An Enhanced DSR Caching Scheme Based on Cross Layer Information

Gaurav Bhatia and Vivek Kumar

Department of Computer Science,
Gurukul Kangri Vishwavidyalya,
Haridwar-249404, India
gbhatia13@gmail.com, vivekdcg@gkvharidwar.org

Abstract. Dynamic Source Routing (DSR) is an efficient reactive routing protocol for mobile ad hoc networks, in which only needed routes are found and maintained. Route caching is employed to avoid the need for discovering a route before each data packet is sent. DSR employs caching of routes to increase the protocol efficiency. However, the absence of any opportune mechanism to remove cache entry causes cache to contain stale information, also the aggressive use of cache cause dissemination of stale route information, which leads to delay and increases loss rate in high mobility scenario. In this paper, we propose a dynamic mechanism which computes Expected Link Expiration Time (ELET) using cross layer parameter which timely removes the stale cache entry from route cache, also an updated route reply method is used to prevent dissemination of stale routes. Simulation results in NS2 show that enhanced DSR (EDSR) cache scheme can swiftly adapt to scenario changes and can perform better than the existing caching scheme.

Keywords: DSR, RSSI, Route Cache, Cross Layer Information, MANET.

1 Introduction

Dynamic Source Routing (DSR) is an on-demand protocol that uses source routing and makes aggressive use of route caches. The current specification of DSR lacks a mechanism to determine the validity of routes in the route caches. DSR uses fixed time interval for cache invalidation, i.e., entry in cache appoints a fixed time and removed when time expired. This mechanism is not efficient as waiting too long to invalidate route introduces stale route cache and its dissemination. Also not waiting long enough removes the routes from cache which are still valid and causes unnecessary retransmission of route request and route reply. The weakness of this scheme is that it cannot adapt to the change of the network topology. Because of these, setting the timeout close to the expected link expiration time is considered to improve the performance. Since the actual lifetime of a link highly depends on node mobility, to achieve good performance, dynamic caching schemes are desired. Our goal is to develop and analyze enhanced cache strategies for reducing number of stale route entries and their dissemination. The basic idea of the scheme is to use the *Expected Link Expiration Time* (ELET) as its cache timeout and preventing the distribution of stale

D. Nagamalai, E. Renault, and M. Dhanushkodi (Eds.): PDCTA 2011, CCIS 203, pp. 191–203, 2011.

information by updated route reply. The ELET is a measure of time duration in which a node will become out of transmission range of another node. ELET is determined dynamically by the *Enhanced DSR* (EDSR) when it receives RREQ from nearby node using the cross layer information. Cross layer design [1] refers to protocol stack that intercommunicate the useful information to collectively achieve the desired optimization goal by allowing the different protocols to share information related to the network status. In this paper, we propose a cross layer based cache mechanism in which DSR computes timeout value of individual links by utilizing received signal strength from physical layer. This method uses locally available network information and does not require any external support.

This paper has been structured as follows. Section 2 gives an overview of DSR protocol and its cache structure. Section 3 describes the problem statement. In section 4, we discuss related work. Section 5 describes proposed work and EDSR cache mechanism, it explains how the cross layer information is collected and how it is used to predict the link expiration time which is further used in cache invalidation. Section 6 presents simulation and result analysis, and finally, in section 7 we present our conclusions and future work.

2 Dynamic Source Routing (DSR) Overview [2]

DSR is a reactive and efficient routing protocol designed specifically for use in multihop wireless ad hoc networks of mobile nodes. It allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The operation of DSR is based on source routing, in which the sender of a packet determines the complete sequence of hops to be used as the route for that packet to its destination. The source route, the complete sequence of hops through which the packet passes, is represented in the header of each packet. DSR protocol operates entirely on-demand and lacks periodic activity of any kind at any layer within the network. This allows the network overhead caused by DSR to minimum only to that needed to react to changes in the currently using routes. DSR protocol consists of following mechanisms:

2.1 Route Discovery

When a source node originates a new packet addressed to a destination node, it will search its Route Cache for a source route. If no route is found in the cache, the sender initializes Route Discovery by broadcasting a *Route Request* (RREQ) packet (Fig. 1), containing destination node address, unique request identification, and an initial empty list which together uniquely identify this Route Discovery.

A node receiving the RREQ, if it is not the intended destination, appends its address to the node list and forwards the packet. However, first it checks whether it has recently seen another RREQ from the same source node with the same request identification and target address, or whether its own address has already presented in the traveled node list of this RREQ. If either check is true, the node silently drops this packet. When the RREQ packet reaches the destination node, this node returns a *Route Reply* (RREP) to the source node (Fig. 2) with a copy of the node list from the RREQ.

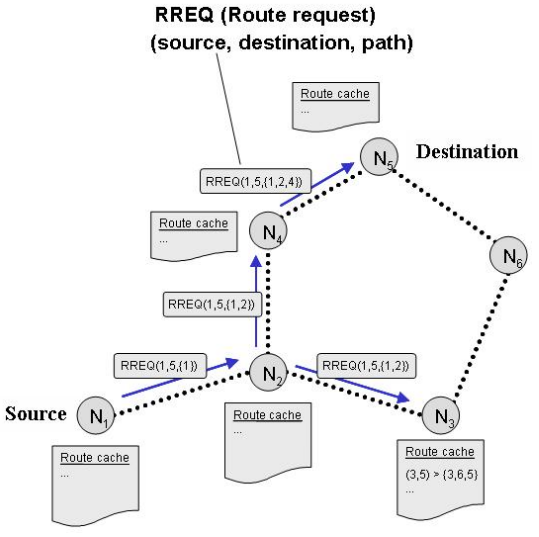


Fig. 1. Node N_1 sends RREQ

If an intermediate node receiving the RREQ contains the route to the destination in its Route Cache then this node returns a RREP to the source node from its own route cache (Fig. 2) rather than forwarding the route request. If the destination node receives the multiple RREQ propagated from different routes, it replies to all RREQ by RREP. As a result of single route discovery to a destination node leads to multiple

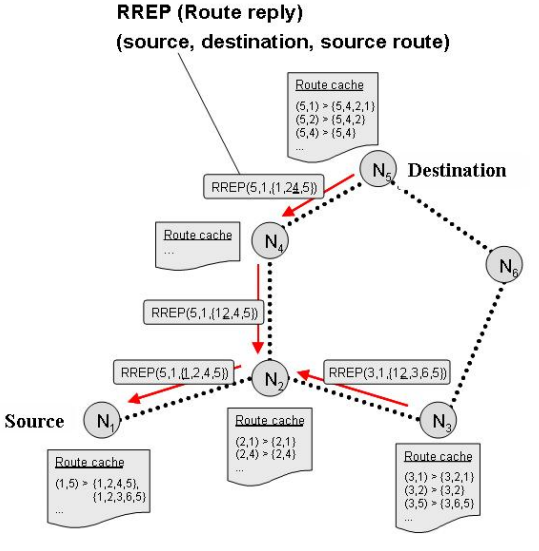


Fig. 2. Nodes N_5, N_3 sends RREP

routes for it. The RREP can be delivered to the initiator by simply reversing the node list, by using a route to the initiator in its own cache, or "piggybacking" the packet on a new Route Request to the original initiator. When the initiator receives the RREP, it adds the source route in its route cache for use in sending subsequent packets to the destination and for future use.

An additional method to learn new routes information is to allow nodes in DSR to add all usable routing information to its own Route Cache by overhearing the source routes on packets sent by other nodes or packets forwarded by it.

2.2 Route Maintenance

While using a source route to send a packet to the destination, each node transmitting the packet is responsible for confirming that it successfully reaches the next hop in the route. The node can confirm by an acknowledgement. If no acknowledgement is received after maximum retransmission, the forwarding node assumes that the next-hop destination is unreachable over this link, and sends a *Route Error* (RERR) to the source of the packet, indicating the broken link.

An additional feature of Route Maintenance is packet salvaging. When an intermediate node forwarding a packet determines the next hop is unreachable over the link, in addition to sending back RERR, it replaces the original route with an alternate route, if it finds any other route to the destination, from its route cache then it forwards the packet to the next hop along with the new route. A node, either source or intermediate, receiving a RERR removes that link from its route cache.

2.3 Route Cache

DSR protocol maintains a Route Cache, containing routing information needed by the node. A node adds information to its Route Cache as it learns new links between nodes in the ad hoc network, for example, a node may learn new links when it receives a packet carrying a Route Request, Route Reply or DSR source route. Likewise, a node removes information from its Route Cache as it learns that existing links in the ad hoc network have broken. The Route Cache can be implemented either in two types of organization [3]:

Path Cache. A path is complete sequence of links leading to the destination node from source node (Fig. 3). By caching each of these paths separately, a path cache organization for the Route Cache can be formed. A path cache is very simple to implement and easily guarantees that all routes are loop-free, since each individual route from a Route Reply or Route Request used in a packet is loop-free. To search for a route in a path cache data structure, the sending node can simply search its Route Cache for any path (or prefix of a path) that leads to the intended destination node.

Link Cache. Each individual link in the routes returned in Route Reply packets (or otherwise learned) is decomposed into individual links and added to a unified graph data structure of this node's current view of the network topology (Fig. 4). To search for a route in link cache, the sending node use a graph search algorithm, such as the well-known Dijkstra's shortest-path algorithm or the breath-first-search (BFS) shortest path algorithm, to find the current best path through the graph to the destination node.

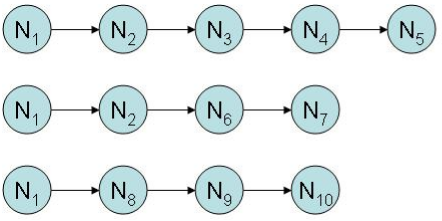


Fig. 3. Path Based Organization

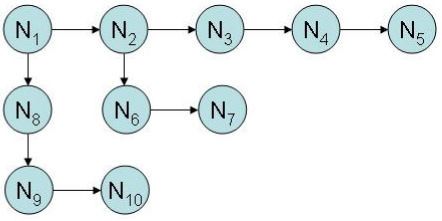


Fig. 4. Link Based Cache Organization

3 Problem Definition

DSR makes very aggressive use of route caching. It adds source route information in route cache when it receives route reply in response to a route request and when it overhears any packet's transmission nearby it. Also, it learns the routes when it forwards any packet. This route cache information is used in sending packets to the destination, sending route reply to another node and for packet salvaging.

DSR protocol uses route cache to avoid the need to rediscover route for each individual packet. If route cache contains stale information, this may degrade performance of DSR rather than improve it.

3.1 Stale Cache Information

The route cache may contain the stale information [3-5] containing links that are no longer exists. When node mobility is high, entries in route caches quickly become invalid as node becomes unreachable when it goes out of transmission range of its nearby nodes. A stale route cache entry is noticed by a node when it sends a packet and receives the route error message by intermediate node. The sender or any other node receiving route error message removes that link from its route cache. This is inefficient error notification as, when a link breaks, route errors are not propagated to all caches that have an entry with the broken link. Instead, the route error is unicast only to the source whose data packet is responsible for identifying the link breakage. Thus only a limited number of caches are cleaned. The failure information is, however, propagated by piggybacking it onto the subsequent route requests from the source. But as the route requests may not be propagated network-wide (because of replies from caches), many caches may remain uncleaned. Thus this is an inefficient method of stale route cache cleaning.

DSR has static timeout mechanisms associated with each entry in the Route Cache to allow that entry to be deleted after a fixed interval of time (RouteCache-Timeout). This leads to stay the stale source route entry for a longer time in route cache when the link becomes invalid. DSR lacks of any timely stale route cache invalidate mechanism. When a stale route is used to send data packets it takes significant time to detect a broken link, which causes data packets to suffer unnecessary longer delays.

3.2 Dissemination of Stale Cache Information

The stale route cache entry can easily spread widely into the network [4], as the DSR has a mechanism through which a node can reply the route request message from its own route cache. However, DSR does not ensure the validity of route which it replies from its cache. Thus a node may reply a route which is stale. When a node uses information from its route cache, the cache staleness problem is compounded, since stale information could circulate in the network indefinitely. For example, one node may use some stale information to route a packet that it sends, allowing a number of nodes to overhear that packet and to cache that stale routing information. If any node that overheard the use of the route does not subsequently overhear the corresponding route breakage notification, that node will be left with a stale link in its route cache, which it may later use in routing its own packets. This may cause a node, which has actually learned that a link no longer exists, to again add the stale route information in its cache.

Thus, DSR protocol has inefficient method of stale route cache cleaning and lacks of any timely stale route cache invalidate algorithm. Also the use of route cache in route reply spread the stale route information in the network. This leads to higher delay latency and reduced throughput.

4 Related Work

Some earlier work [3-5] proposed adaptive timeout for route cache to address the stale cache problem. In [3], author proposed Link-MaxLife timeout mechanism for link cache based on observed link usage and breakages. In [4], Marina and Das use average lifetime of route and time since last link is broken for calculation of timeout value. They multiply average life time of route by some factor then apply MAX function on both values. They also proposed wider error notification and use of negative cache to reduce the distribution of stale cache information. In [5], author proposed adaptive lifetime estimation scheme that adaptively estimate the link lifetime based on the moving average of the previous collected lifetime statistics. These mechanisms predict the timeout of a route cache using pre determined parameters. However, predetermined value of timeout may work for certain scenarios but may not work well for all.

In [6], authors proposed a distributed adaptive algorithm to proactively distribute the broken link information to the nodes that have broken link in their cache. They defined a cache table structure, kept by each node, to maintain the local information necessary for cache updates. Based on it, the proposed algorithm notifies all reachable nodes that have cached the link in a distributed manner. In this work, timeout for route cache entry is not used, thus if nodes become unreachable in some cases then they will not remove the stale route from their caches. In [7], Huang, Chan developed a RERR-Enhance mechanism by transmitting RERR to all nodes that have cached the broken link, they also propose a hierarchical link cache structure, accompanied with a link stability measurement mechanism to determine the stability of a link based on the historical statistic of successful data packets transmission. Ashish Shukla [8] presented a cross-layer approach for predicting the route cache lifetime. The author assigns timeouts of individual links in route cache by utilizing RSSI values received

from physical layer. This scheme requires RSSI thresholds for link timeout on every node of the ad hoc network. In this method the timeout value directly depends on thresholds value selected. In [9], authors proposed an algorithm which performs distributed cache updating by using the local information kept by each node. When a node receives information about a link failure, it checks the field NeighboursToBeInformed and sends a notification to these neighbors. When a neighbor receives a notification, it uses the same algorithm to notify its selected neighbors and so on which quickly propagate broken link information to all reachable nodes whose caches contain that link. The timeout for link cache is also used which is based on Link-Adapt [3] timeout strategy. In [10], a smart packet is generated periodically which travels through the network, collects topology information, and the nodes update their route caches. Route entries then contain new routes reflecting the most recent topology changes. The algorithms [9][10], seem to be effective but they increase the overhead of DSR as they required extra control packets generation.

In this paper, we propose a route cache invalidation mechanism using cross layer parameter. It uses locally available network parameter and does not require any extra control packet generation to get topology information. Our method not only timely removes the stale route cache entry, it also helps in prevention of its dissemination. The novelty of the proposed method is that it is suitable to work with both the cache structures, path and link caches.

5 Proposed Work

An *Enhanced DSR* (EDSR) Route Cache Scheme has been developed to invalidate the route cache entry and updated route reply from route cache. Our approach is to find *Expected Link Expiration Time* (ELET) in between two communicating nodes in the route. Whenever a node adds its link in route request it also adds ELET with respect to nearby requesting node. So that ELET will be used as timeout value for EDSR route cache. When ELET expires, EDSR drops the cache entry to maintain the cache freshness. Also, when a node replies a route request from its cache, it first validates the route by updating the expected link expiration time by subtracting the time spent in the route cache, i.e., the time for which a route stays in the cache, and if it is found that the route is expired or going to be expired it does not reply from cache and further propagate the route request after adding its own information into the source route.

To reduce the problem of stale caching, a route cache invalidation scheme is introduced based on path time out in path cache and link time out in link cache. For path cache, the timeout of source route is considered as minimum of all ELET (from all links in a route). Since if any of links in path fails, it will invalidate the whole route. For link cache, each link is treated individually and path is decided by combining different links, thus, each link is individually invalidated according to its ELET. Also, each time when a node receives the RREP containing already learned link, it updates the cache information from the newer one. So, any link that has not been updated within the ELET period from the time of its addition to the cache is discarded being stale.

For the computation of ELET, each node uses two received signal strength values, one in the table called as neighbor table and other one it gets from the received packet (RREQ). These are used to compute two distances that further predict the relative velocity and the direction of the movement of the nearby nodes.

5.1 Estimation of the Expected Distance to Be Traveled for Separation

When a node receives or overhears a packet from a nearby node at the physical layer, it measures the *receive signal strength indicator* (RSSI). In order to determine link timeout, each node keeps a record of the RSSI of nearby nodes with timestamp.

Assuming a free space path loss model, we have

$$P_i = P_t G_t G_r \left(\frac{\lambda}{4\pi d} \right)^2 \tag{1}$$

where, *Pi* is the RSSI from node *i* and P_t is the default transmission strength, G_t and G_r are the antenna gains of the transmitter and the receiver respectively, *d* is the separation distance in meter and λ is the wavelength in meters.

Thus,
$$P_i = \frac{K}{\mathrm{d_i^2}} \tag{2}$$

and
$$d_i^2 = \frac{K}{P_i} \Rightarrow d_i = \sqrt{\frac{K}{P_i}} \tag{3}$$

where, K denotes a constant that depends on the transmission power, antenna gains of the two nodes and the wavelength of the transmission.

The distance *d* to the transmitter of a packet can be calculated using (3). For simplicity, we assume that nodes of a given link have the same transmission range, which is a maximum communication distance d_{max} (Fig. 5). The distance between the two nearby nodes is approximated by d ($d \leq d_{max}$). Let d_{pre} be the previous distance of nearby node estimated using the RSSI, P_{pre}, at time t_{pre}, (stored in neighbor table) and the d_{cur} is the current distance measured when the route request is received from that node. We need to compute the distance, d_{break}, that two nodes need to be traveled mutually to be out of transmission range.

Case 1 : Nodes are Moving Closer. If $d_{cur} < d_{pre}$, the nodes are moving close together and thus distance traveled to break the link, d_{break}, is

$$d_{break} = \left(d_{\max} + d_{cur} \right) \tag{4}$$

Case 2 : Nodes are Moving Apart. If $d_{cur} > d_{pre}$, the nodes are moving far away and their link will be broken after traveling distance d_{break} , where

$$d_{break} = \left(d_{\max} - d_{cur} \right) \tag{5}$$

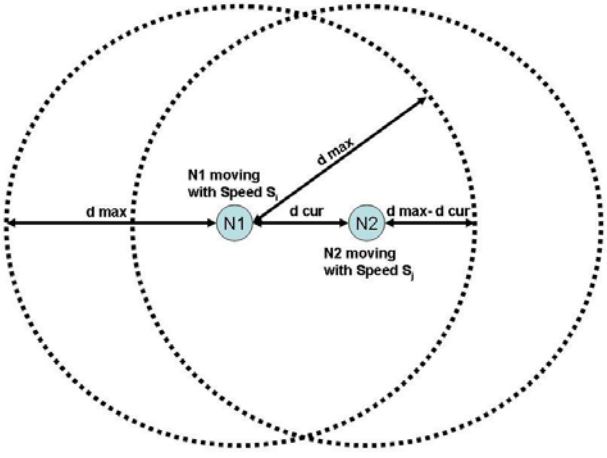


Fig. 5. Two successive nodes constituting a path

5.2 Prediction of Relative Velocity between Two Nodes

We assume that nodes are moving in a straight direction, without any pause and with a constant speed. Using (3), their relative velocity V_{rel} can be predicted as follows:

$$\left| d_{cur} - d_{pre} \right| = \left| \sqrt{\frac{K}{P_{cur}}} - \sqrt{\frac{K}{P_{pre}}} \right| \tag{6}$$

$$V_{rel} = \frac{\sqrt{K}}{t_{cur} - t_{pre}} \left| \left(\frac{1}{\sqrt{P_{cur}}} - \frac{1}{\sqrt{P_{pre}}} \right) \right| \tag{7}$$

where $| (d_{cur} - d_{pre}) |$ is the distance traveled by two nodes mutually with in a time duration $(t_{cur} - t_{pre})$. V_{rel} in (7) allows us to compute the relative velocity. The P_{cur}, P_{pre} are the RSSI measured in between two nearby nodes at time t_{cur} and t_{pre} respectively.

5.3 Expected Link Expiration Time

We estimate the expected link expiration time (τ) as follows:

$$\tau = \left(d_{break} / V_{rel} \right) \tag{8}$$

5.4 Path Expiration Time

Since a route becomes stale as soon as one of its links is broken, thus the *Path Expiration Time* (Γ) for path cache is given by:

$$\Gamma = \min_{i=1}^{N} (\tau_i) \tag{9}$$

where, a path is composed of N links and τ_i represent the expected link expiration time of node i. Only nodes that are in the route from the source to destination are considered in the analysis of path time out duration. If the neighbor table does not have an entry for nearby node then the default value for time out is used.

6 Simulation and Result Analysis

For performance analysis of the proposed scheme, ns-2 [11] simulation tool has been employed. We perform the simulation of the proposed EDSR caching schemes with DSR. We consider both the route cache structures i.e. path cache and link cache.

The scenario consists of 50 mobile nodes which move in an area of 1000×1000 m according to the random way point model. In this model, a node starts in a random position and moves towards in a straight line with a constant velocity and pauses for a specified pause time. We used 0 s as pause time for all scenarios. The nodes move continuously with speed range set to 4, 8, 12, 16, 20 and 24 m/s for different simulation runs. The link layer model is the Distributed Coordinated Function (DCF) of the IEEE 802.11 wireless LAN standard. The default transmission range is 300 meters and channel capacity is 2 Mbits/sec. We considered 15 CBR connections with 4 packets per second and packet size of 512 bytes. Each simulation last for 500 s.

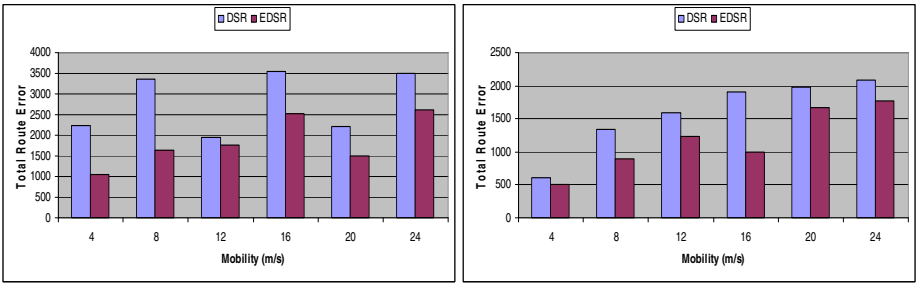
In this simulation, the following performance metrics have been considered for performance evaluation of routing protocol and caching scheme:

- Total Route Error: the total number of route errors generated in the network.
- Stale Cache Ratio: the ratio of stale links present in cache to the total number of links present in the cache.
- End to End Delay: the delay from when a data packet is sent by the source until it is received by the destination.
- Packet Delivery Ratio: the ratio of data packet delivered to the destination and the number of packets generated by the source.

The first two metrics measure the effectiveness of the proposed enhanced DSR caching scheme and the remaining metrics measure the overall performance i.e. latency and loss rate.

6.1 Total Route Errors

Fig.6. shows the total route error generated in network using DSR and EDSR. For both cache structures the total route error for DSR in comparison to EDSR is very high, the reason behind this difference is that the DSR cache structures contain stale routes, which leads to route errors. This is due to the fact that our scheme not only keeps the routes in cache validated but also replies only the valid routes from cache, reducing the possibility of route errors.

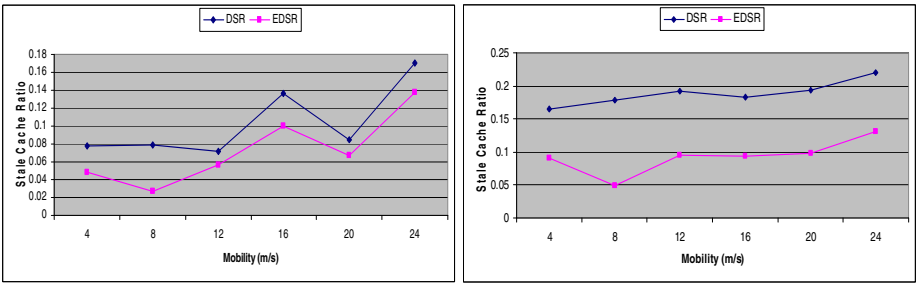


(a) Path Route Cache (b) Link Route Cache

Fig.6. Total Route Errors

6.2 Stale Cache Ratio

Fig.7. shows the stale route cache ratio in path cache and link cache. For EDSR it is much smaller than DSR, this is due to the fact that EDSR timely invalidates the stale route information and does not disseminate it. Such significant improvement shows that EDSR efficiently removes stale routes as the topology changes.

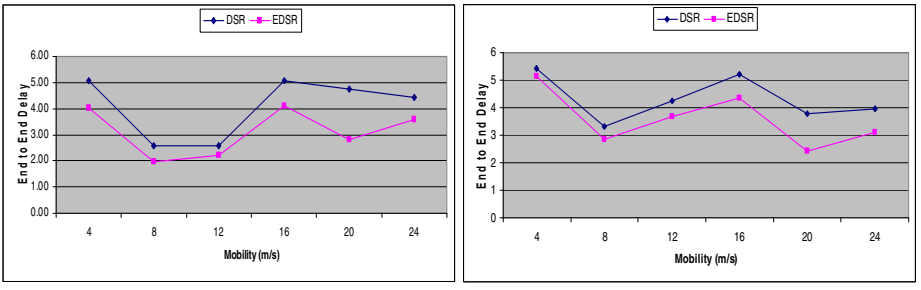


(a) Path Route Cache (b) Link Route Cache

Fig.7. Stale Cache Ratio

6.3 End to End Delay

Fig.8. shows the performance comparison of end-to-end delay for DSR and EDSR protocol. In both path and link cache the end-to-end delay is lower in EDSR as compared to DSR. This is because the time required to recover from broken link due to stale route is very large. It includes the time for a packet travel along the route to the node immediately before the broken link, the time for that node to detect the broken link and the time for a route error message to travel from that node back to the source node. Therefore, we observe that EDSR results in low end to end delay, as delay caused by broken routes are minimized.

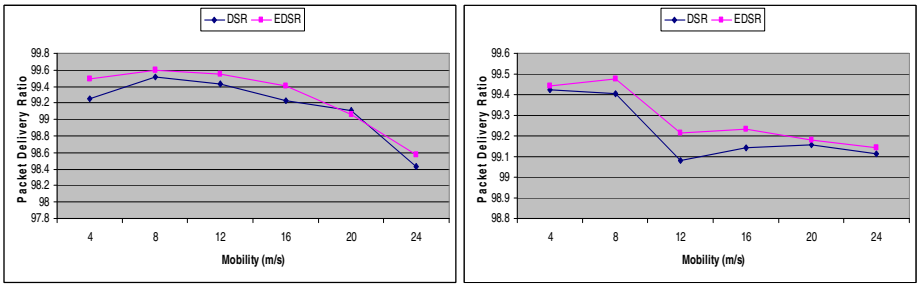


(a) Path Route Cache (b) Link Route Cache

Fig. 8. End to End Delay

6.4 Packet Delivery Ratio

Fig.9. shows that in both cache structures the packet delivery ratio of EDSR is better than DSR. As discussed in section 6.1, DSR reports more route error as compared to EDSR (Fig. 6). DSR drops the data packet due to stale route cache entry which leads to decreased packet delivery ratio for it.



(a) Path Route Cache (b) Link Route Cache

Fig. 9. Packet Delivery Ratio

7 Conclusion and Future Work

In this paper, we present an Enhanced DSR (EDSR) caching scheme for mobile adhoc networks. Towards this, we propose an Expected Link Expiration Time (ELET) that helps in timely removal of stale cache entry. This scheme is based on cross layer information which is gathered at the physical layer (RSSI) and used in DSR to determine the route lifetime.

Our objective is to reduce the stale cache information and its dissemination. With our extension, the Enhanced DSR caching scheme dynamically computes the ELET and adds this value when a route is discovered by source node. The ELET is stored by node in route cache and used to clean the cache entry after its expiration. If a node

replies a route request from its cache then it first updates the ELET value of route by subtracting the time which is spent by route in cache to prevent the stale information dissemination. The simulation results show that Enhanced DSR caching scheme can considerably improve the performance. It reduces the route error which causes lower end to end delay and higher packet delivery ratio. Future work includes implementing the proposed scheme in selection of more stable route for DSR.

References

1. Srivastava, V., Motani, M.: Cross-layer design: a survey and the road ahead. IEEE Communication Magazine 43(12), 1112–1119 (2005)
2. Johnson, D., Maltz, D., Hu, Y.-C.: The Dynamic Source Routing for mobile ad hoc networks. IETF Internet Draft (2004), http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-10.txt
3. Hu, Y., Johnson, D.: Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. In: Sixth Annual International Conference on Mobile Computing and Networking, Boston, MA, USA (2000)
4. Marina, M., Das, S.: Performance of Route Caching Strategies in Dynamic Source Routing, pp. 425–432. IEEE Computer Society, Washington, DC, USA (2001)
5. Lou, W., Fang, Y.: Predictive Caching Strategy for On-Demand Routing Protocols in Wireless Ad Hoc Networks. Wireless Networks 8(6), 671–679 (2002)
6. Yu, X., Kedem, Z.: A distributed adaptive cache update algorithm for the dynamic source routing protocol. In: IEEE INFOCOM 2005, Miami, Florida, USA, vol. 1, pp. 730–739 (2005)
7. Huang, T., Chan, C.: Caching Strategies for Dynamic Source Routing in Mobile Ad Hoc Networks. In: Wireless Communication and Networking Conference (WCNC), pp. 4239–4243 (2007)
8. Shukla, A.: Ensuring Cache Freshness in On-demand Routing Protocols for Mobile Ad Hoc Network: A Cross-layer Framework. In: 4th IEEE Conference of Consumer Communication and Networking, pp. 264–268 (2007)
9. Garrido, J., Marandin, D.: A Linkcache Invalidation Mechanism for Dynamic Source Routing (DSR) in Ad Hoc Networks. In: 18th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 3–7 (2007)
10. Ashokraj, N., Arun, C., Murugan, K.: Route Cache Optimization Mechanism Using Smart Packets for On-demand Routing Protocol in MANET. In: International Conference on Information Technology (ICIT), pp. 141–146 (2008)
11. The Network Simulator – ns-2, http://www.isi.edu/nsnam/ns/