

ROUTING TABLE MAINTENANCE IN MOBILE AD HOC NETWORKS

Seungjin Park*, Seong-Moo Yoo **

* Department of Management, MIS, and CS, University of Southern Indiana, USA

** Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, USA
park@usi.edu, yoos@ece.uah.edu

Abstract— *Due to the mobility of the nodes, finding routes in a wireless mobile ad hoc network is quite a challenging task, and takes up a lot of system resources such as bandwidth and battery power. Although it would be wise to take full advantage of the already discovered paths, many algorithms simply discard the paths if their topology has changed. To overcome this shortcoming, we have proposed a routing table maintenance algorithm that responds to the changes in network topology promptly, and adjusts the paths so that their lifetimes could be maximized..*

Keywords— Mobile ad hoc network, routing, routing table, unicast

I. INTRODUCTION

A mobile ad hoc network (MANET) consists of wireless mobile hosts without any centralized control point or fixed infrastructure. Since a MANET does not require any prior arrangement, it can be quickly deployed and used in such applications including fast establishment of military communication and rescue missions where an established network is neither feasible nor available [1].

Unlike wired stationary networks, finding a path from the source host (or node) to the destination host is quite challenging in a MANET due to the mobility of hosts. Nodes implementing a *proactive* algorithm [2] periodically exchange local information with other nodes so that each node maintains a routing table that contains the routes to all nodes in the network. However, this information exchange takes a lot of bandwidth and battery power. On the other hand, nodes in *reactive* algorithms [3, 4, 5] do not maintain any routing information. Instead, a node dynamically discovers a path to the destination on demand. Therefore, it may reduce considerable overhead to enhance the network throughput.

Although other routing algorithm could be used, this paper is based on DSR [5]. DSR is reactive and has been proved it could be as competitive as any other algorithms in the field.[7, 8, 9]. Like many reactive routing protocol, DSR consists of two phases: *route discovery phase* and *data delivery phase*. When node S has a data for node D, S starts

finding a path to D in the route discovery phase. Once a route is found, S sends data along the path in the data delivery phase. In the route discovery phase S broadcasts a control packet called a *route request packet* (REQ) to all its neighbor nodes. If a neighbor node does not have route information to D, it appends its information to the REQ and relays the REQ to all its neighbor nodes, and so on until the REQ reaches D. (This forwarding of requests from one node to all its neighbors is called “flooding” and consumes a lot of time and bandwidth.) If the REQ reaches either D or a node that contains the path information to D, the node sends a control packet called *route reply packet* (RPY) back to S by reversing the path stored in the REQ. On receiving the RPY, S knows that the path to D has been established, and S sends its data packet along the path. For example, suppose node A in Figure 1 tries to find a path to G that is not found yet. As a first step, A prepares an REQ with its ID appended to it, and sends it to its neighbors including B. If B does not have any information about G, B appends its ID to the REQ and retransmits it to its neighbors including C, and so on until it reaches G. The snapshot of the REQ at E would contain the nodes it passed through thus far, which is A, B, C, D. When the REQ reaches the destination node G, G prepares a RPY which is transmitted to A along the path stored in the REQ.

For example, on receiving the RPY, node A in Figure 1 stores the path (A, B, C, D, E, F, G), and node C stores the path (C, D, E, F, G) in their routing tables so that they can use it later. As mentioned before, it takes a lot of resources to discover a path in a MANET. Therefore, it would be wise to take advantage of it as much as possible. However, most routing protocols do not efficiently utilize the existing paths, because they do not respond properly and promptly to the changes in the network topology. In this paper, we propose a strategy that tries to use the existing paths efficiently by modifying route table as quickly as possible.

Wu et. al. [6] proposed an algorithm that tries to fix some problems in paths such as finding shorter paths and fixing broken links. However, nodes in their algorithm stores only the destination, next hop, hop count, etc. for each path so that the coverage of the algorithm is limited.

This paper is organized as follows. Terminology needed to understand this paper is explained in section 2. Proposed algorithm is presented in section 3, followed by the theoretical analysis in section 4. Section 5 presents the conclusion.

II. TERMINOLOGY

A *Graph* can be represented by $G(V, E)$, where V is the set of nodes and E is the set of links in the graph G . In a wireless network, if a node (or host) B is within the transmission range of node A , then it is said that there is a link from A to B which is denoted as (A, B) . A is an *upstream* node of B , and B is the *downstream* node of A . B is also called as a *neighbor* of A . If (A, B) implies (B, A) , then the link is called *bidirectional*, and *unidirectional* otherwise.

A *path* $P(v_1, v_2, \dots, v_{n-1}, v_n)$, where $v_i \in V, 1 \leq i \leq n$, consists of a set of links $(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_n)$. A *subpath* of $P(v_1, v_2, \dots, v_{n-1}, v_n)$ is the path that is a part of P . Function *rem* (P, v_i) returns the portion of path P starting at node v_i to the end of P . For example, both $P(B, C, D, E)$ and $P(A, B)$ are subpaths of $P(A, B, C, D, E)$, and *rem* ($P(A, B, C, D, E), B$) is $P(B, C, D, E)$.

$|P|$ denotes the number of the links in path P . For example, $|P(A, B, C, D, E)|$ is four, since P consists of four links.

Note that if the links are bidirectional (unidirectional, resp.), then the paths are also bidirectional (unidirectional, resp.). Although links assumed to be bidirectional in this paper, paths are assumed to be unidirectional for simplicity in explanation. It can be easily extended to cover the handling of unidirectional paths.

When a new path is found, each node in the path may contain the path information in its Routing Table for possible later use. This can be done without any extra expense at each node by simply storing the path information when the RPY passes through the node. For example in Figure 1, nodes A, B, C, D, E , and F in the path $P(A, B, C, D, E, F, G)$ store the path information when the RPY passes through them.

III. PROPOSED ROUTING TABLE MAINTENANCE SCHEME

Based on the assumption that all links are bidirectional, the proposed scheme is hybrid, since it discovers new paths reactively but the maintenance of the paths is proactive. In other words, a new path from A to B is discovered only when 1) A wants to send data to B , and 2) there is no existing path from A to B . On the other hand, any change of the existing paths is updated immediately regardless of the usage of the path. This strategy makes sense, because path discovery takes a considerable amount of resources due to the flooding of REQ packets, whereas maintenance requires unicasts only.

In this paper, only the routing table maintenance scheme is presented, since route discovery algorithms can be found in many routing protocols. The routing table maintenance is explained below for different cases.

Case 1) Link breakage is discovered. Consider two paths, $P(A, B, C, D, E, F, G)$ and $P(H, I, C, D, E, F, G)$, in the network as shown in Figure 1. Suppose when node C sends a data packet to node G along $P(A, B, C, D, E, F, G)$, C found out that the link (E, F) is broken. Then, this information is disseminated to all upstream nodes of both paths. On receiving the link breakage information nodes A, B, H , and I delete the path from their routing tables. Otherwise, attempting to use this broken link may cause delay.

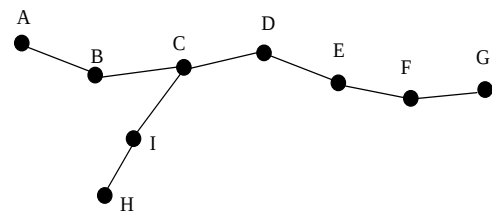
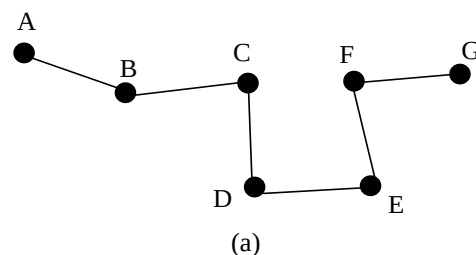


Figure 1. A There are two paths, $P(A, B, C, D, E, F, G)$ and $P(H, I, C, D, E, F, G)$ in the networks.

Case 2) A shorter path is found. There are two possible cases. First, C in Figure 2 (a) found a shorter path (A, B, C, F, G) in the existing path (A, B, C, D, E, F, G) , when F moves into C 's transmission range. In this case, C substitutes (A, B, C, D, E, F, G) with $P(A, B, C, F, G)$ and $P(A, B, C, D, E)$ in its routing table. Note that the upstream nodes of C in the path need not be informed this finding. Instead, if a packet from either A or B destined to either F or G is arrived at C , C directs the packet to F using $P(A, B, C, F, G)$, not to D . If the packet is destined to D or E , then the packet is transferred along $P(A, B, C, D, E)$.

Figure 2 (b) shows the other case where a shorter path is discovered by other newly found path. For example, suppose $P(A, B, C, D, E, F, G)$ is the existing path, and $P(H, C, J, F, K)$ is just discovered. Then, $P(A, B, C, D, E, F, G)$ will be replaced by $P(A, B, C, J, F, G)$ and $P(A, B, C, D, E)$. Note that, as was done in Case 2 (a), this information is not informed to the upstream nodes of C .



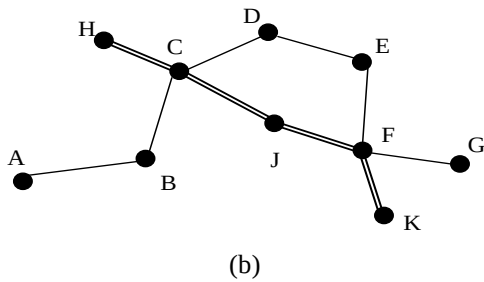


Figure 2. (a) A shorter path $P(A, B, C, F, G)$ is found in the same path. (b) A shorter path $P(A, B, C, J, F, G)$ is found by other path $P(H, C, J, F, K)$.

Case 3) An extension of an existing path, say P_1 , is found by the other node in the path. In this case, the extended portion is disseminated to P_1 . For example, suppose $P_1 = P(A, B, C, D, E)$ in Figure 3 is the existing path, and $P(F, G, H, E, J, K)$ is discovered later. Then, extended portion $P(E, J, K)$ is disseminated to nodes in P_1 . As a result, for example node A, replaces $P(A, B, C, D, E)$ with $P(A, B, C, D, E, J, K)$.

A more general form of Case 3 can be found in Figure 4, where $P_1 = P(A, B, C, D, E)$ is the existing path and $P_2 = P(F, G, H, C, D, J, K)$ is the newly found path. Then, nodes A and B are informed with the newly found portion $P(D, J, K)$ so that, for example, node A can add another entry $P(A, B, C, D, J, K)$ to its Routing Table. Note that nodes C and D already obtained the new path information and modified their routing tables when the RPY from node K to F passed through them.

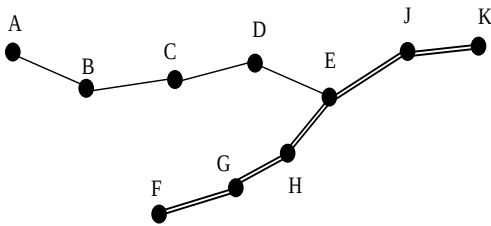


Figure 3. As a result of the discovery of another path $P(F, G, H, E, J, K)$, existing path $P(A, B, C, D, E)$ is extended to $P(A, B, C, D, E, J, K)$ and this information will be stored at nodes A, B, C, D, and E.

Not only the existing paths but also newly found paths can obtain more path information when it receives RPY. For example, node F adds new entry $P(F, G, H, C, D, E)$ to its Routing Table in addition to $P(F, G, H, C, D, J, K)$, if node C appends $P(C, D, E)$ to the RPY from K to F.

Other variations of Case 3 are shown in Figure 5, and 6 with a short explanation. When an existing path is the subpath of newly found path as shown in Figure 5, then the existing path will be replaced by the new path at each node in the existing path. For example, nodes B, C, D, E in the existing path.

existing path in Figure 5 replace their routing table entry of $P(B, C, D, E)$ with $P(A, B, C, D, E, F, G)$.

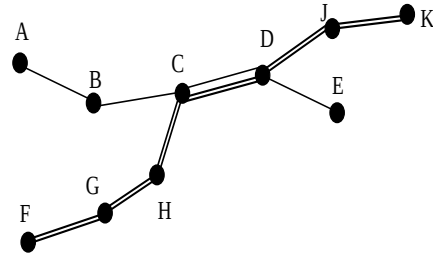


Figure 4. $P(A, B, C, D, E)$ and $P(F, G, H, C, D, J, K)$ are the existing and the newly found paths, respectively, where the intermediate nodes, C and D, in the existing path $P(A, B, C, D, E)$ are the part of newly found path $P(F, G, H, C, D, J, K)$.

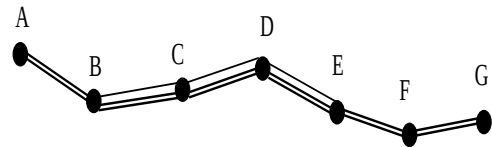


Figure 5. An existing path $P(B, C, D, E)$ is the subpath of newly found path $P(A, B, C, D, E, F, G)$.

As an another example, if two paths share a common subpath such that the first path is the existing path and the second is the newly found path as shown in Figure 6, then all nodes in the existing path should replace their entry in their routing table with the one that includes the newly found path. For example, nodes A, B, C, D, E in the existing path replace their routing table entries of $P(A, B, C, D, E)$ with $P(A, B, C, D, E, F, G)$.

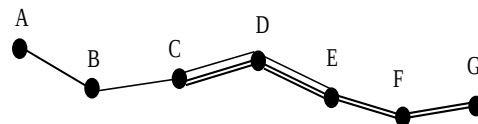


Figure 6. Two paths $P(A, B, C, D, E)$ and $P(C, D, E, F, G)$ share a subpath $P(C, D, E)$.

Summary of the proposed algorithm is presented in Figure 7.

Algorithm RoutingTable_Maintenance

At current node V_1

Case when a broken link is found

if V_1 found out that an existing path P is broken between node A and B then for each path P in the Routing Table that contains a link (A, B)

$P \leftarrow \text{rem}(P, B)$

Sends a Broken Link Message to its upstream nodes, if any, of all paths that contain (A, B) .

if V_1 received a broken link message from its downstream node in an existing path, then for each path P in the Routing Table that contains a link (A, B)

$P \leftarrow \text{rem}(P, B)$

Sends a Broken Link Message to its upstream nodes, if any, of all paths that contain (A, B) .

Case when there is a shorter path

// On receiving a RPY, V_1 check if there is a shorter path for each path in the routing

// table using the path in the RPY.

Let the path in the RPY be $P(v_1, v_2, \dots, v_{n-1}, v_n)$. Note that the current node is v_1 , since the first node in the RPY is v_1 . If there is a path $P(u_1, u_2, \dots, u_{m-1}, u_m)$ in the routing table such that 1) $v_1 = u_i$ and $v_n = u_j$, and 2) $|P(v_1, v_2, \dots, v_{n-1}, v_n)| \leq |P(u_i, u_{i+1}, \dots, u_{j-1}, u_j)|$, then replace the path with $P(u_1, u_2, \dots, u_{i-1}, v_1, v_2, \dots, v_{n-1}, v_n, u_{j+1}, \dots, u_{m-1}, u_m)$, and add new path $P(u_1, u_2, \dots, u_{i-1}, u_i, \dots, u_j)$.

Case when the existing path is extended or modified by the path in the RPY

// On receiving a RPY, call **check_modification_path** (path in the RPY), where each

// path in the routing table is checked if the path should be further modified due to the

// path in the RPY.

Let the path in the RPY be $P(v_1, v_2, \dots, v_{n-1}, v_n)$.

// For the cases shown in Figure 3 and 4

If there is a path $P(u_1, u_2, \dots, u_{m-1}, u_m)$ in the routing table such that $v_1 = u_i$, add $P(u_1, u_2, \dots, u_{i-1}, u_i, v_2, \dots, v_{n-1}, v_n)$ to the routing table.

// For the cases shown in Figure 5 and 6

If there is a path $P(u_1, u_2, \dots, u_{m-1}, u_m)$ in the routing table such that $u_i = v_1$, $u_{i+1} = v_2, \dots, u_m = v_{m-i+1}$, $n > m-i+1$, then modify $P(u_1, u_2, \dots, u_{m-1}, u_m)$ to $P(u_1, u_2, \dots, u_{m-1}, u_m, v_{m-i+2}, \dots, v_n)$.

Figure 7. Pseudo-code for the proposed algorithm.

IV. ANALYSIS OF THE PROPOSED ALGORITHM

This section presents the time complexity of the proposed algorithm in section 4.1, and the impact on the routing in section 4.2.

A. TIME COMPLEXITY OF ROUTING TABLE MAINTENANCE

Let us consider the time complexity of the routing table maintenance at each node. If a node A detects a new neighbor node B , then A checks if B is in any path in its routing table so that A can adjust the existing paths. This searching process takes $O(nm)$, where n is the number of paths in the routing table and m is the average path length. The searching time would be improved to $O(1)$, if the nodes in the paths are stored in a hash table using additional storage. Note that communication takes much longer time and more battery power than computation. Therefore, it would worth the computation if it results in reducing any amount of path length.

Once the path is found, actual adjustment of the routing table would take $O(1)$. If the path falls into one of the Case 3, then the new discovery is transmitted to the source node. Note that the transmission is unicast which is much cheaper than broadcast in wireless communication, because there is a single designated destination node within the source's transmission range in unicast, whereas all nodes are the destinations in broadcasting.

B. IMPACT ON ROUTE DISCOVERY

In this section, we will show how our proposed algorithm can reduce communication overhead and battery power when a new path is being discovered.

Suppose N is the average number of neighbors of each node in the network. If an algorithm abandons the path on its topological changes, then it would take at least $N^0 \cdot N^1 \cdot \dots \cdot N^{L-1} \approx N^{2L}$ REQ transmissions to find a path of length L all over again by using broadcast. On the other hand, if an algorithm can adjust to the minor topological changes of paths like our proposed algorithm, then it would take only L transmissions for the path of length L . Therefore, the advantage of having the proposed algorithm is obvious. This analysis would be true, if the changed path is used, otherwise, the adjustment would be a waste.

V. CONCLUSIONS

Due to the mobility of the nodes, routing in a wireless mobile ad hoc network is quite a challenging task. Many routing algorithms, both proactive and reactive, take up a lot of system resources in discovering paths between hosts. Although it would be wise to take full advantage of the discovered paths, many algorithms simply discard the paths if they have changed

their topology. To overcome this shortcoming, we have proposed a routing table maintenance algorithm that responds to the changes in network topology as promptly yet efficiently as possible to fully utilize existing paths by adjusting them.

Our simple analysis shows that the proposed algorithm reduces number of packet transmissions, and therefore saves resources such as bandwidth and battery power.

REFERENCES

- [1] C. Perkins, "Ad Hoc Networking," *Addison-Wesley*, 2001.
- [2] C. Perkins, and P. Bhagwat, "Highly-dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," pp. 234-244, *SIGCOM'94*, 1994.
- [3] C. Perkins, E. Royer and S. Das, "Ad hoc on demand distance vector (AODV) routing," *Internet draft*, IETF, Oct. 1999.
- [4] V. Park and M. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," *In Proc. IEEE Infocom*, pp.1405-1413, Apr. 1997
- [5] D. Johnson and D. Maltz, "Dynamic source routing in ad-hoc wireless networking," in *Mobile Computing*, T. Imielinski and H. Korth, editors *Kluwer Academic Publishing*, 1996.
- [6] S. Wu, S. Ni, Y. Tseng, and J. Sheu, "Route maintenance in a wireless mobile ad hoc network," *Proc. 33 Hawaii International Conference on System Sciences*, 2000.
- [7] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "Performance comparison of multi-hop wireless ad hoc network routing protocols," *MobiCom '98*, 1998.
- [8] A. Rahman, S. Islam, and A. Talevski, "Performance measurement of various routing protocols in ad-hoc network," *Proc. International Multiconference of Engineers and Computer Scientists*, 2009.
- [9] G. Jayakumar and G. Gopinath, "Performance Comparison of two on-demand routing protocols for ad-hoc networks based on random way point mobility model," *American Journal of Applied Sciences*, 5 (6), pp. 659 – 664, 2008.