Optimized Reparation based on Sequenced Number Routing for Ad Hoc Networks

Zhijun Cai, Mi Lu and Costas Georgiades
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843-3128, U.S.A.

Abstract: An ad-hoc network is a collection of wireless mobile hosts without the required intervention of any centralized Access Point. Traditional routing schemes would not be efficiently executed in such an environment where network topology is dynamic. Without the use of any traditional routers, a new routing method should be provided for mobile hosts to find routes toward destination, and such a routing method should dynamically adjust the routes corresponding to the changes of network topology. Proposed in this paper is a distributed routing method, termed Optimized Reparation based on Sequenced Number Routing (ORSNR), which applies a route repairing approach with optimization efforts to maintain the routes for network communications, and adopts a special technique with sequence numbers and vector representations to update routing information records for mobile hosts quickly and completely. The ORSNR performs better in large ad hoc networks. The performance of the protocol is analyzed. Some future work and applications are also discussed.

Keywords: Ad Hoc network, routing protocol, reparation, optimization.

1. Introduction

An ad-hoc network is a collection of wireless mobile hosts which has no centralized control unit. It is widely used in many cases, such as battlefields, search-and-rescue operations, disaster environments, etc. In case a costly and centralized control unit is impossible to be built, an ad hoc network is desirable.

An ad-hoc network is bereft of base-station and pre-designed routers and characterized by a highly dynamic network topology because of host migration, signal interference and power outage[1]. Due to this, the routing problem in ad-hoc networks is more complicated than traditional networks, so traditional routing methods can not be used efficiently in the type of networks.

We propose a distributed routing method, termed Optimized Reparation based on Sequenced Number Routing (ORSNR), to solve the routing problem in ad-hoc networks which uses an on-demand approach for route discovery, a repairing-with-optimization technique for maintaining routes when an old route is broken, and a sequence number method for updating routing information quickly and completely.

The next section will discuss the background and the related work. The details of the ORSNR will be described in Section 3. Section 4 will give conclusions and the direction of some future work.

2. Background and Related Work

There are two types of routing protocols for the traditional static networks. One is based on the distance vector approach, the other is based on the link-state approach. In the distance vector routing, each router broadcasts to its neighbor routers its distance to all hosts, and each router computes the shortest path to each host based on the information advertised by each of its neighbors[2]. In the link state routing, each router broadcasts the status of its adjacent network links to all the other routers, and then each router can compute the shortest path based on this information. Both of the two methods are based on the hierarchical addressing schemes and routers should pass detailed route information to each other. They are not fit for routing in ad-hoc networks.

Many protocols have been proposed to solve the ad-hoc network routing problem [1-12]. Some protocols have been designed to modify the traditional routing method to make it fit for ad-hoc networks. For example, the Destination Sequenced Distance Vector (DSDV) [10] method, modifies the distributed Bellman-Ford algorithm to prevent looping by including sequence numbers to order the routing information. It requires each mobile host to advertise to each of the current neighbors its own routing table. Most of the other protocols are based on on-demand routing approach in which the route discovery is only activated at the time that there are data to be sent by the source node. The Dynamic Source Routing proposed in [6] uses an address list when source node S needs to find a route to destination host D. Each host forwarding the route discovery packet will add its own address to this address list. When D receives the route discovery packet, it will send a route reply packet to S including a full address-list, and then every data packet which S sends to D has such a full address list in its packet header. The routing method of Signal Stability based on Adaptive Routing (SSA) [3] uses the signal quality of the channel to determine whether the portions of topology are stable. The routing method of Clustered-Based Approach for Routing Protocol (CBRP) [11] divides the whole network.
into non-overlapped groups. Each group has a group header. An intragroup routing can be administered through the group header and intergroup routing is the on-demand routing by group headers. The group header has enough routing information inside the group. Ad hoc On Demand Distance Vector routing (AODV) [8] uses a demand-driven route establishment procedure. Temporally-Ordered Routing Algorithm (TORA) [7] localizes the routing-related messages to a small set of nodes near the change in order to minimize the reaction to topological changes. Zone Routing Protocol (ZRP) [4] initiates the route discovery phase on-demand, and limits the scope of the proactive procedure only to the initiator’s local neighborhood and the topology update propagation to the neighborhood of the change.

Normally, routing process includes two factors: one is the routing information of the current network obtained before a routing discovery process really begins. The other is the delay and network bandwidth spent on a routing discovery process. Normally, the more the routing information obtained, the less the delay and bandwidth. On the other hand, the routing information should be updated according to the change of network topology. Updating the routing information table also uses network resources. In the ad hoc routing, we should find the balance of these two factors. Designing an efficient protocol to decrease the overhead of the route discovery and the maintenance in ad hoc networks is the goal of this paper.

The ORSNR is based on on-demand routing approach in route discovery, but it is significantly different from the above protocols. It uses the repairing with optimization approach in route maintenance, the distance-vector format to store information, and the sequence number along the route to update the routing information table quickly and completely. In each host, it has three tables to store different kinds of information. The ORSNR makes the route discovery quicker and the route maintenance more adjustable to network topology changes.

3.2 Overview of the ORSNR

Our routing method is mainly composed of two parts. One is the route discovery and the other is the route maintenance.

In the route discovery, the source node which has data to send should initiate a route discovery process when the route toward destination cannot be found in the routing information table. In this process, the intermediate hosts along this route should gather routing information as much as they can. The information is stored in a distance vector format to make future routing easy. In our method, when the network topology changes, the route information table can be updated completely and quickly. This is very important, because the out-of-date routing information will make the routing performance much worse. In the route maintenance, we use the route repairing approach to repair the broken route, and use the optimization technique to make the repaired route better.

First, we will discuss how to obtain the routing information, store them efficiently and update the routing information table quickly and completely. Then we will explain the route discovery and route maintenance. Finally, we will give the performance analysis of the ORSNR.

3.3 Obtain Routing Information

In an ad hoc network, periodically broadcasting routing information from one node to another is not a good idea. Our method is based on an on-demand routing approach. We find the route only when the source node has data to send. The only way we obtain routing information is to make full use of the route discovery process. We regard the process that source node S finds a route to destination node D as the process of broadcasting the routing information about a certain part of the network. When node S tries to find a route to D, it sends out a Route Request Packet with an address list. Each host forwarding the packet will add its own address to the address list. When D returns a Reply Packet, it carries the full address list of the intermediate nodes. Every node has its sequence number along this route, starting from the source with 0. Every intermediate node can know the full connections between every two nodes along the route, and all such route information are represented in the distance vector form in each intermediate node, and stored into a passive route table in each node. The route between S and D will be put into an active route table.

For example, when source node S tries to find a route to destination node D, if no route is available, S will send a Route Request Packet to find a route with the address list containing itself only. We assume that this packet reaches destination node D without any loop (to avoid a loop, every intermediate node which is sending a Route Request Packet should check whether it sent that packet in the past.). D will return a Reply Packet to S, including the full address-list. Every intermediate node will receive the full connecting information from this address list.
3.4 Store Routing Information

In every node, there are three tables. One is the active information table. One is the passive information table. The third is the neighbor table. The format of the active table is shown in Table 1. The format of the passive table is shown in Table 2.

The active table includes the routing information that is currently being used, and the passive table includes the routing information that was obtained in the route discovery step. Every item in the active table is also in the passive table.

An example is shown below. Suppose the address list is S-11-12-13-14-15-16-17-18-D. For 13, the active table and passive table are shown in Table 3 and Table 4 respectively. We also show how to form the passive table from the address-list in Figure 1.

3.5 Update the Routing Information Tables

When a link between two routing nodes along the route is broken, how does the system update the routing information table in every node quickly and completely? Suppose the link between 14 and 15 (represented by 14-15) is broken. For 14, it detects that 15 is no longer its neighbor after its neighbor table is changed. 14 can know the SNFS of 15 from the SNFS of itself. 15 is identified by S and D and its SNFS. 14 will perform the following operations: 1) Send the Update Packet to the previous hop. The update packet includes 15's identity (S, D, and SNFS). 2) Mark the item of which next hop is 15 in the active table with “dormant”, shown in Table 5. 3) Delete the address-list field of the item whose next hop is 15, as shown in Table 6. When 13 receives the Update Packet, it will send the Update Packet (the same as what 14 sent to 13) to its previous hop immediately. In this example, it is 12. In the mean time, it will check its active table. From S and D, it will know that the route from S to D is broken. Then it marks the corresponding route item as “dormant”. It will check the passive table subsequently. In the passive table, we use the following conditions to delete an address P in the address list of item(x) if item(x) and P satisfy: a) item(x).S=S; b) item(x).D=D; c) item(x).SNFS - P.15 = 1 ≤ SNFS ≤ item(x).SNFS or item(x).SNFS ≤ SNFS ≤ item(x).SNFS + P.15 + 1 (Note: i_j means the count from previous (or next) hop to P, shown in Figure 2).

Figure 3 shows how we process the passive table for the above example. This process can be done continuously until the Update Packet arrives at S. S will know that the route is broken and it will stop the data transmission and wait for route repair process. The similar process is also executed in 15, 16, 17, .... D. We should point out that we process the active table and the passive table in different manners, because the actual data communication uses the active table and the passive table just provides routing information. As you can see below, we will adopt a special method for “dormant” route.

Note, if an intermediate node receives the Update Packet from both the previous hop and the next hop with the same S and D but different SNFS, it should delete the corresponding route item in the active table. For example, 11-12, 13-14 are broken, and the corresponding route items in the active tables of 12 and 13 will be deleted instead of just marking them dormant.

3.6 Route Discovery

When one node tries to find a route, it checks the active or passive table to obtain the routing information. Hopefully, that route can be found.

If not, S will initiate a route discovery process which uses the broadcast method to find a route. In the forward route discovery, the intermediate node uses the passive table to find a route. If a route is found, the broadcast will become unicast and the Route Request Packet should be continuously sent to destination D. D will then send the Reply Packet to S. The Reply Packet will form through all the intermediate nodes a route corresponding to an item in the routing table, and each intermediate node can also obtain the new routing information from the Reply Packet. If no intermediate node has such a route to D, then the Route Request Packet will be broadcasted to D, and D will return a Reply Packet to S. When S receives the Reply Packet, the route is formed and data communication can begin.

3.7 Route Repairing

When one intermediate node finds that its next hop of one route item in the active table has disappeared, on one hand, it will update the routing information table and notify the previous hop; on the other hand, it will repair the broken route, since repairing a broken route is easier than finding a totally new route. In an ad hoc network, normally, not all the nodes are moved at the same time, so often we can repair the broken route efficiently. Note, the network topology is supposed to be unchanged during the route repairing process.

Following is the process. A node, say A, whose neighbor is moving will compare its SNFS with the moving neighbor. If A’s SNFS is smaller than that of the moving node by 1 and the corresponding route item in A’s active table is marked dormant, A will be the repairing node. If A’s SNFS is bigger than that of the moving node by 1 and the corresponding route item in A’s active table is marked dormant, A will be the connection node. For example, in the route S-11-12-13-14-15-16-17-18-D, if 14-15 is broken, then 14 will be the repairing node and 15 will be the connection node. If 11-12, 13-14 are broken, 11 will be the repairing node and 14 will be the connection node (Note, 12, 13 will delete the corresponding route item in their active tables instead of marking them dormant, as we have discussed above). The repairing node will broadcast the repairing message with its route identity S, D and SNFS as well as the address list containing itself at first. When a node receives this message, it will check if it is
the connection node. In the active table of the connection node, there should also be one item marked dormant which has the same S and D as the repairing node. If the node fits, the route is repaired. After that, the connection node will transmit a wakeup message both to the source and to the destination. This is done by unicast. During this process, the dormant route is waken up and SNFS is updated in every node. The new route is also formed from the repairing node to connection node during this process.

If a node finds that it is not the connection node, it will check whether it has one item marked dormant which has the same S and D as the repairing node. If yes, it will discard the message. Otherwise, it will broadcast continuously, adding its address into the address list.

For example, in the route S-11-S-12-S-13-14-15-16-17-18-D, if S-12-S-13 are broken, then S-12 will broadcast a repairing message. When S-13 (the connection node) receives the message (S-12-S-13), it will send the wakeup message both to S and to D to wake up the dormant item, updating SNFS. When the wakeup message comes to the S, the new route S-11-S-12-S-13-14-15-16-17-18-D will be formed and data transmission can be restarted.

When the source node moves, it will realize that its next hop disappears and will activate the route repair process. When the destination node moves, the first previous hop of the destination node will realize and will activate the route repair process.

The repair process will first use the broadcast. In order to further decrease the broadcast overload, when a broadcast packet for repair is sent out, an effective number will put in the packet. When this packet passes one hop, the number will decrease by one. When the number is 0, the packet should be discarded. The reason we do this is because that the connection node is always nearby. By this method, we can decrease the broadcast overload further. If after a preset time, the repairing node does not receive the reply packets, it will increase the value of the effective number, and broadcast the repair packets again. If the value of the effective number exceeds the gate value and no reply packet comes to the repairing node, the repairing node will notify the source.

### 3.8 Route Optimization

Even if a route is repaired, it may not be a desired route, as shown in Figure 4. So we should conduct the optimization for it. Normally, if the repair process is because of the source node moving or the destination node moving, the bad route can be formed with high probability. If the repair process is because of the intermediate node moving, the bad route can be formed with lower probability. The optimization process is initiated by the source node. The source node will know the reason for every repair process because of the feedback in the repair process. There are two values set in the source node for one route. One is the gate value A. The other is B initially set to 0 when the route is first found. After that, when a repair process occurs, B will be added by the value X (or Y). X is the value set for the repair process because of source or destination node moving. Y is the value set for the repair process because of intermediate node moving. Normally X ≥ Y. If B ≥ A, then the source node will initiate an optimization process and B will be reset to 0. The values of A, X and Y are different according to the different network environments.

The optimization process is given below: The source node will send an optimization packet to the destination node. The packet will also carry a full address-list of this route. Every intermediate node of this route that receives the message will check whether there is a better route to each address of this list. If not, it transfers the packet according to the address list. If yes, it will change the address-list, and send the updated address-list to the next hop according the new address-list. For example, the intermediate node, node I, receives the optimization message from its previous hop. The process that I makes the optimization is as follows: for each address in the address list, say f'. Node I will calculate the hop count between I and f' (directly from the address list). Suppose it is n1. Then I will check the p.f1 item and n.f1 item in its passive table to see if there is any f' in it. If so, I will calculate the hop count between I and f' from the passive table directly. Suppose it is n2. If n2 ≥ n1, no optimization will be needed. If yes, then node I will update the address-list in the optimization packet with a new route from the passive table. After all the addresses in the address-list are checked, node I will send the optimization packet to the next hop according to the updated address-list to continue the optimization process. This process will continue until the address-list reaches the destination node. During this process, the data transmission is going on along the unoptimized route. When the destination receives the optimization packet, it will send a packet carrying the optimized address-list along the optimized route. During this process, a new route will be formed in the active table. When this packet reaches the source node, the data transmission will be switched to the new route. The old route will be deleted at the same time by the source. Because the optimization will be based on unicast, the additional overhead for optimization is small.

### 3.9 Performance Analysis

In the ORSNR, no periodic transmission of the routing information is needed. The routing information in each MN is achieved during the route discovery process. The routing information table can be updated quickly and completely, as we have seen above. No incorrect routing information will exist in the routing information table. This will help us to speed up the route discovery process. The broadcast happens during the route discovery process. In the route repair process, broadcast will finish quickly due to the effective number of the broadcast packets. Most communication overhead will be based on the unicast. So the overhead for the ORSNR is small.

The route maintenance will be quicker because of the repair strategy. Based on the optimization, the better route
will be found. All these are based on our correct routing information table. For large ad hoc networks, the ORSNR will perform much better. In this case, the length of route is always long. Finding a node nearby will be much quicker than discovering a totally new route. With the optimization, better route can be found. The optimization process will never affect the current data transmission. During the optimization process, the data transmission is along the unoptimized route. After the optimization, the data transmission will be on the optimized route. No delay will be caused for data transmission during the optimization. When the repair is in process, the data transmission will be delayed, but the delay is short because the connection node is nearby.

4. Conclusion and Future Work

The ORSNR, which adopts a distance vector form for the information representation, can deliver an efficient routing since it repairs the broken route with optimization and minimizes the broadcast overload. It can also gather the routing information in the network with less communication overhead, and update the table quickly by using a special information structure. However, there is considerable work left for future research. Efficient optimization algorithms and clustering schemes for a large ad hoc network are worth exploring.

(This research was supported by the Texas Advanced Technology Program Grant under Grant No. 97-P-087.)

Reference

**FIG. 1. USING ADDRESS LIST TO FORM PASSIVE TABLE**

Address-list: N.h

Passive table:

<table>
<thead>
<tr>
<th>S</th>
<th>D</th>
<th>SNFS</th>
<th>P.h</th>
<th>p-a-l</th>
<th>N.h</th>
<th>n-a-l</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
<td>3</td>
<td>12</td>
<td>11</td>
<td>S</td>
<td>14</td>
</tr>
</tbody>
</table>

P.h: Previous hop  
N.h: Next hop

**FIG. 2. PROCESSING THE PASSIVE TABLE**

<table>
<thead>
<tr>
<th>S</th>
<th>D</th>
<th>SNFS</th>
<th>P.h</th>
<th>p-a-l</th>
<th>N.h</th>
<th>n-a-l</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
<td>3</td>
<td>12</td>
<td>11</td>
<td>S</td>
<td>14</td>
</tr>
</tbody>
</table>

**FIG. 3. DELETING ITEMS IN THE PASSIVE TABLE**

<table>
<thead>
<tr>
<th>S</th>
<th>D</th>
<th>SNFS</th>
<th>P.h</th>
<th>p-a-l</th>
<th>N.h</th>
<th>n-a-l</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>D</td>
<td>3</td>
<td>12</td>
<td>11</td>
<td>S</td>
<td>14</td>
</tr>
</tbody>
</table>

Because 3<5<3+1+1, delete 15
Because 3<5<3+2+1, delete 16
Because 3<5<3+3+1, delete 17
Because 3<5<3+4+1, delete 18
Because 3<5<3+5+1, delete D

**FIG. 4. ROUTE OPTIMIZATION DIAGRAM**

Original route  
Optimized route  
Repair route  
Moving direction

L_P: the count from previous hop  
L_N: the count from next hop