Optimize DSR Cache with Multilevel Structure

Nada M. Al-Hakkak  
Baghdad College of Economic Sciences University

Issam H. Al-Hadid  
The University of Banking and Finance Science  
Information Technology College

Nawal A. Ibrahim Al-Jarah  
Al-Mustansiria University  
Economic and Administration College
Abstract

Ad-hoc networks consist of nodes that communicate without the need of any centralized infrastructure. Because of the limited range of wireless devices, each node can be a router if it is an intermediate node on a route between any source and destination.

Some of the routing protocols; i.e. Dynamic Source Routing (DSR) protocol depends on the node cache memory to store routes, for future use. We have proposed a new algorithm to deal with the cache of the DSR protocol, for better routing performance.

1-Introduction

Mainly, wireless networks are of two types; e.g. networks with infrastructure; and networks without infrastructure also called Ad-hoc networks (Royer, 1999). This paper deals with the Ad-hoc network’s protocols, the second type. However, in Ad-hoc networks the mobile nodes communicate with each other by radio signals without the need for any central control, because the nodes collaborate with each other for better packet’s transferability in hop-by-hop structure (Johnson, 1999).

2- Routing Protocols in Mobile Ad-hoc Networks

Because of the limited transition rate in the mobile nodes, each node will need the help of its neighbors for connecting with distant nodes. Routing protocols are used to ensure that the cooperation between nodes will take place in a correct way and also to handle the battery limitation of the nodes.
3- Routing Protocols types

There are three main types of routing protocols: (Broadway, 2003)
- **Proactive protocols**: which continuously update its routes.
- **Reactive protocols**: which search just for needed routes.
- **Hybrid Protocols**: which cluster the network into regions and use the proactive methodology inside each region and the reactive methodology between the regions.

4- Dynamic Source Routing protocol (DSR)

The Dynamic Source Routing (DSR) protocol is a simple and efficient routing protocol designed specifically for use in multi-hop wireless Ad-hoc networks of mobile nodes. DSR allows the network to be completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two mechanisms; i.e. Route Discovery and Route Maintenance, which work together to allow nodes to discover and maintain source routes to arbitrary destinations in the Ad-hoc network. The use of source routing allows packet routing to be trivially loop-free by avoiding the need for up-to-date routing information in the intermediate nodes through which packets are forwarded, and allowing nodes forwarding or overhearing packets to cache the routing information for their own future use. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that is needed to react to changes in the routes currently in use (Johnson, 1999).

5- Related Works

DSR is described in detail in (Johnson, 2001). DSR is also one of the few Ad-hoc Routing Protocols that have been implemented and evaluated in a real test. The results are described in (Maltz, 1999). (Aron, 2001) presented an analytical study of the probabilities of successful deliveries and the total amount of traffic generated for a successful delivery. It is argued, that an end-to-end recovery mechanism (as used in DSR) does not scale if the routing path lengths increase. Instead a local recovery mechanism is suggested, that gives much better results (according to the analysis). This algorithm is implemented in the Witness Aided Routing Protocol (WAR, cf. section A.29). Although the analysis is convincing, it was done by the authors of WAR, so the results being strongly in favor of WAR is not surprising. DSR was used in many performance comparisons, evaluating studies, and as a reference for a lot of other protocols. Further, it was used as a reference protocol for investigations to find general improvements for

As we can see from the above studies, there was not a real solution from the cache memory in the DSR, so we hope that our new idea will have a good solution for this problem.

In other words we applied the Ad-hoc On-Demand Distance Vector (AODV) routing protocol’s cache structure, with some modification, on DSR’s cache structure. This made our work different from others; because all previous works did not treat the DSR’s cache directly as two level structures.

6- The Proposed Algorithm

As a result for our study on the behavior of the DSR protocol, we found some problems that the DSR suffers from, those are:

- The cache route size problem; the route cache size will be huge with a lot of route records that hold the whole route, so when the source node wants to send a packet it will check the route cache for best route that can be used to send the packet to the destination.
- The broken route problem; it happened as a result of the following factors: Transmission out of range
  Speed of nodes
  Battery – Empty
- The problem of the route order; when new routes appended to the route cache were the new route will be added to the end of the cache, so the node will pick the old route to send the packet which might be broken.

To solve the above problems we propose a new algorithm that alters the route cache to:

- Solve the size problem
- Solve the broken route (stole routes)
- Search time for route (old and new routes)

The proposed algorithm adds new mechanisms (route order, route status checker, and new route cache structure; i.e. “Master Route cache and Index Route Caches”.

6-1 New Route Cache Structure:

The DSR’s cache will be split into two sub-caches; the first cache called the “Master Route Cache”; which holds information about source node, destination node, number of hops, status of the route and a pointer to the second cache called
“Route’s Index”. The second cache’s part will hold the Index of the route and the route from the source node to the destination node. Master route cache will be ordered based on the shortest path (number of hops) and the newest route which will be provided by the route discovery mechanism. Route Discovery and Route Maintenance will work as suggested in the protocol draft; the proposed version suggests altering the structure of the node’s route cache only.

The following table shows the proposed structure for the DSR’s node; that’s mean each node in the network will use this new structure for storing the information about its neighbors. Mainly, all routes are ordered depending on the most new-short route.

<table>
<thead>
<tr>
<th>Optimized DSR node's cache structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub-cache-1 Master Route Cache</td>
</tr>
<tr>
<td>Source node</td>
</tr>
<tr>
<td>Destination node</td>
</tr>
<tr>
<td>Number of hops</td>
</tr>
<tr>
<td>Route’s status</td>
</tr>
<tr>
<td>Sub-cache-2 Index Route Cache</td>
</tr>
<tr>
<td>Index of the route</td>
</tr>
<tr>
<td>The route from the source to the destination</td>
</tr>
</tbody>
</table>

Table (1): The proposed structure for the DSR’s node

6-2 Working Sections

Mainly, the work of the proposed algorithm will be discussed into four sections; i.e. initialization process, Route Discovery, and Route Maintenance. Hence, we will go through the main steps for the DSR and outline our additions and modifications; where each original step will have (*) as a marker elsewhere other steps will represent the modified parts of the original algorithm.

Section-one: Initialization process

1- Start simulation, with identified environment.
2- Each node will initialize its cache; as it’s a source node and all others are the destination nodes.
   a. Enter source name in the source node field.
   b. Enter destination name in the destination field.
   c. Enter number of hops in the hops field.
d. Enter links status (1=active, 0=broken), but initially all links are supposed to be active.
e. Enter serial number as route-index.
f. Enter full route in route field.

The following example shows the initialization process:

![Network Diagram](image)

Figure (1): A network example for six nodes

### Structure of Master Route Cache

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>No. of Hops</th>
<th>Status</th>
<th>Route Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>1</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>2</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>A</td>
<td>C</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>D</td>
<td>4</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>4</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>2</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>3</td>
<td>0</td>
<td>12</td>
</tr>
</tbody>
</table>

Table (2): Master Route Cache structure for node A

Source: the Source Node. Destination: The Target; the route will be from the source to the destination. No. of hops: number of the hops the packet will jump from node to node to reach the destination node. Status: the status of the
route if it’s broken or not; 1- Active/ 0- Inactive (broken). Route Index: a pointer to the route in the Index Route Cache.

**Structure of Index Route Cache**

<table>
<thead>
<tr>
<th>Route Index</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A-B-C-D</td>
</tr>
<tr>
<td>2</td>
<td>A-B-F-D</td>
</tr>
<tr>
<td>3</td>
<td>A-C-D</td>
</tr>
<tr>
<td>4</td>
<td>A-B</td>
</tr>
<tr>
<td>5</td>
<td>A-B-F-C</td>
</tr>
<tr>
<td>6</td>
<td>A-C</td>
</tr>
<tr>
<td>7</td>
<td>A-B-F-C-D</td>
</tr>
<tr>
<td>8</td>
<td>A-B-C</td>
</tr>
<tr>
<td>9</td>
<td>A-B-F</td>
</tr>
<tr>
<td>10</td>
<td>A-C-F</td>
</tr>
<tr>
<td>11</td>
<td>A-B-C-F</td>
</tr>
<tr>
<td>12</td>
<td>A-C-D-F</td>
</tr>
</tbody>
</table>

Table (3): Index Route Cache structure for node A

*Route Index*: the Index of the route. *Route*: the nodes the packet will pass through from the source node to destination node.

**Section-two: Route Discovery**

1- *Source node has packets to send.*
2- Source node checks its cache, for available active routes.
   a. If route available then get the whole route from the second part of the cache and use it to transmit the packet.
   b. *If route was not available then call the Route Discovery procedure*
      i. Source node broadcast *Route Request (RREQ)* message to all nodes in its radio range, this message contains the source name and destination name.
      ii. For any node that receives RREQ
          1. (YES) if this node was the destination node then it reverse the route, and send *Route Reply (RREP)* to that source.
2. (No) if this node was not the destination then
   a. If it has route to the destination then add its route to the available route and reply by RREP message.
   b. If it does not have any route then just add its address to the RREQ and rebroadcast it.
   c. When source node receives RREP, it checks the route in its cache
      i. (YES) if the route already exists, then change its status to 1
      ii. (NO) if route was new or not exists then
         1. Use route-order program to add the new route to master-route cache depending on its destination and number of hops. With update according to route age.
         2. Add the route to the index-route-cache and use that index as a pointer to master-route-cache related record.
   iii. *The source node uses the route to transmit its packet.

Section-three: Route Maintenance

1- *Any broken link will cause a message to be send to the source node by intermediate nodes.
2- The source node when receives that broken link message:
   a. It will deactivate that link in its master route cache.
   b. Check the master route cache for alternate routes.
   c. If available then copy route from index route cache to packet and retransmit it.
   d. If not available then send RREQ message for new route.

Section-four: Route-Status Checker

Periodically check route-status to delete inactive (broken) routes from index-route-cache and master-route-cache
However, figure (2) describes those main sections in a flowchart. The figure shows the whole process; for the original DSR as regular draws and the optimized DSR as bold draws. Also the strikethrough writing parts represents the initialization section, the underline writing represents route maintenance section, and otherwise will be route discovery section and checkers.

7- Results

This study used the Glomosim simulator, with the following parameters for the simulation environment: transmission radio of 250, 10 source nodes, 2200X600 simulation area, simulation time of 500sec, with the use of IEEE802.11 protocol for Media Access Control (MAC) layer, and the Constant Bit Rate (CBR) as the application layer, with 4pakets/sec as a transmission rate.
Mainly, four main criteria were used: cache size, delivery ratio, broken links, and search time. However, most of them were affected by node speed; hence our best results were on node speed (30m/sec).
Moreover, the figures (3,4,5,6) resulted from the experiment of five-testing rounds. Some of those graph-points had abnormal values, while the rest were normal with 90% confidence according to our statistical results.
Start

Initialize simulation environment

Source node has packet to send

Check node cache for active route

Available

Send RREQ to neighbors

Add its route to the available route

Put its address and retransmit RREQ

Intermediate node receives RREQ

Check if its destination

Reverse route and send RREP to source

Source node checks its cache for RREP route

The route already

Add the new route to cache

Update rout status + 1

Take route from 2nd part of the cache & use it for transition

Check broken link

Send RERR message to source

Source deactivate route (to 0)

End

Periodically check for route status

Figure (2): Optimized DSR
Figure (3): Overhead Measure
From the above figure we can notice that the overhead in the optimized DSR was much less than the original one.

Figure (4): Delivery Ratio Measure
Figure (4) shows that the delivery ratio in the optimized DSR is better than the original DSR.
In figure (5) we did not have optimized results for the broken link issue.

In figure (6) we see that the optimized DSR had a better results in the search time, because of its new structure in string the routes and retrieving them back next time. It’s obvious that the old structure’s search time increased continuously with higher cache size, but with the new structure that time was decreased.
8- Conclusion

The proposed algorithm provides higher speed; Minimum search time for both Master and Index route caches. In addition, the new routes will be stored before the old ones that means decrease the use of the old routes and increase probability that the packet reaches the destination node and solves the broken route problem.

Also, it will solve the cache size problem by dividing the Route cache into two caches (Master and Index); the “Master Route Cache” will be small relatively to the “Index Route Cache”; which will hold the complete route from the source node to destination node and the “Master Route Cache” will only hold a pointer to the master part.

The route status will help in keeping the broken routes for a period of time because some of these routes might be temporary unavailable (battery off or out of radio range) and it might back to the active status again.

Eventually, the proposed algorithm enhanced the delivery ratio, overhead, and search time.
9- References


8- [Das, 1998] Samir R. Das, Robert Casta~neda, Jiangtao Yan, and Ringli Sengupta, Comparative performance evaluation of routing protocols for mobile, Ad-hoc networks," in Proceedings of 7th Int. Conf. on Computer Communications and Networks (IC3N), Lafayette, LA. University of Texas at San Antonio, October 1998,


23