route to the receiving host in its cache, it first unicasts a probe request packet to the receiving host with the cached route as the source routing path. The source host will perceive three possible outcomes.

First, it receives a probe reply from the destination (refer Fig. 16). In this case, the source host will send its message to the receiving host according to the DSR protocol. Second, if the cached route is broken, the source node will receive a RERR message from the intermediate host. In this case, the source host can invalidate its cache entry and initiate route discovery sequence. Third, if the source node has not received a probe reply after the probe timer had expired, it will also initiate route discovery sequence. This case occurs when one of the intermediate hosts in the routing path is heavily congested and cannot forward the probe request before the timer expired.

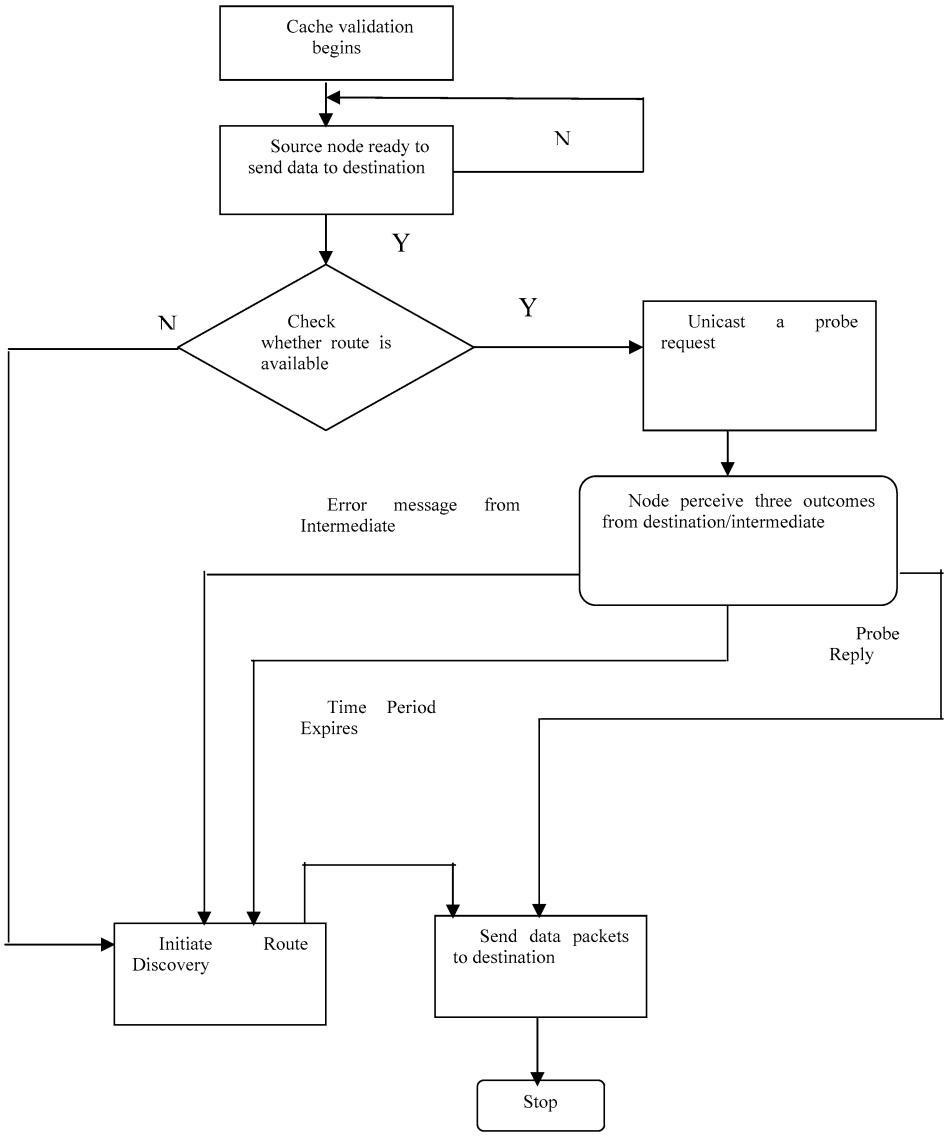


Fig. 16. Flowchart of cache validation by source node.

For an intermediate host, when it receives an RREQ and it has a route to reach the receiving host, it temporarily buffers the RREQ and unicasts a probe request packet to the receiving host using its cached route. The intermediate host will perceive four possible outcomes. First, if it receives probe reply from the receiving host, it will reply to the sender with the route reply using its cached route. Second, if it receives a RERR message from the probe-forwarding host, it will continue to forward the RREQ packet as specified by the DSR protocol. Third, its probe-timer expires, it continues to forward RREQ packet.

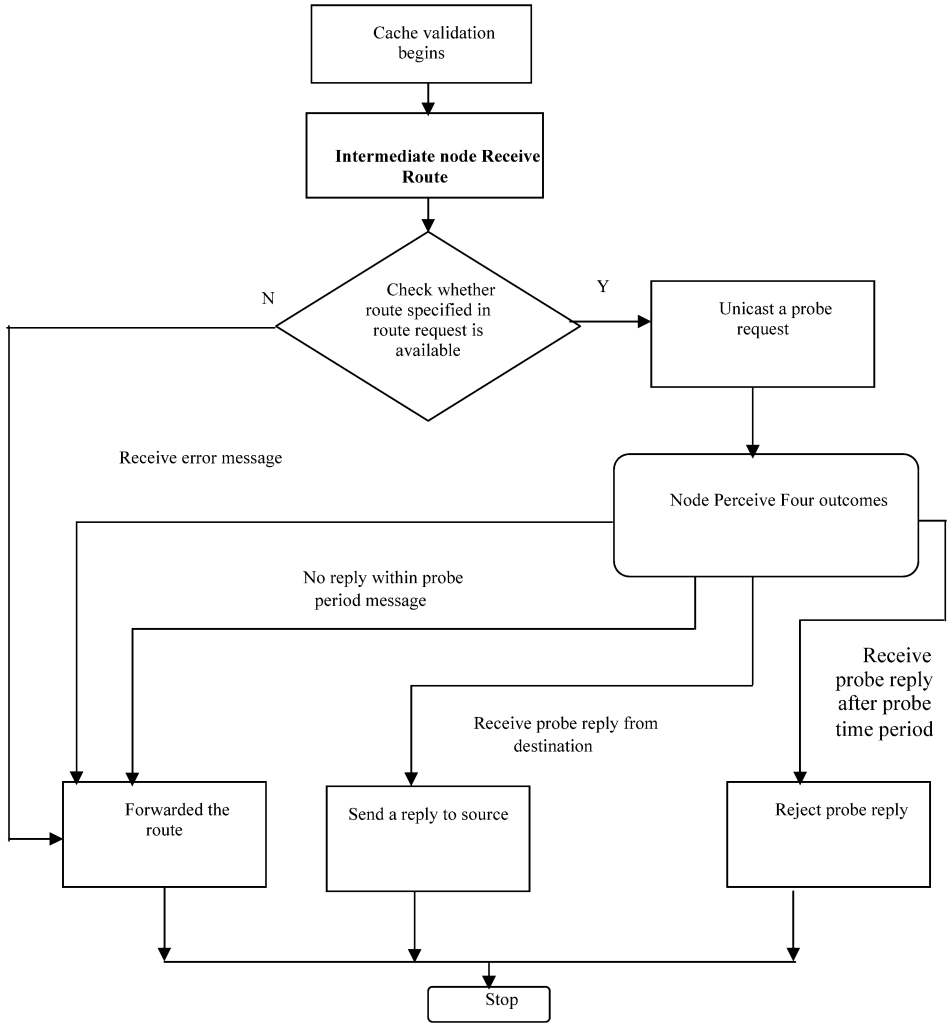


Fig. 17. Flowchart of cache validation by intermediate node.

Fourth, if its timer has expired, the RREQ is forwarded and it receives the probe reply, the host drops the probe reply (Fig. 17).

Cache validation scheme avoids in transit packet losses and increases the throughput. This scheme has some unavoidable problems. The probe packets introduce extra routing overhead into the network, and it increases the time of the route discovery sequence. However, it is believe that these drawbacks are well justified, given the cost that the source host uses an invalid cached route.

2.2.2. *Local Route Repair*

Local Route Repair is a scheme to allow intermediate host to continue to forward data packet despite it encounter broken links. When an intermediate host detected that its next

host cannot be reached (i.e., broken link), instead of immediately sending a RERR back to the source, it temporarily buffers the data packet and broadcast a local route request (LRREQ) packet to its one-hop neighbors by setting the TTL field to one. This message is attempt to locate the next-next hop host specified in the source path of the packet header. If the host receives a local route reply (LRREP) packet from a neighbor, it will update its cached entries and forward the data packet to the neighbor to deliver to the receiving host. If it does not receive an LRREP from its neighbor after its timer expires, it will drop the buffered data packets and send a RERR packet back to the source host.

The benefits of local route repair are fairly intuitive. For a medium Ad hoc network density, 5 to 10 neighbor for each mobile host, this scheme can easily locate a one-hop neighbor that can reach the next-next hop host. If this is possible, this design prevents the source host from reconstruct a new routing path from scratch. This effectively reduces route overhead, transient packet loss and also end-to-end delay. The drawback of this design is that if the host cannot find a neighbor that can reach the next-next hop host, this scheme decrease the frequency of source host to trigger route maintenance.

2.3. *Implementation and Simulation Results*

Both cache validation and local route optimization are implemented in ns-2 network simulator, version 2.1b8a (UCB/LBNL/VINT Network Simulator).

The simulation consists of 50 mobile hosts move about over a rectangular flat space with the size of 2000×600 meters for 900 seconds of simulation time. The movement model in the simulation is based on the random waypoint with the minimum speed of 0 and maximum speed of 20 meters per second. The system simulates the host movement pattern for 7 different pause time, 0, 30, 60, 120, 300, 600 and 900 seconds for UDP and 4 pause time 0, 30, 60 and 120 seconds for TCP.

Simulation result of two optimizations under both UDP/CBR and TCP traffic is shown in this Section. Two important performance metrics namely throughput and delay are analyzed under the various traffic.

2.3.1. *DSR's Cache Usage and Route Maintenance*

The amount of route repair and the amount of RREP that are based on cached routes with different pause time have been observed in this set of simulation. This set of simulation is run with the original DSR. The communication pattern for this set of simulation is UDP/CBR traffic. There are 10 UDP source and sink pair communicates at speed of 4 packets per second. The packet size is 512 bytes. From Fig. 18, the amount of route repair is much higher at high mobility scenarios and significantly reduces as the mobile hosts become more stationary. As mentioned in Section 2.1, route repair triggers route discovery sequence. So the number of route request is high when pause time is low. When an intermediate host receives an RREQ, it first looks through its cache for a cached route to the receiving host. If it does have a cached route, it will send an RREP back to the source node.

As seen from Fig. 19, DSR make heavy use of its cached route in high mobility scenarios because route repair is triggered frequently. So the number of route reply is

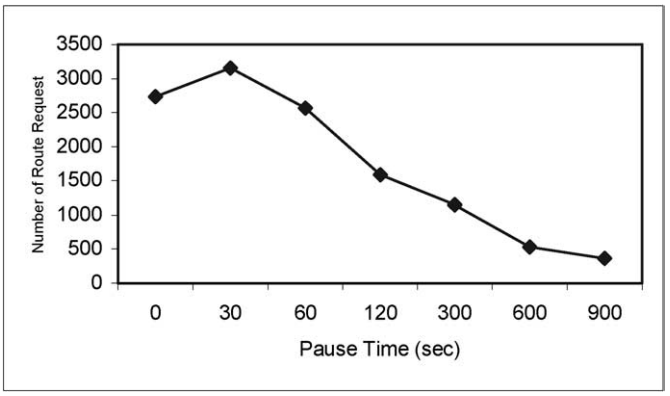


Fig. 18. Number of Route Request.

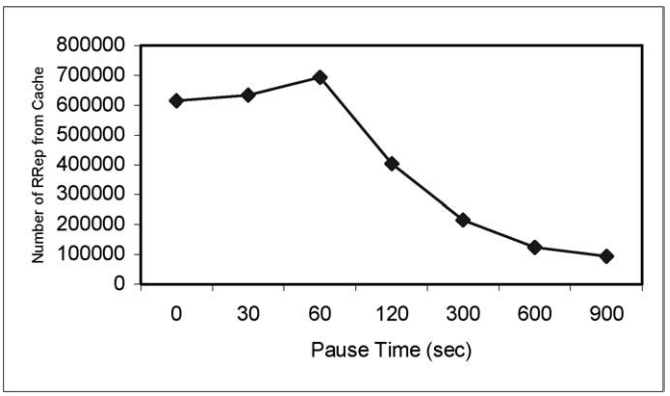


Fig. 19. Number of Route Repair based on cached route.

high when pause time is low. However, in high mobility scenarios, the cached routes expire very quickly, and this increase the chance that a source host uses an invalidate route.

2.3.2. *Comparisons Under CBR Traffic*

In simulation run, performance of original DSR with new cache route validation and local route repair is compared. The traffic pattern of this simulation is CBR/UDP. There are 10 source and sink pairs which send at the rate of 4 packets per second, with packet size of 512 bytes.

In this simulation experiment, local repair performs better than both original DSR and cache validation is observed. This is because local route repair scheme repairs route when links are broken. This method avoids the extra overhead of reconstruct the routing path from scratch.

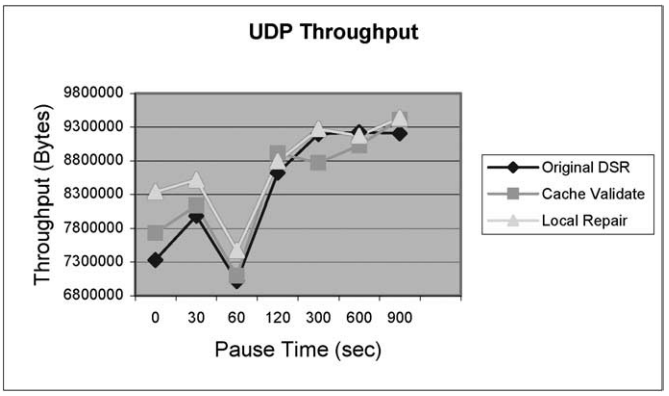


Fig. 20. UDP Throughput.

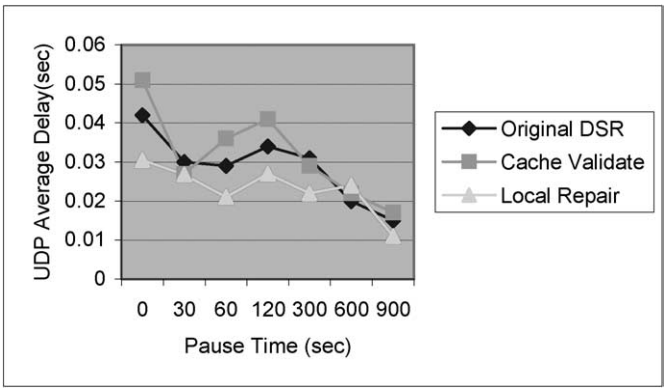


Fig. 21. UDP Delay.

Table 5

UDP Throughput and Delay details of optimization schemes

UDP Metrics	Original DSR	DSR With cache validation	% of savings	Original DSR	DSR With Local Route repair	% of savings
UDP Throughput (Bytes)	8328571.43	8442857.14	1.35 (increase)	8328571.73	8700000.00	4.26 (increase)
UDP delay (sec)	0.030	0.032	5.35 (decrease)	0.030	0.022	25 (decrease)

2.3.3. *Comparisons Under TCP Traffic*

The performance of original DSR is compared with proposed cache route validation and local route repair schemes by simulating using the traffic pattern TCP. There are 10 source and sink pairs which run FTP, with packet size of 512 bytes.

In this set of simulation, again, the local route repair outperforms both original DSR

and cache validation in terms of both throughput and delay. This is the same reason as the previous set of simulation. In this case, cache validation performs better than original DSR in terms of both throughput and delay. This is because cache validation verifies the cached route so that the source host will not use broken links. This effectively avoids some of the TCP timeouts, which result from the route discovery sequence. In the local route repair scheme, significant improvement in the reduction of delay has been achieved by 53.84% and an increase of 8.77% in throughput. Under high mobility situation, local route repair scheme outperforms the conventional DSR in terms of throughput and end-to-end delay.

3. Conclusion

The new Link Cache has given great reduction in the number of control packets and Route Request packets. Difference in performance is too feeble to notice for less number of

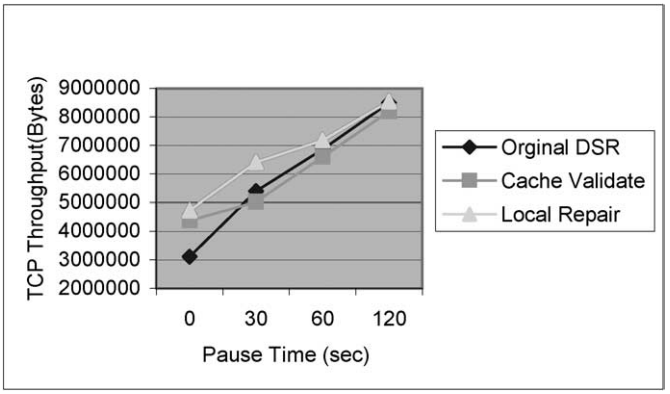


Fig. 22. TCP Throughput.

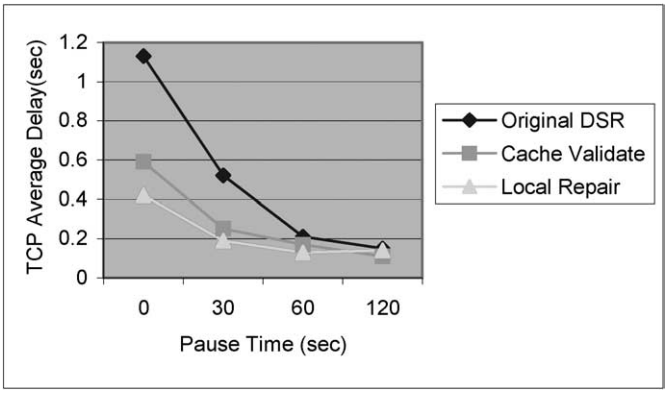


Fig. 23. TCP Delay.

Table 6
TCP Throughput and Delay details of optimization schemes

TCP Metrics	Original DSR	DSR With cache validation	% of savings	Original DSR	DSR With Local Route repair	% of savings
TCP Throughput (Bytes)	5975000	6200000	3.62	5975000	6550000	8.77
TCP delay (sec)	0.52	0.29	43.75	0.52	0.24	53.84

nodes (Less than 10). But when the number of nodes increase, the performance becomes better (20 nodes or more).

The simulation study shows that under high mobility scenarios local route repair is perform well compared to cache validation and. The cache validation scheme performs better for TCP application with reduction of end-to-end delay compared to UDP. No significant is improved on throughput for TCP and UDP applications. The local route repair scheme showed a significant improvement in the reduction of end-to-end delay for TCP and UDP applications. The throughput is improved by local route repair scheme for UDP and for TCP applications.

However, in near static low mobility scenarios, the performance improvement is not noticeable or there is a slight degradation both in terms of throughput and delay for UDP traffic compared to TCP. This is due to slight increase overhead on local route repair.

References

Bai, F., N. Sadagopan, B. Krishnamachari and A. Helmy (2004). Modeling path duration distributions in MANETs and their impact on reactive routing protocols. *IEEE Journal on Selected Areas in Communications*, **22**(7), 1357–1373.

Chin, K.W. *et al.* (2002). Implementation experience with MANET routing protocols. In *ACM SIGCOMM Computer Communications Review*. pp. 49–59.

Johnson, D., and D. Maltz (1996). Dynamic source routing in Ad Hoc Wireless Networks. In T. Imielinski and H. Korth (Eds.), *Mobile Computing*. Kluwer Academic. pp. 153–181.

Johnson, D.B., J. Broch, D.A. Maltz, Y.-C. Hu, J.G. Jetcheva (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking* (*MobiCom '98*). ACM, Dallas, TX.

Johnson, D.B., D.A. Maltz and Y.C. Hu (2003). The dynamic source routing protocol for mobile ad hoc networks (DSR). Internet draft, IETF.

Glomosim Manual (version 1.2).
`http: //pcl. cs. ucla. edu/projects /Glomosim / Glomosim Manual.`

Hou, X., and D. Tipper (2003). Impact of failures on routing in mobile Ad Hoc Networks using DSR. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*. Orlando, FL. pp. 143–149.

Hu, Y.-C., and D. Johnson (2000). Caching strategies in on-demand routing protocols for wireless ad hoc networks. In *Proceedings of the International Conference on Mobile Computing and Networks* (*MobiCom'00*). pp. 231–242.

Lee, S.J., and M. Gerla (2001). Dynamic load-aware routing in ad hoc networks. In *Proceedings of IEEE International Conference on Communications*, Vol. 1. pp. 3206–3210.

Leung, R., L. Jilei, E. Poon, A.-L.C. Chan and L. Baochun (2001). MP-DSR: A QoS-aware multi-path dynamic source routing protocol for wireless ad hoc networks. In *Proceedings of Twenty Sixth Annual IEEE Conference on Local Computer Networks*, Tampa, Florida. pp. 132–141.

Lou, W., and Y. Fang (2002). Predictive caching strategy for on-demand routing protocols in wireless ad hoc networks. *Wireless Networks*, **8**(6), 671–679.

Maltz, D.A., J. Broch, J. Jetcheva and D.B. Johnson (1999). The effects of on-demand behavior in routing protocols for multi-hop wireless Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications* (special issue on mobile and wireless networks). **17**(8), 1439–1453.

Marina, M., and S. Das (2001). Performance of routing caching strategies in dynamic source routing. In *Proceedings of 2nd Wireless Networking and Mobile Computing Workshop* (in conjunction with ICDCS). pp. 425–432.

Perkins, E., and E.M. Royer (1999). Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop. Mobile Computing System and Applications*. pp. 90–100.

Royer, M., and C-K. Toh (1999). A review of current routing protocols for ad-hoc mobile wireless networks. *IEE Personal Communications*, **6**(2), 46–55.

Su, W., S.-J. Lee and M. Gerla (2001). Mobility prediction and routing in ad hoc wireless networks. *International Journal of Network Management*, **11**(1), 3–30.

UCB/LBNL/VINT Network Simulator. `http://www-mash.cs.berkeley.edu/ns/`.

Wang, J., Y. Tang, S. Deng and J. Chen (2001). QoS routing with mobility prediction in MANET Communications. In *Proceedings of IEEE Pacific Rim Conference on Computers and Signal Processing*, Vol. 2. pp. 357–360.

C. Jayakumar has submitted PhD thesis in computer science and engineering at Anna University, Chennai, India. His research interests include wireless network, ad hoc network, software engineering, object oriented analysis and design.

C. Chellappan received the ME and PhD degrees in computer science and engineering from Anna University, Chennai, India, in 1985 and 1987 respectively. He is currently a professor in Department of Computer Science and Engineering at Anna University, Chennai, India. His research area includes mobile computing, distributed computing, parallel and genetic algorithm and neural networks.

Optimizuotas pagal pareikalavimą maršrutizavimo protokolas specialiame mobiliame tinkle

Chinnappan JAYAKUMAR, Chenniappan CHELLAPPAN

Straipsnyje DSR maršruto optimizavimas pasiekiamas naudojant Naujų Ryšių Operatyviosios Buferinės Atminties (NROBA) struktūrą ir specialų maršruto palaikymo metodą. NROBA efektyviai panaudoja atmintį saugodama maršrutus tam tikro tipo duomenų struktūrose. Jis parenka trumpiausius tranzitinius persiuntimus ir mažiausiai perkrautus kelius. Siūlomas metodas sumažina paketų praradimą, persiuntimo laiką ir padidina pralaidumą.