PAPER Special Section on Papers Selected from ITC-CSCC 2005

A Pseudo-Distance Routing (PDR) Algorithm for Mobile Ad-hoc Networks

Min-Gu LEE[†], ***Student Member and*** **Sunggu LEE**[†a)], ***Nonmember***

SUMMARY Previous routing algorithms for mobile ad-hoc networks (MANETs) have focused on finding short-distance path(s) between communicating nodes. However, due to the dynamic and unreliable communication nature of MANETs, previously determined paths can easily become disconnected. Although dynamic routing can be used to circumvent this problem, determining a new route each time a packet needs to be sent involves a lot of overhead. An alternative form of dynamic routing involves maintaining valid routes in routing tables, which can be dynamically updated whenever network changes are detected. This paper proposes a new routing algorithm, referred to as pseudo-distance routing (PDR), that supports efficient routing table maintenance and dynamic routing based on such routing tables.

key words: *routing, mobile ad-hoc networks, pseudo-distance*

1. Introduction

Wireless ad-hoc networks are composed of mobile nodes that communicate with each other using radio transmission without a fixed infrastructure. Such nodes can communicate with other nodes that are within their radio transmission ranges. If a node needs to communicate outside its radio transmission range, intermediate nodes are used to relay packets from the source toward the destination. To relay these packets, wireless ad-hoc network devices should be able to detect the presence of other devices and find suitable paths to their respective destinations. Like conventional routing algorithms for infrastructured networks, the optimality of routing paths in terms of the number of hops in mobile wireless ad-hoc networks (MANETs) is very important. However, unlike conventional infrastructured networks, the network topology is not fixed due to the dynamic nature of the network arising from factors such as the mobility of the nodes, low signal power, suspended states of intermediate nodes (for energy conservation) and interference in the wireless channel. These factors can cause frequent and unpredictable changes in network topology. Therefore, the goal of a routing algorithm for MANETs is not only to route optimally but also to be adaptable to highly dynamic changes in network topology. Furthermore, a routing algorithm for MANETs should provide redundant paths that can be used when the primary route fails or becomes highly congested. Finally, this routing algorithm should be executable in a fully distributed manner.

Manuscript received September 7, 2005.
Manuscript revised December 30, 2005.
Final manuscript received February 16, 2006.

[†]The authors are with the EE Department, Pohang University of Science and Technology, Pohang, Korea.
a) E-mail: slee@postech.ac.kr

DOI: 10.1093/ietfec/e89–a.6.1647

2. Related Works

There has been significant interest in routing algorithms for MANETs in the recent past. Routing algorithms that are developed for ad-hoc networks can be divided into 2 categories: *proactive* and *reactive*. Proactive routing algorithms attempt to maintain consistent and up-to-date routing information from each node to every other node in the network. In order to maintain consistent and up-to-date routing information, proactive algorithms must frequently exchange routing information. On the other hand, reactive algorithms create routes only when desired by the source node. Whenever a node requires a route to a destination, it initiates a route discovery process within the network.

2.1 Proactive Algorithms

Proactive algorithms maintain up-to-date routing information. To maintain routing information, nodes in the network should generate periodic control messages that contain routing information. Therefore, proactive algorithms tend to require a large control message overhead for routing table maintenance.

Destination sequenced distance vector (DSDV) [1] routing is a typical algorithm in the proactive category. DSDV is based on the classical distributed Bellman-Ford routing algorithm. Each node in the network maintains a view of the network topology with a cost for each link and a distance vector table that has the number of hops to all the possible destinations. To maintain consistency, each node periodically broadcasts routing table update messages. If the number of nodes in the network is large, DSDV may generate a lot of control traffic even if the incremental packet scheme is used. Therefore DSDV is not suitable for large scale applications. Furthermore, it may take a long time to converge because changes in routing information need to be propagated to all nodes in the network. Finally, DSDV can provide only a single route to each destination.

Clusterhead gateway switch routing (CGSR) [3] is another proactive routing algorithm that tries to overcome shortcomings of DSDV. A cluster is defined as a group of nodes that can directly communicate with a cluster head that is elected within the cluster. There are three types of nodes in CGSR—normal nodes, gateway nodes and cluster head nodes. Normal nodes belong to one cluster and can directly communicate with the cluster head of that cluster. Gateway nodes can communicate with two or more cluster heads. Therefore gateway nodes can relay packets from a cluster head to another cluster head. Cluster heads control intra-cluster communications and inter-cluster communications. For inter-cluster communication, each cluster head maintains a cluster member table and distance vectors of other cluster heads as in DSDV. Note that only cluster heads maintain distance vector tables and cluster member tables. However, the stability of the cluster head is a critical problem because changing a cluster head is a very expensive process. CGSR also provides only a single path to each destination.

Wireless routing protocol (WRP) [2] is a proactive routing algorithm with four tables—a distance table, routing table, link-cost table and message retransmission list table. An update message is sent after processing updates from neighbors or after detecting a change in a link to a neighbor, such as a loss of a link or a "hello" message from a new node. WRP converges in its route maintenance phases much faster than DSDV, but it still only provides a single path to each destination. In addition, WRP requires a large amount of overhead to maintain shortest path spanning tree information reported by neighboring nodes and to react to failures.

2.2 Reactive Algorithms

Reactive routing algorithms try to find valid routes only when a source node wishes to send packets. The major shortcomings of reactive algorithms are the possibility of significant delay during the route discovery phase. Since route discovery is based on flooding of query messages in most reactive algorithms, it is also very costly. In real-time applications, excessive delays during route discovery may result in deadline misses.

Dynamic source routing (DSR) [5] is an on-demand routing algorithm that implements source routing in MANETs. To find a path to its destination, a source node floods request messages toward the destination. When the destination node receives the request message, it replies to the source with the route that the received request packet passed through. Then the source node can send all data packets along this route. DSR provides only a single path to a destination, and it suffers from a scalability problem because each data packet sent by a source has to contain complete routing information and the size of a control message increases every time it visits an intermediate node.

Ad-hoc on-demand distance vector (AODV) [4] routing is a widely accepted routing algorithm for MANETs. Route discovery is the same as in DSR but the route information is stored in intermediate nodes as in DSDV. When a route becomes disconnected, then a "network broken" message is sent to the source node in order to re-initiate route discovery by the source node. If a node wishes to find a path to a destination that has previously been determined by another node, it still needs to initiate route discovery by flooding route discovery packets. Note that this is a very costly flooding operation that has to be undertaken even if intermediate nodes have cached route information. When an intermediate node detects a disconnected link, it can itself initiate a local route discovery phase. However, if it fails, additional time is required (when compared to source-initiated route discovery only) because the intermediate node reports the failure of local route discovery to the source node only after failure of local route discovery is detected by the intermediate node. AODV also provides only a single path to each destination. However, ad-hoc on-demand multipath distance vector (AOMDV) [10] routing, which is an extension of AODV, can provide multiple disjoint paths to each destination. Nevertheless, AOMDV still retains the other shortcomings of AODV.

Location-aided routing (LAR) [6] utilizes location information to improve routing performance using external devices such as global positioning system (GPS) devices. However GPS is not widely adopted in mobile devices yet, and a GPS device costs over a hundred dollars. In addition, GPS devices do not work in indoor environments, tunnels or forests because GPS devices need to have direct line-of-sight access to satellites.

Temporally-ordered routing algorithm (TORA) [7] is an on-demand routing algorithm that supports multiple paths to each destination. TORA is based on the link reversal algorithms proposed in [8]. The main idea of a link reversal algorithm is to totally order the nodes of a network with respect to a given destination by assigning a height to each node and a direction to each link according to the relative heights of adjacent nodes. By assigning the lowest height to the destination, the algorithm creates a directed acyclic graph (DAG) rooted at the destination. A DAG is *destination-oriented* if, for every node, there exists a directed path originating at that node and terminating at the destination [8]. To build a destination-oriented DAG, each node only needs to maintain routing information for adjacent nodes. Therefore, link reversal algorithms are very scalable and react well in highly dynamic network environments. As long as a node has at least one outgoing link, it has a loop-free route toward the destination, as proven in [8]. When a node loses its last outgoing link, a localized reaction is initiated, which iteratively reverses the links for all nodes whose paths are affected by this link failure until new routes are established. A link reversal algorithm provides multiple paths to each destination, localized maintenance of routing paths and fully distributed execution. However, a link reversal algorithm does not take route optimality into account. Therefore, repeated reconstruction of destination-oriented DAGs can result in long detour paths.

In this paper, the pseudo-distance routing (PDR) algorithm is proposed to overcome the shortcomings of the above algorithms. Like TORA, PDR is also an algorithm that uses a link reversal mechanism to quickly adapt to changes in highly dynamic networks. Therefore, PDR inherits all of the advantages of link reversal algorithms. However, to overcome the main shortcoming of link reversal algorithms (the possibility of long detour paths), PDR employs a pseudo-distance concept instead of the "height" in link reversal algorithms and TORA. Furthermore, if a node initiates a route discovery phase for a destination, then all affected nodes in the network can maintain valid paths to that destination, thereby enabling fast route discovery for that destination during future searches.

3. Pseudo-Distance Routing

3.1 Assumptions

PDR assumes that all links in MANETs are bidirectional. Actual links in MANETs are not bidirectional. However, by building a routing algorithm on top of other protocols that support bidirectional communication, as in the internet MANET encapsulation protocol (IMEP) [9], bidirectional communication links can be realized. Furthermore, communication links of MANETs are very unreliable. Due to the unreliability of MANETs, some control messages may be lost during route discovery or maintenance phases. Because IMEP also provides reliable communication links using an ACK scheme, it is able to overcome the loss of control messages. Therefore we can achieve reliable, bidirectional communication links using an IMEP layer. PDR assumes that detection of neighboring nodes is performed by an underlying layer such as MAC or other layers. Note that IMEP also provides connectivity information to the upper layer protocols. PDR assumes a broadcasting network since most off-the-shelf wireless ad-hoc devices are omni-directional (IEEE 802.11, bluetooth, etc.). PDR also assumes that each node has its own unique identifier (like a MAC address). Finally, PDR does not consider network partitioning.

3.2 Notation

A network is modeled as an undirected graph $G = (V, E)$, where V is a finite set of nodes and E is a set of bidirectional communication links at a given time instant. A (src, dst)-path is a finite sequence of nodes $P = (src = v_0, v_1, \cdots, v_i, \cdots, v_n = dst)$ such that for all $0 \leq i \leq n$, $e_{i,i+1} \in E$ and $v_i \neq v_j$ for all $v_i, v_j \in P$. $n = |P|$ is the length of P, which is the number of nodes in the path P excluding the source node. $D_{i,j}$ is the distance from v_i to v_j, which is the shortest length among all possible (v_i, v_j)-paths. Finally, a neighbor set of node v_i, written as N_i, consists of all nodes that have a bidirectional link to node $v_i \in V$.

3.3 Pseudo-Distance

As stated above, PDR adopts the destination-oriented DAG in [8]. In order to transform a given network graph $G = (V, E)$ into a destination-oriented DAG, each node needs to have its own height value. Unlike TORA and link reversal algorithms, a height in PDR is not a value representing temporal order but a *pseudo-distance* to the destination. The height of a node v_i relative to a node v_j is written as $H_{i,j} =< \lambda, -\alpha, -\beta >$. A pseudo-distance λ is a distance metric between a node v_i and v_j. α is the number of neighbors that have lower λ values than v_i and β is the number of neighbors that have the same λ value.

α represents the number of neighbors that are expected to be closer than the current node to the destination node and β represents the number of neighbors that are expected to be the same distance as the current node to the destination node. α and β are used to find a next-hop node, preferably closer to the destination, that has as many different paths to the destination as possible. Forwarding a packet to a neighboring node that has the largest α or β values should increase the number of possible redundant paths. Clearly, it is better to forward a packet to neighbor that has more possible paths to the destination. Thus, PDR forwards packets to a neighbor with the largest α value, breaking ties with β values if possible.

PDR compares the heights of two adjacent nodes lexicographically. In the height metric, a minus sign is prepended to α and β. This is done to permit simple lexicographic comparisons. Then, in order to forward packets toward the destination, each node simply selects a neighbor with the minimum height as its next hop. Thus, all neighbors with the smallest λ value are considered first. Among all such neighbors, nodes with the smallest $-\alpha$ value are considered next. Then, among of these neighbors, nodes with the smallest $-\beta$ value are considered as next-hop candidates. If there is only one candidate remaining, it is chosen as the next-hop node. If there are still multiple candidates remaining, then the candidate with the lowest ID value is arbitrarily chosen as the next-hop node. To reduce the number of control messages required, temporarily incorrect α and β values are permitted since small deviations in the number of alternate subpaths are not catastrophic.

Two types of links are identified when using the pseudo-distance concept. Primary links are mainly used to route packets along paths that are as short as possible. Auxiliary links are used only when all primary links are broken. If the λ value of any two adjacent nodes are different, then the link is primary; otherwise, the link is auxiliary. When a node loses its last primary link, it need to check whether it has auxiliary outgoing links or not. If auxiliary routing is turned on, then the node does not update its height (in order to reduce the number of control messages being sent). However, if auxiliary routing is turned off, it tries to replace auxiliary links with primary links by increasing its pseudo-distance. Note that primary links are expected to reduce the distance toward the destination but auxiliary links are not. Therefore forwarding a packet along auxiliary links may introduce extra-hop detours in the path to the destination.

There are four cases to be considered when setting the directions of links. For any two neighbor nodes v_i and v_k with destination v_j,

- if $\lambda_{i,j} > \lambda_{k,j}$, then v_i sets its link $e_{i,k}$ as primary outgoing and v_k sets $e_{k,i}$ as primary incoming.
- if $\lambda_{i,j} = \lambda_{k,j}$ and $-\alpha_{i,j} > -\alpha_{k,j}$, then v_i sets its link

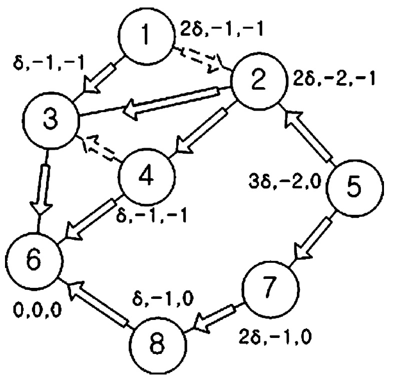


Fig. 1 An example of a destination-oriented DAG using pseudo-distance for destination node v_6.

$e_{i,k}$ as auxiliary outgoing and v_k sets $e_{k,i}$ as auxiliary incoming.

- if $\lambda_{i,j} = \lambda_{k,j}$, $-\alpha_{i,j} = -\alpha_{k,j}$ and $-\beta_{i,j} > -\beta_{k,j}$, then v_i sets its link $e_{i,k}$ as auxiliary outgoing and v_k sets $e_{k,i}$ as auxiliary incoming.
- if $\lambda_{i,j} = \lambda_{k,j}$, $-\alpha_{i,j} = -\alpha_{k,j}$, $-\beta_{i,j} = -\beta_{k,j}$ and $i > k$, then v_i sets its link $e_{i,k}$ as auxiliary outgoing and v_k sets $e_{k,i}$ as auxiliary incoming.

As described in [8], to build a destination-oriented DAG, PDR requires only local (neighbor) routing information (as in link reversal algorithms and TORA). Suppose that a node v_j is the destination. A node v_i should collect $H_{k,j}$ for all k where $v_k \in N_i$ in order to properly set all links $e_{i,k}$.

Figure 1 shows an example of a destination-oriented DAG, with destination node v_6, using the pseudo-distance concept. The numbers in the vertices represent unique identifiers for each node and the numbers beside the vertices are height values $H_{i,6}$ for each node. Solid arrows represent primary links and dashed arrows represent auxiliary links. Note that the pseudo-distance between two nodes is a multiple of δ, which is the default difference in λ between adjacent nodes.

To send packets, each node v_i simply selects a node $v_k \in N_i$ that has the smallest $H_{k,6}$. In Fig. 1, v_1 would select v_3 as its next hop because lexicographically $H_{3,6} < H_{2,6}$. Suppose that v_5 wishes to send a packet to destination v_6. In this case, pseudo-distances of both v_2 and v_7 are the same and both $e_{5,2}$ and $e_{5,7}$ are primary links. However, v_5 should be able to select v_2 because it has more remaining paths to the destination node. v_5 selects v_2 as the next hop because lexicographically $H_{2,6} < H_{7,6}$. Suppose instead that v_5 selects v_7 as the next hop. In this case, if links $e_{7,8}$ or $e_{8,6}$ are broken, then the packet sent from v_5 will fail to be delivered to destination v_6. However, if v_5 selects v_2, then several failure(s) of links, such as $e_{2,4}$, $e_{4,6}$ and others, can be tolerated. Because the heights of both v_3 and v_4 are the same, i.e., $H_{3,6} = H_{4,6}$, and the identifier of v_4 is greater than v_3, v_4 sets $e_{4,3}$ as its auxiliary outgoing link.

3.4 Control Messages

Initially, every node except the destination sets its height to null. The destination node sets its height to zero. In order to assign and maintain height values, three types of control messages are defined.

- QRY is a route query message that is triggered when a node wishes to send a packet to a destination node. QRY is defined as $< DST, ORI, SEQ >$ where DST is the destination, ORI is the source node that initiates the route discovery phase and SEQ is a sequence number used to distinguish QRY messages. QRY is forwarded toward the destination or an intermediate node that has a non-null height value.
- REP, which contains pseudo-distance information, is the reply message for a QRY. REP is defined as $< DST, H, SEQ, ORI >$ where H is the height of the node that sends the REP. REP is triggered by a QRY if a node has valid route information (actually, a node that has a non-null pseudo-distance) for the destination node. REP may also be generated by another REP to forward routing information.
- UPD is a route maintenance message used to reflect topological changes in a MANET. UPD is also defined as $< DST, H, SEQ, ORI >$. UPD is triggered when a local pseudo-distance λ value is changed. Note that for efficiency, in case only α or β values are changed, an UPD is not triggered because temporal inconsistencies in α and β values can be tolerated. The pseudo-distance λ is changed locally when a node loses all of its outgoing links.

Note that PDR assumes that MANETs are broadcast networks. Thus, if v_i sends a control message, all nodes $v_k \in N_i$ can listen to this message and perform the required operations as described in Sects. 3.5 and 3.6.

The algorithm for assigning pseudo-distance values consists of two phases. The first (route discovery) builds a destination-oriented DAG by assigning a height to each node and the second (route maintenance) maintains the destination-oriented DAG whenever there are topological changes.

3.5 Route Discovery Phase

When a source node v_i wishes to send packets to a destination node v_j, v_i first checks its height $H_{i,j}$. If $H_{i,j}$ is null, then v_i initiates route discovery phase by broadcasting QRY=$< v_j, v_i, qryseq_i >$ to its neighbors. After broadcasting QRY, v_i increases its own $qryseq_i$ value by 1. Figure 2 shows the pseudocode of procedure executed when a node v_k receives a QRY message from its neighbor v_l. An intermediate node v_k that receives a QRY from v_i rebroadcasts QRY=$< v_j, v_i, qryseq_i >$ if $H_{k,j}$ is also null (lines 12–13 of Fig. 2). An intermediate node v_l may receive multiple QRYs from its neighbors. In that case, v_l broadcasts only the first

```
 1. recvQRY(message p) {
 2.     if(isReceived(p)) {
 3.         drop(p);
 4.         return;
 5.     }
 6.     if(p.$DST = v_k$) {
 7.         sendREP($H_{k,k}, repseq_k$++);
 8.         return;
 9.     } else if ($H_{k,j}! = NULL$) {
10.         sendREP($H_{k,j}, repseq_k$++);
11.         return;
12.     } else {
13.         sendQRY(p);
14.     }
15.     return;
16. }
```

Fig. 2 Pseudocode for recvQRY.

```
 1. recvREP(message p) {
 2.     if(isReceived(p)) {
 3.         drop(p);
 4.         return;
 5.     } else {
 6.         updateNeighbor(p);
 7.     }
 8.     if(p.$DST = v_k$) {
 9.         drop(p);
10.         return;
11.     } else if ($H_{k,j} = NULL$) {
12.         $\lambda_{k,j} = \lambda_{l,j} + \delta$;
13.         updateHeight();
14.         sendREP($H_{k,j}, repseq_j$);
15.         return;
16.     } else if ($\lambda_{k,j} - \lambda_{l,j} > \delta$) {
17.         $\lambda_{k,j} = \lambda_{l,j} + \delta$;
18.         updateHeight();
19.         sendREP($H_{k,j}, repseq_j$);
20.         return;
21.     }
22. }
```

Fig. 3 Pseudocode for recvREP.

QRY and drops other QRYs because they have the same *ORI* and *SEQ* (lines 2–5 of Fig. 2).

When the destination node v_j receives a QRY from its neighbor v_m, it broadcasts REP=$< v_j, H_{j,j}, repseq_j, v_i >$ where v_i is the source node that initiated the route discovery phase and $H_{j,j} =< 0,0,0 >$ (lines 6–8 of Fig. 2). Figure 3 shows the pseudocode for the procedure executed when v_k receives a REP from v_l with destination v_j. When v_m receives the REP, it updates its neighbor's table with information from the REP (lines 5–6 of Fig. 3) if it has not received this message already. Then v_m updates its pseudo-distance to $\lambda_{m,j} = \lambda_{j,j} + \delta$, where $\lambda_{j,j} = 0$, and the corresponding α and β values because the previous height $H_{m,j}$ was null (lines 11–13 of Fig. 3). After v_m updates $H_{m,j}$, it broadcasts REP=$< v_j, H_{m,j}, repseq_j, v_i >$ to its neighbors (line 14 of Fig. 3). Note that although v_j also receives this REP message, it simply drops the message because v_j is the destination (lines 8–10 of Fig. 3). The neighbors of v_m may receive

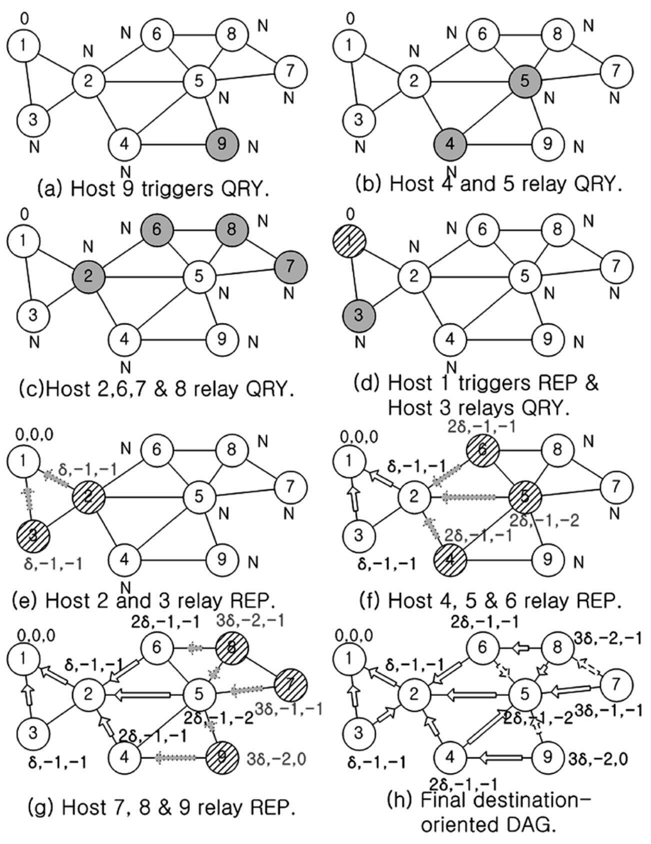

0
N
N
N
N
N
N
N
(a) Host 9 triggers QRY.
(b) Host 4 and 5 relay QRY.
(c)Host 2,6,7 & 8 relay QRY.
(d) Host 1 triggers REP & Host 3 relays QRY.
0,0,0
δ,−1,−1
2δ,−1,−1
2δ,−1,−2
(e) Host 2 and 3 relay REP.
(f) Host 4, 5 & 6 relay REP.
3δ,−2,−1
3δ,−1,−1
3δ,−2,0
(g) Host 7, 8 & 9 relay REP.
(h) Final destination-oriented DAG.

Fig. 4 An example of route discovery.

multiple REPs generated by v_j. Suppose that an intermediate node v_k receives REPs from v_l first and v_n last. When v_k receives a REP from v_l, it updates its pseudo-distance as $\lambda_{k,j} = \lambda_{l,j} + \delta$ (lines 11–15 of Fig. 3) because the previous value of $\lambda_{k,j}$ was null. Afterward, when v_k receives a REP from v_n, it compares $\lambda_{k,j}$ to $\lambda_{n,j}$ (line 16 of Fig. 3). If $\lambda_{k,j} - \lambda_{n,j} > \delta$, then v_k updates its pseudo-distance to $\lambda_{k,j} = \lambda_{n,j} + \delta$ and updates the corresponding α and β values (lines 17–18 of Fig. 3) because the path through v_n is expected to be shorter than the path through v_l. After updating its height, v_k broadcasts the REP to its neighbors as described line 19 of Fig. 3. Note that δ is the default one-hop difference between two adjacent nodes.

Figure 4 shows an example of the route discovery with destination node v_1. Shaded vertices represent nodes that broadcast QRY and slashed vertices represent nodes that broadcast REP. Solid arrows represent primary links, shaded and dotted arrows represent currently updated links and dashed arrows represent auxiliary links. Initially, in Fig. 4(a), v_9 triggers the route discovery phase by broadcasting QRY=$< v_1, v_9, 0 >$ to its neighbors. In (b), v_4 and v_5 forward the QRYs received (lines 12–13 of Fig. 2). In (c), v_2, v_6, v_7, v_8 also forward the QRYs received. In (d), v_1 broadcasts REP=$< v_1, H_{1,1}, 0, v_9 >$ (lines 6–8 of Fig. 2) because v_1 receives a QRY from v_2. In (e), v_2 and v_3 forward the REP received to their neighbors after updating their own routing tables with the new local pseudo-distance and corresponding α and β values (lines 11–15 of Fig. 3). Note that v_1 receives two REPs from v_2 and v_3, but simply drops the later messages (lines 8–10 of Fig. 3). In (f) and (g), other nodes keep forwarding REP to their neighbors and update their height values. Finally, (h) in Fig. 4 shows the resulting destination-oriented DAG directed toward v_1. Note that the pseudo-distances of the initial destination-oriented DAG are exactly proportional to actual distances.

3.6 Route Maintenance

When a node v_k detects that a link is broken, it does not react if it still has primary outgoing links. However, if v_k loses its last outgoing link toward destination node v_j, then v_k triggers a route maintenance phase. There are two options for the route maintenance phase. In order to provide shorter routes, v_k can trigger a route maintenance phase when it loses its last primary outgoing link even if v_k still has auxiliary outgoing links. On the other hand, in order to reduce the number of control messages required, v_k can trigger a route maintenance phase only when it loses all of its outgoing links, including auxiliary links. In the rest of this paper, we will simply assume primary-link-only routing as the alternative cases are simple adaptations from this simple case. Note that a node may lose its outgoing link when all outgoing links become disconnected or when it changes link directions in response to an update message received from a neighboring node.

Figure 5 shows the pseudocode of the procedure executed when node v_k loses its last outgoing link toward destination v_j. Since v_k has lost its last outgoing link, α is zero. If $\beta_{k,j} = 0$, then v_k does not have a neighbor such that $\lambda_{k,j} = \lambda_{l,j}$ for all $v_l \in N_k$. This case occurs when v_k does not have any auxiliary links. Therefore, v_k updates its pseudo-distance as $\lambda_{k,j} = min_{v_l \in N_k}(\lambda_{l,j}) + \delta$ in order to change some of its incoming links to primary outgoing links (lines 2–5 of Fig. 5). On the contrary, if $\beta_{k,j} > 0$ and there exist a node $v_l \in N_k$ such that $\lambda_{k,j} < \lambda_{l,j}$, then v_k updates its pseudo-distance as $\lambda_{k,j} = \lfloor(\lambda_{k,j} + min_{v_l \in N_k}(\lambda_{l,j}))/2\rfloor$, where $\lambda_{k,j} < \lambda_{l,j}$, in order to change its auxiliary links to primary outgoing links (lines 6–9 of Fig. 5). Note that other incoming links remains unchanged in these cases. Finally, if $\beta_{k,j} > 0$ but there is no node that satisfies the condition $\lambda_{k,j} < \lambda_{l,j}$ for all $v_l \in N_k$, then v_k updates its pseudo-distance as $\lambda_{k,j} = min(\lambda_{l,j}) + \delta$ to change all of its incoming links to primary outgoing links (lines 11–13 of Fig. 5). After updating $H_{k,j}$ in either case, v_k broadcasts an UPD to its neighbors to notify them of the change in its pseudo-distance.

The pseudocode of the procedure executed when v_l receives an UPD from v_m, with destination v_j, is shown in Fig. 6. When v_l receives an UPD from its neighbor $v_m \in N_l$, it first updates its own routing table. If $\lambda_{l,j}$ was previously null, v_l updates its $\lambda_{l,j}$ to $\lambda_{l,j} = \lambda_{m,j} + \delta$ (lines 8–12 of Fig. 6). On the other hand, v_l may lose its outgoing link if $\lambda_{l,j} \leq \lambda_{m,j}$ (line 13 of Fig. 6). In that case, v_l has to update its pseudo-distance as described in lines 14–17 of Fig. 6. Finally, if v_l receives an UPD that states that $\lambda_{l,j} - \lambda_{m,j} > \delta$, which indicates that a new shorter distance path to the destination has been found, v_l updates its pseudo-distance to $\lambda_{l,j} = \lambda_{m,j} + \delta$ (lines 18–22 of Fig. 6).

Figure 7 shows a simple example of route maintenance that demonstrates the need for the use of δ in PDR.

```
 1. lostLastPrimaryOutgoingLink() {
 2.     if{$\beta_{k,j} = 0$} {
 3.         $\lambda_{k,j} = min_{v_l \in N_k}(\lambda_{l,j}) + \delta$;
 4.         updateHeight();
 5.         sendUPD($H_{k,j}, updseq_k$);
 6.     } else if (there exist any $v_l$ nodes such that $\lambda_{k,j} < \lambda_{l,j}$ for all
        $v_l \in N_k$ ){
 7.         $\lambda_{k,j} = \lfloor(\lambda_{k,j} + min_{v_l \in N_k}(\lambda_{l,j}))/2\rfloor$, where $\lambda_{k,j} < \lambda_{l,j}$;
 8.         updateHeight();
 9.         sendUPD($H_{k,j}, updseq_k$);
10.     } else {
11.         $\lambda_{k,j} = min_{v_l \in N_k}(\lambda_{l,j}) + \delta$;
12.         updateHeight();
13.         sendUPD($H_{k,j}, updseq_k$);
14.     }
15.     return;
16. }
```

Fig. 5 Pseudocode of procedure executed when last outgoing link is lost.

```
 1. recvUPD(message p) {
 2.     if (isReceived(p)) {
 3.         drop(p);
 4.         return;
 5.     } else {
 6.         updateNeighbor(p);
 7.     }
 8.     if ($H_{l,j} = NULL$) {
 9.         $\lambda_{l,j} = \lambda_{m,j} + \delta$;
10.         updateHeight();
11.         sendUPD($H_{l,j}, p.SEQ$);
12.         return;
13.     } else if ($\lambda_{l,j} \leq$ p.$H.\lambda$) {
14.         if(isLostLastPrimaryOutgoingLink()) {
15.             lostLastPrimaryOutgoingLink(); {
16.             return;
17.         }
18.     } else if ($\lambda_{l,j}$−p.$H.\lambda > \delta$) {
19.         $\lambda_{l,j} = \lambda_{m,j} + \delta$;
20.         updateHeight();
21.         sendUPD($H_{l,j}, p.SEQ$);
22.         return;
23.     }
24. }
```

Fig. 6 Pseudocode for recvUPD.

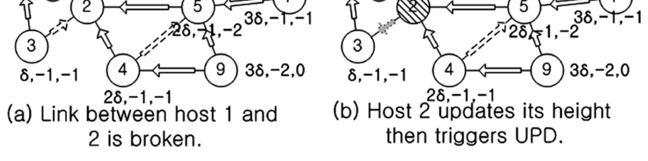


Fig. 7 An example of route maintenance.

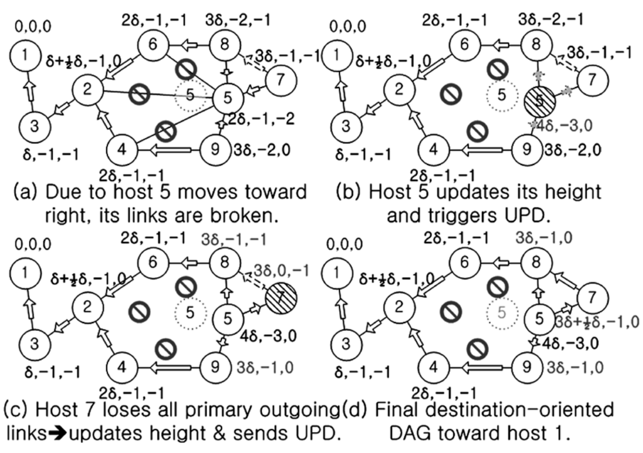

(a) Due to host 5 moves toward right, its links are broken.
(b) Host 5 updates its height and triggers UPD.
(c) Host 7 loses all primary outgoing links➔updates height & sends UPD.
(d) Final destination-oriented DAG toward host 1.

Fig. 8 A second example of route maintenance.

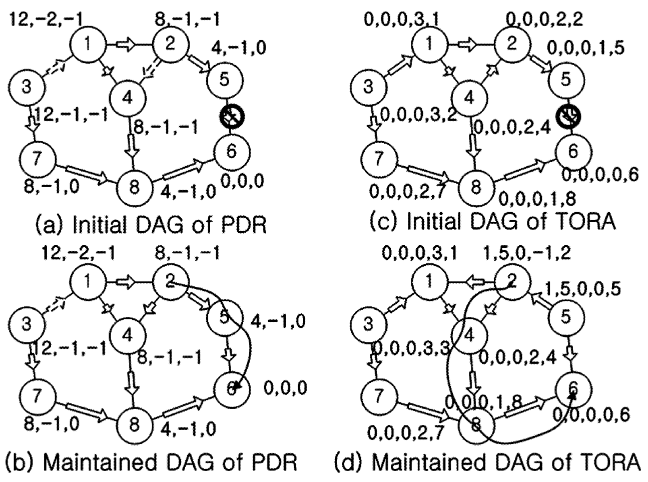


Fig. 9 Comparison of PDR with TORA.

Vertices filled in with backslash lines represent nodes that broadcast UPD. In (a), node v_2 loses its last primary outgoing link toward destination node v_1 because link $e_{2,1}$ becomes disconnected. In (b), v_2 updates its pseudo-distance to $\lambda_{2,1} = \lfloor(\lambda_{2,1} + min_{n_i \in N_2}(\lambda_{i,1}))/2\rfloor$, where $\lambda_{i,1} > \lambda_{2,1}$ (line 7 of Fig. 5) because $\beta_{2,1} = 1$ and there are three neighbors that have greater pseudo-distance values than v_2 and meet the conditions of line 6 in Fig. 5. Note that all neighbors v_i except v_3 have $\lambda_{i,1} = 2\delta$. The node that satisfies $min_{n_i \in N_2}(\lambda_{i,1})$, where $\lambda_{i,1} > \lambda_{2,1}$, is v_5. Suppose that δ is 1. $\lambda_{2,1} = \lfloor(\lambda_{2,1} + \lambda_{5,1})/2\rfloor$ is 1, which is not acceptable in PDR because it can not reverse any links attached to v_2. Thus, PDR has to choose $\lambda_{2,1} = 2$, which results in link reversal topological changes for nodes v_4, v_5, v_6, v_7, v_8 and v_9. In order to deal with such cases, δ should be a sufficiently large integer value. Note that only a single step is sufficient for convergence in this example. For the same example, TORA requires seven steps for convergence as described in the appendix.

Figure 8 shows another example of route maintenance. In (a), v_5 moves toward the right direction. Therefore, links $e_{5,6}, e_{5,2}$ and $e_{5,4}$ become disconnected. For simplicity, suppose that all three links are broken at the same time. Then v_5 updates its pseudo-distance to $\lambda_{5,1} = min_{v_l \in N_5}(\lambda_{l,1}) + \delta = 4\delta$ (lines 2–5 in Fig. 5). Also, $\alpha_{5,1}$ becomes 3 since v_5 has three neighbors with smaller pseudo-distance and $\beta_{5,1}$ becomes zero since v_5 does not have any neighbors with the same pseudo-distance. The intermediate DAG is shown in Fig. 8(b). When v_7 receives an UPD from v_5, it loses its last primary outgoing link because $e_{7,5}$ becomes a primary incoming link due to $\lambda_{5,1} = 4\delta$. Therefore, v_7 also has to update its pseudo-distance to reverse some of its links as shown in (c) of Fig. 8. Since v_7 loses its last outgoing link ($\lambda_{7,1} < \lambda_{5,1}$), it executes the "lostLastPrimaryOutgoingLink()" procedure (lines 13–16 of Fig. 6). Then, v_7 updates its pseudo-distance to $\lambda_{7,1} = \lfloor(\lambda_{7,1}+\lambda_{5,1})/2\rfloor$ (lines 6–9 of Fig. 5) because v_7 has a neighbor v_5 that has $\lambda_{5,1} > \lambda_{7,1}$ and $\beta_{7,1}$ is 1. The final destination-oriented DAG that results is shown in (d).

Figure 9 compares PDR with TORA. (a) in Fig. 9 shows the initial routes used by PDR when $\delta = 4$ and (c) shows the initial routes used by TORA for the same MANET. Suppose that the link $e_{5,6}$ becomes temporarily disconnected. Then v_5 updates its height in both algorithms; $H_{5,6} =< 12, -1, 0 >$ in PDR and $H_{5,6} =< 1, 5, 0, 0, 5 >$ in TORA. If the link $e_{5,6}$ is re-established later, then v_5 updates its pseudo-distance to find shorter paths in PDR as shown in (b) of Fig. 9. However, TORA does not update any height values because both v_5 and v_6 have their own height values. The destination-oriented DAG produced with TORA is shown in (d) of Fig. 9. The (2,6)-path selected by PDR is $P^{PDR}_{2,6} = (2, 5, 6)$ and the length of this path is $|P^{PDR}_{2,6}| = 2$. However, the (2,6)-path selected by TORA is $P^{TORA}_{2,6} = (2, 4, 8, 6)$ with length $|P^{TORA}_{2,6}| = 3$. Note that the distance of a (2,6)-path is $D_{2,6} = 2$ since that is the length of the shortest path from node 2 to node 6. As stated in Sect. 2.2, TORA may produce paths with increased path lengths after a few topological changes.

4. Evaluation

Simulations were conducted to evaluate the benefits and costs of the PDR routing algorithm using ns-2 [11], which is a discrete event simulator tool commonly used in networking research. We compared the performance of PDR with TORA because TORA is a previously proposed link reversal algorithm. The other algorithms discussed in Sect. 2 are not compared because they all have major drawbacks, such as high overhead and being able to provide only one path to each destination, and thus are not directly comparable to our approach.

4.1 Simulation Environment

The distributed coordination function (DCF) of IEEE 802.11 [12] for wireless LANs is used for the MAC and PHY layers. The data rate is set to 11 Mbps that is a rate widely supported by 802.11b devices. The simulation space is a 500 m × 300 m area, and the communication range of

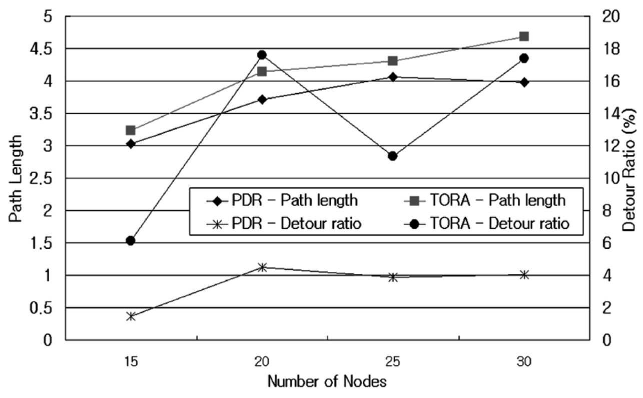


Fig. 10 Path length vs. number of nodes.

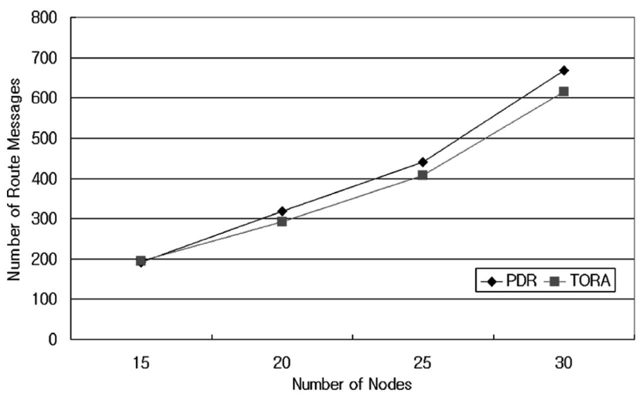


Fig. 11 Number of control messages vs. number of nodes.

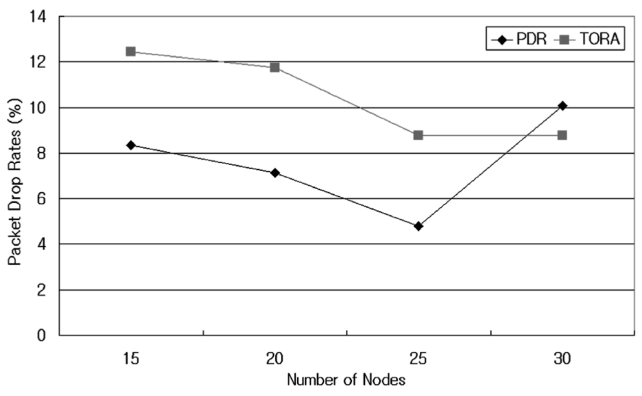


Fig. 12 Packet drop rates vs. number of nodes.

each node is set to 130 m. The mobility of the nodes are controlled by a mobility generator function in ns-2 that uses a random destination model [13]. Each node starts its journey from a random location to a random destination with a randomly chosen speed that is uniformly distributed between 0 and a maximum speed. When a node arrives at its destination, it stays there for some time and then restarts its journey to another random destination with a randomly chosen speed. Finally, the simulation time is set to 130 seconds. A source sends 256 bytes of UDP packet to its destination every 1 second from 10 seconds after the simulation starts to 125 seconds. The maximum speed of each node and the number of nodes are varied for each simulation scenario. Data are collected for ten different simulation scenarios.

4.2 Simulation Results versus the Number of Nodes

Figure 10 shows the average path length of PDR and TORA versus the number of nodes when nodes are moving up to 5 m/s. We track each successfully delivered packet and build the path $P_{src,dst}$ that the packet passes through. The detour ratio is defined as $(|P| - D)/D$, where $|P|$ is the length of the path as defined in Sect. 3.2 and D is the distance of the path as defined in Sect. 3.2. As shown in Fig. 10, the detour ratio of PDR is less than 5% but that of TORA is about 4 times larger than PDR. This is because PDR tries to keep pseudo-distances as small as possible even if it introduces more control messages.

Figure 11 shows the average number of control messages that PDR and TORA generate versus the number of nodes. Due to the effects of δ as described in the previous section, control messages can be decreased for some types of topological changes. However, PDR generates more control messages than TORA because each node exchanges its height information with a neighbor whenever a new link is established between two nodes. Note that there is a trade-off relation between the number of control messages generated and the detour ratio.

Figure 12 shows the average packet drop rates for PDR and TORA. The packet drop rate is calculated as the number of dropped packets divided by the total number of messages sent by the sender. In most cases, PDR shows lower packet drop rates because PDR attempts to choose a paths with more alternative paths to each destination. In the average for all cases, PDR loses 7.587% of its packets but TORA loses 10.435% of its packets. However, the packet drop rate for PDR is slightly greater than TORA when the number of nodes is 30 because the shortest route is unstable in a high-density network due to the edge effect [14]. Note that, unlike TORA, PDR always attempts to choose paths that are as short as possible. Therefore, in some mobility scenarios (two out of ten), PDR shows worse packet drop rates than TORA.

4.3 Simulation Results versus Mobility of Nodes

Figure 13 shows the effect of mobility on path lengths when the number of nodes is fixed at 30. As in Fig. 10, PDR shows better performance than TORA in terms of both path lengths and detour ratios.

Figure 14 shows the average number of control messages generated versus node mobility. As the maximum node speed is increased, the number of topological changes also increases. Therefore, the number of control messages required for both PDR and TORA will also tend to increase. However, due to the effect of δ as described in the previous section, PDR does not require control messages for some types of topological changes. Therefore, the actual number of control messages required for PDR are comparable to

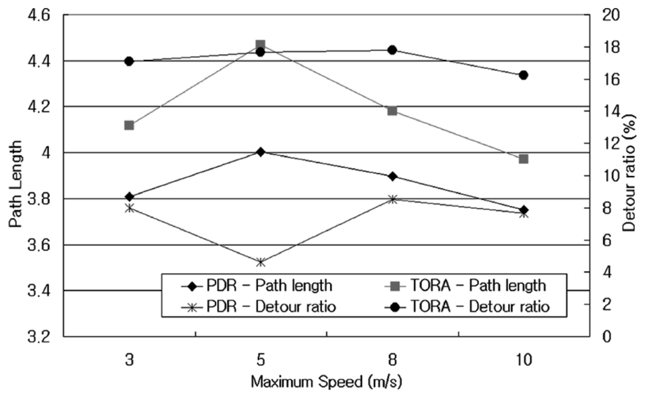


Fig. 13 Path length vs. node mobility.

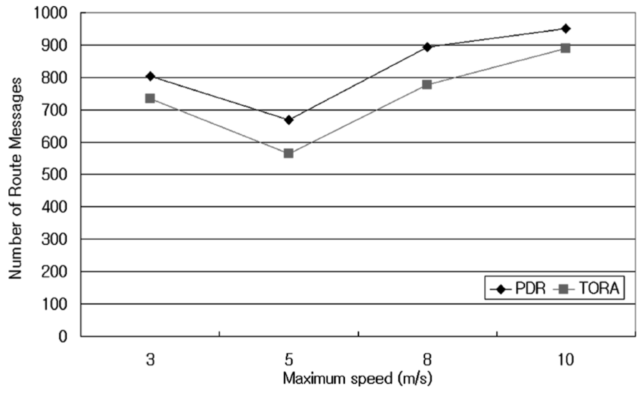


Fig. 14 Number of control messages vs. node mobility.

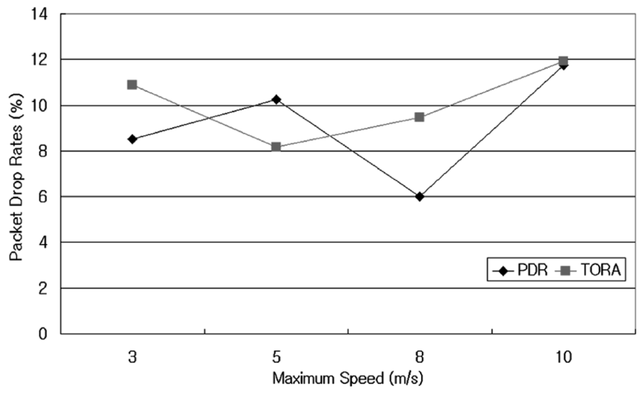


Fig. 15 Packet drop rates vs. node mobility.

TORA even though PDR exchanges messages whenever a new link is established.

Figure 15 shows average packet drop rates versus node mobility. On average, PDR loses 9.13% of its packets while TORA loses 10.109% of its packets. In the 5 m/s case, PDR shows a worse packet drop rate due to the same reason as in Fig. 12.

5. Conclusion

In this paper, we proposed a new routing algorithm, termed *pseudo-distance routing* (*PDR*), that discovers and maintains short-distance, multiple paths to each destination in a mobile ad-hoc network. Analysis of example cases and simulation results show that PDR provides shorter paths than TORA (a previously proposed routing algorithm with features similar to PDR), but that PDR may require a few more control messages. In addition, PDR results in a lower packet drop rate, on average, than TORA because PDR chooses paths with more alternative sub-paths to each destination. Due to these features and the fact that TORA is better than previously proposed algorithms for MANETs, it is claimed that PDR is a practical routing algorithm for MANET environments.

Acknowledgments

This work was supported by Korea Research Foundation Grant (KRF-2004-041-D00586).

References

[1] C. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers,” Proc. ACM SIGCOMM, pp.234–244, Oct. 1994.

[2] S. Murthy and J.J. Garcia-Luna-Aceves, “A routing protocol for packet radio networks,” Proc. ACM/IEEE MOBICOM, pp.86–95, Nov. 1995.

[3] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla, “Routing in clustered multihop, mobile wireless networks,” IEEE Singapore Int’l Conf. on Networks, pp.197–211, 1997.

[4] C. Perkins and E. Royer, “Ad-hoc on-demand distance vector routing,” Proc. IEEE Workshop on Mobile Computing Systems and Applications, Feb. 1999.

[5] D.B. Johnson and D.A. Maltz, “Dynamic source routing in ad hoc wireless networks,” in Mobile Computing, Kluwer Academic Publishers, 1996.

[6] Y.B. Ko and N.H. Vaidya, “Location-aided routing (LAR) in mobile ad hoc networks,” Proc. ACM/IEEE MOBICOM, pp.66–75, 1998.

[7] V.D. Park and M.S. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks,” Proc. IEEE INFOCOM, pp.1405–1413, 1997.

[8] E. Gafni and D. Bertsekas, “Distributed algorithms for generating loop-free routes in networks with frequently changing topology,” IEEE Trans. Commun., vol.COM-29, no.1, pp.11–18, Jan. 1981.

[9] M.S. Corson and V. Park, “An Internet MANET encapsulation protocol (IMEP),” http://tools.ietf.org/html/draft-ietf-manet-imep-spec-00.txt, Internet draft, Aug. 1998.

[10] M.K. Marina and S.R. Das, “On-demand multipath distance vector routing in ad hoc networks,” Proc. IEEE Int’l Conf. on Network Protocols, pp.14–23, Nov. 2001.

[11] VINT Group, http://www.isi.edu/nsnam/ns

[12] IEEE Standards Department, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Standard 802.11-1997.

[13] I. Rubin and Y.-C. Liu, “Link stability models for QoS ad hoc routing algorithms,” Proc. IEEE VTC2003, Fall, pp.3084–3088, Oct. 2003.

[14] G. Lim, K. Shin, S. Lee, H. Yoon, and J.S. Ma, “Link stability and route lifetime in ad-hoc wireless networks,” Proc. IEEE Int’l Conf. on Parallel Processing Workshops, pp.116–123, Aug. 2002.

Appendix: TORA Example

This figure shows the route maintenance phase for the same MANET topology as that shown in Fig. 7. As stated above, PDR only requires a single-step operation, while TORA requires 7 steps for convergence. The details of each step are described in Fig. A· 1.

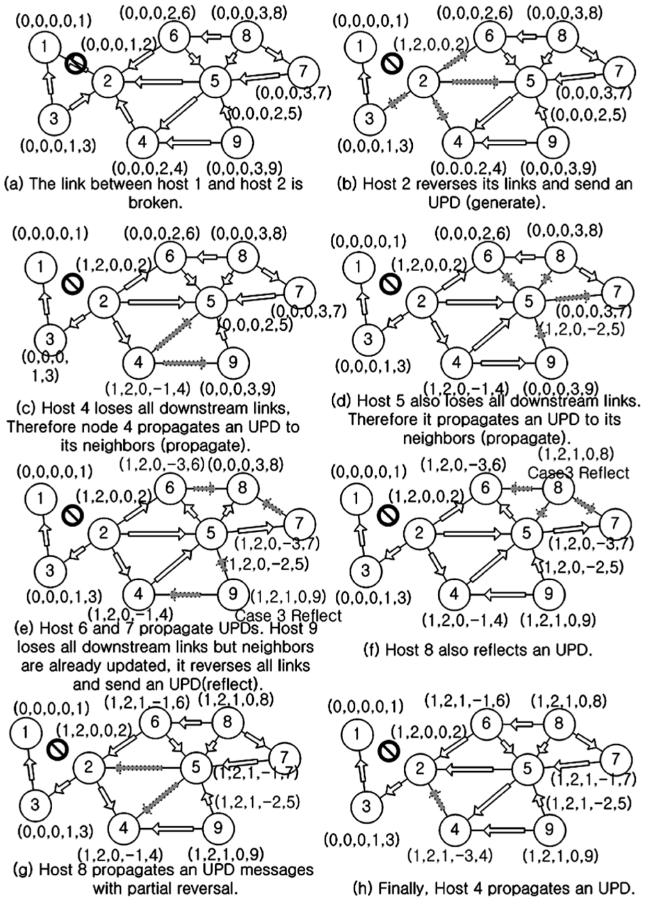
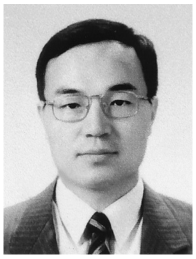

(a) The link between host 1 and host 2 is broken.
(b) Host 2 reverses its links and send an UPD (generate).
(c) Host 4 loses all downstream links, Therefore node 4 propagates an UPD to its neighbors (propagate).
(d) Host 5 also loses all downstream links. Therefore it propagates an UPD to its neighbors (propagate).
(e) Host 6 and 7 propagate UPDs. Host 9 loses all downstream links but neighbors are already updated, it reverses all links and send an UPD(reflect).
Case 3 Reflect
(f) Host 8 also reflects an UPD.
Case3 Reflect
(g) Host 8 propagates an UPD messages with partial reversal.
(h) Finally, Host 4 propagates an UPD.

Fig. A· 1 TORA example with the MANET of Fig. 7.



Sunggu Lee received the B.S.E.E. degree with highest distinction from the University of Kansas, Lawrence, in 1985 and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1987 and 1990, respectively. He is currently a visiting researcher at the University of California at Irvine, where he is spending his sabbatical leave from his full-time position as a professor in the Department of Electronic and Electrical Engineering at the Pohang University of Science and Technology (POSTECH), Pohang, Korea. Prior to this appointment, he was an Assistant Professor in the Department of Electrical Engineering at the University of Delaware in Newark, Delaware, U.S.A. From June 1997 to June 1998, he also spent one year as a visiting scientist at the IBM T.J. Watson Research Center. His current research interests are in mobile ad-hoc networks, grid computing and real-time computing.

Min-Gu Lee received B.S. degree in school of electronic, communication and radio wave engineering from the Hanyang University, Seoul, Korea in 2000 and the M.S. degree in department of electronic and electrical engineering from the Pohang University of Science and Technology (POSTECH), Pohang, Korea in 2002. He is currently a Ph.D. candidate at POSTECH. His current research interests are mobile ad-hoc networks, real-time computing and distributed computing.