A Looping Problem in the Tree-Based Mobility Management for Mobile IP Supported Ad Hoc Networks

Trung-Dinh Han and Hoon Oh

Abstract: **A loop can take place in the process of managing tree topology for mobility management of mobile nodes in infrastructure-based mobile ad hoc networks. The formation of a loop degrades an effective bandwidth of the wireless network by passing an identical message repeatedly within the same loop. Therefore, the loop should be resolved to revert the system back to the normal state. In this paper, we propose a simple and novel mechanism that detects and resolves a loop quickly by tracking the depth of trees. The mobility management approach that employs the loop resolution method is evaluated comparatively with the original tree-based one and the hybrid one. It is shown that the proposed approach far outperforms the other approaches, and it is robust against the rapid changes in network topology.**

Index Terms: **Ad hoc, looping problem, mobile IP, mobility management, tree-based.**

I. INTRODUCTION

Mobility management of mobile nodes (MNs) is required for infrastructure-based mobile ad hoc networks so that they can communicate with other hosts in the Internet or MNs in other mobile ad hoc networks. Since MNs have to register with Internet gateway (IG) and update their mobility periodically via wireless multi-hop, mobility management usually incurs a high network overhead. Various mechanisms to pursue efficiency of mobility management have been proposed during the last decade [1].

The traditional approaches to mobility management are generally categorized into three types: Proactive schemes [2]–[5], reactive schemes [6], [7], and hybrid schemes [8]–[10]. In the proactive schemes, IG periodically floods the network with advertisement messages, establishing registration path. Then, an MN performs registration with the IG along the registration path at regular intervals. Since this method keeps all MNs registered regardless of whether they participate in active communication, it is advantageous that the remote hosts can make a connection request via IG at any time. However, an MN may fail to register since registration path can be broken before the beginning of the next interval, at which the MN initiates registration. In the reactive schemes, only when an MN wants to communicate with remote hosts via the Internet, it floods the network with solicitation messages to search for an IG. When the MN receives an acknowledgement message that sets up registration path, it performs registration along the path. The flooding tends to increase network overhead significantly. Thus, the hybrid scheme was proposed to reduce the range of the flooding [9]. An IG floods the network with advertisement messages, but only within a limited range of hops, say k hops, whereas MNs residing outside k hops flood the network with solicitation messages to locate a registered MN that has a path to an IG so that it can register with the IG via the registered node. This approach reduces the flooding range, but only to a certain degree.

Recently, the tree-based mobility management approaches [11]–[15] have been appealing due to their low network overhead and high scalability since they do not employ flooding. An MN registers with an IG along tree paths to the IG periodically. Despite that the tree-based approach has been studied in various ways for its successful employment, the looping problem has not been addressed yet that takes place in the process of maintaining tree topology. If a loop is formed in tree topology, network overhead increases rapidly, degrading an effective network bandwidth quickly. Therefore, an efficient mechanism to solve the looping problem is required. However, solving the looping problem is not easy because of the synchronization delay[1] of link information between MNs in a tree.

In this paper, we propose a method to detect and resolve the looping problem by tracking and limiting tree depth. The proposed method not only reduces the possibility of loop formation greatly, but also resolves the loop quickly.

The tree-based approach that employs the proposed loop resolution scheme is evaluated by resorting to simulation, comparing with the original tree-based approach [11] and the hybrid approach [9]. According to our simulation results, the proposed loop resolution mechanism improves node registration, latency, and jitter as well as reduces network overhead significantly.

Section II describes the network model, the associated definitions, and a looping problem. The proposed method for detecting and resolving the looping problem is described in Section III. We evaluate performance by resorting to a simulation in Section IV and make concluding remarks in Section V.

II. BACKGROUND

A. Network Model and Definitions

In this paper, we consider an infrastructure-based mobile ad hoc network that consists of IGs and MNs. The considered network is maintained as a set of the trees, each originating in an IG and consisting of one IG and multiple MNs. We assume that nodes can overhear packets that are transferred between other

Manuscript received February 05, 2010; approved for publication by Christos Douligeris, Division II Editor, March 21, 2011.

H. Oh is a corresponding author.

The authors are with the Department of Computer Engineering and Information Technology, University of Ulsan, Korea, email: trungdinhvn@yahoo.com, hoonoh@ulsan.ac.kr.

[1] Synchronization delay indicates that if a link between any two nodes comes to being or is broken, the other nodes know the change only after some time interval, but not immediately.

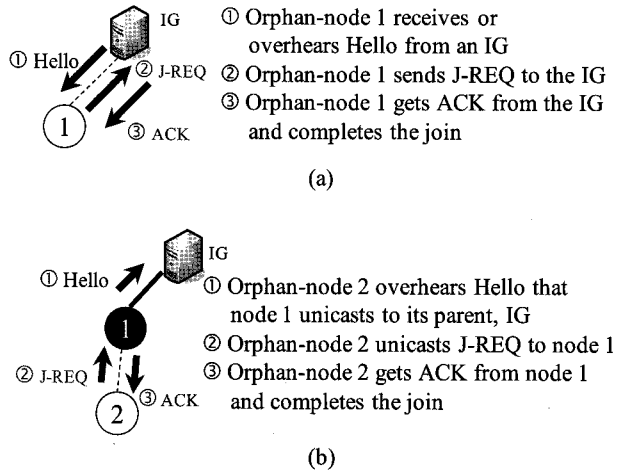


Fig. 1. The join-handshaking process: (a) An orphan-node that joins IG and (b) an orphan-node that joins a tree-node.

nodes [16]. A node is said to be a tree-node if it belongs to any tree and to be an orphan-node if it does not have a parent (even though it has children). A link that represents a parent-child relationship in a tree is specially called a tree-link.

We define some messages for ease of explanation.

- Hello = (HopToIG): IG broadcasts this Hello message periodically. A tree-node always unicasts this Hello message to its parent periodically. The HopToIG indicates the distance in hops from a node that initiates the hello message, to an IG. Thus, the IG has its HopToIG = 0.
- J-REQ = (): MN sends this join request message to its neighbor to join.
- CR-REQ = (): An orphan-node responds with this children release request message to its child that sends Hello.
- REG = (NodeID): A node sends this registration message to register with IG periodically. NodeID indicates IP address of the node. All intermediate nodes maintain tree information upon receiving this message.

The tree information structure (TIS) of node i is expressed as follows:

$$\mathrm{TIS}_i = (i.\mathrm{HopToIG}, i.\mathrm{P}, i.D)$$

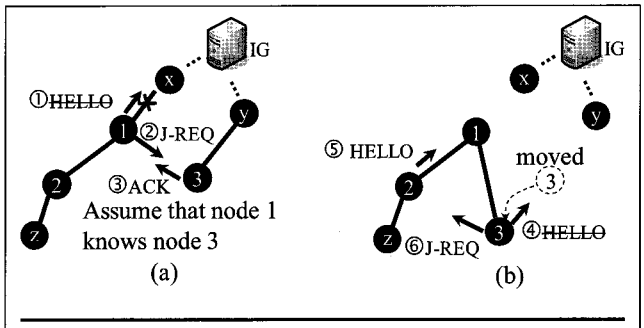
where i.HopToIG indicates node i's HopToIG, i.P does i's parent, and $i.D$ does a set of i's descendants in the tree whose root becomes node i.

B. Tree-Based Approach

In this subsection, we briefly describe the tree-based approach before identifying the problem to be resolved in this paper. An IG broadcasts Hello message periodically. An orphan-node that receives the Hello tries to join the IG by sending J-REQ to the IG. If the IG receives J-REQ, it takes the sender as its child. Upon receiving the acknowledgement (ACK) message in the IEEE 802.11 media access control (MAC) layer, the orphan-node gets the IG as its parent and becomes a tree-node. Every tree-node sends Hello periodically to its parent to assure link availability. An orphan-node that overhears Hello from any tree-node tries to join the sender by sending J-REQ. The tree-node that receives J-REQ takes the sender as its child. Upon receiving ACK for the Hello, the orphan-node gets the tree-node as its parent to become a tree-node. A tree-node becomes an orphan-node (that may or may not have children) immediately when it fails to send Hello to its parent. Every tree-node registers with IG along tree paths periodically. The periodic registration makes sense in that any MN in the network can be requested for connection by other remote hosts or MN's via IG.

Fig. 1(a) shows the join hand-shaking procedure that orphan-node 1 joins IG and Fig. 1(b) shows the hand-shaking procedure that orphan-node 2 joins tree-node 1.

In order to reduce overhead, a node that becomes an orphan does not send a CR-REQ immediately to ask its children to logically leave[2] its parent. Instead, the orphan-node maintains all its descendants as if it has its own parent. If it receives Hello from any child, it responds with CR-REQ to ask that child only to logically leave the current parent. This is because the orphan-node may find a new parent soon.



In Fig. 2(a),
① Node 1 detects the broken link (1, x) when it tries to send Hello to its parent x. In fact, node 1 does not transmit the Hello since it fails to receive clear to send (CTS) in the IEEE 802.11 MAC layer.
②③ Node 1 joins its neighbor 3 through the join process assuming that node 1 already knows the existence of node 3.
In Fig. 2(b),
④ Node 3 moves away from node y and detects the broken link (3, y) upon trying to send Hello.
⑤⑥ When node 2 sends Hello to its parent 1, node 3 overhears the Hello and joins node 2, being its descendant, assuming that node 2 did not issue REG yet. This ends up with the formation of a loop.

Fig. 2. An example of loop formation scenario.

C. Problem Identification

Originating in an IG that is an initial tree-node, orphan-nodes join tree-nodes such that their path length in hops to the IG becomes a minimum. Meanwhile, a tree-node may become an orphan-node when it moves away from and loses connection to its parent. The orphan-node that has just lost its parent will try to join a tree-node again. In this process, the orphan-node may join one of its descendants erroneously, creating a loop, if it does not have information about the descendant. A loop causes the

[2]The logically leave means that a node does not maintain a tree-link for its parent anymore, but a link as a neighbor.

involved nodes to deliver an identical message repeatedly until it is broken, increasing network overhead substantially. This type of abnormal situation takes place because a node sends its REG at regular intervals, incurring an update delay when it either obtains a new child or loses its child. The update delay is referred to as a convergence or synchronization delay.

Fig. 2 shows a part of the network that shows seven nodes $1, 2, 3, x, y, z$, and IG, wherein the scenario of loop formation is illustrated in detail. Assuming that node 3 joins node 2 successfully in Fig. 2(b), if node z sends REG to its parent 2, the REG will circle the loop.

One simple solution (thereafter, referred to as simple solution) to detect the loop is that a node always sends the REG message to its new ancestors immediately after it joins any node successfully. If the originator receives the REG message back, it can be aware of the formation of a loop quickly. Then, the node can break the loop by logically leaving its parent. However, the simple solution incurs two serious problems as follows:

- The way that a node issues the REG message immediately whenever it joins a tree-node will incur a significant control overhead because of the continuous relay of the REG message up to an IG and the high generation frequency of the REG in a high mobility network.
- The transmission delay of the REG message may cause the originator to break a false loop unnecessarily as follows. Referring to Fig. 2, suppose that node 1 has changed its parent to some other node in order to have the shorter HopToIG after it issued the REG message to its current parent, node 3. This change of parent breaks the loop naturally. If node 1 receives the REG message back later on, it will try to break the false loop that is not a loop anymore by logically leaving its new parent. In consequence, node 1 will be left as an orphan-node again.

III. LOOP DETECTION AND RESOLUTION

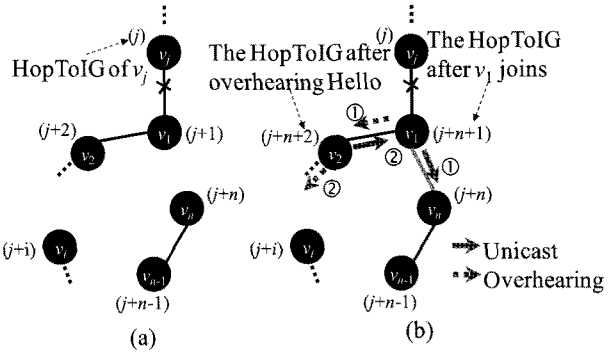
To detect and resolve the looping problem, the newly joined node is required to send Hello immediately to its parent if the following two conditions are satisfied: (1) It has its new HopToIG different from its old one; and (2) it has at least one child.

The first condition indicates that all the children have to change their HopToIG and the second condition indicates that if it does not have any child, it cannot be involved in a loop. The children of the newly joined node can update their HopToIG quickly by overhearing the Hello.

Based on the above rule, a node determines that it is involved in a loop if its parent's HopToIG is equal to a specified value, DEPTH_LIMIT. Once a node detects a loop, it breaks the loop by becoming an orphan-node. The algorithm is detailed in Fig. 4.

We prove that the loop detection and resolution (LDR) algorithm always detects and resolves a loop. Let us assume that an orphan-node does not join a tree-node whose HopToIG is equal to DEPTH_LIMIT. For convenience, let H_i be the HopToIG of node i. We consider two cases separately.

- Case 1: Suppose that a node x has only one child. A node x never joins its child since it does not overhear, but receives Hello from its child. So, it cannot create a loop.



• After v_1 joins v_n
① v_1 sends Hello to its new parent v_n and v_2 overhears it.
② v_2 sends Hello to its parent v_1 and v_3 overhears it.
... (continues)

Fig. 3. Illustration of LDR algorithm: (a) Before a loop is created and (b) after a loop $(v_1, v_2, \cdots, v_i, \cdots, v_{n-1}, v_n, v_1)$ is created.

```
//H_x : is the HopToIG of node x.
//x.P: is the parent of node x.
//x.C: is a set of children of node x.
At a node x that detects disconnection to its parent:
1. IF node x overhears Hello from a tree-node y THEN
2.   IF H_y == DEPTH_LIMIT THEN return; ENDIF;
       // x is not an ancestor of node y or
       // x may not have received REG initiated by y.
3.   IF x's descendant list does not include y THEN
4.     x joins y;
5.     IF (H_x ≠ H_y + 1) THEN
6.       H_x = H_y + 1;
7.       x immediately sends Hello to y if x.C ≠ ∅;
8.     ENDIF;
9.   ENDIF;
10. ENDIF;
At a node x that overhears Hello that its parent sends:
11. IF H_x.P == DEPTH_LIMIT THEN
12.   Node x breaks the loop by becoming an
        orphan-node;
13. ELSE
14.   IF (H_x ≠ H_x.P + 1) THEN
15.     H_x = H_x.P + 1;
16.     x immediately sends Hello to x.P if x.C ≠ ∅;
17.   ENDIF;
18. ENDIF;
```

Fig. 4. LDR algorithm.

- Case 2: Referring to Fig. 3, suppose that a node v_1 has a descendant v_n that is $n-1$ hops away from v_1 and v_1 has not received REG from node v_n yet that has joined v_{n-1} recently. Since node v_1 does not know v_n, it can join node v_n. Now, the group of nodes, $v_1, v_2, \cdots, v_{n-1}, v_n$, forms a loop. Since v_n joined v_{n-1} that is v_1's descendant, $H_{v_1} \neq H_{v_n} + 1$. Thus, according to line 6 and line 7 of the algorithm, $H_{v_1} = H_{v_n} + 1$ and v_1 would always send Hello. In line 15, node v_2 that is a child of v_1 overhears the Hello and changes its HopToIG: $H_{v_2} = H_{v_1} + 1 = H_{v_n} + 2$. Now, node v_2 will send Hello to its parent v_1 and now v_3 will over-

Table 1. Comparison of the simple solution and the LDR.

	Simple solution	LDR
Network overhead	- A high network overhead due to the continuous relay of the REG message up to an IG and the high generation frequency of the REG messages (A node has to issue an REG whenever it joins another tree-node).	- The total number of the generated Hellos is limited by DEPTH_LIMIT $-(j+n)$, $j \geq 0$, $n \geq 3$ where j is HopToIG of the parent of the Hello initiator, and n is the number of nodes involved in a loop. - The limited frequency of Hello initiations since a newly joined node initiates the Hello if the given conditions are satisfied. - Any node stops sending the Hello if its HopToIG is over DEPTH_LIMIT.
Loop resolution speed	- Only the node that issued the REG detects and breaks a loop. - The bigger the loop size, the slower the loop detection.	- Any node in a (possibly) loop can detect the loop by checking its HopToIG. - The bigger the loop size, the quicker the loop detection.
Side effect	- May break a false loop because of the REG transmission delay.	- A node that overhears Hello, but is not involved in a loop can leave its parent logically; however, it contributes to limiting tree depth.

hear it: $H_{v_3} = H_{v_2} + 1 = H_{v_n} + 3$. In this way, for node v_i, $H_{v_i} = H_{v_n} + i$. Thus, HopToIG increases continuously. During this process, a node that finds out the HopToIG of its parent which is equal to DEPTH_LIMIT breaks the loop by becoming an orphan-node itself in line 12 of the algorithm. In line 16, the existence of a loop implies that the node has at least one child and thus surely will issue Hello. Thus, the nodes in the loop continuously increase their HopToIG until any node breaks the loop after it satisfies the condition of the line 11.

With the LDR, the (DEPTH_LIMIT $-(j+n)$) number of Hello messages will be generated at most before breaking the loop where $j \geq 0$ as the distance from the parent of the Hello initiator to an IG and $n \geq 3$ as the number of nodes involved in a loop. Therefore, the bigger the $(j+n)$ is, the smaller the number of Hello messages generated to resolve a loop gets. The LDR can quickly detect and resolve a loop; however it can sometimes have some node break a link to its parent although it is not involved in a loop. Table 1 summarizes the key differences between the simple solution and the LDR.

IV. PERFORMANCE EVALUATIONS

We resort to simulation (use QualNet 3.9) to evaluate the tree-based approach that employs the LDR, named tree-based method (TB)-LDR. This method is compared with the hybrid method (Hybrid), and the original TB. This section consists of three parts: Implementation of Hybrid and TB, evaluation model, and the performance evaluation of the three approaches.

A. Implementation of Hybrid and TB

In implementation, we assume that any wired host or wireless MNs in other ad hoc networks can request connection to an MN in the considered network. This implies that every MN has to do its best to register with IG even though it does not want to have any initiative to make connection to any remote node through an IG.

Every registered MN and its associated IG manage their own registration-timer which is time-synchronized to prevent a stale registration of node. Thus, IG removes a registered node from its node registration list if the corresponding registration-timer expires. Also, every MN has to register with its IG periodically whenever its registration-timer expires. The registration-timer is set to 5 seconds for all the protocols. Other parameters are also used to characterize the operation of each protocol. The values used for the key parameters were carefully chosen through multiple runs of simulation to produce the best outcome.

A.1 Hybrid Method

An IG floods advertisement message (ADV) periodically up to k hops by associating the message with advertisement-timer. During this process, a reverse path from any receiving node to the IG is established. Upon receiving this message, an unregistered node sends REG along the reverse path to register with the IG. A forward path from the IG to the MN is established through this registration process. Each established path is associated with path-timer whose expiration invalidates the path.

An MN that is distanced over k hops and thus does not receive ADV floods solicitation (SOL) message to search for any registered node or an IG by using the expanding ring search method [9]. In this process, another forward path from the receiving registered node to the node that has initiated SOL is established. Then, upon receiving SOL, the registered node sends a response message to the initiator, establishing another reverse path from the unregistered initiator to the registered node. Finally, the node that issued SOL registers with IG along the combined one of two reverse paths. We use 2 seconds for the advertisement-timer and 3 seconds for the path-timer, and k is set to 2.

A.2 TB Method

An IG broadcasts Hello periodically by using hello-timer. An orphan-node either joins the IG if it receives Hello or joins a tree-node if it overhears Hello by sending J-REQ to the tree-node. Every tree-node sends Hello to its parent periodically by using hello-timer and the other neighbors can update their neighbor list by overhearing the Hello. If a node loses its parent, it becomes an orphan-node and then has to initiate the join handshaking procedure. All tree-nodes register with IG along tree

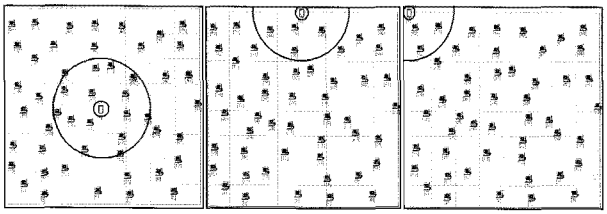


Fig. 5. Deployment scenarios (The node with a small circle indicates IG and the big circle indicates the IG transmission range that is the same as the MN one): (a) S1 with 50 MNs, (b) S2 with 50 MNs, and (c) S3 with 50 MNs.

Table 2. Simulation parameters and values.

Parameters	Values	Remarks
Mobility pattern	Random waypoint	All protocols
Pause time	30	
Number of nodes	Varying density (1 fixed IG)	
Dimension	1000×1000	
Transmission range	250 m	
Wireless bandwidth	2 Mbps	
Registration interval	5 seconds	
Simulation time	600 seconds	
Advertisement interval	2 seconds	Hybrid only
Advertisement zone (IG)	2 hops	
Increment in the expanding ring search (MN)	2 hops	

Table 3. Control message overhead for tree maintenance and mobility management (50 nodes).

Message	TB			TB-LDR		
	S1	S2	S3	S1	S2	S3
Hello	10396	10350	10241	10465	10435	10181
J-REQ	1699	2765	4423	1594	2208	3625
CR-REQ	48	160	535	35	316	769
REG(I)	6030	5993	5920	6029	5908	5806
REG(R)	18256	68163	186384	5433	11351	15709
Avg. KB	7.9	23.2	59.4	3.9	5.7	7.1

(I): Initiated message
(R): Relayed message

paths periodically. In this implementation, we use 3 seconds for the hello-timer.

B. Evaluation Model

In simulation, we use three basic scenarios S1, S2, and S3 that have IG locations in center, top center, and corner, respectively as shown in Fig. 5. The number of nodes (nNodes) varies from 50 to 100, or the maximum speed (mSpeed) changes from 0 m/s to 50 m/s. The other simulation parameters and values are given in Table 2.

Considering the deployment scenarios in Fig. 5, it is easily expected that the average value of HopToIG will be increasing in order of S1, S2, and S3. Thus, it can be easily conjectured that S1 will produce the smallest control overhead in managing node mobility. Table 3 indicates the overheads of pairs of <deployment scenario, mobility management method>.

Basically, control overhead is increasing in order of S1, S2, and S3 as proportional to the expected average size of trees. Looking at the number of REG(R)s, overheads in the table tell that the TB-LDR resolves the loops quickly. One more thing to note is that the number of CR-REQs is large in both S2 and S3 that relatively create more loops. When the TB is used, the orphan-node joins any tree-node and remains a tree-node even within a created loop; however the TB-LDR turns the new tree-node into an orphan-node back after quickly resolving the loop if formed. Thus, the TB-LDR generates more CR-REQs. Note that for the TB-LDR with scenario S3, the less number of Hellos implies the more number of orphan-nodes that in turn, result in the more number of CR-REQs.

For evaluation, we use four metrics: Control overhead, registration ratio, registration latency, and registration jitter. Control overhead is obtained by summing up overheads of the control messages related to tree maintenance and node registration and dividing the sum by the number of nodes. This metric allows us to evaluate the effectiveness of LDR algorithm. Registration ratio is the ratio of the registration-success to the registration-attempt. Whenever an MN sends REG message, it increases its registration-attempt by one; whenever IG receives REG from a node, IG increments registration-success of the node by one. Then, registration ratio is given as

$$\frac{\sum_{i=1}^{\text{nNodes}} \text{registration} - \text{success of node } i}{\sum_{i=1}^{\text{nNodes}} \text{registration} - \text{attempt of node } i}.$$

This metric allows us to evaluate the stability of the paths from the MNs to IG. Also, based on the measured value, we can deduce which protocol can better deliver packets from MNs to IG or vice versus. Registration latency is the difference between the time that an MN issues REG and the time that IG receives the same message. This metric also allows us to evaluate the relationship between network traffic and transmission delay. Registration jitter is the average difference of the latencies of two consecutive packets for all the registration packets that are issued by MNs. This metric allows us to evaluate path stability. All of the metrics are affected by the looping problem in the tree-based approach or the flooding mechanism in the Hybrid approach.

C. Simulation Results and Discussion

According to Fig. 6, the protocol with the LDR shows a low overhead overall. Furthermore, it shows a slightly increasing curve as node speed increases. However, the protocol with the simple solution shows a sharply increasing curve. This is because a node with the simple solution issues REG immediately after it joins a tree-node. Thus, the higher node speed will result in the more REGs. Accordingly the gap becomes larger as node speed increases.

Now, we compare three mobility management approaches,

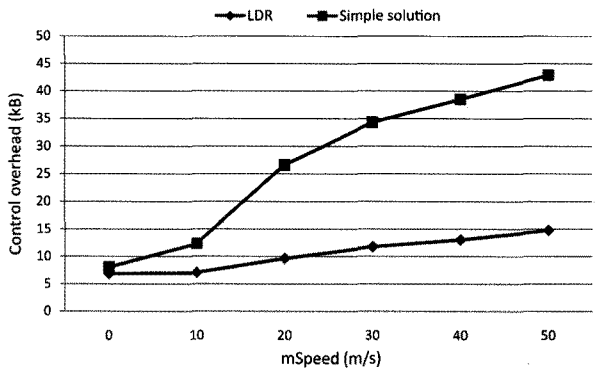


Fig. 6. Control overhead of LDR and simple solution (S3, nNodes = 50).

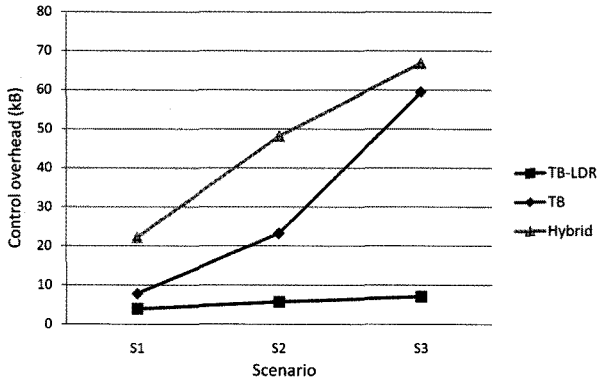


Fig. 7. Control overhead for different scenarios (nNodes = 50, mSpeed = 10 m).

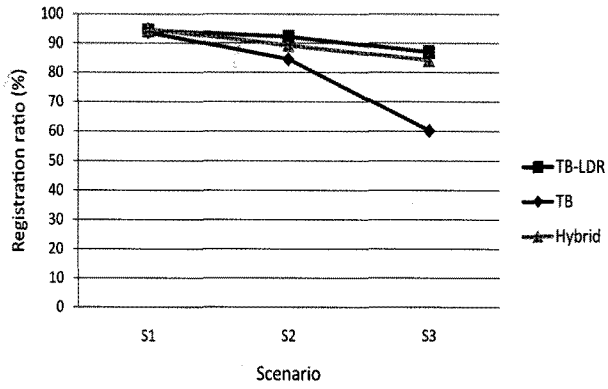


Fig. 8. Registration ratio for different scenarios (nNodes = 50, mSpeed = 10 m).

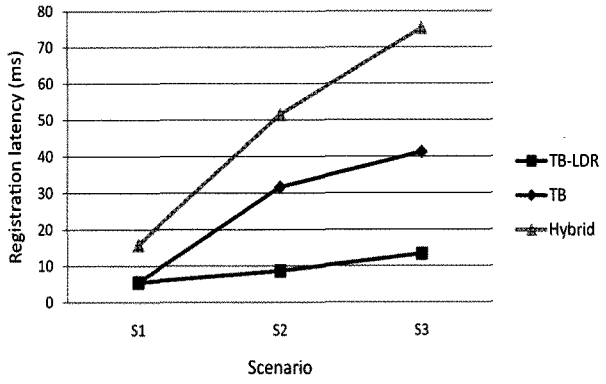


Fig. 9. Registration latency for different scenarios (nNodes = 50, mSpeed = 10 m).

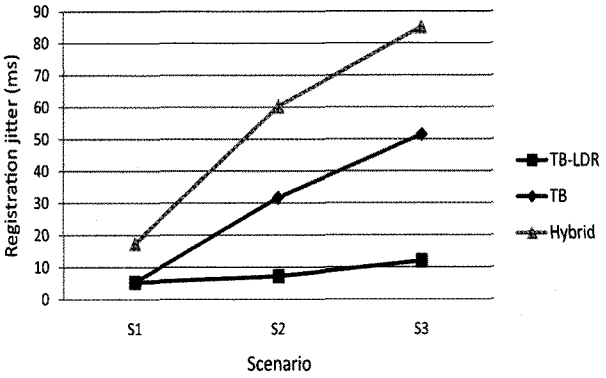


Fig. 10. Registration jitter for different scenarios (nNodes = 50, mSpeed = 10 m).

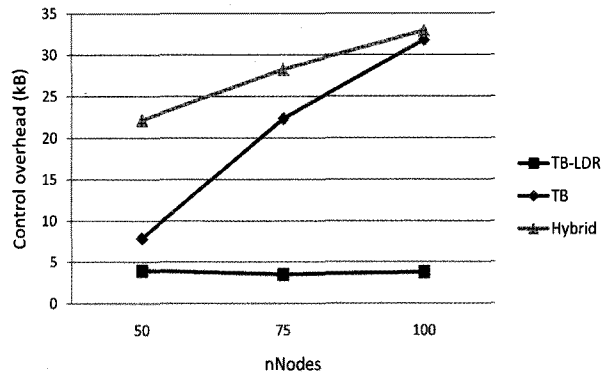


Fig. 11. Control overhead with varying node density (S1, mSpeed = 10 m/s).

TB-LDR, TB, and Hybrid according to the variations of Scenario, nNodes, and mSpeed. According to Figs. 7–10, the Hybrid shows the highest overhead of all since it uses flooding to discover registration path to IG. Control overhead increases proportionally with the increase of average tree size in order of S1, S2, and S3. The TB suffers from the occurrence of loop especially with S3 and thus shows a relatively high overhead and a low registration ratio. However, the TB shows a good performance with the scenario S1 which does hardly generate a loop since its tree depth is small. On the other hand, the TB-LDR shows the very stable and outstanding results for all the scenarios and metrics.

Figs. 11–14 shows the comparison of the three approaches according to the variation of nNodes. In Fig. 11, the overhead of the Hybrid is high and keep increasing according to the increase of nNodes while that of the TB-LDR keeps low. The fact that the TB-LDR sustains very stable against the change of nNodes implies that the tree-based approach works very well unless a loop takes place.

It is natural that the curve of the Hybrid is increasing since the number of nodes increases. Meanwhile, the TB shows a sharply increasing curve since the probability of loop formation increases for the more number of nodes. The occurrences

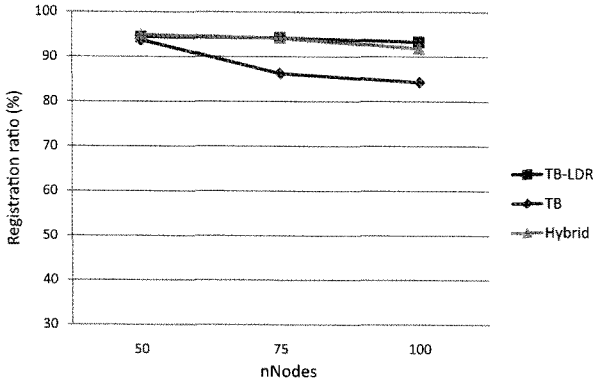


Fig. 12. Registration ratio with varying node density (S1, mSpeed = 10 m/s).

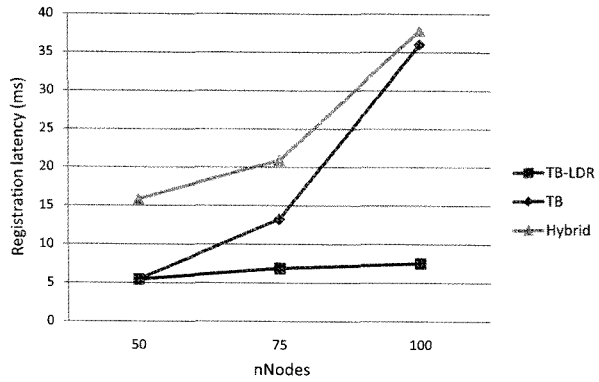


Fig. 13. Registration latency with varying node density (S1, mSpeed = 10 m/s).

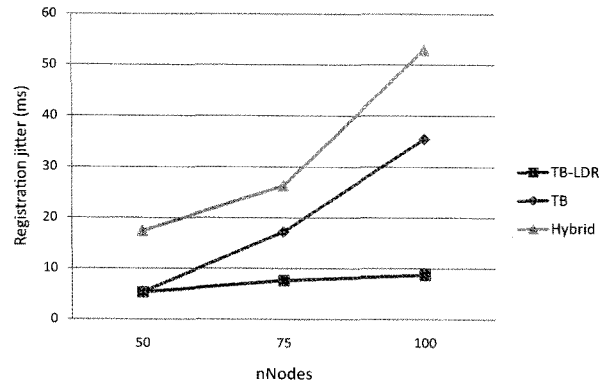


Fig. 14. Registration jitter with varying node density (S1, mSpeed = 10 m/s).

of loop are proved by the lower registration ratio in Fig. 12 and the high registration latency in Fig. 13 for the TB with nNodes = 100.

The increasing overhead due to flooding or looping directly affects the other performance metrics such as latency and jitter in Figs. 13 and 14.

Figs. 15–18 compare the approaches against the increase of mSpeed. We use scenario S3 to effectively compare the scalability of the approaches against the size of tree. According to Fig. 15, the Hybrid and the TB shows higher sensitivity to mSpeed than the TB-LDR. The former experiences the more

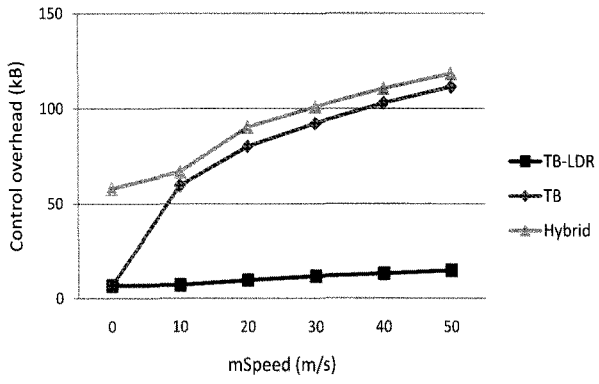


Fig. 15. Control overhead with varying mSpeed (S3, nNodes = 50).

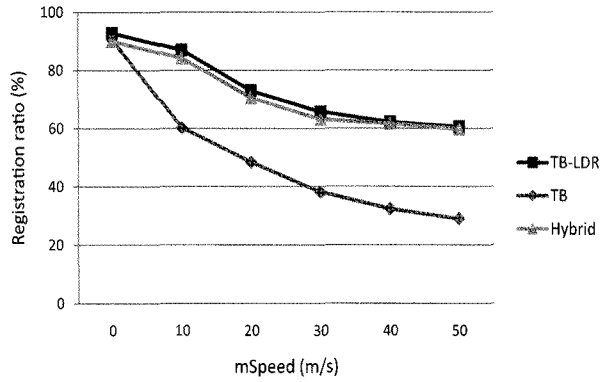


Fig. 16. Registration ratio with varying mSpeed (S3, nNodes = 50).

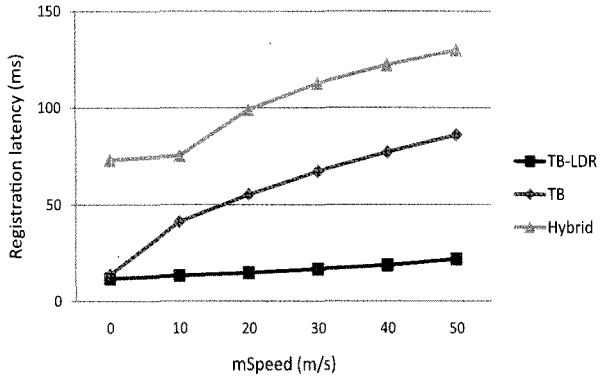


Fig. 17. Registration latency with varying mSpeed (S3, nNodes = 50).

path explorations for registration due to the increased link failures while the latter experiences more loops due to the increased orphan-nodes transiently. However, overhead in TB-LDR is not very sensitive against the increasing node speed. Nevertheless, it is worth noting that control overhead at mSpeed of 50 m/s is almost twice as high as that at mSpeed of 0 m/s in Fig. 15 since the more J-REQs are generated and some additional Hello messages are used to resolve a loop.

The registration ratio of the TB-LDR decreases quite sharply after 10 m/s of mSpeed because the high node speed increases link failure rate. We can tell that the TB-LDR is highly stable against the increase of mSpeed overall.

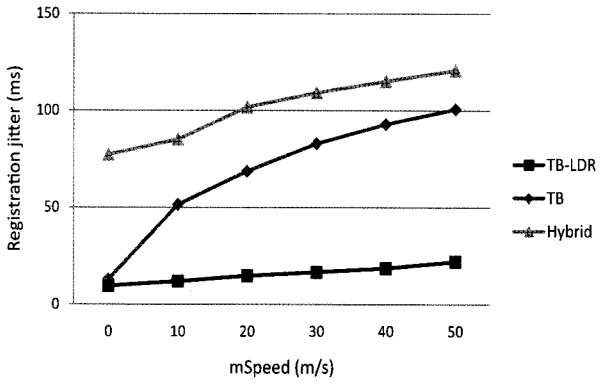


Fig. 18. Registration jitter with varying mSpeed (S3, nNodes = 50).

V. CONCLUSIONS

We identified a looping problem in the tree-based mobility management and presented a LDR method to solve the looping problem. We proved that the LDR method detects and resolves a loop quickly by tracking the depth of trees and then we evaluated its performance by resorting to simulation. We compared the TB-LDR with the TB and the Hybrid. The simulation results show that the TB-LDR far outperforms the others and is highly scalable against the large size of trees and the high mobility of nodes.

REFERENCES

[1] A. George, A. Kumar, D. Cavalcanti, and D. P. Agrawal, "Protocols for mobility management in heterogeneous multi-hop wireless networks," *Pervasive and Mobile Comput.*, vol. 4, no. 1, pp. 92–116, Feb. 2008.

[2] K. U. R. Khan, A. V. Reddy, and R. U. Zaman, "An efficient integrated routing protocol for interconnecting mobile ad hoc networks and the Internet," *Int. J. Comput. Elec. Eng.*, vol. 1, no. 1, pp. 1793–8198, Apr. 2009.

[3] R. Kumar, M. Misra, and A. K. Sarje, "An efficient gateway discovery in an ad hoc networks for Internet connectivity," in *Proc. ICCIMA*, 2007, pp. 275–281.

[4] U. Jonsson, F. Alriksson, T. Larsson, P. Johansson, and G. Q. Jr. Maguire, "MIPMANET-mobile IP for mobile ad hoc networks," in *Proc. IEEE/ACM MobiHoc*, USA, Aug. 2000, pp. 75–85.

[5] Y. Sun, E. M. Belding-Royer, and C. E. Perkins, "Internet connectivity for ad hoc mobile networks," *Int. J. Wireless Inf. Netw.*, vol. 9, no. 2, pp. 75–88, Apr. 2002.

[6] J. Broch, D. A. Maltz, and D. B. Johnson, "Supporting hierarchy and heterogeneous interfaces in multi-hop wireless ad hoc networks," in *Proc. ISPAN*, Perth, Australia, June 1999, pp. 370–375.

[7] R. Wakikawa, J. T. Malinen, C. E. Perkins, A. Nilsson, and A. J. Tuominen. *Global connectivity for IPv6 mobile ad hoc networks*. [Online]. Available: http://tools.ietf.org/id/draft-wakikawa-manet-globalv6-05.txt

[8] H. Ammari and H. El-Rewini, "Integration of mobile ad hoc networks and the Internet using mobile gateways," in *Proc. IEEE IPDPS*, vol. 13, Apr. 2004, p. 218b .

[9] P. Ratanchandani and R. Kravets, "A hybrid approach to Internet connectivity for mobile ad hoc networks," in *Proc. IEEE WCNC* 2003, Mar. 2003, pp. 1522–1527.

[10] P. Ruiz and A. G. Skarmeta, "Enhanced internet connectivity for hybrid ad hoc networks through adaptive gateway discovery," in *Proc. IEEE LCN*, Los Alamitos, USA, Nov. 2004, pp. 370–377.

[11] H. Oh, "A tree-based approach for the Internet connectivity of mobile ad hoc networks," *J. Commun. Networks*, vol. 11, no. 3, pp. 261–270, June 2009.

[12] M. Caleffi, G. Ferraiuolo, and L. Paura, "Augmented tree-based routing protocol for scalable ad hoc networks," in *Proc. IEEE MASS*, Oct. 2007, pp. 1–6.

[13] M. Caleffi and L. Paura, "M-DART: Multi-path dynamic address routing," *Wireless Commun. Mobile Comput.*, vol. 11, no. 3, pp. 392–409, Mar. 2011.

[14] T.-D. Han and H. Oh, "A topology management routing protocol for mobile IP support mobile ad hoc networks," in *Proc. AdHoc-Now*, Murcia, Spain, Sept. 2009, pp. 341–346.

[15] W. Peng, Z. Li, and F. Haddix, "A practical spanning tree based MANET routing algorithm," in *Proc. ICCCN*, Oct. 2005, pp. 19–24.

[16] J. Jubin and J. D. Tornow, "The DARPA packet radio network protocols," *Proc. IEEE*, vol. 75, no. 1, pp. 21–32, Jan. 1987.

Trung-Dinh Han is a Ph.D. candidate in the School of Computer Engineering and Information Technology, University of Ulsan, South Korea. He received the B.E. and M.E. degrees in Electrical and Electronic Department, Ho Chi Minh City University of Technology, Vietnam in 1999 and 2002, respectively. He worked as a head of Computer Systems Division, Information Technology Faculty, Ho Chi Minh City University of Industry. His research interests include wireless sensor networks, mobile ad hoc networks, embedded systems, and digital image processing.

Hoon Oh received the B.S.E.E. degree from the Sung Kyun Kwan University, Seoul, and the M.S. degree and the Ph.D. degree in Computer Science from the Texas A&M University at College Station, Texas, in 1993 and 1995, respectively. From 1983 to 1989 and 1996 to 2000, he worked as a Software Engineer and Software Architect in the Corporate Research Center of Samsung Electronics. He was involved in developing the communication protocols for data services of the CDMA and IMT2000 handset products. Currently, he is an Associate Professor of the School of Computer Engineering and Information Technology and a Director of the Vehicle IT Convergence Technology Research Center in the University of Ulsan, Korea. He is a Member of IEICE, ISCA, KICS, and ICASE, and has been a Lifetime Member of the Korea Information Society since 1989. He received a Best Paper Award from the National Academy of Science, USA in 1995. His research interests lie in wireless sensor networks, mobile ad hoc networks, real-time computing, and ubiquitous computing.