ELSEVIER

Computer Communications 25 (2002) 1684–1695

computer communications

www.elsevier.com/locate/comcom

SOM: spiral-fat-tree-based on-demand multicast protocol in a wireless ad-hoc network ☆

Yuh-Shyan Chen[a,*], Tzung-Shi Chen[b], Ching-Jang Huang[a]

[a]*Department of Statistics, National Taipei University, Taipei 10433, Taiwan, ROC*
[b]*Department of Information Management, Chang Jung University, Tainan, Taiwan, ROC*

Received 20 November 2000; revised 7 January 2001; accepted 29 January 2002

Abstract

A mobile ad-hoc network (MANET) is characterized by multihop wireless links, in the absence of any cellular infrastructure, as well as by frequent host mobility. Existing on-demand multicasting protocols are classified into tree-based and mesh-based schemes. This paper presents a robust multicast routing protocol, called the Spiral-fat-tree-based On-demand Multicast (SOM) protocol, which is a means of dynamically establishing a special tree data structure, namely the *spiral-fat-tree*, in the MANET. The *spiral-fat-tree* is constructed by appending some backup paths to a multicast tree constructed by our scheme for the purpose of improving the robustness of the multicast tree. The contribution of the spiral-fat-tree is to maintain the stability and increase the robustness of the multicast tree. A performance study shows that our proposed scheme outperforms existing on-demand multicast protocols.

Keywords: Mobile ad-hoc network; Mobile computing; Multicast; Wireless networks

1. Introduction

A mobile ad-hoc network (MANET) [9] consists of wireless mobile hosts that communicate with each other in the absence of a fixed infrastructure. Due to some factors, such as radio power limitations, power consumption, and channel utilization, a mobile host may not be able to communicate directly with other hosts. A multihop scenario occurs when packets sent by a source host are re-transmitted through several intermediate hosts before reaching the destination host. In a MANET, host mobility results in frequently unpredictable topology changes. Designing a routing protocol is more complicated than that for traditional wired networks. Under low-cost considerations, there are many schemes proposed for designing MANET unicast routing protocols [5,8,11,12,14] and developing MANET multicast routing protocols [1,3,4,6,10,13]. This paper investigates the MANET multicast protocol by examining the robustness of building a multicast tree.

The MANET multicast results include proactive and reactive schemes [1,3,4,6,10,13]. The key difference in proactive and reactive schemes is whether a shared multicast tree is constructed or not. Proactive schemes require high communication time due to high tree-maintenance costs for having mobility-tolerant capabilities. Proactive protocols are not suitable for the MANET. Therefore, reactive schemes have recently received a lot of attention, and have been classified into tree-based and mesh-based schemes. Tree-based multicast protocols consist of a *multicast-tree* structure connecting to all destination hosts. Mesh-based multicast protocols offer a *multicast-mesh* structure with multipaths from a source to all destinations.

Existing tree-based multicast protocols, as depicted in Table 1, comprise CBT (core-based trees) [1], DVMRP [4], and AODV [13] protocols. The CBT scheme [1] is a proactive scheme which maintains a shared-tree whose structure is periodically refreshed. Any source node first sends a multicast message to the root node, and then forwards the message down the CBT tree. The disadvantage of the CBT scheme is that communication latency is high and multicast operation is halted if any node in the CBT fails. The CBT-based scheme provides no mobility-tolerant strategy. The DVMRP scheme [4] is a reactive scheme which dynamically constructs a multicast tree; fortunately, it is possible to construct a loop-free shortest-path tree. With the DVMRP scheme, it is easier to re-configure the multicast

☆ A preliminary version of this paper received the *Best Paper Award* at *IEEE ICOIN-15*: The 15th International Conference on Information Networking, 2001, Beppu City, Japan.

* Corresponding author.
E-mail addresses: yschen@mail.ntpu.edu.tw (Y.-S. Chen), chents@mail.cju.edu.tw (T.-S. Chen).

0140-3664/02/$ - see front matter
PII: S0140-3664(02)00051-8

Table 1
Comparison table for various MANET multicast routing protocols

	Protocol	Proactive/reactive	Multiple path	Location-aware
Tree-based multicast protocols	CBT [1]	Proactive	×	×
	AODV [13]	Reactive	×	×
	DVMRP [4]	Reactive	×	×
Mesh-based multicast protocols	CAMP [6]	Proactive	√	×
	FGMP [3]	Reactive	√	×
	ODMRP [10]	Reactive	√	√
	SOM	Reactive	√	×

tree than it is for the CBT scheme. A hybrid scheme, namely the AODV multicast protocol, was designed by Royer and Perkins [13]. The hybrid scheme inherits advantages of both the proactive and reactive schemes. Conventional tree-based schemes achieve better performance in wired networks; however, tree-based schemes do not perform well in the MANET due to the node-mobility problem.

Existing mesh-based multicast protocols include CAMP [6], FGMP [3], and ODMRP [10] protocols. Basically, mesh-based schemes are more stabile than tree-based schemes because of their use of multipath approach. The CAMP protocol [6] is a proactive scheme which is an extended version of the CBT [1] protocol. The multipath approach adopted in the CAMP protocol aims to provide stable routing. The FGMP protocol [3], which is a reactive approach, introduces the concept of *forwarding group* which is a set of forwarding nodes. Each node in the forwarding group is responsible for re-forwarding the message. This approach produces better performance since all multicast packets only flood the forwarding group. The forwarding group is periodically refreshed to handle topology/membership changes. But it is not easy to recognize the forwarding group. A reactive approach, namely the ODMRP scheme, was presented by Lee et al. [10], and is an extended version of FGMP [3]. The ODMRP scheme adds a node/link-stability consideration based on the associativity-based routing (ABR) protocol [14]. Consequently, the ODMRP protocol is more stable than the FGMP protocol. Unfortunately, the *forwarding group* is not easily maintained or identified.

This paper aims to find a way to enhance the robustness of a *multicast-tree* structure. We developed a novel multipath approach to overcome the drawback of existing multicast protocols. In this paper, we propose a robust multicast routing protocol by dynamically establishing a special tree data structure, called the *spiral-fat-tree*, in the MANET. The *spiral-fat-tree* was established by adding some small backup paths to the conventional multicast tree for the purpose of improving the robustness and stability of the multicast tree. The main advantage of the spiral-fat-tree is that it maintains the stability and increases the robustness of the *multicast-tree*. Our approach is more formally and easily constructed than the *forwarding group* scheme which is adopted in FGMP [3] and ODMRP [10] protocols. Our proposed protocol is a special tree structure with more links (multipath approach), which belongs to the mesh-based multicast protocol as shown in Table 1. Finally, a performance study supports the fact that our proposed scheme outperforms existing on-demand multicast protocols.

The rest of this paper is organized as follows. Section 2 presents the basic idea of the SOM protocol. The SOM protocol is presented in Section 3. Section 4 discusses the performance analysis, and in Section 5, conclusions are presented.

2. Basic idea

A MANET [8] is modeled as an undirected graph $G = (V, E)$, where V is set of $|V|$ mobile hosts (or nodes) and E is a set of $|E|$ undirected links connecting mobile hosts (or nodes) in V. Each node has an unique identifier to represent a distinct mobile host. Each mobile host has a wireless communication device with a fixed transmission range, R. Each node may move away, and change its speed and direction independently. Any node is assumed to know all one-hop neighbors' statuses. An undirected link (i, j) connecting nodes i and j is formed if the distance of nodes i and j is less than R. Link (i, j) is removed from E if the distance of the two nodes is larger than R.

This paper presents a new on-demand multicast tree structure, namely the *spiral-fat-tree*. A spiral-fat-tree is a tree, which has more links near the root than in any other part of the tree. Notably, our scheme belongs to a mesh-based multicast protocol due to adopting the multipath approach. The spiral-fat-tree is a variation of the conventional *fat tree*. The advantage of the spiral-fat-tree is discussed below. Given a tree as illustrated in Fig. 1(a), the bottleneck for transmitting messages in the tree is near the root. This is because the traffic toward the root is heavier. A fat tree as proposed by Hwang [7], has a channel bandwidth which increases as one ascends from the leaves to the root. The increased channel bandwidth is achieved by using a greater number of communication links. The more communication links there are, the more-robust the link will be. For instance, the channel bandwidth of a conventional tree in Fig. 1(b) is double that of the tree in Fig. 1(a). It is impossible to increase the channel bandwidth between two

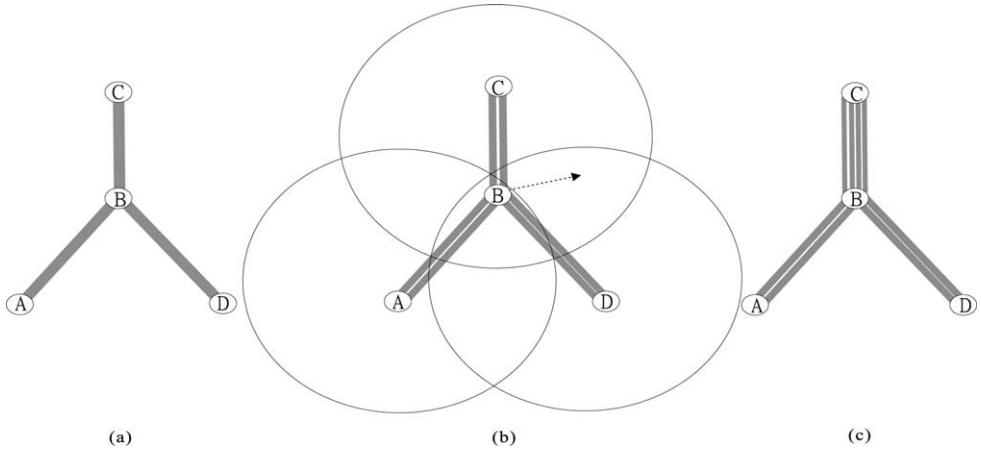

Fig. 1. The fat-tree structure.

one-hop neighboring nodes, as illustrated in Fig. 1(b) for nodes *A* and *B*. In the MANET, if we increase the channel bandwidth to obtain robust links, we then possibly construct multiple communication links.

A fat tree is like a conventional tree, but any link of the fat tree gets thicker toward the root as illustrated in Fig. 1(c). We observe that if node *B* is moving away, the tree structure is broken, even if there exist multiple paths between one-hop neighboring nodes. This implies that it is impractical to directly construct a fat tree in a wireless ad-hoc network. To overcome this limitation, we propose the spiral-tree and spiral-fat-tree in the MANET. The basic idea of the spiral-fat-tree approach is to use a spiral-path instead of the links in the traditional fat tree. We initially introduce the concept of the spiral-path. The spiral-path is a special structure of multiple paths, denoted as spiral-path P_k, which is formally defined below.

Definition 1. *Spiral-path* [2]. Given a conventional path *P*, every node of the path *P* connects to the next *k*-hop node in path *P* by a disjointed link, joining all extra links into the original path *P* to form spiral-path P_k.

This paper mainly uses the spiral-path P_2, as shown in Fig. 2(a), to develop the SOM protocol. This idea can be easily further extended to construct P_k in order to obtain higher route robustness, as illustrated in Fig. 2(b). The route robustness of a spiral-path is achieved by maintaining multiple links. If a path fails, the failed path is immediately replaced with a backup path. The higher the value of *k* is, the more-robust the route will be. This is because when using P_k, multiple consecutive-faults can be tolerated, although the success rate of constructing P_k is lower than that of P_2, where $k \geq 3$ [2]. The spiral-tree and spiral-fat-tree are formally defined below.

Definition 2. *Spiral-tree*. A tree is said to be a spiral-tree if all links in the tree are replaced by spiral-paths.

Definition 3. *Spiral-fat-tree*. A spiral-tree is said to be a spiral-fat-tree if the channel bandwidth of a spiral-tree increases when ascending from the leaves to the root.

Definition 4. *Branch-node*. A node is said to be a branch-node if at least two disjoint paths exist from the same node.

A branch-node is useful for constructing our spiral-tree and spiral-fat-tree. Examples of a spiral-tree and spiral-fat-tree are illustrated in Fig. 3(b) and (c), respectively. The SOM protocol first constructs the spiral-tree and then possibly constructs the spiral-fat-tree if additional extra links exist. If any node in the spiral-tree fails, the tree-maintenance

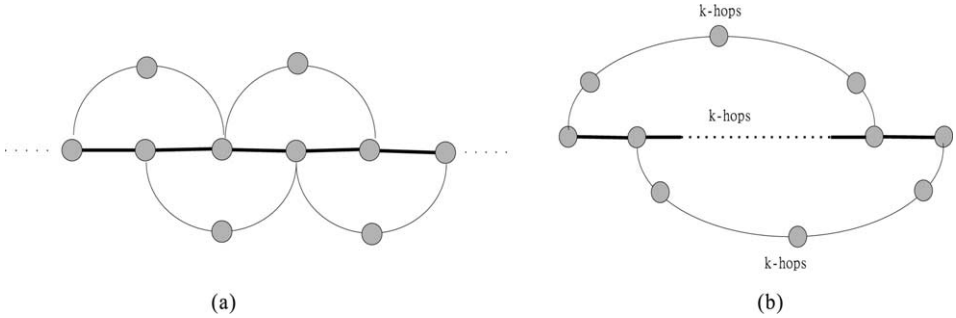

Fig. 2. The spiral path (a) P_2 and (b) P_k.

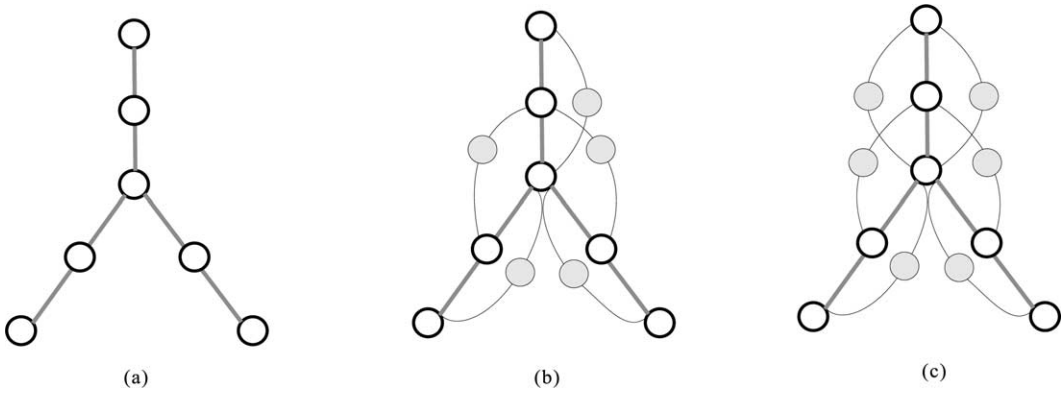

Fig. 3. Construction of a spiral-fat-tree.

operation immediately recovers the tree-structure by using backup paths. The spiral-fat-tree aims to further enhance the robustness and stability of the spiral-tree. Observe that the spiral-fat-tree is a spiral-tree if we increase the number of multiple paths ascending from the leaves to the root. A node near the root failing always takes more costs to re-configure in a multicast-tree than one failing near a leaf. The robustness of links near the root of the spiral-fat-tree has been enhanced.

3. The SOM protocol

We first give an overview of our proposed SOM: Spiral-fat-tree On-demand Multicast routing protocol. The SOM protocol mainly constructs a spiral-tree and a spiral-fat-tree to perform the on-demand multicast routing operation. The SOM protocol is achieved by three phases of *branch-node identification*, *spiral-fat-tree construction*, and *multicast-tree maintenance*. The *branch-node identification* phase tries to identify all branch-nodes in the MANET. The *spiral-fat-tree construction* phase constructs the spiral-tree and spiral-fat-tree by merging multipaths and backup paths from a source to all destinations. The *multicast-tree maintenance* phase aims to maintain the structure of the spiral-tree and spiral-fat-tree for the sake of maintaining high robustness of the multicast operation since multicast-tree stability has been strengthened.

3.1. Routing information

Given node e, assume that there are k gateway nodes between branch-node e'. That is, there are k disjointed paths between a pair of branch-nodes e and e'. Assume that the path between e and e' is a sub-path of the primary path from a source to one destination. A beacon packet is used to maintain the link stability. The routing table of node e is formally defined below.

- RoutingTable.*DestinationGroup* is the multicast group ID to represent a set of all destination nodes.
- RoutingTable.*PrimaryNextHop*[i] records the next-hop node, for node e, for one of the primary paths. Usually $i = 1$; a branch of the multicast tree occurs if $i > 1$.
- RoutingTable.*SecondNextHop*[i] records the next-hop node, for node e, of the backup path for the corresponding primary path.
- RoutingTable.$Click_j$ represents the stability status for the jth *gateway node* between nodes e and e', where $j \leq k$.
- RoutingTable.$TwoHopPath_j$ denotes the two-hop path from e to e' and through the jth *gateway node*, where $j \leq k$.
- RoutingTable.*BranchNode*: is TRUE if node e is a *branch-node*.

Observe that, $Click_j$ is maintained in the *spiral-fat-tree construction* phase by using beacon packets to keep track of the connecting status of the jth gateway nodes. Using the beacon packet allows us to construct a more-robust spiral-path and spiral-fat-tree for multicasting.

3.2. Phase I: branch-node identification

Our algorithm needs to ordinarily identify the *branch-nodes*. Fortunately, better robustness is obtained by means of using some extra overhead.

If a node is a *branch-node*, then at least two disjointed paths exist from other branch-nodes. This paper only considers the effect of using spiral-path P_2 to construct the spiral-tree and spiral-fat-tree. This means that we just identify a branch-node if multiple distinct paths from other branch-nodes exist, where the path length is equal to 2.

Given a pair of two-hop adjacent nodes, e and e', a node is said to be a *gateway node* if the gateway node can communicate with both nodes e and e'. Since k gateway nodes possibly exist between nodes e and e', nodes e and e' are branch-nodes. This operation mainly identifies the branch-nodes in the MANET. This task is achieved by periodically flooding a beacon packet within 2-hop neighbors. Denote the beacon packet as Beacon(*HopNumber*, *NodeList*), where *HopNumber* is the life time of the beacon packet, and *NodeList*, $|NodeList| \leq HopNumber = 2$, is a node list which represents the traversal path of the beacon packet. Consider

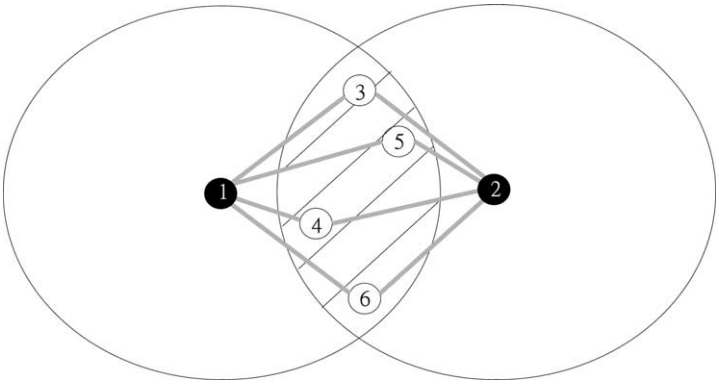

Fig. 4. Identifying branch-nodes.

a pair of branch-nodes, e and e', with k gateway nodes between nodes e and e', then the identifying operation is formally defined below.

1. Each node e floods and re-forwards a Beacon(*HopNumber*−−, *NodeList*) if $HopNumber > 0$. Initially, $HopNumber = 2$ and $NodeList = \{e\}$, and *NodeList* will record the node list of the traversal path of the Beacon. This information is also kept in Routing Table. $TwoHopPath_j$, where the beacon packet goes through the jth gateway node, and $j \leq k$.
2. Denote $Beacon_j.NodeList$ [1] as a distinct beacon packet. Suppose that node e' receives two beacons, $Beacon_1$ (*HopNumber*, *NodeList*) and $Beacon_2$(*HopNumber*, *NodeList*), which are sent from the j-gateway node and node e, respectively. If $Beacon_1.NodeList$ [1] = $Beacon_2.NodeList$ [2] then increase RoutingTable.$Click_j$, where j-gateway node = *NodeList* [1] in $Beacon_1$. This implies that the j-gateway node is still at the gateway area. The higher the value $Click_j$ is, the higher the stability will be.
3. If no beacon packet is sent from the jth gateway node for a period of time, set RoutingTable.$Click_j = 0$, because that jth gateway node has run away.
4. Suppose that $Beacon_a$(*HopNumber*, *NodeList*) and $Beacon_b$(*HopNumber*, *NodeList*) exist. If $Beacon_a$. *NodeList* [1] = $Beacon_b$.*NodeList* [1], then *NodeList* [1] is a branch-node, and set RoutingTable.*BranchNode* = TRUE.

The information kept in RoutingTable.$TwoHopPath_j$ is used in the next phase to determine a stable spiral-path. Generally, the larger the value $Click_j$ is, the higher the stability of the corresponding gateway will be. Our strategy, by Step 3, tries to eliminate the ping-pong effect which was discussed in Ref. [5]. Our path selection strategy is performed by constructing the spiral-tree and spiral-fat-tree with a maximum $Click_j$ value. The beacon packet is used to identify the branch-node, in Step 4, if the packets are delivered from the same node. Fig. 4 illustrates an example in which node 1 receives beacon packets from node 2 through gateway nodes 3, 4, and 5; thus node 1 is a branch-node. The status remains stable between nodes 1 and 2 via gateway nodes 3, 4, and 5.

3.3. Phase II: spiral-fat-tree construction

A spiral-fat-tree is constructed from a given source node. We assume that each branch-node knows all of its two-hop neighboring nodes after the branch-node identification phase. This information is available for constructing the spiral-tree and spiral-fat-tree.

1. *Multiple-path searching*. Multiple paths from source to destination nodes are found based on the total number of branch-nodes. Each destination node will identify a group of paths from the source node. One selected path is satisfied by the following consideration. One non-shortest path with a greater number of branch-nodes gives a higher priority to the shortest path with a fewer number of branch-nodes. This implies that a path is searched for while considering its mobility-tolerance. This path information is sent back to the source node, which is called the reversed-path in this paper.
2. *Multiple-path merging*. A spiral-fat-tree is constructed by merging all reversed-paths. At the same time, an important merging criterion is used to merge all reversed-paths to establish the stable spiral-tree and spiral-fat-tree.

3.3.1. Multiple-path searching

This section presents the operation for exploiting multi-paths from a source to its destinations. This work is achieved by maintaining a counter, which denotes the number of branch-nodes in a search path. The larger the value of the counter is, the higher the stability of the path will be. A path from a source to a destination node with a high counter value will be selected, even if this path is not the shortest path.

The request packet is denoted as MREQ(*DestinationSet*, *Counter*, *PathRecord*), where *DestinationSet* represents the set of destination nodes, *Counter* indicates the number of

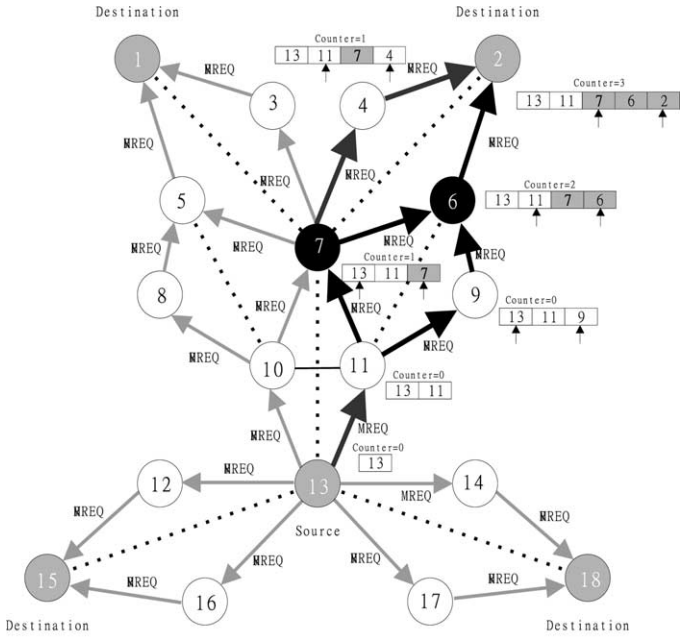

Fig. 5. Example of multiple-path searching.

branch-nodes in the path, and *PathRecord*, a path stored as an array, records the history-path from the source node to the current node. Here we use the notation $\langle s_1, s_2, \ldots, s_i \rangle$ to denote a directed path which needs $i-1$ hops starting from s_1 to s_i. The operation, including the following four steps, is formally developed below.

1. The source node S initiates a packet, MREQ(*DestinationSet*, $Counter = 0$, $PathRecord\ [1] = S$), where *DestinationSet* denotes the set of all destinations.
2. If node $E \notin DestinationSet$, after receiving a MREQ in MANET, it performs Step 3 repeatedly.
3. Assume that each node E knows all its two-hop neighboring nodes. If E and $PathRecord[i-2]$ are a pair of two-hop neighboring nodes, node E forwards packet MREQ(*DestinationSet*, $Counter{+}{+}$, $PathRecord[{+}{+}i] = E$) into MANET. Otherwise, node E forwards packet MREQ(*DestinationSet*, *Counter*, $PathRecord[{+}{+}i]\ E$). E).
4. If node $E \in DestinationSet$, it waits for a period of time in order to receive multiple paths from the source node. One shortest path with the maximum value of *Counter* is eventually selected.

Fig. 5 shows a scenario which assumes that the source node is node 13 and the destination set $D = \{1, 2, 15, 18\}$. We show how to select multiple paths from source node 13 to node 2. First, source node 13 initiates a MREQ (D, 0, ⟨13⟩) to node 11, and node 11 forwards the MREQ(D, 0, ⟨13,11⟩) to its neighboring nodes except for node 13. Then nodes 7 and 9 receive the same packet MREQ(D, 0, ⟨13,11⟩) from node 11. The counter of node 7 becomes 1 to represent that another two-hop path between nodes 13 and 7 exists from $PathRecord = \langle \underline{13}, 11, \underline{7} \rangle$. This information is maintained by the identifying branch-node operation. On the contrary, the counter of node 9 is not increased because nodes 13 and 9 do not have another two-hop path from $PathRecord = \langle \underline{13}, 11, \underline{9} \rangle$. Finally, destination node 2 receives two paths: $\langle 13, 11, 7, 6, 2 \rangle$ and $\langle 13, 11, 7, 4 \rangle$. The counter of the former path is 3 and its path length is 5, but the counter of the latter one is 1 and its length is 4. Our scheme selects the former one. Similar results can be derived from the source to all other destinations.

3.3.2. Multiple-path merging

The reply packet is denoted MREP(*DestinationSet*, *PathRecord*, *Counter*, *HopNumber*). The initial value of MREP.*DestinationSet* is one of the possible destination nodes, which initiates the reply packet. After completing the merging operation, MREP.*DestinationSet* records all destination nodes in the merged path. MREP.*PathRecord* denotes the path-history. MREP.*Counter* denotes the number of branch-nodes in the path. MREP.*HopNumber* represents the longest hop number from the source to the destination nodes.

In fact, each destination node may reply to many MREP packets if the node receives many disjointed paths from the source node. Therefore, we describe the merging criterion as follows.

1. A path with more destination nodes takes a higher priority over a path with fewer destination nodes (as shown in Fig. 6(b) and (c).

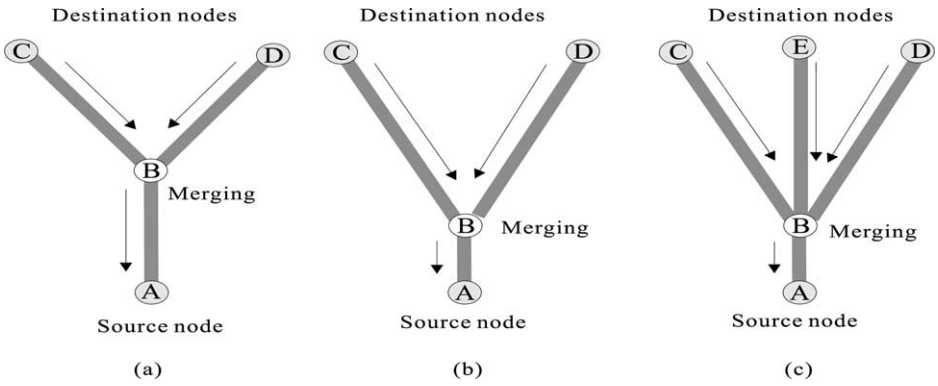

Fig. 6. Merging criteria.

2. Consider two paths containing the same number of destination nodes.

   The merging operation of one path occurs nearer to the source node than does the other one, so it takes a lower priority. For instance, the result of Fig. 6(a) is better than the result of Fig. 6(b).

For simplicity in describing our work, we only let each destination node reply to one path. That is, we obtain a better multicast tree if we let destination nodes reply to all possible paths. Now we return to discuss the multiple-path merging operation. The multiple-path merging operation is used to merge reversed paths from all destination nodes to the source node. The main work in this operation is to fill out the next-hop information of primary and secondary paths in the routing table. The operation is formally described below.

(1) Each destination node D replies to a packet MREP($DestinationSet = \{D\}$, $PathRecord$, $Counter = 0$, $HopNumber = 0$) when the reply path is the reversed path of $PathRecord$.

If node E receives a MREP from node E', set RoutingTable.$PrimaryNextHop$ [1] to be path $\langle E', \rangle$, where E' is a destination node.

Example: Fig. 7 shows that node 2 replies to node 6 with a MREP($\{2\}$, $\langle 13, 11, 7, \underleftarrow{6, 2} \rangle$, 0, 1), since node 2 is a destination node; therefore, node 6 sets RoutingTable.$PrimaryNextHop$ [1] = $\langle 2 \rangle$.

(3) Suppose that node E receives a MREP from node E', but E' is not a destination node. Let index i be the position of $PathRecord$ of node E. If E and $PathRecord[i + 2]$ are a pair of two-hop neighboring nodes and a two-hop path $\langle E, E_1, PathRecord[i + 2] \rangle$ exists, where $E_1 \neq PathRecord[i + 1]$, we set RoutingTable.$PrimaryNextHop$ [1] to be node $\langle E' \rangle$ and RoutingTable.$SecondNextHop$ [1] = $\langle E_1, PathRecord[i + 2] \rangle$.

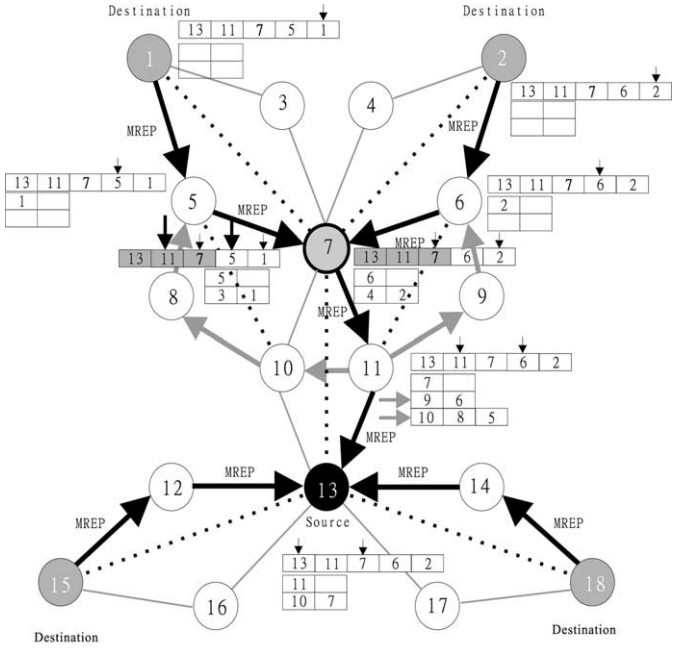

Fig. 7. Example of multiple-path merging.

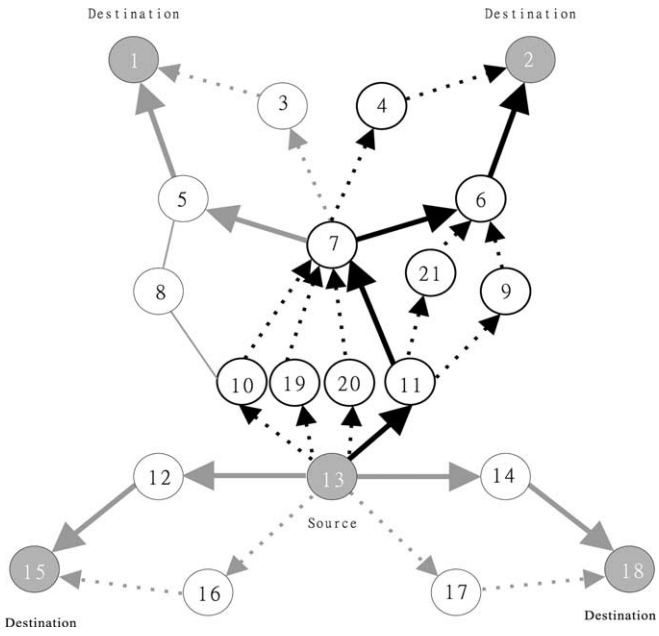

Fig. 8. Example of the spiral-fat-tree.

Example: Fig. 7 shows that node 6 replies to node 7 with a MREP({2},⟨13, 11,$\overleftarrow{7, 6, 2}$⟩, 0, 2), since node 6 is not a destination node; so, node 7 sets RoutingTable.*PrimaryNextHop* [1] = ⟨6⟩. We also set RoutingTable.*SecondNextHop* [1] = ⟨4,2⟩, because nodes 7 and 2 have another two-hop path ⟨7,4,2⟩.

(4) Suppose that node E receives a MREP from node E', while E' receives k MREPs. Let $PathRecord_j$, $1 \le j \le k$, denote the history-paths in E's MREPs. First, RoutingTable.*PrimaryNextHop* [1] and RoutingTable.*SecondNextHop* [1] are obtained in a similar way as described in Steps 2 and 3. In addition, RoutingTable.*SecondNextHop*[j] is a disjointed path $\langle E, E', PathRecord_j[i+2]\rangle$, where index i is the position of *PathRecord* of node E and $j = 2...k$.

Example: Fig. 7 illustrates that node 7 replies to node 11 with a MREP({1,2}, ⟨13,$\overleftarrow{11, 7}$,$\overleftarrow{6}$, 2⟩, 1, 3), but node 7 has two packets MREP({2}, ⟨13, 11,$\overleftarrow{7, 6, 2}$⟩, 1, 3) and MREP({1}, ⟨13, 11,$\overleftarrow{7, 5, 1}$⟩, 3, 4). We set RoutingTable.*PrimaryNextHop* [1] = ⟨7⟩, RoutingTable.*SecondNextHop* [1] = ⟨9,6⟩, and RoutingTable.*SecondNextHop* [2] = ⟨10,8,5⟩ since a disjointed path ⟨11, 10, 8, 5⟩ exists between nodes 11 and 5 in the path ⟨13, 11, 7,$\overset{5}{\uparrow}$, 1⟩.

(5) Suppose that node E receives multiple MREPs from different nodes, which are not the destination nodes. Assume that there are k MREPs, and RoutingTable.*PrimaryNextHop*[2…k] and RoutingTable.*SecondNextHop*[2…k] are obtained in a similar way as described in Steps 2 and 3. Let index i be the position of *PathRecord* of node E. Two cases are discussed below.

(a) If all received paths have the same *PathRecord*[1…i], we only reply to node *PathRecord*[i − 1] with a MREP.

(b) If not all of the received paths have the same *PathRecord*[1…i], we only reply to node *PathRecord*[i − 1] with a MREP if *PathRecord*[1…i] has the shortest path to the source node and also has the maximum number of branch-nodes.

Example: Fig. 7 shows that node 7 indeed receives a second MREP packet from node 5; therefore, we may set RoutingTable.*PrimaryNextHop* [2] = ⟨5⟩ and RoutingTable.*SecondNextHop* [2] = ⟨3,1⟩. The two MREPs have the same paths ⟨13, 11, 7⟩, so only one packet is sent back to node 11.

This example shows that node 11 has three paths, ⟨11, 7⟩, ⟨11, 9, 6⟩, and ⟨11, 10, 8, 5⟩, to maintain the multicast operation. That is why our multicast tree is called the spiral-fat-tree. Generally speaking, for high mobility-tolerance, we also let the node near the root node have a greater number of disjointed backup-paths. Another example of the spiral-fat-tree is given in Fig. 8, in which there are four disjointed paths, ⟨13, 11, 7⟩, ⟨13, 20, 7⟩, ⟨13, 19, 7⟩, and ⟨13, 10, 7⟩ between source node 13 and node 7. The main advantage of the spiral-fat-tree is that it enhances the robustness of the multicast tree for nodes near the root, which aims to reduce the possibility of re-configuring the multicast tree.

3.4. Phase III: multicast-tree maintenance

Our main contribution, in this work, is to provide on-line route-recovery capability. A node is said to have failed if it is outside of the original transmission radius or if it really fails. We show how to achieve on-line recovery capability such that the multicast operation can be continually performed. The on-line recovery capability of the SOM

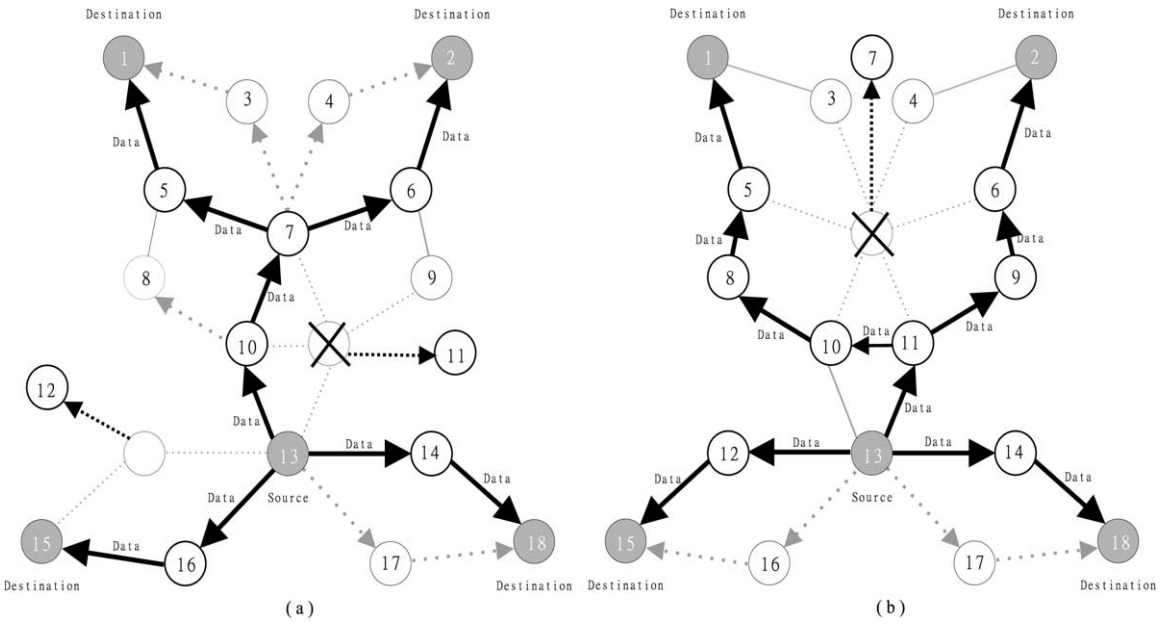

Fig. 9. Examples of multicast-tree maintenance.

protocol is achieved using a path replacement strategy. The path replacement strategy is given according to the different roles of a failed node.

1. If the failed node is not a branch-node, a backup path RoutingTable.*SecondaryNextHop* [1] is used to replace the failed path. We now initiate a local searching-path operation between a pair of branch-nodes to find other new backup paths to preventRoutingTable.*SecondaryNextHop* [1] becoming a failed path. For instance, Fig. 9(a) shows that if node 11 is a failed node and path $\langle 13, 11, 7\rangle$ is broken, node 13 can use RoutingTable.*SecondaryNextHop* [1] = $\langle$13,10,7$\rangle$ to replace the failed path. We further check the new backup path to see if there is any two-hop path between nodes 13 and 7.
2. If the failed node is a branch-node, two backup paths, RoutingTable.*SecondaryNextHop* [1] and RoutingTable.*SecondaryNextHop* [2], can immediately be used to recover the failed paths. A similar searching-path operation is performed herein. For instance, Fig. 9(b) illustrates that if node 7 is a failed node and both paths $\langle 11, 7, 5\rangle$ and $\langle 11, 7, 6\rangle$ are broken, node 11 may use RoutingTable.*SecondaryNextHop* [1] = $\langle$11,9,6$\rangle$ and RoutingTable.*SecondaryNextHop* [2] = $\langle$11,10,8,5$\rangle$ to proceed with data transmission.

4. Performance analysis

We have developed a simulator using Java language as a experimental platform. The core of this simulator is a discrete event-driven engine designed to simulate systems that can be modeled by processes communicating through signals. The parameters used in our simulator are listed below.

- The number of mobile hosts ranges from 50 to 100.
- The mobile speed of each host is from 0–10 to 0–90 km/h.
- The transmission radius ranges from 50 to 150 m.
- The routing protocols we designed include AODV, DVMRP, FGMP, ODMRP, and the SOM.
- The data transmission rate is set to 2 Mb/s.
- The maximum size of the control packet is 2K.
- The message length ranges from 1 to 30K.
- The number of destination nodes ranges from 3 to 10.

The area in which hosts move is bounded by 500×500 m^2. To simulate host mobility, each host is simulated by generating a series of turns. For each turn, a direction, a velocity, and a time interval are uniformly generated. The direction is uniformly distributed from 0 to 360°, and the time interval is uniformly distributed from 1 to 100 s. The velocity of a mobile host is randomly chosen from 0 to V km/h, where $10 \leq V \leq 90$. The performance metrics contain:

- *REachability (RE)*: the number of all destination nodes receiving the data message divided by the total number of all destination hosts that are reachable, directly or indirectly, from the source host.
- *ReBroadcast (RB)*: the number of MREQ packets for all mobile hosts in the MANET.
- *Average Latency (AL)*: the interval from the time the multicast is initiated to the time the last host finishes its multicasting.

It is worth mentioning that an efficient multicast protocol is achieved by having a high *RE*, a low *RB*, and a low *AL*. In the

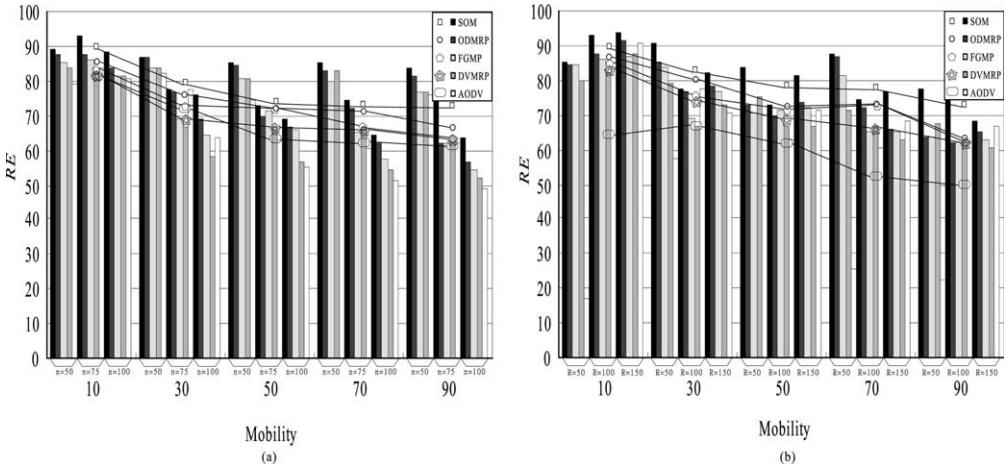

Fig. 10. Performance of *RE* vs. effect of (a) the number of mobile hosts, and (b) the transmission radius.

following, we illustrate our simulation results of *RE*, *RB*, and *AL* from various prospects.

4.1. Performance of reachability vs. mobility

The simulation results of AODV, DVMRP, FGMP, ODMRP, and SOM routing protocols are shown in Fig. 10 to reflect the performance of *RE* vs. mobility. The average *RE* is obtained by calculating the average value of all estimated *RE* values. Two kinds of effects are discussed below.

(1A) Effect of the number of mobile hosts. Each value in Fig. 10(a) was obtained by assuming a transmission radius of 100 m, while the number of mobile hosts ranged from 50 to 100. A higher *RE* indicates that a better scheme was achieved. Fig. 10(a) shows that the SOM scheme has a higher *RE* than do the other schemes even with various numbers of mobile hosts and values of mobility. For example, the average *RE*s of SOM, ODMRP, FGMP, DVMRP, and AODV are 80, 77, 73, 69, and 72%, respectively, when the mobility of hosts was 0–30 km/h. To see the effect of the number of mobile hosts, two situations can be observed. For low mobility (at 0–10 km/h), the greater the number of mobile hosts is, the higher the *RE* will be. For high mobility (at 0–90 km/h), the greater the number of mobile hosts is, the lower the *RE* will be. This is because the probability of re-constructing a route path increases with a greater number of mobile hosts and higher mobility.

(1B) Effect of Transmission Radius. Each value in Fig. 10(b) was obtained by assuming the number of mobile hosts to be 75. Average *RE*s of AODV, DVMRP, FGMP, ODMRP, and SOM were obtained with a transmission radius which varied from 50 to 100. Observe that our scheme has a lower *RE* compared to other schemes under various transmission radii, as shown in Fig. 10(b). For instance, the average *RE*s of SOM, ODMRP, FGMP, DVMRP, and AODV were 83, 80, 76, 75, and 68%, respectively, when the mobility of the hosts was 0–30 km/h. Observe that the average *RE* of SOM is 8% better than those of the AODV, DVMRP, FGMP, and ODMRP protocols.

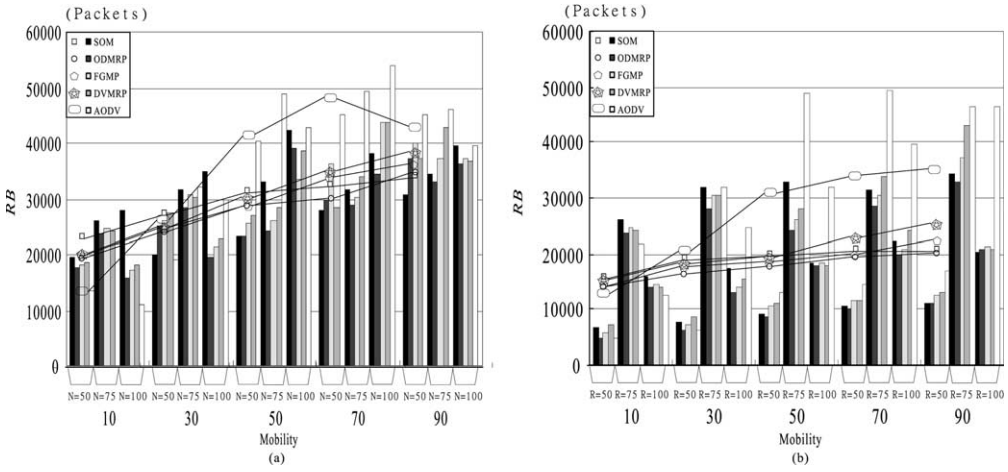

Fig. 11. Performance of *RB* vs. effect of (a) the number of mobile hosts, and (b) the transmission radius.

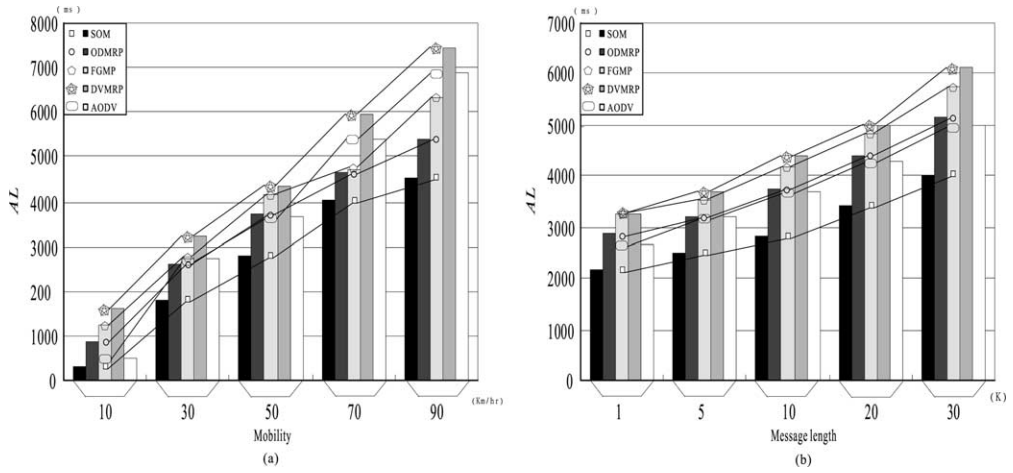

Fig. 12. Performance of *AL* vs. (a) mobility, and (b) message length.

4.2. Performance of rebroadcast vs. mobility

The simulation results shown in Fig. 11 illustrate the performance of *RB* vs. mobility. The average *RB* was obtained by calculating the average value of all estimated *RB* values. Two kinds of effects are discussed below.

(2A) Effect of number of mobile hosts. The simulation assumption is the same as in case *1A*. A lower *RB* indicates a better scheme. Fig. 11(a) shows that the SOM scheme has a better *RB* for high node-mobility and has a worse *RB* for low node-mobility. For instance, average *RB*s of AODV, DVMRP, FGMP, ODMRP, and SOM were 13,001, 20,145, 20,032, 19,354, and 23,443, respectively, when the mobility was 0–30 km/h. But with high node-mobility, the SOM scheme has an improved average *RB*. The reason is that the SOM scheme needs to rebroadcast extra packets to maintain stability. It is worth mentioning that the effect of *RB* can be dominated using the robust path, for high node-mobility. For instance, the average *RB*s of AODV, DVMRP, FGMP, ODMRP, and SOM were 43,150, 38,145, 35,121, 33,878, and 34,127, respectively, for high mobility (0–90 km/h).

(2B) Effect of transmission radius. The simulation assumption is the same as case *1B*. Fig. 11(b) illustrates that our scheme has a lower *RB* under a large transmission radius. For instance, average *RB*s of AODV, DVMRP, FGMP, ODMRP, and SOM were 36,132, 26,157, 22,103, 20,945, and 20,321, respectively, when the mobility was 0–50 km/h. We see that our scheme has a better *RB* under a large transmission radius and high node-mobility as shown in Fig. 11(b).

Comparing Fig. 10 with Fig. 11, we observe that the SOM scheme provides better average reachability with about 20% extra re-broadcast packets than do the other schemes.

4.3. Performance of average latency vs. mobility and load

The simulation results of AODV, DVMRP, FGMP, ODMRP, and SOM routing protocols are shown in Figs. 12 and 13 to reflect the effects of average latency. To

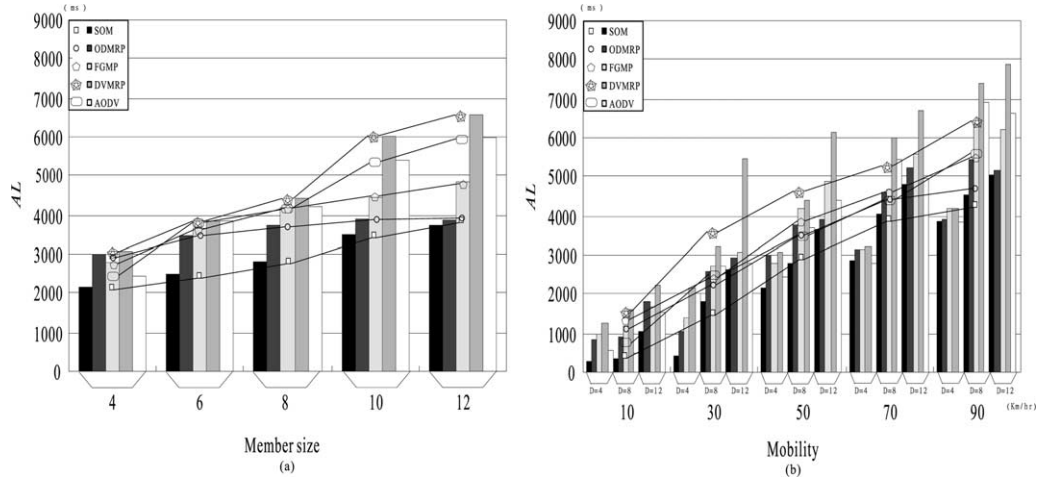

Fig. 13. Performance of *AL* vs. (a) mobility/number of destination nodes, and (b) number of destination nodes.

estimate the performance of *AL*, four kinds of effects are discussed.

(3A) Effect of mobility. Fig. 12(a) shows *AL* when message length was 10K. The SOM scheme incurs lower latency than do the other schemes. This reflects the fact that our scheme has a better performance than the other schemes at various mobilities. For instance, *AL*s of SOM, ODMRP, FGMP, AODV, and DVMRP were 4505, 5419, 6319, 6884, and 7417 ms, respectively, when the mobility was 90 km/h.

(3B) Effect of message length. Fig. 12(b) shows *AL* when the transmission radius was 150 m and the message length ranged from 1 to 30K. We can see that our scheme has a better performance than do the other schemes at various message lengths. For instance, *AL*s of SOM, AODV, ODMRP, FGMP, and DVMRP were 2817, 3638, 3751, 4175, and 4370 ms, respectively, when the message length was 10K.

(3C) Effect of number of destination nodes. Fig. 13(a) shows *AL* when the transmission radius was 150 m and the number of destination nodes varied from 4 to 12, and mobility was 50 km/h. The SOM scheme also has a good performance. For instance, as shown in Fig. 13(a), average *AL*s of SOM, AODV, ODMRP, FGMP, and DVMRP were 2135, 2420, 2810, 2986, and 3069 ms, respectively, when the number of destination nodes was 4. On the other hand, average *AL*s of SOM, AODV, ODMRP, FGMP, and DVMRP were 3910, 4054, 5850, 5960, and 6613 ms, respectively, when the number of destination nodes was 12. Therefore, our scheme has a better performance than do the other schemes even if there are a greater number of destination nodes.

(3D) Effect of mobility vs. number of destination nodes. The simulation assumption is the same as in case *3A*. Fig. 13(b) shows that the SOM scheme has better *AL* than do the other schemes. This reflects the fact that our scheme has the best performance of all schemes for various mobilities. For instance, as shown in Fig. 13(b), average *AL*s of SOM, ODMRP, FGMP, AODV, and DVMRP were 402, 1010, 1314, 2107, and 2001 ms, respectively, when the number of destination nodes was 4 and the mobility of each host was 30 km/h. On the other hand, average *AL*s of SOM, ODMRP, FGMP, AODV, and DVMRP were 5004, 5113, 6235, 6687, and 7913 ms, respectively, when the number of destination nodes was 12 and the mobility of each host was 90 km/h. Thus, our scheme has a better performance than do the other schemes, under a greater number of destination nodes and high mobility.

Apparently, it is desirable to have a high *RE* as well as a high *AL*. Generally, the higher the *RE* is, the higher the *AL* will be. It is always beneficial to adopt our proposed scheme as demonstrated by the simulation results.

5. Conclusions

This paper presents a robust multicast routing protocol by means of identifying a special tree data structure, namely *spiral-fat-tree*, in the MANET. The *spiral-fat-tree* was constructed for the purpose of improving the robustness of the multicast tree. The contribution of the spiral-fat-tree is to maintain the stability and increase the robustness of the multicast tree. Finally, a performance studies illustrate that our proposed scheme outperforms the existing on-demand multicast protocols. Future work will extend the SOM scheme to the Quality-Of-Service (QoS) multicast protocol in the MANET.

Acknowledgements

This work was supported by the Ministry of Education, Taiwan, under Grant 90A-H-FA07-1-4 (Learning Technology), and the National Science Council of the Republic of China, Taiwan, under Grant NSC90-2213-E-305-001.

References

[1] T. Ballardie, P. Francis, J. Crowcroft, Core-based tree (cbt) an architecture for scalable inter-domain multicast routing, ACM SIGCOMM (1993) 85–95.
[2] Y.-S. Chen, K.-C. Lai, MESH: multi-eye spiral-hopping protocol in a wireless ad hoc network, IEICE Transactions on Communications E84-B (8) (2001) 2237–2248.
[3] C.-C. Chiang, M. Gerla, L. Zhang, Forwarding group multicast protocol (fgmp) for multihop, mobile wireless networks, Cluster Computing October (1998) 187–196.
[4] S.E. Deering, D.R. Cheriton, Multicast routing in datagram internetworks and extended lans, ACM Transactions on Computer Systems 8 (2) (1990) 85–110.
[5] R. Dube, C.D. Rais, K.-Y. Wang, S.K. Tripathi, Signal stability-based adaptive routing (ssa) for ad hoc mobile networks, IEEE Personal Communications February (1997) 36–45.
[6] J.J. Garcia-Luna-Aceves, E.L. Madruga, The core-assisted mesh protocol, IEEE Journal on Selected Areas in Communications 17 (8) (1999).
[7] K. Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, McGraw-Hill, New York, USA, 1993.
[8] D.B. Johnson, D.A. Maltz, Dynamic source routing, Mobile Computing (1996) 81–153.
[9] J. Jubin, J. Tornow, The darpa packet radio network protocols, Proceedings of the IEEE 75 (1) (1987) 21–32.
[10] S.-J. Lee, W. Su, M. Gerla, On-demand multicast routing protocol (odmrp) for ad hoc networks. IETF manet (draft-ietf-manet-odmrp-02.txt), 2000.
[11] S. Murthy, J.J. Garcia-Luna-Aceves, An efficient routing protocol for wireless networks, ACM Mobile Networks and Applications October (1996) 183–197.
[12] C.E. Perkins, P. Bhagwat, High dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers, Computer Communications Review October (1994) 233–244.
[13] E.M. Royer, C.E. Perkins, Ad-hoc on-demand distance vector routing, Second IEEE Workshop on Mobile Computer Systems and Applications February (1999) 25–26.
[14] C.K. Toh, Associativity-based routing for ad-hoc mobile networks, Wireless Personal Communications Journal 4 (2) (1997) 103–139.