A Tree-Based Approach for the Internet Connectivity of Mobile Ad Hoc Networks

Hoon Oh

Abstract: We propose a tree-based integration of infrastructure networks and MANETs (TIIM) to efficiently extend the scope of mobile Internet protocol to mobile ad hoc networks and devise a tree-based routing protocol (TBRP) that is suitable for the TIIM architecture. Infrastructure networks have a number of fixed Internet Gateways that connect two heterogeneous networks. Mobile nodes form a number of small trees named pMANETS, each of them growing from anchor node which can communicate directly with an Internet Gateway. A new node registers with foreign agent and home agent along the tree path without resorting to an inefficient flooding. Furthermore, the TBRP sets up a routing path efficiently by exploiting the tree information without relying on flooding. We show by resorting to simulation that our approach is competitive against the conventional AODV based approach.

Index Terms: Mobile ad hoc network (MANET), mobile IP, mobility management, routing protocol, tree structure.

I. INTRODUCTION

Mobile ad hoc network (MANET) is temporarily deployed among mobile nodes in a place void of infrastructure elements such as access point (AP) or base station (BS) to which the mobile nodes can directly connect. Despite its broad applications of areas, MANET lacks its applicability in the areas in which mobile nodes need an access to infrastructure networks or communication with mobile nodes of another MANET, or are requested for connection by nodes on other wired or wireless networks. Thus, the scope of mobile Internet protocol (IP) need be extended to include multi-hop wireless nodes; however, it requires management for the current locations of mobile nodes, resulting in a high traffic of control messages.

Recently, a number of researches have been conducted for efficient node mobility management in integrated infrastructure networks and MANETs. Broch et al. assume that Internet gateway (IG) is equipped with two network interface cards [1]. One interface that executes standard IP mechanism communicates with Internet nodes while another that does MANET routing mechanism connects to mobile nodes. They use the DSR protocol [2] for a MANET routing. Jonsson et al. focus on when to register with FA using MIPMANET Cell Switching algorithm in order to reduce control overhead and adopt the AODV protocol [9] for a MANET routing. In [3], Sun et al. describe how mobile IP and AODV routing protocols can cooperate to discover multi-hop paths between mobile nodes and foreign agents. Prashant et al. [4] present a hybrid scheme that combines the techniques such as agent advertisements, TTL scoping and caching of agent advertisements, eavesdropping, and agent solicitation. In [5], Ammari et al. propose a three-layer approach that uses mobile IP and DSDV [6], and uses a mobile gateway as an interface between MANET and Internet.

Most of the approaches discussed above resort to an inefficient flooding that causes considerable network overhead for mobility management and/or route discovery. We propose a tree-based integration of infrastructure networks and MANETs (TIIM) to support mobility management and routing and then present a mobility management protocol and a tree-based routing protocol (TBRP) that does not use an inefficient flooding by exploiting the TIIM architecture. The TIIM architecture is composed of a number of small trees, each of them growing from a node that can communicate directly with an Internet Gateway (IG). A new node joins a tree as being a child of some node in the tree and registers with foreign agent (FA) and home agent (HA) along the tree path to which it belongs.

In the design of the proposed protocol, we are pursuing the four aspects: Reducing control overhead, reducing latency to establish a route, balancing the use of two heterogeneous network resources, and increasing communication efficiency. We try to keep the trees as small as possible in order to reduce overhead incurred by various control messages used in network architecture construction and maintenance, mobility management, and route discovery. We can quickly set up a route by exploiting tree information managed at each node and utilizing reliable infrastructure network resources. We notice that infrastructure network can be overloaded if the number of mobile nodes increases largely. Thus, a routing protocol is designed such that a path consists of only mobile nodes if possible. Lastly, we take the distance of two mobile communication parties into consideration upon setting up a routing path. The route is formed by including gateways to increase communication efficiency only if the distance is judged to be relatively long.

Section II describes the network model and some useful definitions. Section III and IV details network management protocol with a new tree-based routing protocol. Performance evaluation is given by resorting to simulation in Section V and followed by concluding remarks in Section VI.

II. PRELIMINARY

A. Network Model

An immobile IG, acting as a FA and/or a HA, has two infrastructure network interfaces, IF1 to communicate with mobile nodes and IF2 to communicate with fixed nodes. Two nodes are "connected" when they stay within each other's transmission range. All mobile nodes and IGs can act as a router. We assume

Manuscript received September 14, 2007; approved for publication by Abbas Jamalipour, Division II Editor, July 10, 2008.

The author is with the Department of Computer Engineering and Information Technology, University of Ulsan, Korea. email: hoonoh@ulsan.ac.kr.
Table 1. Tree information structure (TreeInformationTablei).

<table>
<thead>
<tr>
<th>IG</th>
<th>Anchor</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of hops to anchor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node Status: Anchor, member, orphan</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_1$</td>
<td>Next node leading to $D_1$</td>
<td>$d_{D_1}$</td>
</tr>
<tr>
<td>$D_2$</td>
<td>Next node leading to $D_2$</td>
<td>$d_{D_2}$</td>
</tr>
</tbody>
</table>

that nodes can overhear packets that are transferred among the other nodes [7]. The network is modeled as an undirected graph G: IGs and mobile nodes are modeled as nodes, and connection of two nodes is as an edge. $G = (V, E)$, $V$ is a set of nodes and $E$ is a set of edges. $V = V_1 \cup V_2$, $V_1$ is a set of IGs and $V_2$ is a set of mobile nodes. $E = E_1 \cup E_2$. $E_1 = \{(x, y) | x \in V_1, y \in V_1, x$ and $y$ are connected $\}$ and $E_2 = \{(x, y) | x \in V_2, y \in V_2, x$ and $y$ are connected $\} \cup \{(x, y) | x \in V_1, y \in V_2, x$ and $y$ are connected $\}$.

- (Neighbor, Neighbor Set) Neighbor set, $N_i$ of node $i$ is a collection of neighbor node $x$ such that $(i, x) \in E$.
- (pMANET) pico MANET, shortly pMANET, is a connected tree formed by $V_2$ where its root node is connected to a node in $V_1$. A node that belongs to a pMANET is said to be a member and a node which is not a member is an orphan. A root node being a member is especially called an anchor.
- (NeighborTablei) Every node $i$ keeps NeighborTablei to manage information for each node in $N_i$. An entry of neighbor $j$, NeighborTablei,$(j) = (j$’s ID, $j$’s relationship, $j$’s IG, $j$’s anchor, $j$’s parent, $j$’s distance to $j$’s anchor, $j$’s last updated time), $j$’s relationship indicates whether $j$ is child or not.
- (TreeInformationTablei) Every node $i$ maintains TreeInformationTablei, as shown in Table 1. It holds its IG, its anchor, its parent, its distance to anchor, its node status (anchor, member or orphan), and for each of its descendants, next child node and distance to reach it.
- (Broadcast, Unicast) A message is broadcast if it is sent to all its neighbors without appointing any specific neighbor; it is unicast if it is sent to a specific neighbor. In our implementation, we mostly use “unicast” in transmission that can not only avoid collision, but also detect transmission success or failure quickly by examining the reception of ACK from the MAC layer.

Fig. 1 shows an example THM architecture with twenty two nodes (IG100, IG200, 1, 2, 3, · · · , 20). The thick-solid edges and the dotted (or dashed) edges indicate wired connection and wireless connection, respectively. The dashed big circle indicates the transmission range of mobile node and IFI in IG and white, shaded, and shaded thick circles represent orphan, member, and anchor, respectively. Nodes 2, 5, 10, and 14 are anchors, forming four pMANETs. Each pMANET is represented by connected thick dashed edges.

1 $D_n$: descendant $n$.
2 $d_{D_n}$: distance to $D_n$ in hops.

B. Message Definition

We define some messages to describe network architecture management protocol and routing protocol.

- MN-Hello, IG-Hello: Member nodes unicast MN-Hello to their parents periodically (orphan node does not send MN-Hello). Anchor being a root node unicasts MH-Hello to its IG periodically. The other neighbors overhear the message. MN-Hello = (MsgId, Nodeld, IGId, AnchorId, ParentId, OverCapacity, HopsToAnchor).

○ MN-Hello message structure (tocm: Type of control message, O: OverCapacity, R: Reserved)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MsgId is a unique message number that the sender generates. A pair <Nodeld, MsgId> defines a message uniquely throughout the network. IGId, AnchorId, and ParentId are IG, anchor, and parent to which the sender belong. OverCapacity is set to 1 if the sender can not accept child anymore. HopsToAnchor is the number of hops from the sender to its anchor. IG unicasts IG-Hello periodically to an anchor selected from all the anchors that it manages. If no such anchor does exist, it broadcasts the message. IG-Hello = (MsgId, IGId, OverCapacity). IGId is the sender’s identification number. OverCapacity is set to 1 if IG can not accommodate an anchor anymore.

○ IG-Hello message structure:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- AREQ, ARES: A mobile node that wants to be an anchor selects an IG with its OverCapacity set to 0 among the IG list in its neighbor table and then unicasts anchor request message (AREQ) to the selected IG. AREQ = (MsgId, SrcId, DstId, UrgentFlag, IGId).
If a member node is disconnected from its parent or IG and searches for any IG, it sets UrgentFlag to 1 and sends the AREQ to an IG. In this case, the receiving IG accepts it as an anchor regardless of its capacity. However, an orphan node does not send AREQ if it does not find an IG with its OverCapacity set to 0. If a receiving IG accepts a new anchor, it responds with anchor response message (ARES). ARES = (MsgId, SrcId, DstId, AcceptedFlag, IGId) where IG sets AcceptedFlag to 1 if it allows a new anchor.

**AREQ message structure (UF: UrgentFlag)**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>MagId</td>
</tr>
<tr>
<td>6</td>
<td>DstId</td>
</tr>
<tr>
<td>7</td>
<td>SrcId</td>
</tr>
<tr>
<td>8</td>
<td>UFlag</td>
</tr>
</tbody>
</table>

**ARES message structure (AF: AcceptedFlag)**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>MagId</td>
</tr>
<tr>
<td>6</td>
<td>DstId</td>
</tr>
<tr>
<td>7</td>
<td>SrcId</td>
</tr>
<tr>
<td>8</td>
<td>UFlag</td>
</tr>
</tbody>
</table>

**JREQ, JRES: An orphan node that wants to be a member selects a node with its OverCapacity 0 among the neighbor member nodes in the neighbor table and sends a join request message, JREQ, to the selected node. If it does not receive join response message (JRES), it sends JREQ to another node with its OverCapacity 0 after some interval. JREQ = (MsgId, SrcId, DstId, UrgentFlag, IGId) and JRES = (MsgId, SrcId, DstId, AcceptedFlag, IGId, AnchorId, HopsToAnchor). The usage of AcceptedFlag and UrgentFlag is the same as in AREQ and ARES.**

**JREQ message structure:**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>MagId</td>
</tr>
<tr>
<td>6</td>
<td>DstId</td>
</tr>
<tr>
<td>7</td>
<td>SrcId</td>
</tr>
</tbody>
</table>

**JRES message structure:**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>MagId</td>
</tr>
<tr>
<td>6</td>
<td>DstId</td>
</tr>
<tr>
<td>7</td>
<td>SrcId</td>
</tr>
</tbody>
</table>

**PCREQ, PCRES: A member that wants to change its parent sends parent change message (PCREQ) to a selected neighbor with OverCapacity 0. PCREQ = (MsgId, SrcId, DstId, IGId). A receiving node replies with parent change response, PCRES = (MsgId, SrcId, DstId, IGId, AnchorId, HopsToAnchor, AcceptedFlag), where the node sets AcceptedFlag to 1 if it accepts a new child.**

**UREQ message structure (DF: DirFlag)**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>MagId</td>
</tr>
<tr>
<td>6</td>
<td>DstId</td>
</tr>
<tr>
<td>7</td>
<td>SrcId</td>
</tr>
</tbody>
</table>

**III. NETWORK ARCHITECTURE MANAGEMENT PROTOCOL**

A. Neighbor Management

IG selects an anchor among its neighbors and unicasts IG-Hello every Hello-Interval to the selected node. If it does not receive ACK for IG-Hello-Reply-Interval, it selects another anchor and repeats the process. Member (Anchor) sends MH-Hello to its parent (IG) periodically. Member or anchor confirms the availability of its upstream link by checking ACK. In this process, all receiving or overhearing nodes update its NeighborTable.

B. Anchor Request Procedure

An orphan node, upon overhearing IG-Hello from an IG, sends AREQ to the IG in order to become an anchor only if OverCapacity for the IG is 0. The receiving IG responds with ARES if it can accommodate a new anchor. An anchor node becomes an anchor if AcceptedFlag in the ARES is 1, and then updates its TreeInformationTable by using ARES and the previously received IG-Hello. IG registers the new anchor if it receives ACK against the ARES.

C. Join Procedure

Fig. 2 illustrates the join procedure in case that node 4 joins member 3. Node 3, upon receiving JREQ, responds with JRES and updates topology information when it receives MAC ACK. Node 3 sets DirFlag to 2 (bidirectional) and sends
1. Orphan 4 overhears MN-Hello with OverCapacity = 0 from node 3.
2. Send JREQ to node 3.
3. Receive JREQ with AcceptedFlag = 1 from node 3.
4. Node 3 sends UREQ with DirFlag = 2, Action = Update, IGId = 100, AnchorId = 1, and MemberList = {4}.
5. Node 4 overhears UREQ.
6. Node 5 sends UREQ with DirFlag = 1, Action = Add, MemberList = {4}.
7. Node 1 sends the same msg as node 5 to IG100.

Fig. 2. Join procedure.

\[ \text{UREQ with Action = Update, IGId = 100, AnchorId = 1, and MemberList = \{4\} to node 5.} \]
\[ \text{The new node 4 updates its IGId and AnchorId by means of overhearing.} \]
\[ \text{Node 5 removes IGId and AnchorId, changes DirFlag to 1, and sets Action to "Add" from the received UREQ and then forwards it upward.} \]
\[ \text{Node 1 forwards the same message to IG100.} \]

D. Parent-Change Procedure

If member \( x \) receives MN-Hello with OverCapacity = 0 from its neighbor \( y \), \( x \) compares its distance to its anchor and \( y \)'s distance to \( y \)'s anchor plus 1. If the difference is not less than \( T_{\text{change}} \), \( x \) decides the change of its parent and sends PCREQ to \( y \). If \( x \) receives PCRES of AcceptedFlag = 1, it changes its tree information (parent, anchor, and IG) and sends UREQ that includes its updated information and descendants to its new parent. Its parent sends the UREQ that includes only the MemberList to its ancestors and lets its ancestors and its IG update their tree information. Its children that receive UREQ update the UREQ such that they include the changed anchor and IG and then continue to forward to their respective children. At last, all descendants update a new anchor and IG. The previous parent continues to forward the UREQ that takes MemberList only to its own parent and all the previous ancestors, and IG deletes the nodes by looking up MemberList. If \( T_{\text{change}} \) is set to 1, all nodes will try to keep the shortest distance to IG. The bigger \( T_{\text{change}} \) is, the more packet transmission overhead the network gets, but the less overhead by parent change it gets.

Let us take a look at information update process when node 4 changes its parent from 5 to 3. Node 4 unicasts UREQ to its new parent 3 with DirFlag = 2, Action = Update, IGId = 200, AnchorId = 3, and MemberList = \{6\}. Node 3 and IG200 that receive this message add nodes 4 and 6. Node 6 that overhears the same UREQ updates its IG and anchor. Node 5, upon overhearing it, deletes nodes 4 and 6 if its AnchorId and the sender's AnchorId are different and then forwards the message to its parent with the following modifications: DirFlag = 1 and Action = Delete. Its ancestors 2, 1, and IG100 remove nodes 4 and 6. In this way, node 4 completes topology update by using one UREQ.

E. Tree Link-Failure Repair Procedure

A tree link can be broken due to exhausted power, system failure or node movement. It is extremely critical to detect a broken link quickly in order to correctly maintain tree topology information. A node regards a downlink as broken if it does not receive MN-Hello from the corresponding child for a specified interval, HelloInterval + e, where \( e \ll HelloInterval \) is a maximum transmission delay. A node confirms availability of its uplink by checking MAC ACK against its MH-Hello. Note that an anchor sends MN-Hello to its IG. An anchor regards its uplink as broken if it does not receive ACK (MAC Layer) from the IG. Thus, the worst case detection time of broken link for up-link and down-link is HelloInterval + e. IG sends its IG-Hello periodically to one of the anchors that it manages. Because of the downlink failure to the selected anchor, it can either fail to send IG-Hello or it may not receive ACK. In this case, it selects another anchor after some short interval and repeats the same process. The other anchors overhear the IG-Hello. Of course, IG has to broadcast IG-Hello if it does not manage any anchor. Note that broadcast messages being a cause of collision are not used except for this special case.

A node that detects a downlink failure sends UREQ to its ancestors and IG. A node that detects uplink failure becomes a temporary root node (that is not a member but has some descendants) and tries to become an anchor of another IG or join a member by looking up its NeighborTable. At this emergency case, UrgentFlag is set to 1 and the node can join any IG or member regardless of OverCapacity value. If it succeeds, it sends UREQ to its ancestors and descendants. It also unicasts UREQ to its new IG or parent with DirFlag = 2. If it fails, it sends UREQ to one of its children and remains an orphan until it overhears MN-Hello or IG-Hello. Each node that receives or overhears the UREQ takes the same process as its parent and can be an anchor, a member or an orphan.

Suppose that link (12, 13) of pMANET1 is disconnected in Fig. 4. Node 12 that detects the downlink failure sends UREQ to its ancestor 2 and IG100. Node 13 that detects the uplink failure finds its neighbor 11 by looking up its NeighborTable and performs emergency join. If it succeeds, node 13 sets DirFlag to 2 and unicasts the UREQ to its new parent 11. Node 11 continues to forward the UREQ after changing DirFlag to 1 that is finally forwarded up to IG100. Its children, 3 and 4 overhears the same UREQ and checks the direction flag. Since DirFlag is 2, node 3 that receives UREQ forwards it to node 18 after changing DirFlag to 0. 

\[ ^4 \text{We assume that a node sends MN-Hello to its parent successfully every HelloInterval as far as the link is not disconnected, because it does not broadcast a message.} \]
IV. TREE-BASED ROUTING PROTOCOL

In this section, we propose a tree-based routing protocol, TBRP for the TIM architecture. TBRP protocol consists of path establishment, path improvement, and path recovery. In TBRP, path establishment methods vary according to the relative locations of source (src) and destination (dst).

A. src on Internet and dst on MANET

In this case, the routing path is defined as \((src, \ldots, dst.IG, \ldots, dst)\) where src is a wired host on Internet. Supposed that all mobile nodes including dst are registered with FA and HA, no route establishment is needed since the partial path \((dst.IG, \ldots, dst)\) already was established by the network architecture management protocol. Therefore, packets are delivered to dst with the help of Mobile IP.

B. src on MANET and dst on Either MANET or Internet

In this case, the routing path is simply defined as \((src, \ldots, src.IG, dst.IG, \ldots, dst)\). Two path segments, \((src, \ldots, src.IG)\) and \((dst.IG, \ldots, dst)\), are maintained by the network architecture management protocol. The routing from src.IG to dst.IG is performed by Mobile IP. Therefore, we do not need any control message to set up the path.

However, the above routing always relies on infrastructure networks. If a number of mobile nodes are very large, it can impose considerable overhead on infrastructure networks. So, if two nodes are mobile, it may be desirable to set up a routing path by utilizing MANET resources only. However, it may not be desirable to set up such a path if two mobile nodes are distant far away because it will consume MANET resources too much and thus degrade performance in terms of end-to-end delay and delivery ratio. Consequently, we need a method to setup paths such that the utilization of two heterogeneous resources is well balanced by taking performance into consideration.

As a result, instead of using the simple path that always relies on infrastructure resources, we set up various paths depending on the locations of two communication parties as follows:

- **Type1**: dst is a descendant or neighbor of src.
- **Type2**: src and dst belong to the same pMANET and dst is not a descendant of src (that is, dst is either an ancestor of src or a descendant of its ancestor).
- **Type3**: src and dst belong to two different pMANETs whose gateways are identical.
- **Type4**: src belongs to a pMANET and dst belongs to either Internet or another pMANET whose gateway is different.

B.1 Route Establishment

A node that wants to setup a path looks up its TreeInformationTable and NeighborTable to see if dst is either its descendant or its neighbor. If that is the case, the path type is a Type1. Type1 path need not be established. Otherwise, src sends route request (RREQ) message toward its IG by following a tree path. While the RREQ is moving upward, every receiving node checks if dst is either itself or one of its descendants. If that is the case, the node replies with route reply (RREP) message, setting up a Type2 path as \((src, \ldots, co-ancestor, \ldots, dst)\), where co-ancestor is a common ancestor of both src and dst. Both Type1 path and Type2 path are established within the identical pMANET, utilizing MANET resources only. Considering the locality of MANET communication, the frequency of these

5In MANET, it is highly possible that two communication parties reside nearby. This is referred to as a locality of communication.
two path types is expected to be high in reality. The bigger the
tree is, the higher the probability of these two path types can be.

If dst is not found until RREQ arrives at src.IG, src.IG ex-
amines whether or not dst belongs to any other pMANET under
its management. If dst belongs to another pMANET, it estab-
lishes Type3 path as (src, · · · , src.anchor, src.IG, dst.anchor, · · · , dst) and sends RREP including dst.anchor to src. Oth-
erwise, src.IG sends RREQ to dst with the help of Mo-
 bile IP and the receiving dst.IG replies with RREP includ-
 ing dst.IG and dst.anchor. Type4 path is given as (src, · · · , src.anchor, src.IG, dst.anchor, · · · , dst).

B.2 Route Improvement

We improve the path length by using overhearing. While
RREQ is moving upward, overhearing nodes checks if dst is ei-
ther its descendant or neighbor. If that is the case, it can imme-
cdiately reply with RREP. For example, suppose that (src, dst) =
(1, 13) in Fig. 5. Nodes 10, 14, and 2 can overhear the RREQ
that node 9 forwards to node 5. Node 2 knows that node 13 is
its neighbor and each of nodes 10 and 14 knows that node
13 is its descendant. Thus, they reply with RREP that includes
their distance to dst. However, if they send RREP at the same
time, delay can be caused by a possible collision. So, we apply
grace-delay function d [2]

\[ d = h \times (h + r). \]

Here, h is a number of hops from the reply node to dst, r is
a value between 0 and 1, and h is a constant value that indicates
an inter-hop transmission delay. By this function, a priority of
reply is given to a node that has the shortest distance to dst.
In consequence, a shorter path (1, 9, 14, 13) or (1, 9, 2, 13) is
established instead of the long Type3 path (1, 9, 5, 100, 10, 14,
13).

Consider another case (src, dst) = (6, 13) where two nodes
have a common neighbor 2. Node 2 can overhear the RREQ that
node 6 forwards to node 9 and then replies with RREP imme-
diately.

B.3 Route Recovery

A routing path can be broken due to link failure in the pro-
cess of packet transmission. Its recovery is described accord-
ing to the path types.

• (Type1 path recovery) Suppose that a link (x, y) on Type1
  path was broken. In this case, x can detect this broken link
  by checking ACK immediately after sending a packet to
  y. Then, x saves the packet in a buffer and sends RREQ
  of TTL = 2 with dst (distance to dst) waiting for RREP
  from any node that has a path to dst, possibly a node on
  (y, · · · , dst), but has distance not larger than dst in order
  to avoid a looping problem. At this case, note that if y is
  alive, it is highly possible that y stays within two hops.
  If path recovery succeeds, node x sends the saved packet.
  Otherwise, x sends route error (RERR) message to src.\(^6\)

Then, src initiates route re-establishment after some inter-
val, Link-Recovery-Time which takes to fix the broken tree
by the network architecture management protocol.

• (Type2 path recovery) Suppose that a link (x, y) on path
  Type2 was broken. Then, x saves the packet in a buffer and
  sends RREQ of TTL = 2 with dst, waiting for RREP
  from any node that has a path to dst, but has distance no
  larger than dst. If path recovery succeeds, node x sends
  the saved packet. Otherwise, x sends route error (RERR)
  message to src. Then, src re-initiates route establishment
  after some interval, Link-Recovery-Time which takes to fix
  the broken tree by the network architecture management
  protocol.

• (Type3 path recovery) In case of Type3 path, a link on
either path segment (src, · · · , src.anchor) or
(dst.anchor, · · · , dst) can be broken. The recovery pro-
cess of these two segments is the same as that of Type1 and
Type2. However, if (x, y) is (src.anchor, IG), x sends
RREQ of TTL = 2, waiting for RREP from any anchor or
any node that has a path to the IG. The responding nodes
use the grace-delay function with respect to IG being an in-
termediate destination. On the other hand, if a broken link
(x, y) is (IG, dst.anchor), x sends RREQ of TTL = 2,
waiting for RREP from any node that knows a path to dst.
If it fails, IG sends RERR to src to re-initiate route discov-
ery.

• (Type4 path recovery) Suppose that a link (x, y) on
a path segment (src, · · · , src.anchor, src.IG). If x is
src.anchor, it sends RREQ with TTL = 1, waiting for
RREP from some other IGs or members that know src.IG
but are not its descendant. If x is one of the other mobile
nodes, the recovery process is the same as that of Type2
path.

V. PERFORMANCE EVALUATION

We used the QualNet 3.9 network simulator, a commercial
version of Glomosim. 103 nodes (100 mobile nodes and three
IGs) are placed in a square zone: 100 mobile nodes are randomly
distributed and 3 IGs take the predefined positions. In the terrain
of 1000x1000(m²), scenarios 1, 2, 3, 4 use different IG positions
as in Figs. 6(a)–(d), respectively while the 5th scenario is the
same as scenario 1, but has the bigger terrain, 1500 x 1500(m²).

Considering the IG distribution patterns for different scenar-
ios, an IG with the bigger coverage will have more anchors and
thus get more trees. Consequently, it will have the smaller size
of trees, shortening the distance from each member to IG. From
this viewpoint, we can judge that the IG distribution in scenario
1 is the best of all. On the contrary, note that the IG located at
the corner has only one quarter of coverage that in the middle.
Thus, scenario 4 is considered the worst.

A. Scenario Properties and Performance

Simulation was run to evaluate the structural characteristics of
TIM for the five scenarios with the parameter values of Table 2
and variations of maximum node speed from 0 to 25m/s. Each
metric was given an average value from ten runs of simulations

\(^6\) Route recovery can be performed after sending RERR to src; however, since
the local route recovery is performed with TTL = 2 without delay, doing local
recovery first will make packets flow smoothly.
with different seed values. In graph notation, we denote scenario \( n \) with \( k \) dimension as \( S_{n,k} \) where \( n = 1, \ldots, 4 \) and \( k = 1000 \times 1000 \) or \( 1500 \times 1500 \).

Take a look at Figs. 7 and 8. According to Fig. 7, it is shown that the number of trees decreases slightly with the increase of node speed. This is due to the transiently increased number of orphan nodes. \( S_1 \) to \( S_4 \) in which IGs cover the biggest area shows the largest number of trees, leading to the smallest average tree size. As we expected, \( S_4 \) to \( 1000 \times 1500 \) shows the smallest number of trees, resulting in the largest average tree size. The number of trees in \( S_1 \) to \( 1500 \times 1500 \) is almost half as large as that of \( S_1 \) to \( 1000 \times 1000 \) because its low node density allows the less number of anchors.

In proportion to the coverage area of IGs, the average number of trees is given in the order of \( S_1 \), \( S_3 \), \( S_2 \), and \( S_4 \), but the size of trees is in the reverse order. In general, the size of tree is closely related to network overhead as shown in Fig. 9. However, comparing \( S_2 \) to \( 1000 \times 1000 \) and \( S_4 \) to \( 1000 \times 1000 \), the former with the smaller size of tree shows the higher overhead than the latter. The reason is that nodes are prone to join the IG in the middle in view of the IG deployment of \( S_2 \) to \( 1000 \times 1000 \). Accordingly, the pMANETs that belong to the middle IG can be large. Referring to Fig. 10, delivery ratio is in a reciprocal proportion to network overhead. Also, note that \( S_1 \) to \( 1500 \times 1500 \) shows overhead much higher than \( S_1 \) to \( 1000 \times 1000 \) because of its high probability of link destruction as in Fig. 9.

Fig. 11 shows how much infrastructure network resource is used in the communication of two mobile nodes. According to the simulation result, we can see that 25% to 58% of communications uses only MANET resource depending on node speed, even in \( S_1 \) to \( 1000 \times 1000 \). The reason that we have the largest value in \( S_2 \) to \( 1000 \times 1000 \) is that we can have the largest number of Type1 or Type2 paths due to the increased size of the trees grown from the IG in the middle. However, in case of \( S_4 \) to \( 1000 \times 1000 \), the sizes of trees will be almost even since three IGs are all located in the corner, resulting in the lower MANET-Only utilization. Furthermore, we can see that the MANET-Only resource utilization ratio decreases since MANET becomes unstable with the increased speed.

According to the simulation study, it was proven that \( S_1 \) has the best IG distribution. Now, let us compare \( S_1 \) to \( 1000 \times 1000 \) and \( S_1 \) to \( 1500 \times 1500 \). \( S_1 \) to \( 1500 \times 1500 \) has less number of anchors (trees) due to its reduced node density. So, \( S_1 \) to \( 1500 \times 1500 \) shows the increased average size of tree and thus the increased overhead compared to \( S_1 \) to \( 1000 \times 1000 \), resulting in much less delivery ratio as in Fig. 10. Also, \( S_1 \) to \( 1500 \times 1500 \) shows a slightly higher decrease in delivery ratio as node speed.
increases. This is because the network becomes unstable due to the loose connectivity among mobile nodes. The MANET-Only resource utilization in $S1 - 1500 \times 1500$ with the big trees is higher than that of $S1 - 1000 \times 1000$ since the former will have the increased number of Type1 and Type2 paths. However, node connectivity in $S1 - 1500 \times 1500$ is more likely to be broken because of its loose connectivity. So, its MANET-Only utilization decreases rapidly because communication will have a higher dependency on IGs with increased node speed.

### B. Performance Comparison

We evaluated our approach against the AODV based protocol [3]. It uses a hybrid mobility management approach which mixes both proactive and reactive ones for the efficiency of mobility management. IG floods an agent advertisement message periodically up to $k_1$ hops. A reverse path from any receiving node to the IG is established while the advertisement message is being moved. Any unregistered mobile node that receives the advertisement message sends a registration request message along the reverse path to register with HA and FA. A forward path from IG to a mobile node is established while the registration request message moves. A data packet is delivered to a registered mobile node along the forward path. The IG that receives a registration request message registers the mobile node with FA and HA. After completion of registration, the path is regarded as a valid one until the corresponding timer expires.

A mobile node that is distanced over $k_1$ hops and thus does not receive an agent advertisement message floods agent solicitation message with $TTL = k_2$ in order to find any previously registered node. In this process, the forward path from the receiving node to the node that initiated the agent solicitation message is established. The previously registered node that receives the agent solicitation message sends a response message whose moving establishes the reverse path. Accordingly, a mobile node that has not been registered yet can register with FA and HA along the combined two partial reverse paths.

A node that wants data transmission first examines whether the destination is in its routing table or not. If the path is fresh, it starts sending packets. Otherwise, if it is stale, the source floods RREQ with the old TTL in the routing table to explore a new path. Meanwhile, if a source has not known the destination so far, it floods RREQ with $TTL = 1$. If it does not receive RREP from either the destination or any IGs, it increases TTL by 2 and floods RREQ. This repeats until TTL reaches $TTL_{threshold} (= 7)$. If source still fails to receive RREP, it sets TTL to AODV_DEFAULT_NET_DIAMETER ($= 35$) and floods RREQ as a last resort.

We used two scenarios $S1 - 1000 \times 1000$ and $S1 - 1500 \times 1500$ to compare two approaches. In case of TBRP, we took three types of messages into account in overhead computation: Hello
Fig. 13. Delivery ratio with variation of maximum node speed: TBRP vs. AODV.

Fig. 14. Control overhead with variation of nodes: TBRP vs. AODV.

Fig. 15. Delivery ratio with variation of nodes: TBRP vs. AODV.

Fig. 16. Control overhead with variation of sessions: TBRP vs. AODV.

Fig. 17. Delivery ratio with variation of sessions: TBRP vs. AODV.

message that each node sends every two seconds, messages to form and maintain the TIIM architecture, and messages to establish a route. In Fig. 12, we see that TBRP induces much less overhead than the AODV approach (for AODV, we set $k_1$ and $k_2$ to 3 and 2, respectively which show the best performance). This is because our approach does not use a flooding in both mobility management and routing protocol. As a result, TBRP showed good improvement in delivery ratio by about 12% overall.

Another simulation was conducted by varying the number of nodes as 50, 100, 150, 200 nodes with terrain 1000m $\times$ 1000m and maximum speed 5 m/s. Fig. 14 shows that TBRP induces much less overhead than AODV, showing about 12% gain of delivery ratio overall as shown in Fig. 15. When the number of nodes is 50, AODV shows a low overhead because nodes can be isolated or network can be partitioned transiently, reducing the range of RREQ. However, delivery ratio is quite sensitive to the traffic due to the increase of overhead as in Fig. 16 and Fig. 17. We can easily identify that the increase of overhead is due to the route overhead caused by RREQ, RREP, and RRER. However, overhead by RREQ in TBRP is relatively low compared with that by the other messages (include architecture management control messages). This is because we exploit formation in establishing a path. Considering that almost 50% of nodes participate in active communications for 25 sessions, our protocol is quite competitive.

VI. CONCLUDING REMARKS

We proposed the TIIM architecture to expand Mobile IP for the support of MANET, network architecture management protocol to form and manage the TIIM architecture, and then the TBRP as a routing protocol suitable for the TIIM architecture. Mobile nodes form a number of small trees called pMANETs, each of them growing from an anchor node that is able to communicate directly with an IG. Mobile node can easily register with FA and HA along the pMANET paths without using an in-
efficient flooding. pMANETs are maintained such that all members maintain the shortest path to their IG. The TBRP sets up and optimizes a routing path efficiently by exploiting the tree information of the TIIM architecture. It does not use an inefficient flooding, either. We showed the competitiveness of the proposed protocol by comparing with the well-known AODV based approach.

REFERENCES


Hoon Oh received the B.S.E.E. degree from the Sung Kyun Kwan University, Seoul, and the M.Sc. 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 in the University of Ulsan, Korea. He received a Best Paper Award from the National Academy of Science, USA in 1995. He completed some joint industry-academy projects such as “Applying Ubiquitous Computing Technology to the Steel-Plate Piling Process of Ship Construction” and “Developing Crane Anti-Collision System for the Ship-Building Safety” which was successfully applied to the Hyundai Heavy Industries, Ltd. His research interests lie in IT applications to industrial fields, embedded systems, mobile ad hoc networks, real-time computing, context-aware computing. He is a Member of ACM, ISC, IEICE, KICS, and ICASE, and has been a Lifetime Member of the Korea Information Society since 1989.