Telecommunication Systems 18:1–3, 61–84, 2001

Route Maintenance in a Wireless Mobile Ad Hoc Network

SHIH-LIN WU, SZE-YAO NI and JANG-PING SHEU
Department of Computer Science and Information Engineering, National Central University, Chung-Li 320, Taiwan

YU-CHEE TSENG
Department of Computer Science and Information Engineering, National Chiao-Tung University, Hsin-Chu 300, Taiwan

Abstract. A *mobile ad-hoc network* (MANET) is formed by a cluster of mobile hosts, without the infrastructure of base stations. To deal with the dynamic changing topology of a MANET, many routing protocols have been proposed. In this paper, we consider the route maintenance problem, which includes two parts: *route deterioration* and *route breakage*. 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 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. So the routing paths can be adjusted on-the-fly while they are still being used for delivering packets and can be patched in minimum wireless bandwidth and 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 [3,17], DECT [19], and GSM [8,11]. 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 [1,2,10,23,27,28].

One wireless network configuration that has received a lot of attention recently is the *mobile ad hoc network* (MANET) [12]. 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 [9]. 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) [15], OSPF [16], and Destination Sequenced Distance Vector (DSDV) [21]. 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) [13], Signal Stability-based Adaptive Routing(SSA) [6], and Ad Hoc On Demand Distance Vector Routing (AODV) [20], have been proposed based on such on-demand philosophy. Recently, a hybrid of these two approaches, called the Zone Routing Protocol (ZRP) [9], is also proposed.

Routing in a reactive protocol typically consists of three parts: *route discovery*, *data forwarding*, and *route maintenance*. This paper considers the route maintenance problem in a reactive protocol. 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, under our definition, should address two issues: *route deterioration* and *route breakage*. Route deterioration refers to the situation where an existing route becomes worse than other routes due to host mobility. However, in existing protocols, such as [6,9,13,20], 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 is typically done by network flooding [18]. 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.

To deal with the route breakage problem, most existing protocols will send a route error packet back to the source node from the position where breakage is found, which will then invoke another route discovery procedure. However, since route discovery is very costly (in terms of both bandwidth and delay), it should be used with caution. Several recent works have targeted in the direction of reducing the cost of one route discovery [4,14,18]. In this paper, we propose to exploit the *locality* of a route to reduce the number of times that route discovery is activated. We observe that it is very likely that a route is broken because only one relay node leaves its neighbors. This is very

similar to the "spatial locality" discussed in [4], which uses prior routes to rebuild new routes. In this paper, we suggest to use a *local route recovery* to patch a broken route before a route error packet is sent back to its source node.

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.

Simulation results are presented to demonstrate the effectiveness of our route maintenance. The current implementation status at the High-Speed Communication and Computing Laboratory, National Central University, will also be reported. One interesting result that we have observed is that the length of a route can significantly affect the bandwidth that can be offered by the route. This justifies the importance of route optimization. However, route optimization can only be effective when there exist (yet-to-be-discovered) better routes around the route currently being used. Our simulation reveals an interesting phenomenon: using route optimization *alone* will not provide good performance, but if route optimization is used together with local route discovery, then significant improvement can be obtained. The reason is that local route recovery will usually extend the lengths of routes.

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. Simulation results are in section 4. Our implementation experience is presented in section 5. Conclusions are drawn in section 6.

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 rebroadcast 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 [15], OSPF [16], and DSDV [21].

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 [6,13]. 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) [13] is derived based on the concept of *source routing* (refer to [5, chapter 7]). 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 figure 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

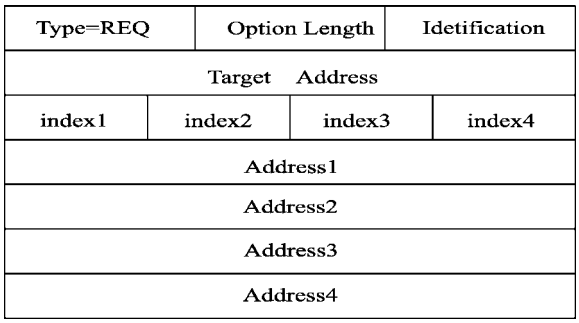

Figure 1. The ROUTE_REQ packet used in DSR. The *option length* field specifies the packet length after itself, the *identification* field is the sequence number, and the *index* and *address* fields define the path that has been tracked so far. Note that the index and address fields can be expanded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

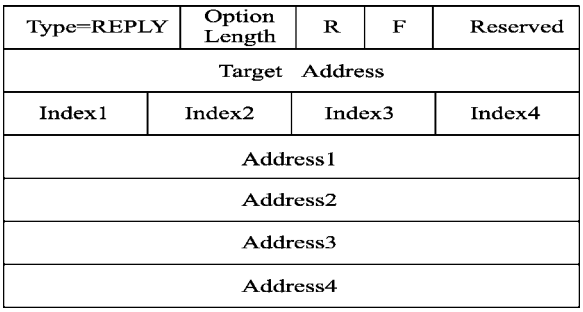

Figure 2. The ROUTE_REPLY packet used in DSR.

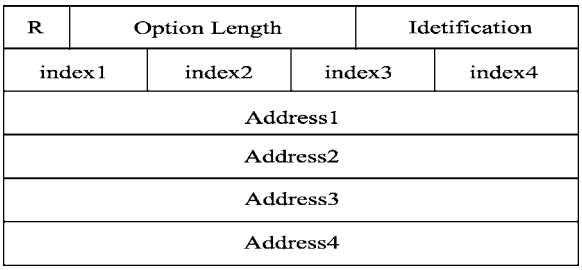

Figure 3. Header of data packets in DSR. If necessary, the index and address fields can be duplicated for longer routes.

a ROUTE_REQ is received by the destination, it returns to the source node a route reply (ROUTE_REPLY as shown in figure 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 figure 3.

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 (figure 4) to the source node. On

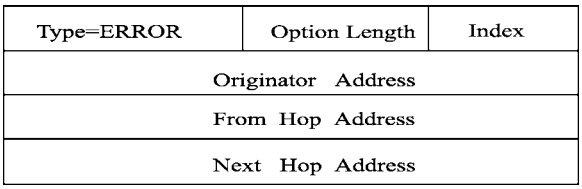

Figure 4. The ERROR packet used in DSR. The *originator address* field indicates the source of the broken route, and the *from hop* and *next hop* identify the two end nodes of the broken link.

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.

Destination	Next Hop
t	o
m	x

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

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) [6] 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 figure 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 figure 6). Below, we summarize the SSA protocol.

A. Route discovery. On needing a route, a source node broadcasts a ROUTE_REQ packet as shown in figure 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

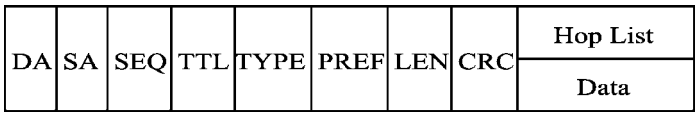

Figure 7. Packets used in SSA. According to the TYPE field, the same format can be used by data, ROUTE_REQ, and ROUTE_REPLY packets.

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 $a \to \cdots \to b \to \cdots \to d$, 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 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 [20] 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

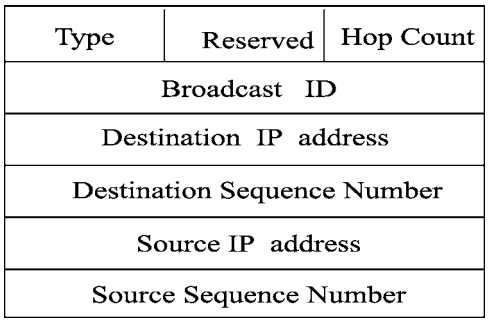

Figure 8. The ROUTE_REQ packet used in AODV.

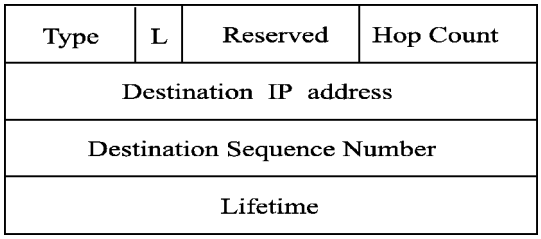

Figure 9. The ROUTE_REPLY packet used in AODV.

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 figures 8 and 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.

2.4. ZRP: zone routing protocol

The Zone Routing Protocol (ZRP) [9] 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 [21]. 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.

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 figure 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

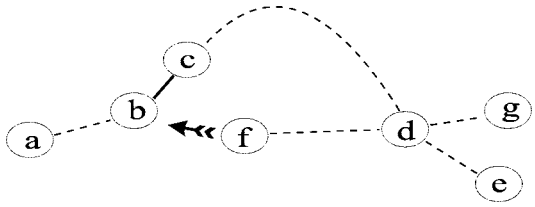

Figure 10. An example of route optimization in DSR.

a promiscuous mode, node f may hear the packets from b to c. If f has a route $f \rightarrow \cdots \rightarrow d \rightarrow \cdots \rightarrow g$. If the hop count from c to d is longer than f to d, then f can suggest a better route $a \rightarrow \cdots \rightarrow b \rightarrow f \rightarrow \cdots \rightarrow d \rightarrow \cdots \rightarrow e$ 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 $P = a \rightarrow \cdots \rightarrow b \rightarrow c \rightarrow \cdots \rightarrow e$.
2. Let $Tmp = P$.
3. **for** (each path P' in f's routing cache) **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*.

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 figure 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 figure 12,

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.

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

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

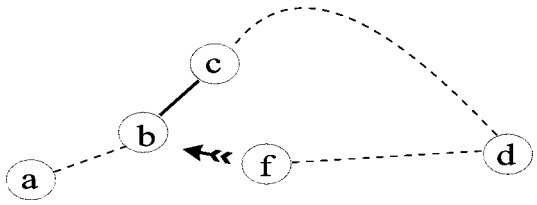

Figure 13. An example of route optimization in protocols using next-hop routing such as SSA and AODV.

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: $a \rightarrow \cdots \rightarrow b \rightarrow c \rightarrow \cdots \rightarrow d$ (refer to figure 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 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** $(i > i' + 1) \wedge (s \leqslant s')$ **then**

   if ($s \leqslant$ 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** $(i > i' + 1) \wedge (s \geqslant s')$ 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 intra-zone 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 figure 14). Note that only border nodes are registered in a route, so in figure 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.

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

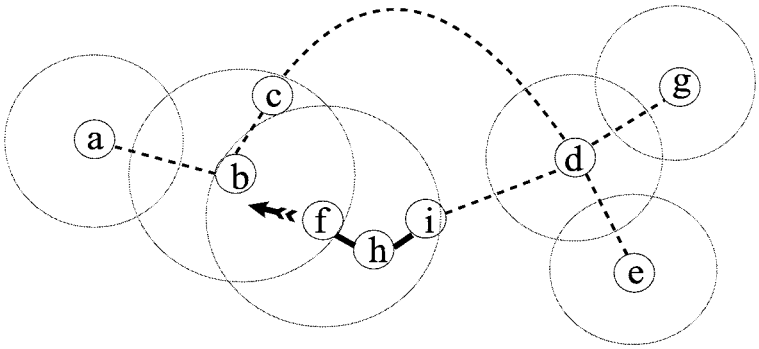

Figure 14. A route optimization example for the ZRP protocol.

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 figure 15(a), which contains a route from a to f: $a \to \cdots \to b \to c \to e \to \cdots \to f$. If host c moves out the transmission range of host b as in figure 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. Note that this is in fact a special case of route discovery except that the ROUTE_REQ packet has a hop limit (or time-to-live) to constrain its searching range. 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

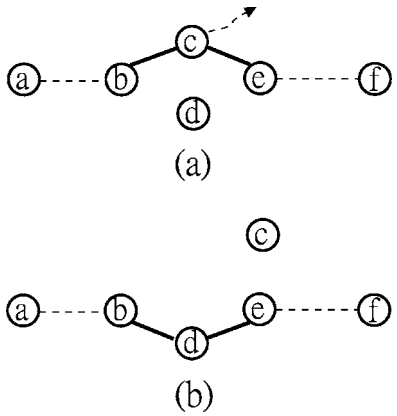

Figure 15. An example of local route recovery.

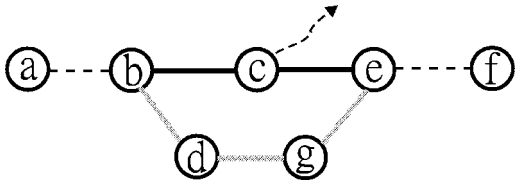

Figure 16. The recovery of least hops of one node leave when a active route is broken.

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.

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 centered at itself. Thus, the closer a route is to this node, the more likely the route can be rebuilt locally.

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. Figure 16 shows 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 [25]. 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.

4. Simulation results

We have developed a simulator to verify the performance of our scheme and compare our result to the DSR protocol. Our simulator was designed by C++. Central to the simulator is a discrete-event engine designed to simulate systems that can be modeled by processes communicating through signals. A simplified version of the MAC specification by IEEE 802.11 was referenced to simulate CSMA/CA behavior among hosts.

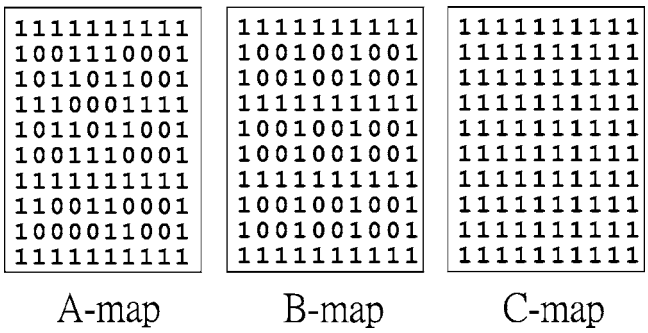

Figure 17. The maps used in our simulation. A "0" represents an obstacle unit, and a "1" a normal unit where hosts can stay.

The communication parameters were: transmission radius = 500 meters, packet size = 256 bytes, and radio transmission rate = 1 Mbit/s. We followed the DSSS physical layer timing parameters suggested by the IEEE 802.11 (PLCP overhead, the slot time, DIFS, backoff window size, etc.). The mobility model was a follows: each mobile host first randomly chose a direction and a speed, and then moveed in that direction and speed for a randomly selected duration. A number of communication pairs were generated, each randomly selected among all simulated mobile hosts. For each communication pair, 500 data packets were generated each separated by a random interval between 0–2000 ms.

The performance metrics to be observed are:

- Number of successfully delivered data packets.
- Number of control packets per data packet. By this ratio, we can understand how many control packets are needed per transmitted data packet.

The MANET was located in a physical area of size 10 × 10 units as shown in figure 17, where one unit is a square of side length 500 meters (equal to the transmission radius). There are three maps used in our simulation: the A-map contains 34 randomly chosen units as obstacles (denoted by 0's) in which areas hosts can not stay, the B-map simulates streets in a city, and the C-map simulates a plain field where there are no obstacles. The following simulation results (items A, B, C, and D) are all based on the A-map, and in the last simulation (item E) we will compare these 3 maps. Each simulation was run 40,500 s (communication pairs were generated in the first 40,000 s, and the rest 500 s were for these communications to terminate). Unless otherwise stated, 200 mobile hosts were simulated in the physical area and 1,000 communication pairs were randomly selected.

A. Effect of mobility. In the first simulation, we varied the speed of mobile hosts to observe its effect. In figure 18, we show the number of data packets delivered by DSR and our protocols. Two versions of our protocol were simulated: one with only route optimization (denoted by "Opt") and one with both route optimization and local route recovery (denoted by "Opt + LRR"). The performance is normalized to the number of

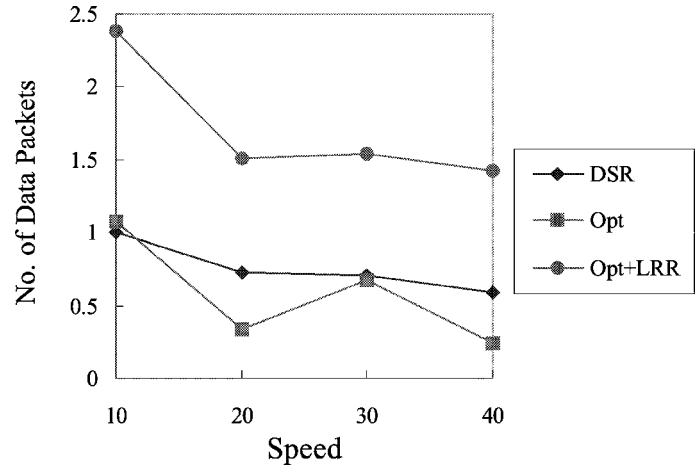

Figure 18. Bandwidth vs. mobility.

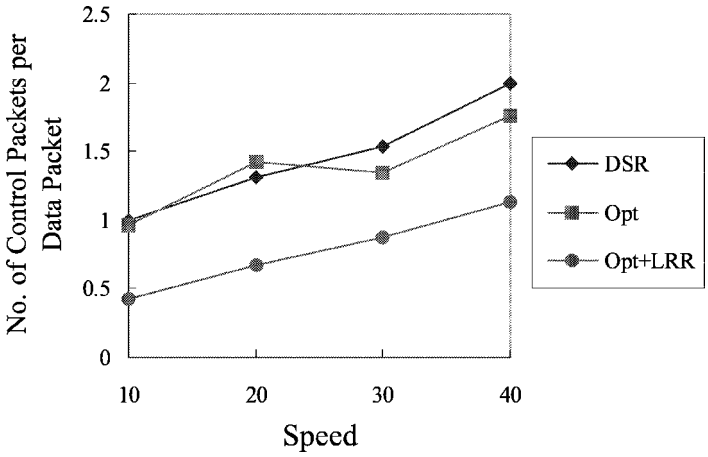

Figure 19. Cost of control packets vs. mobility.

data packets delivered by the DSR at speed = 10 km/h. We can see that in all cases the performance is degrading as the host mobility increases. An interesting phenomenon is that using route optimization *alone* in fact has a negative effect on performance as compared to DSR. The reason is that the initial route is usually the shortest among all available routes, and thus there is little chance for route optimization to take effect. However, when combining with our local route recovery, route optimization has more chances to succeed because a local route recovery is more likely to extend the length of the route. Overall, with both route optimization and local route recovery, our protocol outperforms the DSR protocol by about 2 times. In figure 19, we show the number of control packets incurred per data packet. The cost of control packets increases as mobility increases. However, our cost is about half that of the DSR.

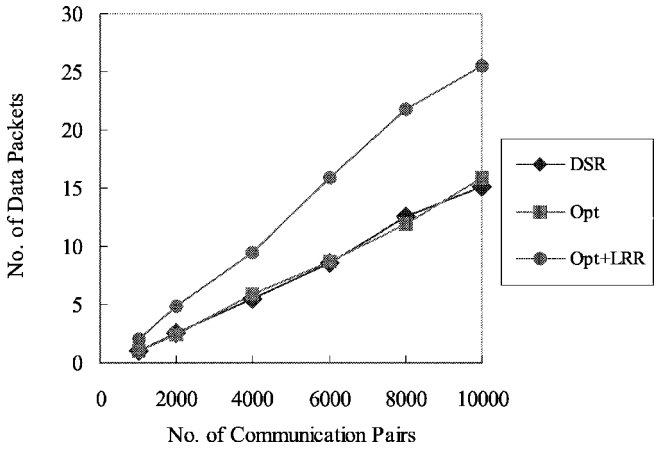

Figure 20. Bandwidth vs. traffic load.

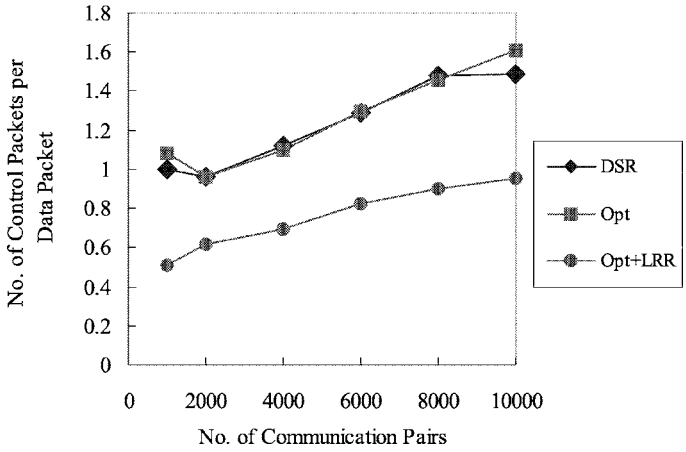

Figure 21. Cost of control packets vs. traffic load.

B. Effect of traffic load. In this simulation, we vary the number of communication pairs. The result is in figures 20 and 21 (note that these numbers are normalized to the DSR's performance with 1,000 communication pairs). The similar trend as in the previous simulation can be observed. Also, a factor that can contribute to our protocol is that with more communication pairs, there is more route information that can be collected by mobile hosts, making route optimization and local route recovery easier to succeed.

C. Effect of host density. In this simulation, we vary the number of mobile hosts between 100 and 300. The result is in figures 22 and 23. The result, as compared to earlier ones, is in fact more favorable for our protocol since there is higher probability for route optimization and local route recovery to succeed with higher host density.

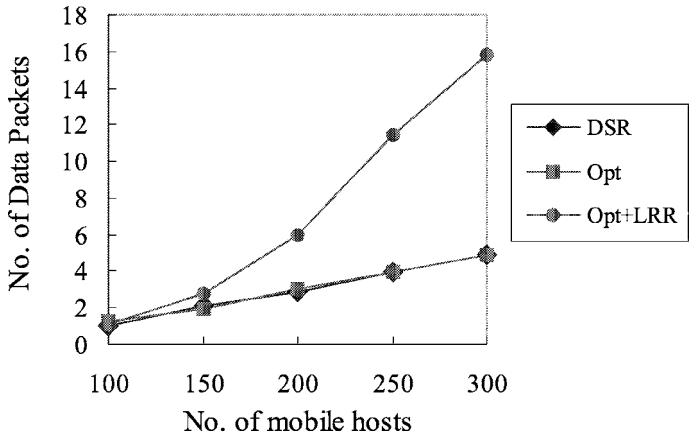

Figure 22. Bandwidth vs. host density.

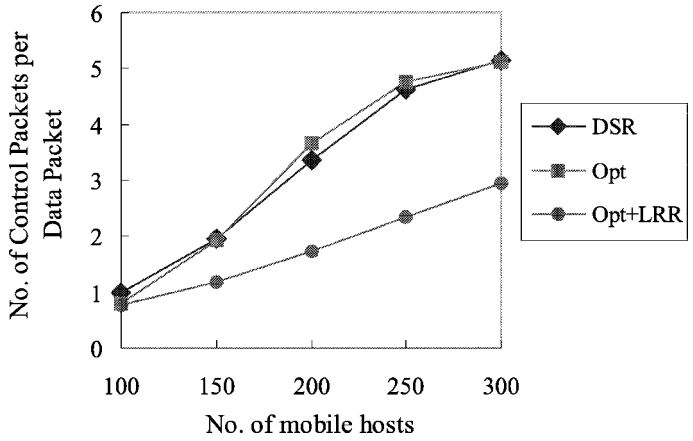

Figure 23. Cost of control packets vs. host density.

D. Effect of range of local route recovery. In all the above simulations, we have used 2 hops (i.e., TTL $= 2$) to broadcast local route recovery packets. To understand the effect of hop counts of these packets, we further vary the hop count between 1 and 7 and observe the effect. The result is in figures 24 and 25. As can be seen, this does not have significant effect on performance. Using a hop count of 2 or 3 will lead to a slightly better throughput. But using a hop count of 4 will give the lowest control cost.

E. Effect of topology of physical area. All the above simulations have used the A-map in figure 17. In this simulation, we compare the 3 maps in figure 17. The result, as shown in figures 26 and 27, shows an interesting phenomenon: it is more favorable to perform route maintenance in a wide open area than an area with obstacles. From the result, we see that our protocol performs comparably the best in C-map, which is followed by A-map, which is in turn followed by B-map.

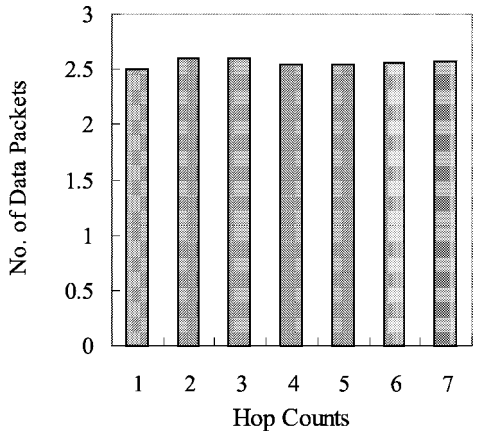

Figure 24. Bandwidth vs. hop count used in local route recovery.

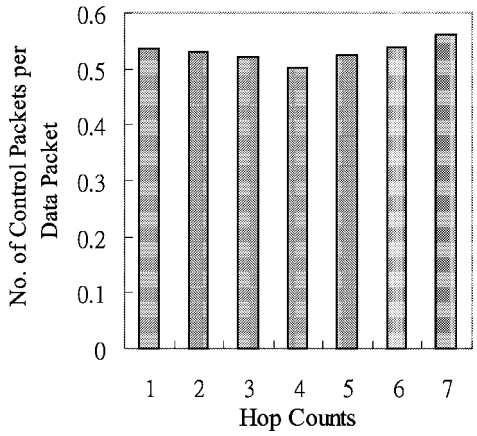

Figure 25. Cost of control packets vs. hop count used in local route recovery.

5. Implementation results and observations

We have implemented a next-hop routing protocol on top of the Linux operating system. The platform has a number of notebooks of a variety of speeds (Pentium 200MMX, Pentium 233MMX, Pentium II 350, etc.), 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/s.

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.

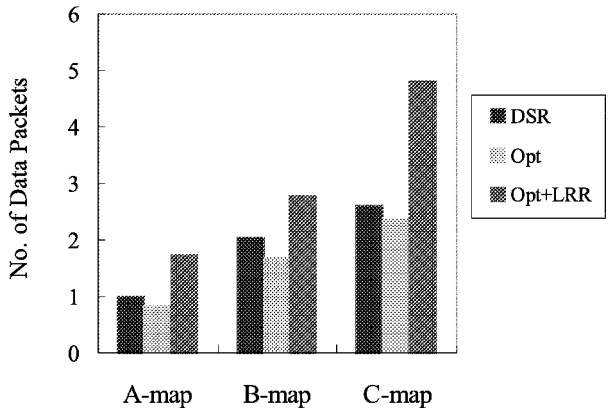

Figure 26. Bandwidth vs. different maps.

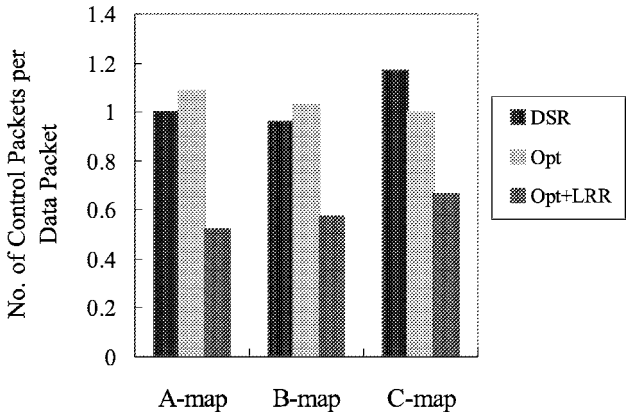

Figure 27. Cost of control packets vs. different maps.

With this platform, we try to observe the effect of hop count on the performance of MANET. We placed the mobile hosts in a linear manner such that each host can hear only its one or two neighbors. Our first experiment was to use the *ping* command at a certain host to contact another host to observe the delay to discover a new route. Also, this experiment was done in an environment in which all mobile hosts have no entry in their route caches. The result is in figure 28. As can be seen, the delay is quite small (in a scale of 10^{-3} s). The time needed to find a route will increase linearly with respect to the hop count, which is reasonable.

Our second experiment was to use the *ftp* command (under binary mode) to determine the communication bandwidth at different hop counts. The result is shown in figure 29. In the "simplex" experiment, we initiated a *ftp* from a source host to a destination host separated by a certain number of hops. In the "duplex" experiment, we initiated a *ftp* in both sides. One interesting observation is that the bandwidth degrades by half when the hop count changes from 1 to 2. The bandwidth further degrades by

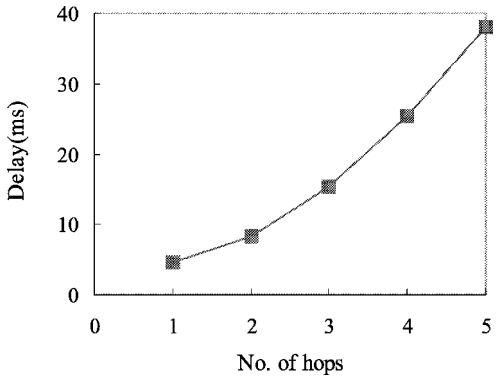

Figure 28. The delay to discover a new route vs. route length.

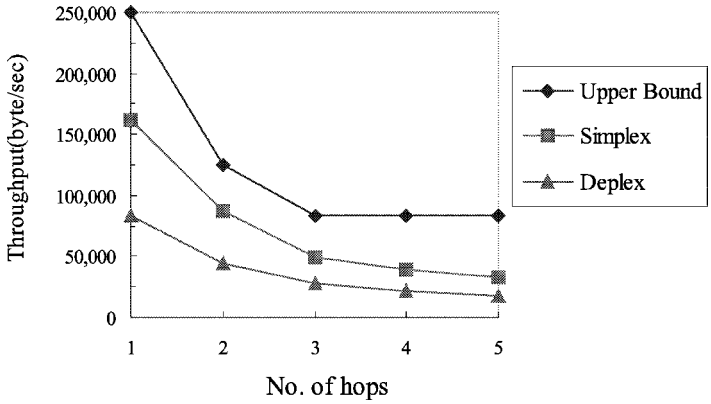

Figure 29. The bandwidth of a route vs. route length.

1/3 when the hop count changes from 1 to 3. After 3 hops, the bandwidth still keeps on degrading, but in a slower speed. Hence, this justifies the need of performing route optimization in a MANET to help improving the end-to-end bandwidth. Of course, the level of contention on medium can also be reduced if routes are shorter.

It is also worth commenting on the line "upper bound" in figure 29. Obviously, given a sender–receiver pair that are next to each other, the theoretical bound on bandwidth is 2 Mbit/s. Given a sender-receiver pair that are 2 hops away, the theoretical bound will suddenly reduce to 1 Mbit/s. The reason is that none of the hosts in the (2-hop) route can transmit at the same time. Following the same line of reasoning, given a sender-receiver pair that are 3 hops away, the theoretical bound will reduce to 2/3 Mbit/s. This is resulting from the effect of signal interference and the hidden terminal problem [24]. However, after 3 hops, these factors will disappear and a pipelining effect may appear. Specifically, two hosts separated by $\geqslant 3$ hops may be able to send at the same time. For instance, in figure 30, we show 10 mobile hosts arranged in a linear array. Hosts 1, 4 and 7 can send simultaneously, hosts 2, 5 and 8 can send simultaneously, and hosts 3, 6 and 9 can send simultaneously. This in fact can be formulated by the

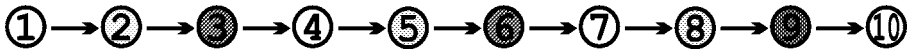

Figure 30. An illustration of the pipelining effect to derive the theoretical upper bound of bandwidth in a multihop path.

well-known *graph-coloring* problem. Thus, if the a "perfect" pipeline can be formed, then the theoretical upper bound on bandwidth will be $2/3$ Mbit/s.

6. Conclusions

In this paper, we have shown how to perform route optimization for four MANET routing protocols, 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 even under 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. From simulations, we have seen that local route recovery is quite effective, but will usually lengthen routes. So route optimization is necessary to save the wireless bandwidth. By combining route optimization and local route recovery, significant improvement can be obtained.

Our route optimization may lead to unstable routes due to host mobility (e.g., going back and forth between two routes), thus wasting the wireless bandwidth. Hence one direction that deserves further investigation is to perform route optimization only when the new route is shorter as well as more stable (which may be determined from, say, signal strength). Recently, an independent work [7] also suggests a similar approach. As a route becomes broken, route recovery can be performed. Since such an event will be sent to the source host, the source host will initiate a route optimization request when some criteria is satisfied.

Acknowledgements

This research is supported by the Ministry of Education, ROC, under grant 89-H-FA07-1-4 (Learning Technology).

References

[1] G.D. Abowd et al., Cyberguide: A mobile context-aware tour guide, Wireless Networks 3(5) (1997) 421–433.

[2] A. Archarys and B.R. Badrinath, A framework for delivering multicast messages in networks with mobile hosts, Mobile Networks and Applications 1(2) (1996) 199–219.

[3] K. Budka, Cellular digital packet data: Channel availability, in: *Proc. IEEE Personal Indoor and Mobile Radio Communications (PIMRC'95)*, September 1995.

[4] R. Castaneda and S.R. Das, Query localization techniques for on-demand routing protocols in ad hoc networks, in: *Proc. of ACM MOBICOM '99*, August 1999.
[5] D.E. Comer, *Internetworking with TCP/IP*, Vol. 1: *Principles, Protocols, and Architecture* (Prentice-Hall, Englewood Cliffs, NJ, 1991).
[6] R. Dube, C.D. Rais, K. Wang and S.K. Tripathi, Signal stability based adaptive routing for ad-hoc mobile networks, IEEE Personal Communications (February 1997).
[7] Z. Gai, M. Lu and C. Georghiades, Optimized reparation based on sequenced number routing for ad hoc networks, in: *IASTED Internat. Conf. on Parallel and Distributed Computing and Systems*, 1999, pp. 302–306.
[8] J. Geier, *Wireless Networking Handbook* (New Riders Publishing, Indianapolis, USA, 1996).
[9] Z.J. Haas and M.R. Pearlman, The zone routing protocol for ad-hoc networks, Internet draft (November 1997); `http://www.ietf.org/html.charters/manet-charter.html`.
[10] A. Harter and A. Hopper, A distributed location system for the active office, IEEE Network 8(1) (January 1994).
[11] A. Hills and D.B. Johnson, Wireless data network infrastructure at Carnegie Mellon University, IEEE Personal Communications 3(1) (February 1996).
[12] IETF MANET Working Group, `http://www.ietf.org/html.charters/manet-charter.html`.
[13] D.B. Johnson and D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: *Mobile Computing*, eds. T. Imielinski and H. Korths (Kluwer Academic, Dordrecht, 1996) chapter 5, pp. 153–181.
[14] Y.-B. Ko and N.H. Vaidya, Location-aided routing (LAR) in mobile ad hoc netwroks, in: *Proc. of ACM MOBICOM '98*, August 1998.
[15] G. Malkin, RIP version 2 carrying additional information, RFC 1723 (1994); `http://www.nexor.com/public/rfc/index/rfc.html`.
[16] J. Moy, OSPF version 2, RFC 1583 (1994); `http://www.nexor.com/public/rfc/index/rfc.html`.
[17] S. Nanda, K. Chawla and K. Budka, CDPD over shared AMPS channels: Interference analysis, in: *Proc. IEEE Personal, Indoor, and Mobile Radio Communications*, September 1995.
[18] S.-Y. Ni, Y.-C. Tseng and J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, in: *Proc. of ACM MOBICOM '99*, August 1999.
[19] H. Oschesner, DECT-Digital European Cordless Telecomunications, in: *Proc. of the 39th IEEE Vehicular Technology Conf.*, 1989, pp. 718–721.
[20] C.E. Perkins, Ad-hoc on-demand distance vector routing, Internet draft (November 1997); `http://www.ietf.org/html.charters/manet-charter.html`.
[21] C. Perkins and P. Bhagwat, Highly dynamic destination-sequenced distance vector routing for mobile computers, Computer Communications Review 24(4) (October 1994) 234–244.
[22] R. Prakash, M. Raynal and M. Singhal, An efficient causal ordering algorithm for mobile computing environments, in: *Internat. Conf. on Distributed Comput. Systems*, 1996, pp. 744–751.
[23] R. Prakash and M. Singhal, Low-cost checkpointing and failure recovery in mobile computing systems, IEEE Transactions on Parallel and Distributed Systems 7(10) (October 1996) 1035–1048.
[24] A.S. Tanenbaum, *Computer Networks* (Prentice Hall, Englewood Cliffs, NJ, 1996).
[25] C.-K. Toh, Associativity-based routing for ad-hoc mobile netwroks, in: *Proc. of IPCCC '96*, February 1996.
[26] Y.-C. Tseng and C.-C. Tan, On termination detection protocols in a mobile distributed computing environment, in: *Internat. Conf. on Parallel and Distributed Systems*, 1998.
[27] R. Want, A. Hopper, V. Falcao and J. Gibbons, The active badge location system, in: *ACM Internat. Conf. on Supercomputing*, 10(1) (January 1992) pp. 91–102.
[28] 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) (1997) 365–372.