Route Maintenance in a Wireless Mobile Ad Hoc Network [*]

Shih-Lin Wu, Sze-Yao Ni, Yu-Chee Tseng, and Jang-Ping Sheu
Department of Computer Science and Information Engineering
National Central University
Chung-Li, 32054, Taiwan
E-mail: {ken, nee, yctseng, sheujp}@csie.ncu.edu.tw

Abstract

A mobile ad-hoc network (MANET) is formed by a cluster of mobile hosts, each installed with a wireless transceiver, without the assistance of base stations. Due to the transmission range constraint of transceivers, two mobile hosts can communicate with each other either directly, if they are close enough, or indirectly, by having other mobile hosts relay their packets. Several routing protocols, such as DSR, SSA, AODV, and ZRP, have been proposed for a MANET with a dynamically changing topology. In a MANET, a route may suddenly become broken because only one host roams away. Even if a route remains connected, it may become worse due to host mobility or a better route newly being formed in the system. Existing protocols, however, will stick with a fixed route between a source-destination pair once it is discovered, until it is expired or broken. In this paper, we show how to enhance several existing protocols with route optimization and local route recovery capability, such that the routing paths can be adjusted on-the-fly while they are still being used for delivering packets or can be patched in minimum wireless bandwidth and packet transmitting delay while route errors occur.

Keywords: mobile ad hoc network (MANET), mobile computing, routing, route recovery, wireless network.

1. Introduction

Mobility has become a new issue on today's computing systems. The maturity of wireless transmissions and the popularity of portable computing devices have made the dream of "communication anytime and anywhere" possible. Users can move around, while at the same time still remaining connected with the rest of the world. Many wireless communication products are available commercially, such as WaveLAN by Lucent, AIRLAN by Solectek, BreezeNET by BreezeCOM, RangeLAN and RangeLINK by Proxim, AirLink Bridge by Cylink, ARDIS, CDPD [4, 21], DECT [10], and GSM [8, 13]. Small, light-weight, economic hand-held *mobile hosts*, such laptop PCs, palmtop PCs, and PDAs, are also widespread. *Mobile computing* (or *nomadic computing*) has received intensive attention recently [2, 3, 12, 20, 23, 24].

One wireless network configuration that has received a lot of attention recently is the *mobile ad hoc network* (MANET) [1]. A MANET consists of a set of mobile hosts operating without the assistance of base stations. Mobile hosts must communicate with each other either directly or indirectly by relaying by intermediate mobile hosts. The applications of MANETs appear in places where infrastructure networks are difficult to build (e.g. fleets in ocean, armies in march, and battle fields) or unavailable (e.g., convention centers, festival field grounds, or natural disasters where wired networks are down).

Routing protocols for a MANET can be classified as *proactive* and *reactive*, depending on how they react to topology changes [11]. A host running a proactive protocol will propagate routing-related information to its neighbors whenever a change on its link state is detected. The information may trigger other mobile hosts to re-compute their routing tables and further propagate more routing-related information. The amount of information propagated each time is normally proportional to the scale of the MANET. Examples of proactive protocols include RIP (or distributed Bellman-Ford, DBF) [9], OSPF [16], and Destination Sequenced Distance Vector (DSDV) [19]. Observing that a proactive protocol may pay costs to constructing routes even if mobile hosts do not have such need, thus wasting the limited wireless bandwidth, many researchers have proposed to use reactive-style protocols, where routes are only constructed on-demand. Many reactive protocols, such as Dynamic Source Routing (DSR) [14], Signal Stability-based Adaptive Routing(SSA) [7], and Ad Hoc On Demand

[*] This work is supported by the National Science Council of the Republic of China under Grants #NSC88-2213-E-008-023, #NSC88-2213-E-008-024 and #NSC88-2213-E-008-025.

Distance Vector Routing (AODV) [18], have been proposed based on such on-demand philosophy. Recently, a hybrid of these two approaches, called the Zone Routing Protocol (ZRP) [11], is also proposed.

Routing in a reactive protocol typically consists of three parts: *route discovery*, *data forwarding*, and *route maintenance*. This paper studies the route maintenance problem in a MANET. When a mobile host wants to communicate with another host, it first tries to discover a good route to the destination, on which the data packets are forwarded.

Route maintenance, by its name, should address the problem when a route becomes worse or even broken due to host mobility. However, in existing protocols, such as [7, 11, 14, 18], a sending host will stick with the discovered route until it is expired or broken, even if some better routes are newly being formed in the system. One straightforward solution is to run the route discovery procedure more frequently to detect such possibility. However, this is very costly as route discovery will typically activate a network flooding [17]. This observation has motivated the first work in this paper: we propose to use *route optimization* to refine or improve the routes *on-the-fly* while they are being used for transmission. Not only can the data packets be sent with less hops and latencies, but also may the chances of route breakage be reduced, lowing the number of times the costly route discovery process being called.

Another issue is to rebuild broken routes. In existing protocols, whenever a node finds that its link to the next hop is broken, it will send a route error packet back to the source node, which will then invoke another route discovery procedure (which, as stated earlier, involves costly flooding). These actions will result in waste of scarce wireless bandwidth as well as long delay. Several recent works have targeted in the similar direction to reduce flooding packets [5, 15, 17]. Therefore, it is important that a protocol will use route discovery with care. Moreover, many real-time applications may not tolerate such long delay. Further, these problems will be worsened when mobility is high (more frequent route error/discovery packets) or the network is large (longer way for route error/discovery packets to travel). We observe that it is highly possible that a route is broken because only one relay node leaves its neighbors. This is very similar to the “spatial locality” discussed in [5], in which it is proposed to use prior routes to rebuild new routes. This observation has motivated the second work in this paper: we will first employ a *local route recovery* to patch a broken route before a route error packet is sent back to its source node.

Another motivation for local route recovery is: when we look at a destination node which has multiple connections going into it, these connections are likely to form a star graph center at itself. Thus, the closer a route is to this node, the more likely the route can be rebuilt locally.

To achieve the above goals (route optimization and local route recovery), mobile hosts must have more up-to-date route information. Since wireless transmissions are inherently broadcast, we try to enable mobile hosts to take full advantage of their *promiscuous receiving* capability by overhearing all surrounding packets. Thus, “free” route-related information may be retrieved without sending new packets, making our approach very attractive even in large and highly mobile networks.

In this paper, we show how to enhance several existing protocols with route optimization and local route recovery capabilities. We confine our work to one as an enhancement to the original protocols. Hence there will be no change on the original protocols’ behavior. For instance, in an “on-demand (or reactive)” routing protocol, this assumption should not be violated when conducting route optimization and local route recovery. Thus, although the term “optimization” is used, it by no means implies that an optimal route will always be found. Instead, better routes are formed in a best-effort manner.

The current implementation status at the High-Speed Communication and Computing Lab., National Central University, will also be reported. One of the results we observed from this implementation is that the length of a route can significantly affect the bandwidth that can be offered by the route. This further justifies the importance of route optimization proposed in this paper.

The rest of this paper is organized as follows. Section 2 reviews four existing routing protocols for a MANET. The proposed enhancements (route optimization and local route recovery) are in Section 3. Our implementation experience is presented in Section 4. Conclusions are drawn in Section 5.

2. Review of Some MANET Routing Protocols

Routing protocols for MANET can be categorized according to how they react to the link state changes: *proactive* or *reactive*. In a proactive protocol, a mobile host will broadcast its link state information whenever a change on such state is detected. A host, on receiving such information, may re-broadcast such change based on the received information and its own link state. The amount of link state information is normally proportional to the scale of the MANET. Examples of proactive protocols include RIP [9], OSPF [16], and DSDV [19].

On the other hand, a reactive protocol only tries to construct a route on demand. Several studies have shown that such approach is more efficient because routes are constructed when necessary [7, 14]. A reactive protocol typically consists of three components:

- *Route Discovery:* describes how to request for routes and respond to such requests.

- *Data Forwarding:* describes how packets are delivered to their destinations, such as the format of data packets and routing tables.

- *Route Maintenance:* explains how route problems (such as link breakage) are reported and recovered.

Below, we review four routing protocols for MANET. In Section 3 we will show how to enhance these protocols with route optimization and local route recovery capabilities.

2.1. DSR: Dynamic Source Routing Protocol

The Dynamic Source Routing (DSR) [14] is derived based on the concept of *source routing* (refer to Chapter 7 in [6]). Each data packet specifies in its header the whole route to be traversed. A node, on receiving a data packet, only needs to forward the packet to the next node on the route. The advantage is that the intermediate hosts do not need to maintain any routing information locally. However, the overhead is on the longer packet headers, which may traverse several hops.

The DSR is a reactive protocol. Its operations are summarized in the following.

A) Route Discovery: On a source node needing a route to a destination node, it broadcasts a route request (ROUTE_REQ, as shown in Fig. 1) packet containing the address of the destination node. On a node receiving this request, two cases may happen. If it does not know a route to the destination, it appends its own address to the packet and propagates the ROUTE_REQ packet to its neighbors. Thus, paths leading to the destination can be tracked by ROUTE_REQ packets. Loops can also be avoided. When a ROUTE_REQ is received by the destination, it returns to the source node a route reply (ROUTE_REPLY as shown in Fig. 2) packet containing the route indicated in the ROUTE_REQ. The ROUTE_REPLY then travels, through unicast, in the reverse direction of the discovered route or a path already known by the destination to the source. The source node, on receiving the ROUTE_REPLY, will place the route in its route cache.

In addition to the destination node, an intermediate node can also return ROUTE_REPLY if it already knows a route fresh enough in its route cache. In this case, it concatenates the route in ROUTE_REQ and that in its route cache, and supplies this new route to the source. Also note that an intermediate node should register the ROUTE_REQ it has received to discard duplicate ROUTE_REQ's.

B) Data Forwarding: To send a data packet, a source node should specify the complete route to be traveled by the packet. Each intermediate node, on receiving the data packet, should look at the route and forward the packet to the next node. The format of data packets is shown in Fig. 3.

<table>
<tr><td colspan="4">Type=REQ</td><td colspan="4">Option Length</td><td colspan="4">Idetification</td></tr>
<tr><td colspan="12">Target Address</td></tr>
<tr><td colspan="3">index1</td><td colspan="3">index2</td><td colspan="3">index3</td><td colspan="3">index4</td></tr>
<tr><td colspan="12">Address1</td></tr>
<tr><td colspan="12">Address2</td></tr>
<tr><td colspan="12">Address3</td></tr>
<tr><td colspan="12">Address4</td></tr>
</table>

Figure 1. The ROUTE_REQ packet used in DSR. The *option length* field specifies the packet length after itself, the *identification* field is the sequence number, and the *index* and *address* fields define the path that has been tracked so far. Note that the index and address fields can be expanded at the end of the packet, if necessary.

<table>
<tr><td colspan="5">Type=REPLY</td><td colspan="3">Option Length</td><td colspan="2">R</td><td colspan="2">F</td><td colspan="4">Reserved</td></tr>
<tr><td colspan="16">Target Address</td></tr>
<tr><td colspan="4">Index1</td><td colspan="4">Index2</td><td colspan="4">Index3</td><td colspan="4">Index4</td></tr>
<tr><td colspan="16">Address1</td></tr>
<tr><td colspan="16">Address2</td></tr>
<tr><td colspan="16">Address3</td></tr>
<tr><td colspan="16">Address4</td></tr>
</table>

Figure 2. The ROUTE_REPLY packet used in DSR.

<table>
<tr><td colspan="2">R</td><td colspan="8">Option Length</td><td colspan="6">Idetification</td></tr>
<tr><td colspan="4">index1</td><td colspan="4">index2</td><td colspan="4">index3</td><td colspan="4">index4</td></tr>
<tr><td colspan="16">Address1</td></tr>
<tr><td colspan="16">Address2</td></tr>
<tr><td colspan="16">Address3</td></tr>
<tr><td colspan="16">Address4</td></tr>
</table>

Figure 3. Header of data packets in DSR. If necessary, the index and address fields can be duplicated for longer routes.

Type=ERROR	Option Length	Index
Originator Address		
From Hop Address		
Next Hop Address		

Figure 4. The ERROR packet used in DSR. The *originator address* **field indicates the source of the broken route, and the** *from hop* **and** *next hop* **identify the two end nodes of the broken link.**

Host	Signal Strength	Last	Clicks	Set
c	S	10:33	7	SC
g	W	10:26	5	WC

Figure 5. The signal stability table of SSA. Each row is for one link. The *signal strength* **and the** *last* **fields indicate the signal strength and the time, respectively, that the last beacon was received. The** *clicks* **field registers the number of beacons that have been received continuously. Each link is classified as SC (strongly connected) or WC (weakly connected) in the** *set* **field, according to the last few clicks received.**

C) Route Maintenance: When an intermediate node forwards a data packet to the next node, the former node should snoop at the latter's traffic for some pre-defined time. If the former hears no transmission from the latter, it assumes the link to the next node is broken, in which case it will send an error packet (Fig. 4) to the source node. On knowing such event, the source will invoke the route discovery process to construct a new route.

2.2. SSA: Signal Stability Adaptive Routing Protocol

The Signal Stability Adaptive protocol (SSA) [7] tries to discover longer-lived routes based on signal strength and location stability. Each link is differentiated as *strong* and *weak* according to the average signal strength at which packets are heard. The location stability criteria further biases the protocol toward choosing a path which has existed for a longer period of time. Beacons are sent periodically by each host for its neighbors to measure these criteria. Each host maintains a *signal stability table* as shown in Fig. 5.

Unlike DSR, which uses source routing, SSA follows the *next-hop* routing. Each host keeps a routing table which indicates the next host leading to each destination known to it (refer to Fig. 6). Below, we summarize the SSA protocol.

Destination	Next Hop
t	o
m	x

Figure 6. The routing table of SSA, which uses next-hop routing.

DA	SA	SEQ	TTL	TYPE	PREF	LEN	CRC	Hop List
								Data

Figure 7. Packets used in SSA. According to the TYPE field, the same format can be used by data, ROUTE_REQ, and ROUTE_REPLY packets.

A) Route Discovery: On needing a route, a source node broadcasts a ROUTE_REQ packet as shown in Fig. 7. The source can also specify in the PREF field the quality of the route it desires: STRONG_LINK_ONLY, STRONG_PREFERRED, or NO_PREFERENCE. It is suggested that the STRONG_LINK_ONLY option be used in the first attempt. A receiving node should help propagating the request if (1) the ROUTE_REQ is received over a strong link, and (2) the request has not been forwarded previously. Like DSR, the path traversed by ROUTE_REQ is appended at the packet (in the Hop List field). The propagation stops when the destination is reached or a node having a non-stale route to the destination is reached, on which event a ROUTE_REPLY will be sent. If multiple ROUTE_REPLYs are received by the source, it can choose the one with the best quality to use.

The ROUTE_REPLY should travel on the reverse direction of the ROUTE_REQ. On its way back, each intermediate node can set up the next hop leading to the destination. Besides, there are also some "gratuitous" routes that can be added to the routing table. Specifically, if the discovered route is $a \rightarrow \cdots b \rightarrow \cdots \rightarrow d$, then host b can learn a route to each node in its downstream.

If the source fails to receive a ROUTE_REPLY before a timeout period, it can send another ROUTE_REQ with other PREF options (such as STRONG_PREFERRED and NO_PREFERENCE) to find a weaker route.

B) Data Forwarding: Since the next-hop routing is used, each data packet only indicates its destination node in the DA field. A host simply looks at its routing table to determine the next host where packets should be forwarded to.

C) Route Maintenance: A link may become broken due

Type	Reserved	Hop Count
Broadcast ID		
Destination IP address		
Destination Sequence Number		
Source IP address		
Source Sequence Number		

Figure 8. The ROUTE_REQ packet used in AODV.

to host mobility. When a link is broken, to reflect this fact, mobile hosts in both sides of the link will remove entries in their routing tables that use this link. When a data packet arrives at a node lack of a route to the destination, a notice will be sent to the source node, which will then invoke another route discovery. An ERASE packet may be needed to invalidate stale route entries in each intermediate node.

2.3. AODV: Ad Hoc On Demand Vector Routing Protocol

The main purpose of the AODV protocol [18] is to avoid the "counting to infinity" problem associated with the Bellman-Ford algorithm by offering quick convergence when the MANET topology changes. This is done by using a *destination sequence number* associated with each route entry. Using the number ensures loop freedom. Given the choice of multiple routes to a destination, a source node always selects the one with the greatest sequence number.

The AODV protocol is summarized below.

A) Route Discovery: A node broadcasts a ROUTE_REQ when it determines that it needs a route to a destination but does not have one available. A destination sequence number is associated with the packet. The number is used to compare the freshness between routes. The destination node or a node with a route of a destination sequence number no less than the sequence number in the packet, can reply the request using a ROUTE_REPLY. The formats of these packets are shown in Fig. 8 and Fig. 9.

B) Data Forwarding: AODV also follows the next-hop routing style. The procedure is similar to the SSA protocol.

C) Route Maintenance: Each nodes will broadcast a HELLO packet periodically. Through such packets a node knows its neighbor nodes. On a node finding a link becoming broken, it will send a ROUTE_REPLY with an infinite metric traveling along the reverse direction of each route that uses the broken link to invalidate the route. At the same time the destination sequence associated with the route is also incremented and sent together with the ROUTE_REPLY. This number will be used by the source node to request for a new route.

Type	L	Reserved	Hop Count
Destination IP address			
Destination Sequence Number			
Lifetime			

Figure 9. The ROUTE_REPLY packet used in AODV.

2.4. ZRP: Zone Routing Protocol

The Zone Routing Protocol (ZRP) [11] is a hybrid of proactive and reactive scheme. From each node, the set of nodes within a pre-defined r hops is called a *zone*. Thus, the network has a number of zones equal to its size. Routing inside a zone will follow the proactive style, while routing across zones will follow the reactive style.

The ZRP is summarized below.

A) Route Discovery: In intra-zone routing, it is suggested to use the proactive DSDV protocol [19]. Whenever a node's link state is changed, a notice will be sent as far as r hops away (i.e., the zone of this node). Hence, a node always knows how to reach another node in the same zone. This also limits the number of updates triggered by a link state change.

On the other hand, for inter-zone routing, it is suggested to use a modified DSR protocol as follows. When a node needs a route to a node outside its zone, it performs a *bordercasting* by sending a ROUTE_REQ to each node on the "border" of this zone. On receiving such a packet at a border node, it first checks its intra-zone routing table for existence of a route to the requested destination node. If so, a ROUTE_REPLY can be sent; otherwise, it performs another bordercasting in its zone. This is repeated until a route is found.

B) Data Forwarding: A modified source routing is used for inter-zone routing. A routing path only contains the border nodes that have to be traversed. Once a data packet reaches a border node whose zone contains the destination, its intra-zone routing table (which follows next-hop routing) will be used to forward the packet.

C) Route Maintenance: Whenever a link is broken or newly formed, the information will be propagated with a hop limit r. If a route failure is detected, the same operation as in DSR is used to inform the source node.

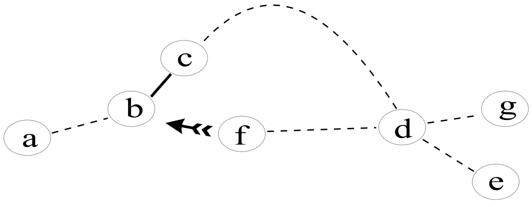

Figure 10. An example of route optimization in DSR.

3. The Enhancements

3.1. Route Optimization

To help with route optimization, mobile hosts should collect as much fresh information as possible with the least cost. Wireless transmission is broadcast in nature. Also, one property of wireless transmission is that only one pair communication can exist in an area. That is, all other nodes covered by this transmission must stay idle. In our proposal, these idle nodes should configure their network interfaces in the promiscuous receive mode to collect fresh routing information. Packets that may carry routing information include data packets (such as in DSR), ROUTE_REQ, and ROUTE_REPLY.

Below, we show how to enhance DSR, SSA, AODV, and ZRP with route optimization capability.

3.1.1. Route Optimization for DSR

In DSR, since source routing is used, the routing paths are only kept two places: those source nodes that are currently active in sending messages, and data packet headers. As a result, route optimization should be achieved based on information therein.

As the MANET topology changes, it is possible for a source node to find out a better route for another source. For instance, consider the scenario in Fig. 10. Suppose there is a route from a to e: $a \to \cdots \to b \to c \to \cdots \to d \to \cdots \to e$. Under a promiscuous mode, node f may hear the packets from b to c. If f has a route $f \to \cdots \to d \to \cdots \to g$. If the hop count from c to d is longer than f to d, then f can suggest a better route $a \to \cdots \to b \to f \to \cdots \to d \to \cdots \to e$ to the source node a.

The route optimization protocol is formally developed below. It is executed by any node f receiving a data packet from node b to node c such that $c \neq f$.

1. Let the route in the packet header be $P = a \to \cdots \to b \to c \to \cdots \to e$.

2. Let $Tmp = P$.

3.
   ```
   for (each path $P'$ in $f$'s routing table) do
       for (each $d \in P'$ such that $d$ is a downstream node of $c$) do in $P$
           Let $P''$ be the path obtained from $P$ by replacing the subpath from $c$ to $d$ by the path from $f$ to $d$ in $P'$.
           If the length of $P''$ is less than $Tmp$, then let $Tmp = P''$.
       end for;
   end for;
   ```

4. If $Tmp \neq P$, then send a ROUTE_REPLY packet to the source node a with Tmp as the suggested new route.

5. When a receives the ROUTE_REPLY, it replaces the entry P in its routing table for destination node e by Tmp.

Destination	Next Hop	Hop Count	Route Quality
t	o	5	SC
m	x	3	WC

Figure 11. The modified routing table used in SSA. The modified part is shown in gray.

3.1.2. Route Optimization for SSA

There are two directions to optimize routes in SSA: (i) to find a route of a less hop count, and (ii) to find a route of higher quality (such as from SC to WC). To achieve these goals, we modify the routing table used in the SSA to one as shown in Fig. 11. The *hop-count* field is the length to the corresponding destination. The *route quality* field indicates the quality of the route. These two fields can be filled when the ROUTE_REPLY packet returns from the destination to the source, using the hop_list and PREF fields.

Also, to help finding a shorter route, each data packet must indicate the remaining hops that it has to traverse. Thus, we modify the data packet format as shown in Fig. 12, by adding a hop_count field. Through this information, it is possible for other nodes to tell if they can suggest shorter routes or not.

For instance, suppose there are data packets being transmitted along the path: $a \to \cdots \to b \to c \cdots \to d$ (refer to Fig. 13). When a node, say f, hears a data packet sent by b and it has a better route from f to d, node f can send a packet to recommend node b to forward its data packet destined for d to it instead of to node c. The protocol is

DA	SA	SEQ	TTL	TYPE	PREF	LEN	CRC	Hop Count	Hop List
									Data

Figure 12. The new format of data packet used in SSA.

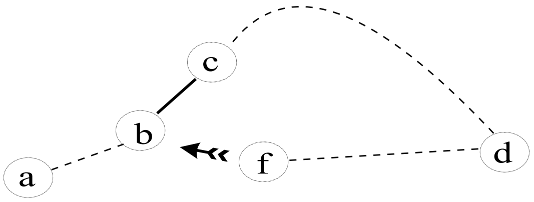

Figure 13. An example of route optimization in protocols using next-hop routing such as SSA and AODV.

formally developed as follows. It is executed by any node f when receiving a data packet from node b to node c such that $c \neq f$.

1. Retrieve the hop count (say i) and the PREF (say s) from the packet header.

2. Let i' and s' be the hop count and route quality, respectively, to node d recorded in f's routing table, if any.

3. **if** $(i > i' + 1) \wedge (s \leq s')$ **then**

    if ($s \leq$ the signal strength from b to f) **then**

    Broadcast a packet ROUTE_REPLY with hop_limit=1 to indicate that "node f has a route of length i' and with quality s' to node d."

    end if;

   end if;

4. Any node (including node b), on receiving the ROUTE_REPLY packet, updates its routing table for destination d (including hop-count and route quality) if the route is better than its current one. Note that to follow the on-demand route discovery notion, only those routes that are currently active are updated. If there is any update on its routing table, the node should in turn broadcast anohter ROUTE_REPLY to indicate that it has a route of length $i' + 1$ and quality s' to node d.

3.1.3. Route Optimization for AODV

The AODV protocol also follows the next-hop routing style. So the route optimization is similar to the SSA protocol. Both the routing table and the data packet need to have a hop-count field. The only difference is that the AODV does not have the link quality information, but it has a destination-sequence-number field. A larger sequence number means a fresher route, so this may also mean a route of less chance being broken.

The protocol shown below is a slight modification of the one for SSA.

1. On f receiving the data packet from b destined for node d, it will retrieve the hop count (say i) and the destination sequence number (say s) from the packet header.

2. Let i' and s' be the hop count and the sequence number for node d recorded in f's routing table.

3. **if** $(i > i' + 1) \wedge (s \geq s')$ then

    Broadcast a packet ROUTE_REPLY to indicate that "node f has a route of length i' and with destination sequence number s' to node d."

   end if;

4. Any node (including node b), on receiving the ROUTE_REPLY packet, updates its routing table for destination d (including hop-count and destination sequence number) if the route is better than its current one. Note that to follow the on-demand route discovery notion, only those routes that are currently active are updated. If there is any update on its routing table, the node should in turn broadcast anohter ROUTE_REPLY to indicate that it has a route of length $i' + 1$ and destination sequence number s' to node d.

3.1.4. Route Optimization for ZRP

In ZRP, a node always knows the best route to any node in its local zone, so no route optimization is needed for intrazone routing. On the inter-zone part, a modified DSR protocol is used.

For instance, consider a MANET using ZRP with radius = 2. Suppose there is a route from a to e: $a \rightarrow \cdots \rightarrow b \rightarrow c \rightarrow \cdots \rightarrow d \rightarrow \cdots \rightarrow e$ (refer to Fig. 14). Note that only border nodes are registered in a route, so in Fig. 14 node c is a border node of b's zone. Suppose there is another route $f \rightarrow i \rightarrow \cdots \rightarrow d \rightarrow \cdots \rightarrow g$, and the border count from f to d is less than that from c to d. If f hears the packets from b to e, f can figure out a better route $a \rightarrow \cdots \rightarrow b \rightarrow f \rightarrow i \rightarrow \cdots \rightarrow d \rightarrow \cdots \rightarrow e$ and recommend the route to the source a.

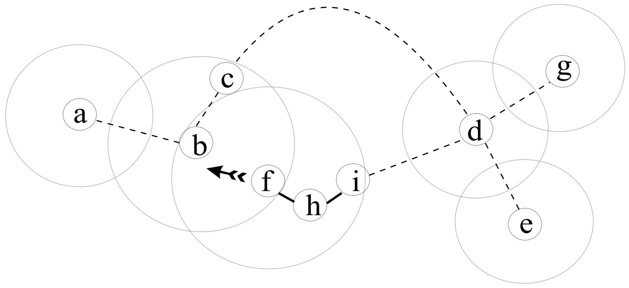

Figure 14. A route optimization example for the ZRP protocol.

Although the new route known by a can successfully deliver data packets to the destination e, the route has violated the original ZRP protocol's definition because node f is not a border node of node b (the hop count is 1, which may be less than the radius). This can be resolved as follows. Node a still transmits data packets using the new route recommended by f. When f receives a data packet from b with itself as the border, it will find out from its intra-zone routing table that h should be a border node of b leading to i. So f can replace itself by h in the packet header and forward the data packet to h (now the new route will become $a \to \cdots \to b \to h \to i \to \cdots \to d \to \cdots \to e$). Then intra-zone routing will be used to forward the data packet to h. On h receiving the data packet, the similar scenario will be discovered by h, who will further modify the route in the data packet. This will keep on going until the packet arrives at e. Now the route already follows the ZRP style, so node e will send a ROUTE_REPLY with this modified route to a.

We now formally present the protocol. The protocol is executed by any node f receiving a data packet from b to c destined to e such that $c \neq f$.

1. Let the route in the packet header be $P = a \to \cdots \to b \to c \to \cdots \to d \to \cdots \to e$.
2. Let $Tmp = P$.
3. **for** (each path P' in f's inter-zone routing table) **do**

```
    for (each $d \in P'$ such that $d$ is a downstream node of $f$ in $P$) do
        Let $P''$ be the path obtained from $P$ by replacing the subpath from $c$ to $d$ by the path from $f$ to $d$ in $P'$.
        If the length of $P''$ is less than $Tmp$, then let $Tmp = P''$.
    end for;
end for;
```

4. If $Tmp \neq P$, then send a ROUTE_REPLY packet to the source node a with Tmp as the suggested new route.
5. When a receives the ROUTE_REPLY, it replaces the entry P by Tmp in its inter-zone routing table for destination node e.
6. The following operations should be added to the data forwarding part when any node f receives a data packet from node b.

```
    if ($f$ is not a destination node and $f$ is not a border node of $b$) then
        Find out from the route in the data packet the next border node, say $i$.
        Compute from $f$'s intra-zone routing table the path from $f$ to $i$ (let the path be $P'$).
        Compute the border node of $b$ on the path $P'$ (let the result be $h$).
        Replace the entry $f$ in the route of the data packet by $h$.
        Forward the data packet to node $h$.
    endif;
```

3.2. Local Route Recovery

When a route is found to be broken, it is sometimes desirable to remedy the problem as soon as possible with the least cost (in terms of both time and bandwidth). So we propose to perform a local route recovery before the problem is reported to the source. Consider Fig. 15(a), which contains a route from a to f: $a \to \cdots \to b \to c \to e \cdots \to f$. If host c moves out the transmission range of host b as in Fig. 15(b), the route will become broken. In fact, it is very likely that a host nearby the c's original location, such as host d in the figure, can serve as the relay node to pave the gap.

To resolve the problem, we can let host b, on finding its connection to c becoming broken, broadcast a "local" ROUTE_REQ packet with a small hop limit, in hope of rebuilding the route with little effort. Then d, g, or e can send a normal ROUTE_REPLY packet to b to rebuild the route. We believe that this problem possesses a "location locality", so only a very small hop limit (such as 2 or 3) will be sufficient. Also, the initiator of ROUTE_REQ should set up a timer, so that if the broken route can not be rebuilt within the timeout period, it can send a normal route error packet to the source node so that a "global" ROUTE_REQ packet can be sent.

Next, we comment on the hop limit. Suppose a hop limit of 2 is used for local route recovery. It is still possible to discovery a partial path of 3 hops. Fig. 16 shows

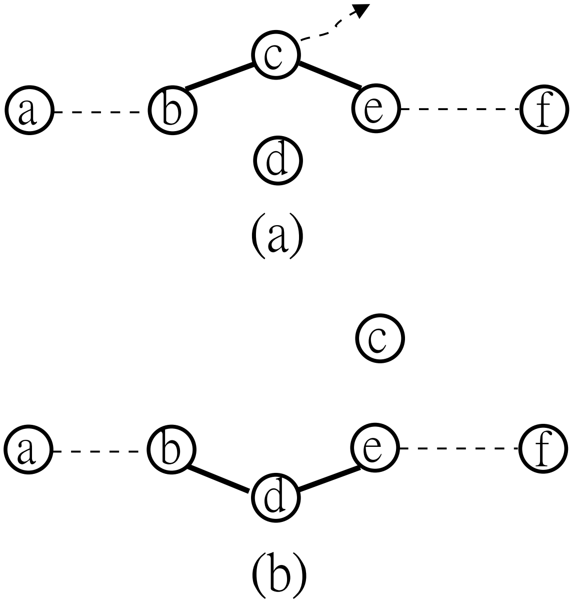

Figure 15. An example of local route recovery.

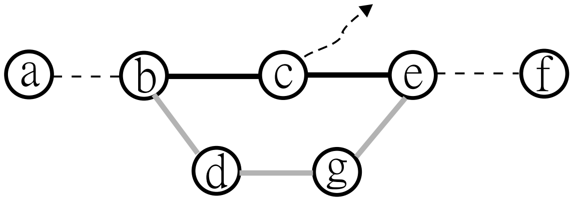

Figure 16. The recovery of least hops of one node leave when a active route is broken.

such an example, where the gray route $b \rightarrow d \rightarrow g \rightarrow e$ is the one to remedy the broken route due to c's leaving. The reason is that host g will help broadcasting the local ROUTE_REQ (it is at 2 hops from b). So e has a chance to reply a ROUTE_REPLY. Similarly, if a hop limit a 3 is used, then a partial path of 4 hops can be built.

Our proposal here is similar to, but of a different intention from, the ABR protocol [22]. In ABR, if a host x finds that its connection to the next host is broken, two cases could happen. If x is located at the first half of the route (i.e., it is nearer to the source than the destination), then a route error is reported to the source. Otherwise, it will broadcast a ROUTE_REQ with a hop limit equal to the remaining number of hops that it was supposed to travel before the route is broken. If this succeeds, this route is remedied and no route error will be reported. Otherwise, a route error will be reported to the host preceding x, which will in turn repeat trying the above two cases again. This is recursively repeated until either the broken route is remedied or one host at the first half of the original route is reached. As can be seen, this approach may take more bandwidth and longer delay if the above recursion kept on failing. In our proposal, we only try to remedy the broken route locally with the least efforts. So it is insensitive to the route length and scalable to the network size.

Table 1. The observed delays and bandwidths at different hop counts.

	hop = 1	hop = 2	hop = 3
Delay (sec)	0.00385	0.00772	0.01076
Bandwidth (Mbyte/sec)	0.16191	0.087512	0.048221

4. Implementation Results and Observations

We have implemented a next-hop routing protocol on top of the Linux operating system. The platform has four notebooks (Pentium 166MMX, Pentium 233MMX, Pentium II 333 and Pentium II 350), each equipped with a Lucent WaveLAN wireless card conformed to the IEEE 802.11 MAC protocol operating at the 2.4 GHz band. The highest transmission rate is claimed to be 2 Mbit/sec.

The system design is divided into two parts: one in Linux kernel to deal with queue management and route error, and one as daemons to account for data packets transmission and routing table establishment and maintenance. The system already can operate correctly with TCP/UDP-based programs. With this platform, we first try to observe the effect of hop count. Three numbers of hop counts, 1, 2, and 3, were tested. Our first experiment was to use the *ping* command to determine the delay when a route does not exist. The result is in the second row of Table 1. As can be seen, the delay is quite small. But the time needed does increase linearly with respect to the hop count. Our second experiment was to use the *ftp* (using binary mode) to determine the communication bandwidth at different hop counts. The result is in the third row of Table 1. One interesting observation is that the bandwidth degrades by half when the hop count changes form 1 to 2. The bandwidth further degrades when the hop count changes from 2 to 3. This does justify the need of performing route optimization in a MANET — not only the end-to-end bandwidth can be increased, but also the level of contention on medium can be reduced.

5. Conclusions

In this paper, we have shown how to perform route optimization for four routing protocols for MANET, namely DSR, SSA, AODV, and ZRP. The original protocols will use a fixed route between a pair of node to deliver data packets, until it is broken. We show how to enhance these protocols with route optimization such that better routes can be formed on-the-fly while the original route is being used for transmission. So data packets will not experience delays because of the route optimization. We have also proposed to use local route recovery to patch broken routes, so the costly route discovery will be executed less frequently.

References

[1] IETF MANET Working Group. http://www.ietf.org/html.charters/manet-charter.html.

[2] G. D. Abowd *et al.* Cyberguide: A Mobile Context-Aware Tour Guide. *ACM/Baltzer Wireless Networks*, 3(5):421–433, 1997.

[3] A. Archarys and B. R. Badrinath. A Framework for Delivering Multicast Messages in Networks with Mobile Hosts. *ACM/Baltzer J. of Mobile Networks and Applications*, 1(2):199–219, 1996.

[4] K. Budka. Cellular digital Packet Data: Channel Availability. In *Proc. IEEE Personal Indoor and Mobile Radio Communications(PIMRC'95).*, Sept. 1995.

[5] R. Castaneda and S. R. Das. Query Localization Techniques for On-Demand Routing Protocols in Ad Hoc Networks. In *Proc. ACM MOBICOM '99.*, Aug. 1999.

[6] D. E. Comer. *Internetworking with TCP/IP, Volume 1: Principles, protocols, and architecture*. Prentice Hall, Englewood Cliffs, N.J., 1991.

[7] R. Dube, C. Rais, K. Wang, and S. Tripathi. Signal stability based adaptive routing for ad–hoc mobile networks. In *IEEE Personal Communications*, Feb. 1997.

[8] J. Geier. *Wireless Networking Handbook*. New Riders Publishing, Indianapolis, USA, 1996.

[9] G.Malkin. RIP Version 2 Carrying Additional Information. In *RFC 1723 (http://www.nexor.com/public/rfc/index/rfc.html)*, 1994.

[10] h. Oschesner. DECT-Digital European Cordless Telecomunications. In *Proc. 39th IEEE Veh. Tech. Conf.*, pages 718–721, 1989.

[11] Z. J. Haas and M. R. Pearlman. The zone routing protocol for ad–hoc networks. In *Internet draft RFC (http://www.ietf.org/html.charters/manet-charter.html)*, Nov. 1997.

[12] A. Harter and A. Hopper. A Distributed Location System for the Active Office. *IEEE Network*, 8(1), Jan. 1994.

[13] A. Hills and D. B. Johnson. Wireless Data Network Infrastructure at Carnegie Mellon University. *IEEE Personal Communications*, 3(1), Feb. 1996.

[14] D. Johnson and D. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks in Mobile Computing, T. Imielinski and H. Korth eds.*, pages 153–181. Kluwer Academic, 1996.

[15] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Netwroks. In *Proc. ACM MOBICOM '98.*, Aug. 1998.

[16] J. Moy. OSPF Version 2. In *RFC 1583 (http://www.nexor.com/public/rfc/index/rfc.html)*, 1994.

[17] S.-Y. Ni, Y.-C. Tseng, and J.-P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *Proc. ACM MOBICOM '99.*, Aug. 1999.

[18] C. Perkins. Ad–Hoc on–Demand Distance Vector Routing. In *Internet draft RFC (http://www.ietf.org/html.charters/manet-charter.html)*, Nov. 1997.

[19] C. Perkins and P. Bhagwat. Highly Dynamic destination–sequenced distance vector routing for mobile computers. *Computer Communications Review*, 24(4):234–244, Oct. 1994.

[20] R. Prakash and M. Singhal. Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems. *IEEE Trans. on Paral. and Distrib. Sys.*, 7(10):1035–48, Oct. 1996.

[21] N. S., C. K., and B. K. CDPD over Shared AMPS Channels: Interference Analysis. In *Proc. IEEE Personal, Indoor, and Mobile Radio Communications.*, Sept. 1995.

[22] C.-K. Toh. Associativity-Based Routing For Ad-Hoc Mobile Netwroks. In *Proc. IPCCC '96.*, Feb. 1996.

[23] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACMTOIS*, 10(1):91–102, Jan. 1992.

[24] L.-H. Yen, T.-L. Huang, and S.-Y. Hwang. A protocol for casually ordered message delivery in mobile computing systems. *Mobile Networks and Applications*, 2(4):365–372, 1997.