Family ACK Tree (FAT): Supporting Reliable Multicast in Mobile Ad Hoc Networks

Wanjiun Liao and Ming-Yu Jiang

***Abstract*—In this paper, a new protocol, called Family ACK Tree (FAT), is proposed to support reliable multicast service for mobile ad hoc networks. For each reliable multicast protocol, a recovery scheme is used to ensure end-to-end delivery of unreliable multicast packets for all group members. FAT employs a tree-based recovery mechanism that localizes ACKs and retransmissions to avoid feedback implosion. To cope with node movements, FAT constructs an ACK tree on which each node maintains reachability information to three generations of nodes on the ACK tree. When a tree is fragmented due to a departed node, the fragments will be glued back to the tree using the underlying multicast routing protocol. FAT then adopts an adaptive scheme to recover missed packets that have been multicast to the group during fragmentation and are not repaired by the new reliability agent. We have conducted simulations to compare the performance of FAT with existing solutions. The results show that FAT achieves better performance for the provision of reliable service in ad hoc networks, in terms of reliability, scalability, and delivery efficiency.**

***Index Terms*—Family ACK tree (FAT), mobile ad hoc network, reliable multicast.**

I. Introduction

A MOBILE ad hoc network [1] is an autonomous system composed only of mobile nodes. In such a network, each node plays both roles of a terminal and a router and communicates with one another without the support of any wired infrastructure. Nodes may move in and out of the transmission ranges of the other nodes. This may cause disruptions of ongoing connections.

IP multicast is an efficient group communications mechanism. It avoids transmitting packets from a sender to each recipient separately. With a class D group address in the destination address field of the IP header, a multicast packet is delivered to group members with the same "best effort" delivery as unicast IP transmission. Reliable multicast protocols operate on multicast delivery trees constructed by multicast routing protocols. They ensure reliable end-to-end delivery of unreliable multicast datagrams for all group members. Each reliable multicast protocol needs a recovery mechanism to cope with occasional losses, errors, duplications, and out-of-order delivery of datagrams. The recovery mechanism may rely on feedback messages using either an acknowledgment (ACK) to report correct data reception or a negative acknowledgment (NAK) to request retransmissions. It is a challenge for reliable multicast protocols to satisfy simultaneously the requirements of efficiency and scalability while ensuring reliability. Feedback implosion is one of the problems. As the number of group members grows, the number of feedback messages increases dramatically. This leads to a heavy burden on data sources and causes more severe congestion and packet losses. There has been much work on reliable multicasting, including ACK-based sender-initiated protocols [2], NAK-based receiver-initiated protocols [3], ring-based protocols [4], [5], and tree-based protocols [6]–[8]. As shown in [9], of these protocols, tree-based protocols, especially in combination with NAK-suppression schemes, perform the best in terms of efficiency and scalability.

Reliable multicast in mobile ad hoc networks adds the dimension of host mobility within the scope of reliable multicast. The semantics of group model and reliable multicast transport remain unchanged. Multicast datagrams are still delivered via best effort, unreliable service to mobile group members. Reliable delivery is then ensured by mobile reliable multicast protocols. In other words, once having joined a multicast group, a mobile node will not experience packet losses or duplications due to roaming. This allows mobile nodes to enjoy the service quality of reliable multicast as if they were fixed hosts. They can receive data streams continuously and reliably even while roaming. The challenge to extend reliable multicast protocols for mobile ad hoc networks is to cope with node movements even while forwarding packets. Node movement may cause some nodes in the same group to be disconnected from the multicast tree, hence missing some multicast packets, even though they will eventually be glued back to the tree.

Reliable multicast for fixed hosts has been an active research area [2]–[9]. However, providing reliable multicast service for ad hoc networks has attracted little attention. Reference [10] studies multicast in ad hoc networks. The authors suggest a scheme that support reliable multicast for a set of predefined group members. They introduce the concept of "forwarding regions" that limit the flooding of messages to the immediate vicinity of the nodes which experienced topology changes due to host mobility. Each group member sends ACK messages to the core node of the tree (i.e., tree root). Upon receiving the acknowledgment to a packet from all members in the group, the core notifies the packet sender of correct reception. The sender then stabilizes the packet. Once the packet has been stabilized, the core piggybacks the information of message's stability on the successive multicast datagrams to all group members. Each

Manuscript received September 16, 2002; revised March 9, 2003 and April 10, 2003. This work was supported in part by the MOE program Promoting Academic Excellence of Universities under Grant 89E-FA06-2-4-7 and in part by the National Science Council, Taiwan, R.O.C., under Grant NCS91-2213-E-002-057.

W. Liao is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., and the Graduate Institute of Communication Engineering, National Taiwan University, Taiwan, R.O.C. (e-mail: wjliao@cc.ee.ntu.edu.tw).

M.-Y. Jiang is with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TVT.2003.816633

group member saves a copy of each unstable message but removes the messages that have been stabilized. This protocol essentially needs each member to send feedback (to acknowledge reception of packets or to request for retransmissions) directly back to the source. Thus, it suffers from the problem of feedback implosion.

In this paper, we propose a tree-based protocol, called Family ACK Tree (FAT), to provide reliable multicast for mobile ad hoc networks. Unlike [10], which requires each node to acknowledge a reception directly back to the source, thus suffering from ACK implosion, FAT adopts a tree-based recovery mechanism to localize retransmissions. To cope with node movements, FAT constructs an ACK tree on which each node maintains reachability information to the families of three generations of nodes, i.e., the families of the node's parent, grandparent, and the node itself, on the ACK tree. When a tree is fragmented due to a departed node, the fragments will be glued back to the tree using the underlying multicast routing protocol. FAT then adopts an adaptive mechanism to recover missed packets that have been multicast to the group during fragmentation. Note that FAT is not concerned with the data link layer[1] and multicast routing issues, but with the transport layer issue (i.e., reliable multicasting). It runs on top of a constructed multicast tree using arbitrary multicast routing protocols, such as [11]–[14]. FAT is designed to work with any existing ad hoc multicast routing protocols.

The rest of this paper is organized as follows. Section II describes the proposed FAT protocol. Section III presents the simulation study conducted to evaluate the performance of FAT. Finally, concluding remarks are included in Section IV.

II. The Family ACK Tree (FAT) Protocol

In this section, the proposed FAT protocol is described in details. FAT is a tree-based reliable multicast protocol for mobile ad hoc networks. It runs on any multicast trees established by the underlying multicast routing protocol such as [11]–[14]. The typical approach of tree-based reliable multicast protocols works as follows. The system maintains an acknowledgment (ACK) tree organized in a way of "region hierarchy" to manage ACKs and to localize retransmissions. In each region, one node is selected as the reliability agent for the other nodes. Each node in a region cumulatively acknowledges the reception of packets or requests retransmissions to the agent node. For FAT, lost packets are recovered by a tree-based local recovery scheme. The semantics of a tree-based protocol for reliable multicast remains unchanged as it is in the wireline environment except that nodes (i.e., members in a group and the intermediate multicast routers) are mobile. FAT then introduces a repair mechanism to recover packet losses due to node movements. In summary, FAT works as a typical tree-based mechanism when nodes remain connected. When nodes become disconnected due to movements, FAT reconnects the fragments back to the tree based on the underlying multicast routing protocol and recovers the loss due to node movements by the proposed repair mechanism.

A. The Basic Mechanism

FAT is the proposed ACK tree to provide reliable multicast for mobile ad hoc networks. A family ACK tree is overlaid on the underlying multicast tree. On the FAT tree, the parent node (i.e., immediate upstream node) serves as the reliability agent for its child nodes. Each child node cumulatively acknowledges the reception of packets or requests retransmissions to their parent node. Once the parent node has moved away, the new parent found by the underlying multicast routing protocol will become the new reliability agent. To recover the packet losses not reparable by the new agent, the node may request retransmissions from its former parent on the old ACK tree. In the worst case, this retransmission node may be the root (i.e., the source). As such, we can avoid sending retransmission requests all the way back to the source.

1) Definition: In FAT, each node maintains an ACK table that contains reachability information to the families of three generations of nodes on the ACK tree. These three families of a node are identified with a grandparent ID (GID), a parent ID (PID), and a children ID (CID), respectively. A GID, a PID, and a CID indicate a multicast group of nodes rooted at the grandparent, the parent, and the node itself, respectively.

1) GID (i.e., the grandparent's family) identifies a group of nodes containing the node's parent, the node's grandparent (the parent's parent), and the parent's siblings.[2] In other words, a GID describes the family of the node's grandparent.
2) PID (i.e., the parent's family) indicates a collection of nodes including the node itself, the node's parent, and the node's siblings. In other words, a PID describes the family of the node's parent.
3) CID (the node's family) indicates a group of nodes including the node itself and the node's children. In other words, a CID describes the family of the node itself.

Fig. 1 shows the sets of nodes corresponding to the GID, the PID, and the CID of node D. Node D's GID includes nodes A, B, and C, corresponding to the grandparent, the parent, and the uncle (i.e., parent's sibling), respectively. Node D's PID includes nodes B and D, which are its parent and itself, respectively. Node D's CID includes itself, D, and its children, nodes E and F.

2) Message Types: There are two types of control messages to be exchanged among the three generations of each node on a family ACK tree.

1) Subgroup ID advertisement (Ad) is used to configure the ACK table of each node on a family ACK tree. In ad hoc networks, each mobile node periodically broadcasts beacons to its neighbors. The beacons from those nodes involved in the multicast tree are associated with Ad messages. An Ad message sent by a node contains a PID and a CID to configure the ACK table of the node's child nodes. The PID and CID of the Ad received from the parent of a node will become the GID and PID, respectively, for the node. The node generates its CID, the value of which must be different from its GID or PID, and different from

[1]As long as the data link protocol can provide the identities of a node's neighbors and informs the upper layers of any creation or deletion of logical links.

[2]Two nodes are siblings when these two nodes have a common parent node.

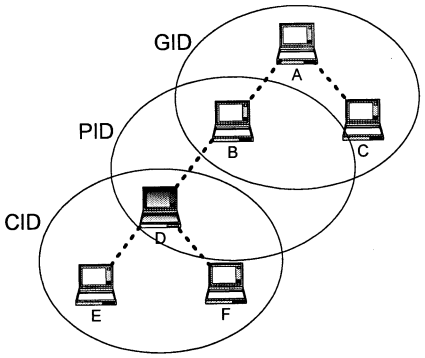


Fig. 1. An example of a family ACK tree.

its siblings' CIDs.[3] Once the CID is determined, the node updates the received Ad with its PID and CID, and then rebroadcasts the updated Ad to its neighbors. Note that those nodes not involved on the multicast tree just ignore the received Ads.

2) Retransmission request (RTQ) is used by a node to report a loss gap to its reliability agent and to request a retransmission. In the normal state (i.e., no node is disconnected), the child node unicasts an RTQ to its parent node. When a node is disconnected, a tree may be fragmented into a forest. Each subtree rooted at a child of the departed node tries to be glued back using the underlying multicast routing protocol. Once reconnected to the tree, and upon detecting the lost packets not reparable by the new parent, a node multicasts an RTQ message with a time-to-live (TTL) value to its "old" GID (i.e., using the old GID as the destination address of the request.). The TTL value limits the range of forwarding messages. An RTQ is resent if the node has not received the requested packets within a predefined timeout. The TTL value is incremented by one at each retry to broaden the range of message forwarding and to increase the probability of reaching a node that can grant the RTQ and retransmit the packets.

3) Directional Recovery: For FAT, the repair process is directional backtracking toward the root (i.e., the source). In other words, if a node's parent has moved away, the grandparent will become the foster parent for the node [4] i.e., it is able to recover the loss for the node by a rollback process. Each child with its descendants reconnects back to the multicast tree. For those packets not reparable by the new reliability agent found by the multicast routing protocol, the node sends an RTQ to the foster parent on the old ACK tree for loss recovery. To cope with further node movement, the RTQ is multicast to its old GID (i.e., the GID in the ACK table used before the departure of the parent), instead of directly to the foster parent, because the foster parent may also move away. In case the old grandparent (i.e., foster parent) has also left, the parent of the foster parent will become the new foster parent. The RTQ will be intercepted and forwarded one more level up the tree, with the assistance of an uncle node. This process repeats until the final foster parent is found, from where the loss can be recovered. In the worst case, the request will go to the root. Fig. 2 illustrates how FAT's recovery mechanism works. Fig. 2(a) shows the original ACK tree. In Fig. 2(b), E_1 is gone, and F_1's request is granted by the old grandparent, i.e., D_1. In Fig. 2(c), both the parent and the grandparent are gone (i.e., E_1 and D_1). Thus, the request goes to C_1, i.e., the parent of D_1. In Fig. 2(d), E_1, D_1, and C_1 have all moved away. In this case, the request will go up one more level, i.e., B_1.

FAT directional repairing performs well in dense (i.e., the number of nodes is large within a given area), slow-moving ad hoc networks. In a fast-moving mobile ad hoc environment, we may consider using an omnidirectional repairing mechanism similar to scalable reliable multicast (SRM) [3], instead of directional repairing, to reduce the FAT maintenance overhead. With the omnidirectional loss recovery algorithm, RTQs are multicast to the entire multicast group instead of the multicast subgroup identified by the GID. The nodes who have a copy of the requested packet will answer the request. The repair will also be multicast to the entire multicast group. For nodes sending requests or repairs, this recovery scheme provides a suppression mechanism to prevent duplicate requests from triggering duplicate repairs (i.e., at most one node will send a request or answer the request). As a result, when the number of nodes affected by the packet loss is small, this mechanism allows local loss recovery and reduces unnecessary use of bandwidth by limiting the scope of multicasting requests and repairs.

The following compares these two recovery mechanisms.

a) For *omnidirectional recovery,* RTQs are multicast to the entire multicast group. The goal is to find any node in the group that is able to grant the request. If no repair is received within the timeout period, the requesting node will resend a request with a larger TTL. Increasing TTL has the effect of widening the scope of multicasting. As the number of nodes involved in request multicasting increases, the possibility of reaching a node that is able to repair also increases, at the expense of unnecessary bandwidth consumption.

b) For *directional recovery,* RTQs are multicast with a limited TTL to the GID of the requesting node in order to find the grandparent node. If the grandparent parent node is also gone, an uncle intercepts the request and multicasts to the uncle's GID. In this way, the original RTQ is backtracked up one more level of the ACK tree until a node able to repair is found. The RTQ forwarding is directional, moving up level by level along the direction of the ACK tree. In the worst case, the request will go to the data source. This guarantees that a node able to repair can be found. For the directional recovery, a node able to repair is found due to RTQ backtracking directionally, rather than a wide scope request multicasting as in the omnidirectional scheme. The limited TTL multicast in this approach broadens the searching range to reach any node in the GID group, from where directional backtracking starts.

Fig. 3 illustrates the difference between omnidirectional and directional recovery using the same example as in Fig. 2(a). In the omnidirectional approach, all nodes involved in a TTL value must process the request, while in the directional recovery

[3]In Section II-C, we will give an example to determine the CID of each node uniquely.

[4]The details of the rollback process will be described in Section II-B.

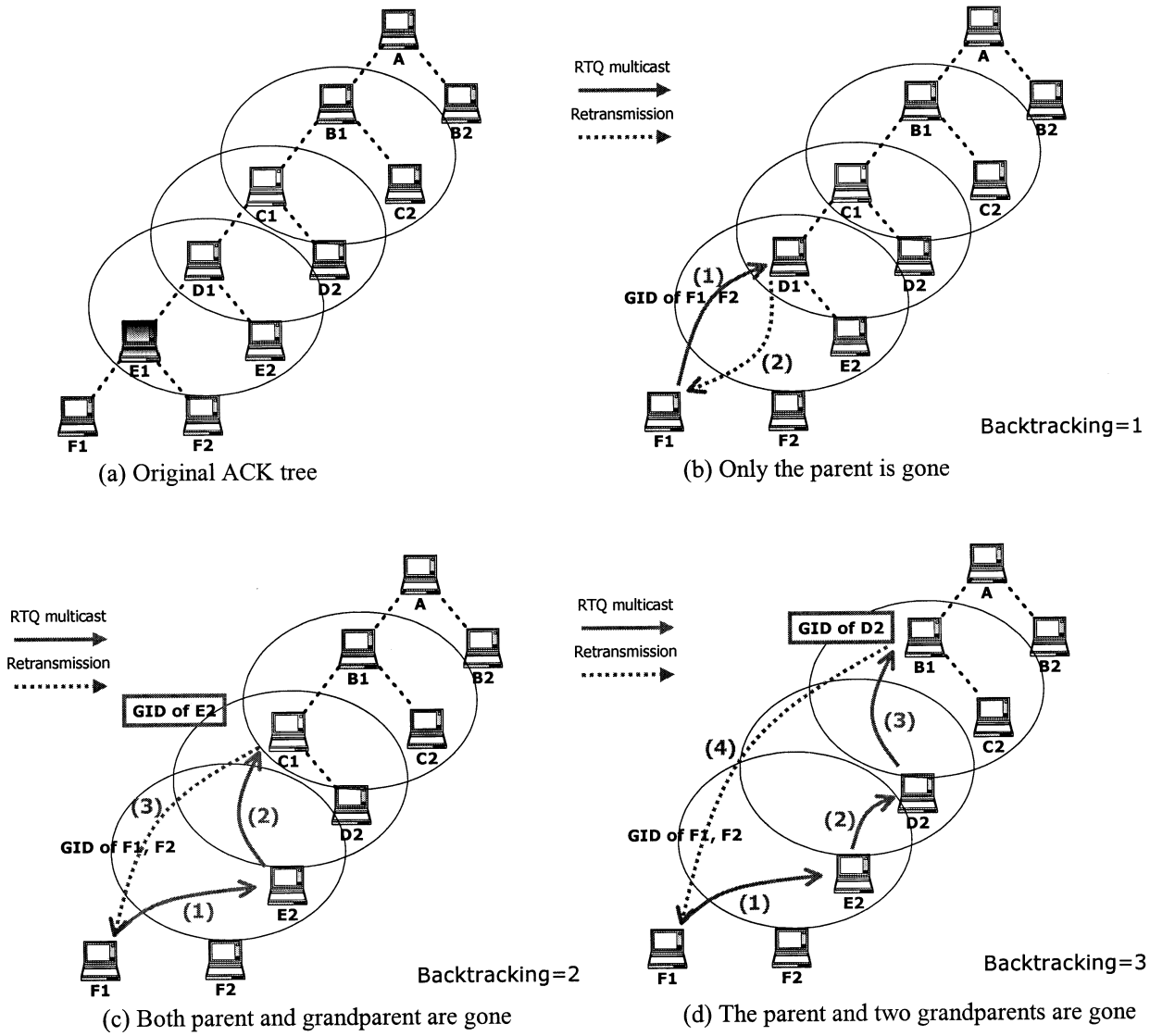


Fig. 2. An example of FAT recovery. (a) Original ACK tree; (b) only the parent is gone; (c) both parent and grandparent are gone; and (d) the parent and two grandparents are gone.

mechanism, only the nodes in the GID subgroup need to process the request. In Section III, we will provide simulation results to show the advantage of directional backtracking over omnidirectional recovery in ad hoc networks.

B. Operation Overview

1) Family ACK Tree Construction:

a) Once a multicast tree has been constructed by an ad hoc multicast routing protocol, the subgroup ID advertisement process starts. The tree root (i.e., the source) does not have a GID and a PID. Thus, the source generates its CID and duplicates the value of CID to the PID on an Ad. The source broadcasts a beacon with the Ad to its neighbors. Upon receiving an Ad from the parent of a node, the node records the PID and CID in the Ad as its GID and PID, respectively, and generates its CID to be the PID of its children. It then updates the Ad received and rebroadcasts the updated Ad message to its neighbors. All the tree nodes store their GIDs, PIDs, and CIDs in the respective acknowledgment tables, except the root that stores a CID only.

b) All nodes on the multicast tree are involved in the subgroup ID advertisement. Through this process, all nodes are able to acquire their GIDs, PIDs, and CIDs to configure their respective ACK tables.

c) The subgroup ID advertisement can only be initiated by the root, once per an integer multiple of periodic beacons. Each child node then broadcasts a beacon with an updated Ad message to its transmission range once an Ad is received from its parent node.

2) Family ACK Tree Maintenance:

a) Each node may be in one of two states: normal state and repairing state.

i) Usually, a node is in the normal state.

ii) Upon detecting the departure of the parent, the node enters the repairing state. Each node in the repairing

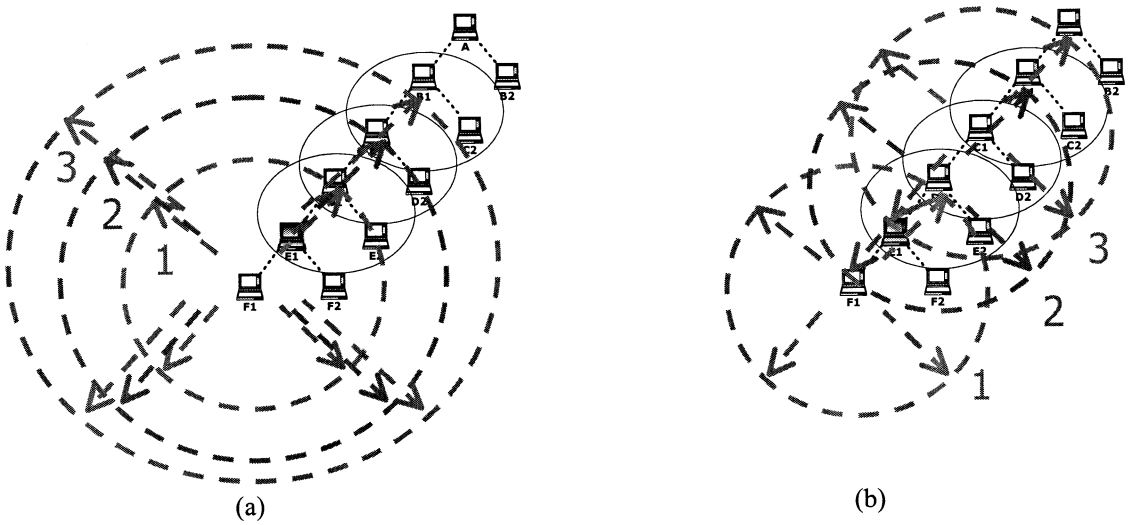


Fig. 3. Two recovery mechanisms. (a) Omnidirectional and (b) directional recovery.

state maintains two ACK tables: the old ACK table (i.e., for the "old" ACK tree) and the new ACK table (i.e., for the "new" ACK tree). Once the node has reliably received all lost packets due to node movements, it enters the normal state, when only the new ACK table is maintained.

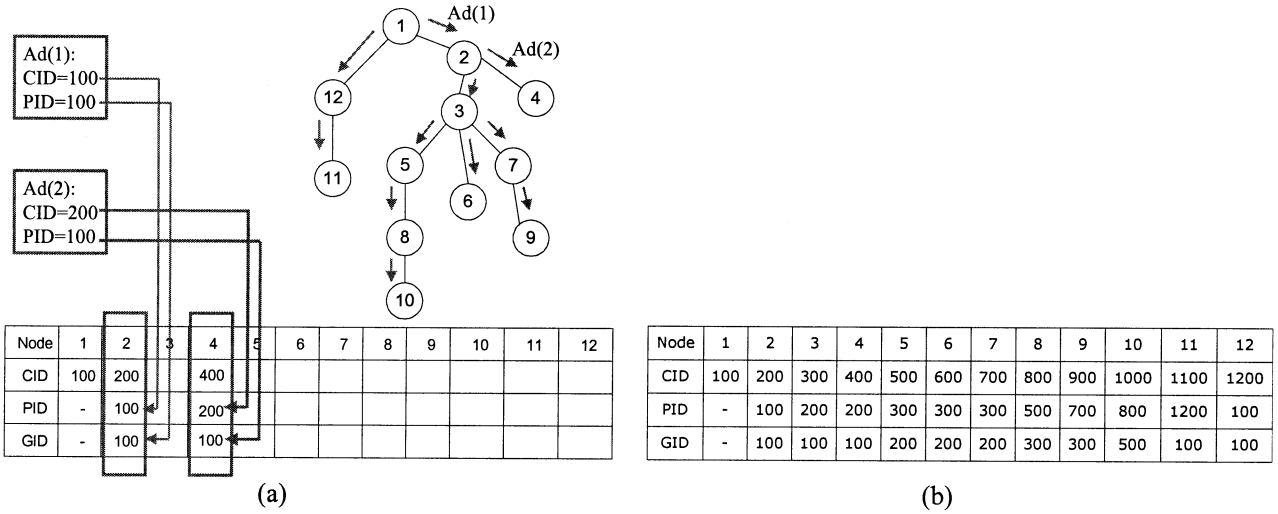
b) If a node in the normal state receives a multicast packet reliably, the packet is cached with a predefined timer T_c; otherwise, the node negatively[5] acknowledges the packet upstream to its parent and starts a predefined timer T_r.
   i) If the node has not received a unicast RTQ for the packet from any of its children on expiry of timer T_c, the node removes the packet from its buffer; otherwise, the packet is retransmitted to those children from whom a NAK (i.e., a unicast RTQ) is received.
   ii) If the node has not received a retransmission from its parent on expiry of timer T_r, a unicast RTQ is resent.

   This approach allows reliability agents in the upper hierarchical levels of a family ACK tree to have smaller buffers, compared to the approach that acknowledges a reception only after feedbacks from all the children have been received. In the latter approach, the buffer size in a reliability agent increases as we move upstream toward the root.

c) When a node departs, the nodes within the transmission range of the departing node will be informed of the node's departure by the data link layer protocol. The nodes notified may include the parent node, the child nodes, and possibly other on-tree sibling nodes. The departing node forwards the packets in its cache upstream to its parent. This rolls back the parent's buffer and allows the grandparent (i.e., the parent node of the departed node) to serve as the foster parent for its grandchildren (i.e., the children of the departed node). Note that the packets to be rolled back are multicast to the departing node's GID if the departing node is in the repairing state; otherwise, the packets are unicast to the departing node's parent. If a node in the repairing state receives the rolled-back packets whose GIDs match its PID, the node starts a suppression timer T_s, randomly selecting a value between (min, max). If the node detects any other node relaying the rolled-back packets upstream before its timer expires, it stops the timer and ignores the received packets. Otherwise, on expiry of its timer, the node modifies the packets' destinations to its old GID and multicasts the packets to the group identified by the old GID. This backtracking process may repeat several levels upstream. (It may reach the source in the worse case.)

d) Once a node has departed, all the children of the departed node become orphans. Each orphan node along with its descendants (i.e., a subtree rooted at an orphan node) attempts to glue back to the tree independently using the underlying multicast routing protocol. During the "glue back" process, an orphan node sends a request message downstream to prevent its descendant nodes from leaving the subtree (e.g., trying to reconnect to the multicast tree by themselves or leaving the group reluctantly). Once an orphan node has reconnected to the tree, it reconfigures its ACK table and identifies the missing packets unrepairable by the new parent node. The node then negatively acknowledges the loss gaps to its former GID in the old ACK table and requests retransmissions, using an RTQ with a limited TTL multicast. The repairing node (i.e., the node sending the RTQ) starts a predefined timer T_m. If the repairing node has not received the requested packets upon expiry of timer T_m, a new RTQ with a larger TTL is resent. This repairing process continues until all the loss gaps are fully recovered, when the old ACK table is deleted and the repairing node enters the normal state.

[5]To further improve reliability, FAT can also be modified to positively acknowledge correct reception, so that a packet in the buffer is removed only when a node has received feedback from all its children before the timer expires. Here, we just demonstrate how FAT works with NAKs.



Node	1	2	3	4	5	6	7	8	9	10	11	12
CID	100	200		400								
PID	-	100		200								
GID	-	100		100								

(a)

Node	1	2	3	4	5	6	7	8	9	10	11	12
CID	100	200	300	400	500	600	700	800	900	1000	1100	1200
PID	-	100	200	200	300	300	300	500	700	800	1200	100
GID	-	100	100	100	200	200	200	300	300	500	100	100

(b)

Fig. 4. Subgroup ID advertisement. (a) Subgroup ID advertisement and (b) final ACK table.

e) Each node performs the following upon receipt of a multicast RTQ.

i) When a node in the normal state receives an RTQ whose GID matches its PID, the request is discarded.

ii) When a node in the repairing state receives an RTQ whose GID matches its PID, the node starts a suppression timer Ts randomly selecting a value between (min, max). If the node detects any other node relaying an updated RTQ upstream before its timer expires, it stops the timer and ignores the received RTQ. Otherwise, on expiry of its timer, the node modifies the RTQ's destinations to its previous GID and multicasts the updated RTQ to the group identified by the previous GID. This backtracking process may repeat several levels upstream. (It may reach the source in the worse case.)

iii) Upon receiving an RTQ with the GID equal to its GID, the request is discarded.

iv) When a node, say, F receives a multicast RTQ whose GID matches its CID, node F must be the grandparent of the requesting node. If the received request is encapsulated, node F decapsulates the request to identify the original requesting node, say, node O; otherwise, the request sender must be the original requesting node. Node F then retransmits the requested packets directly to node O.

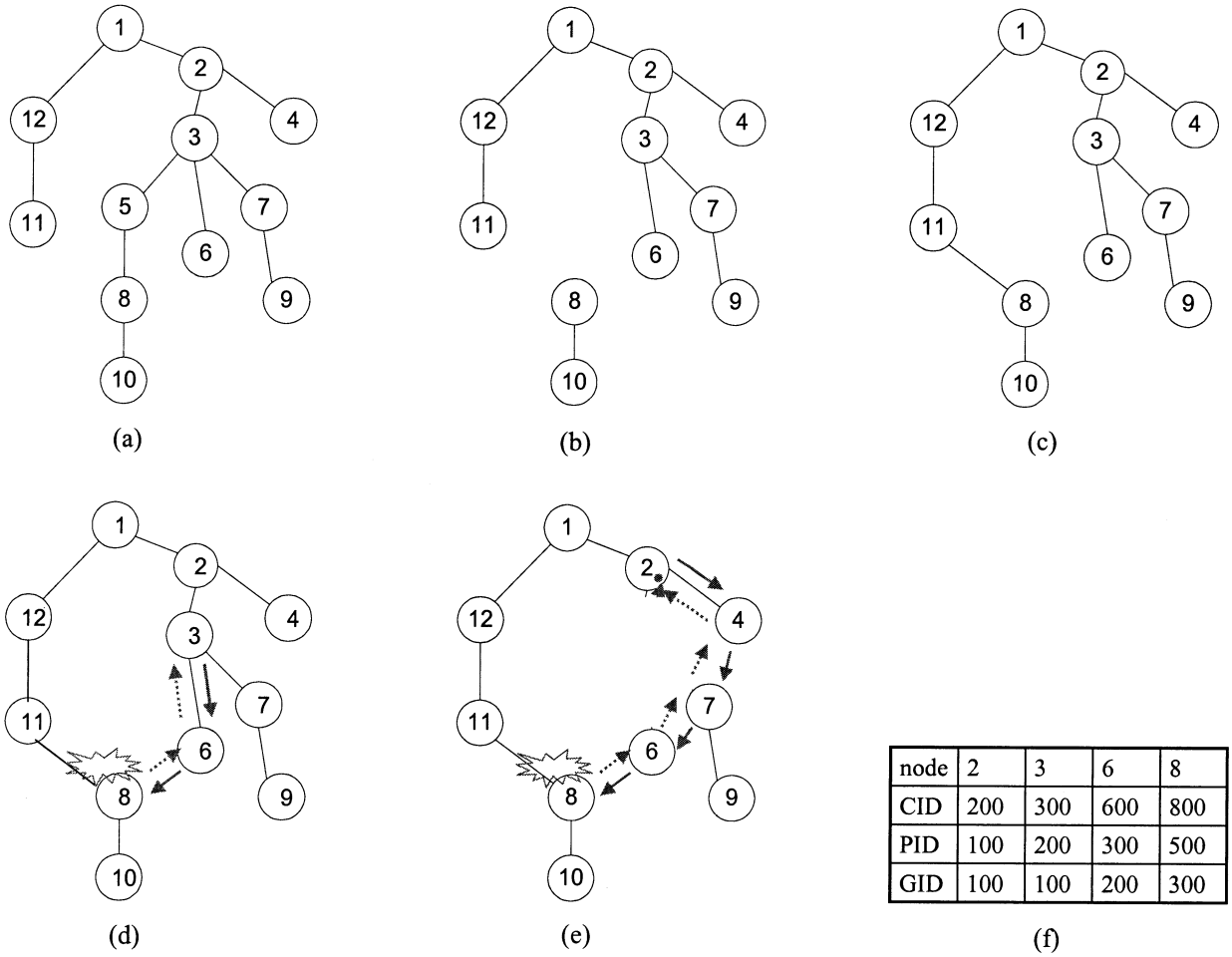
Note that if a node responsible for an RTQ (both unicast and multicast) cannot repair a loss (e.g., a failure in rollback or a failure in the ACK table maintenance), it forwards the request level by level up toward the source until a node able to answer the request is reached, from where the packet is retransmitted. If the multicast RTQ has reached the source, the request will always be granted directly from the source.

3) An Example: Figs. 4 and 5 demonstrate how FAT works. Fig. 4(a) shows the original multicast tree (and the ACK tree as well), which consists of 11 mobile nodes rooted at node 1. It also shows the subgroup ID advertisement process. Each column in Fig. 4(a) shows the ACK table stored at each node. Note that the numbers in Fig. 4(a) are chosen for ease of illustration. In reality, each GID, PID, and CID may be an administrative scoped address as defined in [14]. Multicast addresses 239.0.0.0 to 239.255.255.255 are reserved for administrative scoped IP multicast. Data of administrative scoped IP multicast are confined in a scope region whose boundary is formed by a set of routers. Addresses in the same scoped region are kept unique but may overlap in different scoped regions. In the context of ad hoc networks, each ad hoc network can be a scope region. Each mobile node in the network then uses the address allocation protocol (AAP) [15] to obtain an administrative scoped address of the scope region as its CID. As such, each subgroup multicast address is unique in the network. Fig. 4(b) shows the resulting ACK table of each node in the multicast group.

Fig. 5 shows the ACK tree maintenance of FAT. Fig. 5(a) shows the original ACK tree. Fig. 5(b) shows the tree when node 5 departs. Upon detecting the departure of node 5, node 8 with its descendant (i.e., node 10) needs to reconnect back to the tree [in Fig. 5(c)]. Node 8 identifies the loss offset in packet sequence numbers between what it has and what it is receiving from the new branch. Then, it multicasts an RTQ destined to its old GID, i.e., 300, to request a retransmission [i.e., the dashed curves in Fig. 5(d)]. On receiving the RTQ, node 3 finds that the destination GID is equal to its CID. It sends the requested packets back to node 8 [i.e., the solid arrows in Fig. 5(d)]. If node 3 departs before receiving the RTQ, as shown in Fig. 5(e), nodes 6 and 7 enter the repairing states and try to reconnect to the tree. Upon receiving an RTQ with the GID equal to its PID, node 6 updates the RTQ with a GID of 200 [i.e., the dashed arrows in Fig. 5(e)]. Upon receiving the RTQ with a GID of 200, node 2 retransmits packets as requested back to node 8 [i.e., the solid curves in Fig. 5(e)]. Fig. 5(f) shows the ACK tables of the nodes involved in this example.

C. FAT With Ad Hoc Multicast Routing Protocols

In this section, we further describe how FAT works with layer-3 multicast routing protocols in mobile ad hoc networks.



node	2	3	6	8
CID	200	300	600	800
PID	100	200	300	500
GID	100	100	200	300

Fig. 5. An example to illustrate FAT operations. (a) Original tree, (b) node 5 leaves, (c) node 8 reconnects to the tree, (d) only the parent leaves, (e) the parent and grandparent both leave, and (f) ACK tables of nodes involved.

In particular, we consider two types of protocols: tree-based and mesh-based multicast routing protocols.

1) Tree-Based Protocols: When source-based forwarding trees are used for multicast data delivery, each multicast packet is forwarded from data source S along the shortest path through the tree to the members of the multicast group, say, G. For example, the adaptive demand-driven multicast routing protocol (ADMR) [16] is a source-based multicast routing protocol. In ADMR, a source-based forwarding tree for a group is created whenever there is at least one source and one receiver active for the group. The multicast state is set up when a new multicast sender starts sending packets to a group or when a receiver joins a group. ADMR exploits the mechanism of multicast receiver discovery and multicast source discovery to create a membership table at each node. The membership table contains one entry for each group, no matter whether the node is a receiver or a sender. Once the multicast state has been set up and the source has started sending packets, the subgroup ID advertisement of the FAT mechanism is activated and a family ACK tree rooted at data source S is constructed. ADMR uses the techniques of link break detection and link break repair to maintain multicast states. Once a node detecting a disconnection reconnects to the multicast tree, the node uses the FAT loss recovery mechanism to request retransmissions.

For shared tree protocols, such as the multicast operation of the ad hoc on-demand distance vector (MAODV) routing protocol [11], the members of a group form a shared tree. Each data source finds a route to the group by the route discovery mechanism and sends multicast packets through the route. A family ACK tree rooted at a data source is constructed by FAT's subgroup ID advertisement. Upon detecting a broken link, MAODV's repair mechanism is used to reconstruct the tree. Once a node detecting disconnection reconnects to the multicast tree, the node uses the FAT loss recovery mechanism to request retransmissions.

2) Mesh-Based Protocols: For mesh protocols, such as on-demand multicast routing protocol (ODMRP) [12], a set of nodes forming a forwarding group is responsible for multicast data delivery. Multicast packets are forwarded along the shortest paths from the source to the receivers. Flooding redundant packets to the forwarding group can help combat packet losses during node movements, thanks to a redundant route between any node pairs. As a result, when a link breaks, nodes on the broken link can still receive multicast packets from other routes. Considering a multicast packet being sent from the source, although a mesh (i.e., some redundant paths exist) is used to forward the packet, the packet is delivered through the shortest path from the source to the receivers. Thus,

the result is still in the form of a delivery tree. Under moderate node mobility, packets are delivered through the shortest path from the source to the destinations. It is beneficial to construct a family ACK tree, rooted at the data source and based on the hierarchy of the multicast delivery path, to provide reliable multicast service. When node mobility is very high so that the delivery path is changed on a packet-by-packet basis, it is not necessary to construct an ACK tree since the tree hierarchy does not exist.

III. Performance Evaluation

This section describes the simulation conducted to evaluate the performance of FAT. We investigate the behavior of FAT and compare FAT with a protocol in which feedbacks are sent directly back to the source (denoted as source_ACK in the rest of this paper), as in [10], in terms of reliability, scalability, and delivery efficiency. We also compare the efficiency of omnidirectional and FAT directional recovery in the simulation.

A. Simulation Environment

In the simulation, we focus on the impact of nodal mobility on the performance of the protocols, and assume that packet transmissions are error-free and all losses are caused by node movements. We consider a 100-by-100 mesh in which nodes are roaming in the mesh during the simulation. The distance between two adjacent intersection points (i.e., coordinates) in the mesh is 1 m. We randomly determine a set of N mobile nodes, initially located at N randomly selected coordinates in the mesh, where N varies from 200 to 600. We assume that there is only one sender, randomly selected one from the N mobiles, and the rest of the N-1 mobiles are the group members. Each node has a transmission range of 20 m, within which mobile nodes can directly communicate with one another. A multicast delivery tree is rooted at the source and spans over all other nodes. The sender generates data packets at a constant rate of one packet per second.

Each node moves according to the mobility model defined in [17]. Initially, each node randomly selects a location in a 100-by-100 mesh. On expiry of a pause time, a node moves to another randomly selected coordinate in the mesh at a speed uniformly distributed between 10 and 20 m/s. Once having reached the destination, the node pauses again for another pause time. Then it selects another destination and speed and moves again. We use six different pause times in the simulation: 10, 13 16, 20, 40, and 80 s. The shorter the pause time, the higher the mobility.

B. Simulation Results

1) Source_ACK Versus FAT: This experiment is conducted to compare FAT with the source_ACK approach. To simplify the comparison, a parameter called the reliability index is defined as the ratio of the number of the granted requests to the total number of the received requests. A request is granted if the requested packet losses can be retransmitted successfully. The larger the reliability index, the larger the number of lost packets that can be recovered, and the better the performance. Note that the reliability index of any reliable multicast protocol must be equal to one. To concentrate on the backtracking effect of FAT,

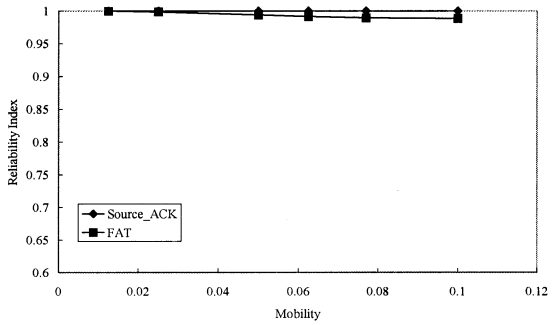


Fig. 6. Reliability index versus mobility.

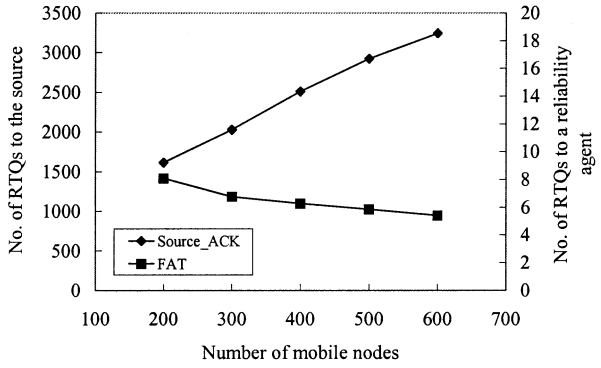


Fig. 7. Retransmission numbers.

we only count those RTQs granted by backtracking as "granted requests" in the calculation of the reliability index of FAT in the simulation. In fact, when a node retries a couple of times for a retransmission and fails to get a response, it will send an RTQ directly back to the source, from where the data will be retransmitted as in the source_ACK protocol. Thus, the reliability index of FAT should also be equal to one. We define "mobility" as the inverse of the mean time that a node stays at a location. The shorter a node stays at a location, the more frequently a node moves, and the higher the mobility.

For the source_ACK approach, each member negatively acknowledges packet losses directly back to the source. Any lost packets can be retransmitted by the source. Thus, the source_ACK approach has a reliability index of one, irrespective of host mobility, as shown in Fig. 6. This approach, however, does not scale well due to the problem of ACK implosion. Fig. 7 shows that the total number of feedbacks increases dramatically for the source_ACK approach as the number of mobile nodes becomes large (see the left scale in the y-axis). This is because for Source_ACK, all these requests must go directly to the source, causing the source to become a bottleneck and incurring large overhead to the system. On the other hand, FAT is a tree-based local recovery mechanism. Each parent node serves as the reliability agent for its child nodes, thus distributing the overhead of retransmissions to each parent node. This renders lower overhead as shown in Fig. 7 (see the right scale in the y-axis). In addition, FAT employs the family ACK tree mechanism to cope with node movement. The backtracking mechanism provides the reliable multicast service almost as good as in source_ACK (as shown in Fig. 6).

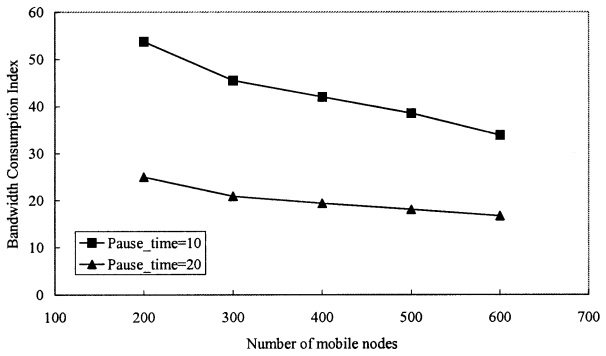


Fig. 8. Bandwidth consumption.

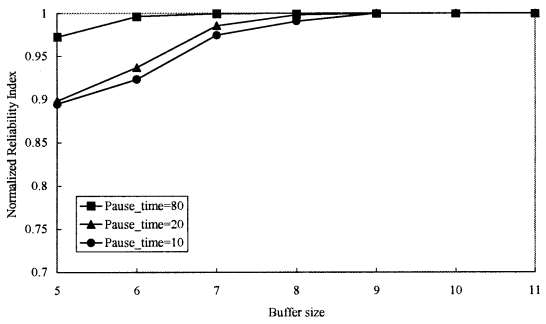


Fig. 9. Buffer size.

2) FAT Performance:

a) Bandwidth consumption (scalability): This simulation is performed to measure the bandwidth consumption of FAT. A parameter called bandwidth consumption index is defined as the total number of retransmission requests multiplied by the average path length (the number of hops) traversed by each retransmitted packet, normalized by the total number of mobile nodes. We vary the number of nodes from 200 to 600, and let the pause time be fixed at 10 and 20 s, respectively. Fig. 8 shows that the bandwidth consumption index decreases as the number of nodes increases, thanks to retransmissions being localized by each parent node. Thus, FAT is scalable, providing reliable multicast services even in a large ad hoc network.

b) Normalized reliability index versus buffer size: This experiment is conducted to evaluate the relationship between the normalized reliability index and buffer size. The normalized reliability index is defined as the ratio of the reliability index with a certain buffer size to the reliability index with an infinite buffer. For example, suppose that the pause time is fixed at 10 s. If a reliability index with an infinite buffer is 0.987 and a reliability index with a buffer size of five packets is 0.884, the normalized reliability index is $0.884/0.987 = 0.896$.

Fig. 9 shows the three curves of the normalized reliability index, varying the buffer size from five to ten packets. The three curves in the figure correspond to pause times of 10, 20, and 80 s, respectively. The longer a node stays at a location, the better reliability FAT can achieve. It can be observed that even with a small buffer size, say, five packets, for a pause time of 20 s, the normalized reliability index reaches above 0.9. As the buffer size increases, the normalized reliability index is further improved. For all three pause times, FAT with a buffer size of eight packets is sufficient to reach a normalized reliability index of 0.999, achieving the performance of an infinite buffer. Interestingly, using a buffer size larger than ten packets cannot provide a higher normalized reliability index because the curves have saturated. Thus, in this simulation, a buffer size of eight to ten packets is sufficient for FAT to reach a performance corresponding to an infinite buffer.

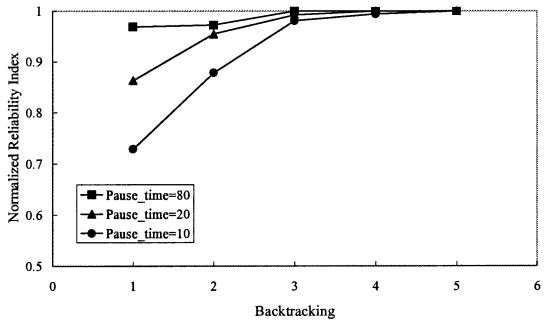


Fig. 10. Backtracking.

c) Normalized reliability index versus backtracking: This experiment is conducted to observe the relationship between the reliability index and the average path length of a retransmission. For FAT, when a node is about to leave, it forwards its buffer to its parent so that the grandparent can serve as a foster parent for the grandchildren. Each child with its descendants is glued back to the multicast tree independently and requests retransmissions for those packets unable to be repaired by its new parent. To cope with node movements while ensuring reliable multicasting, each request is destined to the old GID (i.e., the GID in its ACK table before the departure of the parent) of the requesting node, instead of directly to the old grandparent, because the original foster parent may also have moved away. In case a foster parent is gone, FAT makes a member in the subgroup of the GID encapsulate the request and forward the encapsulated request to its GID (i.e., uncle node's GID). This operation is repeated until reaching the final foster parent, from where a retransmission is sent directly back to the requesting node. The more frequently a foster parent moves, the farther a retransmission travels, consuming more network bandwidth. We define "backtracking" as the number of changes in foster parents. The larger the backtracking, the more frequently foster parents are changed, causing a longer retransmission path.

Fig. 10 shows two normalized reliability index curves for the pause times of 10, 20, and 80 s, respectively, varying backtracking from one to five. The normalized reliability index here is defined as the ratio of the reliability index with a certain backtracking to the reliability index with infinite backtracking (i.e., going back to the source). Again, the longer a node stays at a location, the higher the normalized reliability index for FAT. It can be observed that when a retransmission goes only to the first level of backtracking (i.e., always goes to the original grandparent), FAT reaches a reliability index of only 0.86 for the curve with a pause time of 20 s (i.e., for high mobility nodes) and that of 0.95 for the curve with a pause time of 80 s (i.e., for low mobility nodes). When a retransmission travels to the third level of backtracking, the reliability index approaches one for low mobility nodes and to 0.99 for high mobility nodes, respectively.

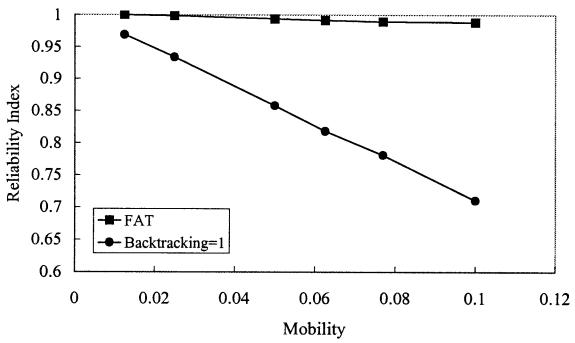


Fig. 11. Reliability index versus mobility.

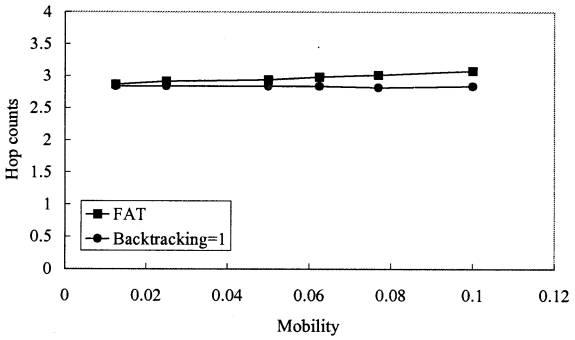


Fig. 12. Retransmission path versus mobility.

Fig. 10 shows that FAT can fully recover lost packets (i.e., reliability index $= 1$) as backtracking reaches four. Thus, it can achieve performance as well as that with requests directly going back to the source, while avoiding the ACK implosion that a Source_ACK approach suffers from (in Fig. 6).

Figs. 11 and 12 compare FAT with the approach in which the requests of repairing nodes always go to the grandparent (i.e., a backtracking of one). This approach allows the shortest retransmission path if the grandparent does not move, but suffers from low reliability index as host mobility increases (shown in Fig. 11). Fig. 12 shows the average number of hops traversed by each retransmitted packet with a backtracking of one. In this case, irrespective of host mobility, the path length curve stays close to three, roughly equal to the distance between a grandparent and a grandchild. Although FAT requires a slightly longer retransmission path as mobility increases, its reliability index always stays close to one, showing that it provides reliable multicast service for ad hoc networks in an efficient way.

3) Directional Versus Omnidirectional Repairing: This experiment is conducted to compare the efficiency of directional FAT with an omnidirectional recovery in which an RTQ is multicast to the whole group and any node on the ACK tree with the packet is able to retransmit it. For the directional FAT, an RTQ is multicast with a limited TTL value, where the TTL is increased if no retransmission is received. It is the same as in the omnidirectional approach except that the destination of the RTQ in the latter approach is the whole group. The difference is that for FAT directional recovery, the increment of TTL at each try increases the probability for the RTQ to reach the members of GID, while for the omnidirectional approach, the purpose is

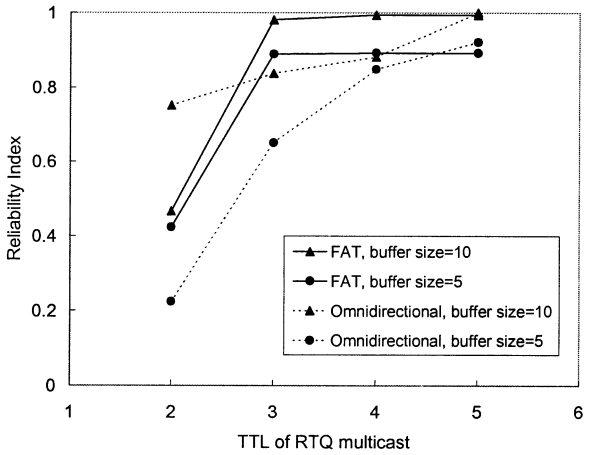


Fig. 13. Reliability index versus TTL.

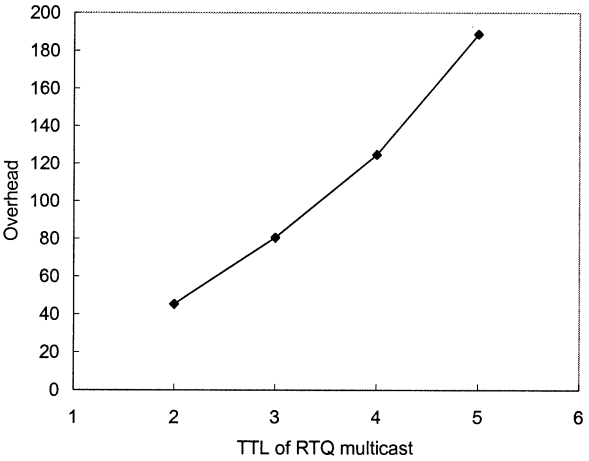


Fig. 14. RTQ retransmission overhead.

to increase the probability to reach any node that is able to retransmit. In the worst case, the omnidirectional approach floods the whole network.

Fig. 13 shows the reliability index curves of the directional and omnidirectional approaches, with different buffer sizes for different values of TTL (in RTQ multicast). For FAT with buffer sizes of five and ten, the reliability index saturates at TTL $= 3$, which is sufficient for the RTQ to reach the grandparent or uncles in the GID subgroup. The reliability index of the omnidirectional approach with buffer sizes of five and ten increases as TTL increases and reaches the same performance level as that in FAT at TTL $= 5$. To measure the overhead of RTQ multicasting, we count the number of times that an RTQ message is forwarded. Fig. 14 shows that an RTQ is forwarded for more than 80 times at TTL $= 3$ and almost 190 times at TTL $= 5$ in a network with 200 mobile nodes. This means that the omnidirectional approach incurs much more overhead than FAT to provide the same level of reliability. In addition to the RTQ forwarding overhead shown in the figure, the RTQ processing overhead also defeats scalability. For FAT, only GID members are required to process the RTQ in terms of repairing packet loss or remulticasting the request. However, for the omnidirectional approach, each member receiving the RTQ is required to check if it can repair the loss itself. As the TTL value increases, the coverage area

of RTQ multicast increases and the number of affected group members increases. In other words, overall, the omnidirectional approach forces more nodes to be involved in the retransmission operation and incurs much higher processing overhead.

IV. Concluding Remarks

In this paper, we have proposed an efficient tree-based protocol, the family ACK tree protocol, to support reliable multicast service for mobile ad hoc networks. FAT performs well in dense, slow-moving ad hoc networks. The proposed mechanism is composed of two parts: ACK tree construction and ACK tree maintenance. A family ACK is built as follows. To cope with node movements, each node maintains an ACK table to store the reachability information to three generations of nodes on the tree, i.e., a GID, a PID, and a CID. Normally, each node serves as the reliability agent for its child nodes. When a tree is fragmented due to a departed node, the fragments will be glued back to the tree using the underlying multicast routing protocol, and a new ACK tree will be formed accordingly. FAT then adopts a directional recovery mechanism to speed up retransmissions for packets not reparable by the new agent. We compare the difference between the omnidirectional repairing and the FAT directional recovery, and show the advantages of the directional over the omnidirectional mechanisms by simulation. We have also conducted simulations to evaluate the performance of FAT and to compare FAT with the existing solution. The results show that FAT achieves the best performance in providing reliable service for ad hoc networks, in terms of reliability, scalability, and delivery efficiency.

In this paper, we mainly focused on the protocol description, and demonstrated the performance of the FAT protocol by simulation. In the future, we will further analyze reliable multicast protocols for ad hoc networks.

Acknowledgment

The authors would like to thank the three anonymous reviewers. Their comments have significantly improved the quality of this paper.

References

[1] IETF Mobile Ad Hoc Networks (MANET) Working Group Charter.. [Online]http://www.ietf.org/html.charters/manet-charter.html

[2] W. T. Strayer, B. J. Dempsey, and A. C. Wever, *XTP: The Xpress Transfer Protocol*. Reading, MA: Addison-Wesley, 1992.

[3] S. Floyd *et al.*, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Trans. Networking*, vol. 5, pp. 784–803, Dec. 1997.

[4] B. Whetten, T. Montgomery, and S. Kaplan, "A high performance totally ordered multicast protocol," in *Theory and Practice in Distributed Systems*. Berlin, Germany: Springer-Verlag, LNCS 938, 1994.

[5] J.-M. Chang and N. F. Maxemchuk, "Reliable broadcast protocols," *ACM Trans. Comput. Syst.*, vol. 1, no. 3, pp. 151–173, Aug. 1984.

[6] S. Paul *et al.*, "Reliable multicast transport protocol (RMTP)," *IEEE J. Select. Areas Commun.*, vol. 15, pp. 407–421, Apr. 1997.

[7] B. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves, "The case for concurrent reliable multicasting using shared ACK trees," in *Proc. ACM Multimedia '96*.

[8] L. Rizzo and L. Vicisano, "A reliable multicast data distribution protocol based on software FEC techniques (RMDP)," in *Proc. IEEE HPCS '97*, pp. 115–124.

[9] B. N. Levine and J. J. Garcia-Luna-Aceves, "A comparison of known classes of reliable multicast protocols," in *Proc. IEEE ICNP '96*, 1999.

[10] S. K. S. Gupta and P. K. Srimani, "An adaptive protocol for reliable multicast in mobile multi-hop radio networks," in *Proc. IEEE Workshop Mobile Computing Systems and Applications*.

[11] E. M. Royer and C. E. Perkins, "Multicast operation of the ad-hoc on-demand distance vector routing protocol," in *Proc. ACM/IEEE MOBICOM '99*, 1999.

[12] S.-J. Lee, M. Gerla, and C.-C. Chiang, "On-demand multicast routing protocol," in *Proc. IEEE WCNC '99*.

[13] J. J. Garcia-Luna-Aceves and E. L. Madruga, "A multicast routing protocol for ad-hoc networks," in *Proc. IEEE INFOCOM '99*, 1999.

[14] D. Meyer, "Administratively scoped IP multicast,", IETF RFC 2365.

[15] M. Handley and S. R. Hanna, "Multicast address allocation protocol (AAP),", IETF Internet Draft, draft-ietf-malloc-aap-04.txt.

[16] J. Jetcheva and D. Johnson, "The adaptive demand-driven multicast routing protocol for mobile ad hoc networks (ADMR),", IETF Internet Draft, draft-jetcheva-manet-admr-00.txt.

[17] J. Broch *et al.*, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *Proc. ACM MOBICOM '98*, 1998, pp. 85–97.

Wanjiun Liao received the B.S. and M.S. degrees from National Chiao Tung University, Taiwan, in 1990 and 1992, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1997.

She joined the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, as an Assistant Professor in 1997. Since August 2000, she has been an Associate Professor. Her research interests include wireless networks, optical networks, and broadband Internet. She is also actively involved in the international research community and serves on the program committees of many international conferences.

Dr. Liao is currently an Associate Editor for IEEE Transactions on Wireless Communications. She has received many research awards and the Outstanding Research Paper Award in Electrical Engineering from the University of Southern California in 1997. Two papers she coauthored with her students received the Best Student Paper Award from the First IEEE International Conferences on Multimedia and Expo (ICME) in 2000 and the Best Paper Award from the First International Conference on Communication, Circuits and Systems (ICCCAS) in 2002. She was elected as one of Ten Outstanding Young Women in Taiwan in 2000 and is listed in *Marquis Who's Who in 2001–2003* and *Contemporary Who's Who in 2003*.

Ming-Yu Jiang received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taiwan, R.O.C., in 1999 and 2001, respectively.

Since October 2001 he has been with the Networking and Communications business group, BenQ Corporation, Taiwan, as an R&D Engineer working on the development of wireless communications products. His research interests are in ad hoc networks and wireless communications.