

An Integrated Approach to Efficiently Providing Fault Tolerance for Mobile Multicast*

JENN-WEI LIN

Department of Computer Science and Information Engineering

Fu Jen Catholic University

Hsinchuang, 242 Taiwan

E-mail: jwin@csie.fju.edu.tw

The paper presents an efficient approach to providing fault-tolerant capability for mobile multicast. In multicast communication, packets can be concurrently sent from a source node to all members by a multicast tree. If failures occur in nodes (links) of the multicast tree, the faulty tree will be partitioned into several disconnected subtrees. For the subtrees without the source node, it cannot deliver multicast packets to the members under its delivery range again. The main goal of this paper is to make members immune from failure affection. The proposed fault-tolerant approach contains two schemes. The first scheme uses the redundant resources of a mobile network to reconnect all the disconnected subtrees. Compared to previous approaches, the first scheme does not generate loops. In addition, it can control the maximum delivery delay of the new reconnected multicast tree. The second scheme is initiated when the first scheme cannot reconnect all the subtrees. It extracts the failure-free part of the faulty multicast tree to form a safe multicast subtree. Then, multicast packets are only delivered along the safe multicast subtree to all the members. Unlike the first scheme, the second scheme is not based on the network topology support to achieve fault tolerance. Finally, simulations are performed to compare the proposed approach and previous approaches in terms of the fault-tolerant capability and various performance overheads.

Keywords: fault-tolerant capability, mobile multicast, failures, mobile network, simulations

1. INTRODUCTION

With the rapid progress of wireless technology, users have been able to access Internet applications via wireless mobile systems. Nowadays, many popular Internet applications need the support of multicast communication, such as video conferencing, distance learning, resource discovery, etc. This results in a great demand to provide multicast in wireless mobile systems. The solutions for mobile multicast have been proposed in the IETF Mobile IP [1].

In multicast communication, if failures occur in a multicast tree, the multicast tree will be partitioned into several disconnected subtrees. The subtrees without the source node (*failure-affected subtrees*) cannot deliver multicast packets to their covering members. For example, in Fig. 1, the multicast tree is partitioned into two *failure-affected*

Received May 26, 2004; revised August 31, 2004; accepted September 23, 2004.

Communicated by Yu-Chee Tseng.

* This paper "An Efficient Fault-Tolerant Multicast Approach for Wireless Mobile Systems," has been presented at *The 9th IEEE Symposium on Computers and Communications*, pp. 556-561, June 2004, Alexandria, Egypt.

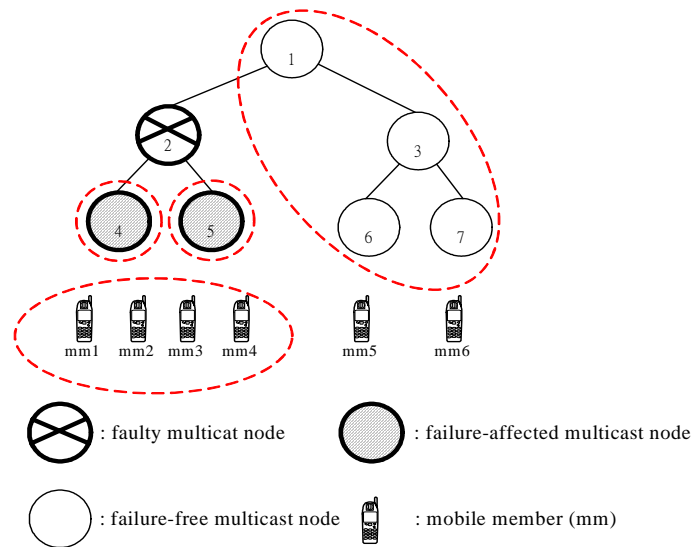


Fig. 1. The affection of a faulty multicast tree.

subtrees: {4} and {5} and one *failure-free subtree*: {1, 3, 6, 7}. Multicast packets cannot be delivered to members (*mm1*, *mm2*, *mm3*, and *mm4*) since they are under the delivery ranges of the two failure-affected subtrees.

Many papers have addressed the reliable issue of multicast communication [2-6], which focus on the problem how to efficiently retransmit lost multicast packets. As for the fault-tolerant problem, previous approaches [7-10] mainly research on the fixed network environment, and they can be classified into two basic methods: *on-demand fault tolerance* (ODFT) and *pre-planned failure restoration* (PPFR) [10]. The ODFT method utilizes the intrinsically redundant paths of a network to reconnect all the disconnected subtrees. The subtree reconnection may generate loops. To eliminate loops, some members need to rejoin the multicast group for establishing their respective new multicast paths. The loop elimination is not a trivial task. Obviously, the ODFT method is unacceptable for real-time and time-critical multicast applications. For the PPFR method, it predefines some backup paths in a multicast network. A backup path is activated when a node (link) failure is detected in the multicast tree. In a wireless mobile system, the structure of a multicast tree may be changed due to handing off in addition to joining and leaving. Compared to a fixed network, more backup paths are required to be predefined in a mobile network to cope with various failures occurring in a highly changeable mobile multicast tree. This complicates the topology of a mobile multicast network. In addition, the PPFR method does not allow failures to occur in backup paths. If there is also a failure in a used backup path, the PPFR method will fail.

The main goal of the paper is to propose an efficient approach to providing fault-tolerant capability for mobile multicast. The proposed fault-tolerant approach contains two schemes. The first scheme attempts to establish several fault-recovery paths to reconnect all the disconnected subtrees. The establishment of the fault-recovery paths is also supported by the intrinsically redundant resources of a mobile network. Unlike the

ODFT method, the first scheme does not generate any loop while reconnecting subtrees. Due to not performing loop elimination, the fault-tolerant overhead can be significantly reduced. The first scheme also constrains the maximum delivery delay of the new reconnected multicast tree. However, if the original mobile network does not have any redundant path between a disconnected subtree and the source node, the fault-recovery path of the subtree cannot be established. In such situation, the first scheme cannot wholly repair the faulty multicast tree. The ODFT and PFR methods also have this problem. This failure situation can be successfully handled by the second scheme of the proposed approach. The second scheme extracts the failure-free part of a faulty multicast tree to form a safe multicast subtree. Then, multicast packets are only delivered along the safe multicast subtree to all the members.

The rest of this paper is organized as follows. Section 2 describes the background materials of this paper. Section 3 proposes our approach. Section 4 gives the implementation algorithms of the proposed approach. Section 5 evaluates the overhead of the proposed approach. Section 6 compares the proposed approach with previous approaches. Finally, concluding remarks are made in section 7.

2. BACKGROUND

This section gives the system model used in this paper. Then, the mobile multicast is briefly introduced. Next, the fault assumption is made. Last, previous work is reviewed.

2.1 System Model

The system model considered in this paper refers to the architecture of a 3G wireless system [11], as shown in Fig. 2. It contains four major components: mobile node (MN), radio access network (RAN), interconnection network, and Mobile IP network. The MN acts as a multicast member, which interacts with a RAN to obtain radio resources for performing multicast service. The RAN provides the data transmission across air interface. The interconnection network is used to relay packets between RANs and the Mobile IP network. With the Mobile IP network, it is divided into several serving areas and has the following main entities: foreign agent (FA), home agent (HA), and intermediate multicast router. Each FA is the default multicast router for the members within its serving area. In addition, it also cooperates with the HA to support the Mobile IP functionality [1]. Intermediate multicast routers assist FAs to deliver multicast packets in the Mobile IP network.

2.2 Mobile Multicast

Providing multicast communication in a mobile network environment is more difficult than that in a fixed network environment due to frequent movements of MNs. The current IETF Mobile IP working group has proposed two basic methods to support mobile multicast: *foreign agent based multicast* and *home agent based multicast* [1]. In this paper, the foreign agent (FA) based multicast method is adopted. Based on this method, a

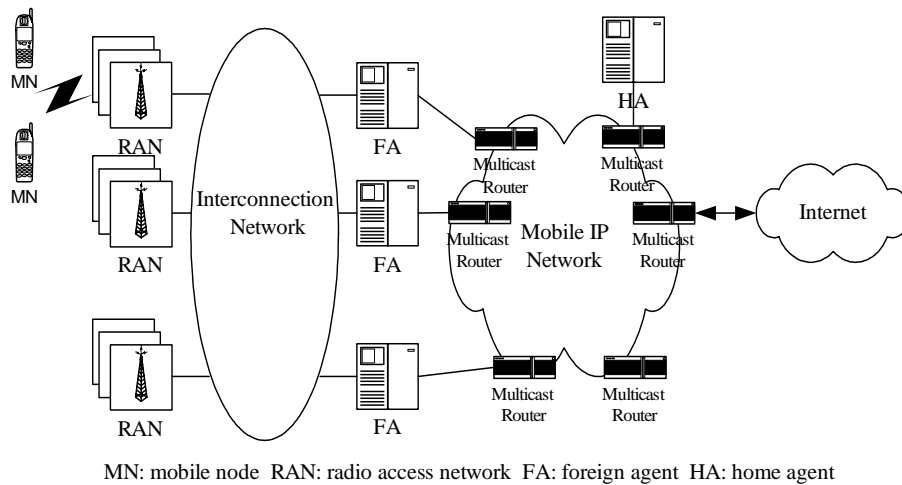


Fig. 2. Wireless mobile system model.

multicast tree is built among the FAs having members within their serving areas. All the termination nodes of a built multicast tree are FAs. When an FA receives a multicast packet, it forwards the packet to its serving members (the members within its serving area) through the interconnection network and an RAN by using multiple times of unicast transmission (multiple times of point-to-point transmission). This multicast transmission mode is also adopted in the 3G UMTS system [12].

There are many protocols to build a multicast tree, e.g. DVMRP [13], MOSPF [14], PIM [15], and CBT [7]. The CBT protocol is used in this paper since it is independent of underlying unicast routing protocol and has scaling advantage, which builds the multicast tree as follows.

- Selecting a node from the network as the core instead of the multicast source node.
- Locating the shortest path from the core to each FA with members within its serving areas.
- Merging the shortest paths.

After building a multicast tree, the structure of the multicast tree may be changed as members arrive, leave, or move (hand off). When a member newly arrives at or moves in the serving area of an FA, a *Join-Request* message is sent to the FA to add a multicast routing entry corresponding to the member. If the delivery path of the member has not been established in the multicast tree, a *graft* message is sent from the FA towards the core for adding multicast entries corresponding to the delivery path [16]. When a member leaves or moves out the serving area of an FA, a *Quit-Notification* message is sent to the FA to remove the multicast entry with respect to the member. If a segment on the delivery path of the left member is not required for other members, a *prune* message is sent along the unused path segment to delete the corresponding multicast routing entries [16].

2.3 Fault Assumption

Although the CBT protocol is used, the core failure is not discussed in detail. We assume that if the core fails, one of its child nodes will be selected to be the new core and all the members can continuously receive multicast packets from the new core. This assumption is also made in [8]. In this paper, if failures occur in a multicast tree, we first consider the situation that a portion of members incur failure affection. The multicast tree is divided into two parts: failure-free part and failure-affected part. The failure-free part contains one *failure-free subtree*. The members under the delivery range of this subtree do not incur failure affection since the subtree contains the core. The failure-affected part contains one or more *failure-affected subtrees*. The members under the delivery ranges of those subtrees incur failure affection.

With regard to the failure detection, it is assumed that the faulty state of a node (link) in a multicast tree (on-tree node (link)) can be detected by the neighboring nodes in the multicast tree. This can be achieved by operating a “keep-alive” mechanism (e.g. ICMP echo request/reply messages) between two adjacent on-tree nodes [17]. If an on-tree node fails, the root of each failure-affected subtree and one termination node of the failure-free subtree will be aware of the failure event since they are the neighbors of the faulty node.

2.4 Previous Work

As described in section 1, the existing fault-tolerant multicast approaches were classified into two basic methods: *ODFT* and *PPFR* [10]. In [7], [9], and [18], their proposed approaches belong to the ODFT method. The approach of [7] tries to find a useful redundant path to reconnect the root of each failure-affected subtree with the core (Note that the core is the root of the failure-free subtree). The useful redundant path is found by the assistance of unicast routing protocol since the unicast routing tables will be updated after some nodes (links) fail. However, the unicast routing update will take a few seconds to several minutes. In addition, the reconnection may introduce a loop if the redundant path passes through a descendent node of the failure-affected subtree. To eliminate the loop, each member under the delivery range of the failure-affected subtree (each *failure-affected member*) is asked to rejoin the multicast group by reestablishing its new multicast path. The approach of [9] attempts to reduce the loop elimination overhead in the approach of [7]. To avoid reestablishing multicast paths, it modifies the parent-child relationship of nodes in the failure-affected subtree several times. Based on the change of parent-child relationship, if the loop cannot be wholly removed, the multicast paths for the failure-affected members are re-established. In some failure cases, the overhead in the approach of [9] is larger than the approach of [7]. The approach of [18] is similar to the approaches of [7] and [9]. It considers a specific multicast environment using the PIM, IGMP, and OSPF protocols. When a failure occurs in a multicast tree, the three protocols interact to bypass the failure. The OSPF updates unique routing tables to create new loop-free routes. Then, the PIM issues join messages to reconnect all failure-affected subtrees by using new loop-free routes. The recovery time in this approach is dominated by the convergence time of OSPF routing update.

For the PPFR method, such as [8] and [10], they work under a complicated network topology. The approach of [8] utilizes the predefined backup paths to bypass the faulty

nodes (links). In the approach, many backup paths are predefined off-line for tolerating various possible failures. The approach of [10] presents a “dual-tree” scheme to build the primary and secondary multicast trees. The links and internal nodes in the primary tree are disjointed with those in the secondary tree. During the failure-free period, multicast packets are delivered along the primary tree. The secondary tree is used to deliver multicast packets only when a node (link) failure is detected in the primary tree.

3. THE PROPOSED APPROACH

This section presents a new fault-tolerant approach for mobile multicast. There are two schemes in the proposed approach. Unlike the previous approaches, the fault-tolerant capability of the proposed approach cannot fully depend on the network topology support.

3.1 Basic Idea

In mobile multicast, the delivery tree of a multicast group does not contain all the nodes (links) of a mobile network. The nodes (links) of a mobile network can be divided into two types: on-tree nodes (links) and non-tree nodes (links), as shown in Fig. 3 (a). For any two on-tree nodes, if a failure occurs between them, the non-tree nodes (links) can be used to assist the establishment of an alternative path for reconnecting the two on-tree nodes. Utilizing the intrinsically redundant resources of the mobile network (the non-tree nodes (links)), the first scheme establishes a fault-recovery path with a constrained delay for each failure-affected subtree. As shown in Fig. 3 (b), there are two failure-affected subtrees $\{4, 7\}$ and $\{5, 8\}$ after node 2 fails. The fault-recovery paths of the two failure-affected subtrees are $1, 11, 12, 4$ and $1, 3, 6, 5$, which reconnect the two subtrees with the core again. To avoid generating loops, any descendent nodes (links) in a failure-affected subtree are not allowed to be involved in the corresponding fault-recovery path. For the details about the loop avoidance and the delay constraint of the fault-recovery path, they will be elaborated in section 4.1.

Like the ODFT and PPFT methods, the first scheme is based on the network topology support. In some failure cases, the first scheme cannot work successfully. For example, in Fig. 3 (c), when node 3 fails, the first scheme can only establish the fault-recovery path for the failure-affected subtree $\{6, 9\}$. The fault-recovery path for the failure-affected subtree $\{10\}$ cannot be established via the assistance of non-tree nodes (links).

Unlike the first scheme, the second scheme is not based on the network topology support. According to the assumption made in section 2.3, if there is a failure in a multicast tree, the members under the delivery range of the failure-free subtree (*the failure-free members*) are still able to receive multicast packets again since the subtree contains the core. The basic idea of the second scheme is to use the failure-free subtree to deliver multicast packets to all the members. Recall that the mobile multicast in this paper is according to the FA based multicast method. The multicast communication is terminated at FAs (see section 2.2). Therefore, if the *failure-affected members* (the members under the delivery ranges of failure-affected subtrees) change their serving FAs to the FAs in the failure-free subtree (*failure-free multicast FAs*), the multicast packets can

if an FA receives a multicast packet, it can send the packet to any RAN and any member since a member is located within the radio coverage area of one RAN. In practice, an FA sends its received multicast packets to the RANs with members located. The identifiers of the RANs are recorded in the managing RAN record of the FA. By modifying the managing RAN record, the failure-affected members can change their serving FAs without moving their locations, as shown in Fig. 3 (c). In Fig. 3 (c), FAs 7 and 8 add the identities of the RANs 3 and 4 in their managing RAN records. Thereafter, whenever FAs 7 and 8 receive a multicast packet, they can additionally transmit the multicast packet to the four failure-affected members. The details about the technique of changing serving FA can also refer to [19] (section 3.1).

3.2 Extension

In the second scheme, if the structure of the failure-free subtree is a line as shown in Fig. 4, all the members will receive multicast packets from the only one FA. As mentioned in section 2.2, an FA forwards multicast packets to its serving members using the unicast transmission way. In Fig. 4, when FA 7 receives a multicast packet, it needs to repeatedly send the multicast packet to all the members. The performance of FA 7 may incur significant degradation.

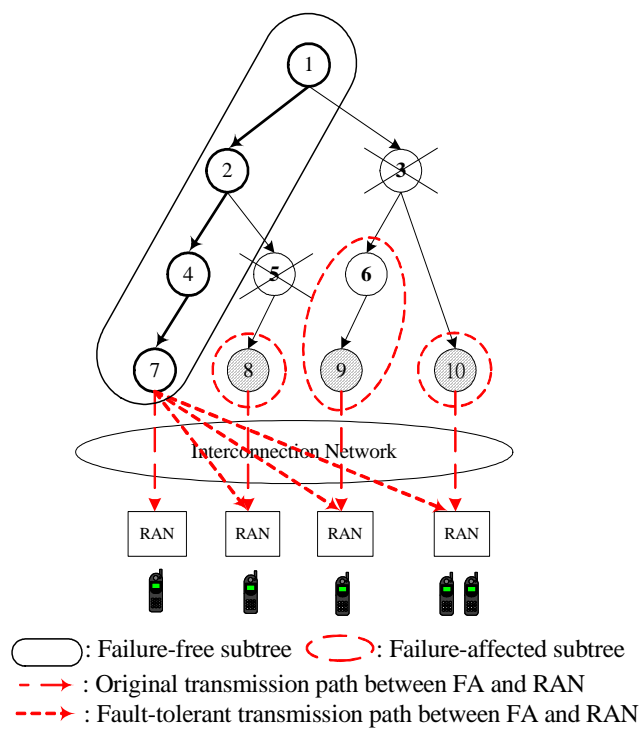


Fig. 4. A skew failure-free multicast subtree.

To avoid concentrating all multicast traffic on a single FA, the second scheme is enhanced to involve more failure-free FAs in the failure-free subtree for extending it. If the FAs are randomly selected, the extension of the failure-free subtree may take a long time for establishing the corresponding multicast paths of the randomly selected FAs. To rapidly extend the failure-free subtree, the multicast paths of the pre-visited FAs are appropriately preserved without pruning them (Note that multicast packets are not forwarded along the preserved multicast paths during the failure-free period). The pre-visited FAs indicate the FAs that previously visited by some members but now no members are in them. Thereafter, if an FA is beneficial for extending the failure-free subtree, its corresponding multicast path can be quickly added without re-establishing them. The selection of benefic FAs and the extension detail will be described in section 4.2.

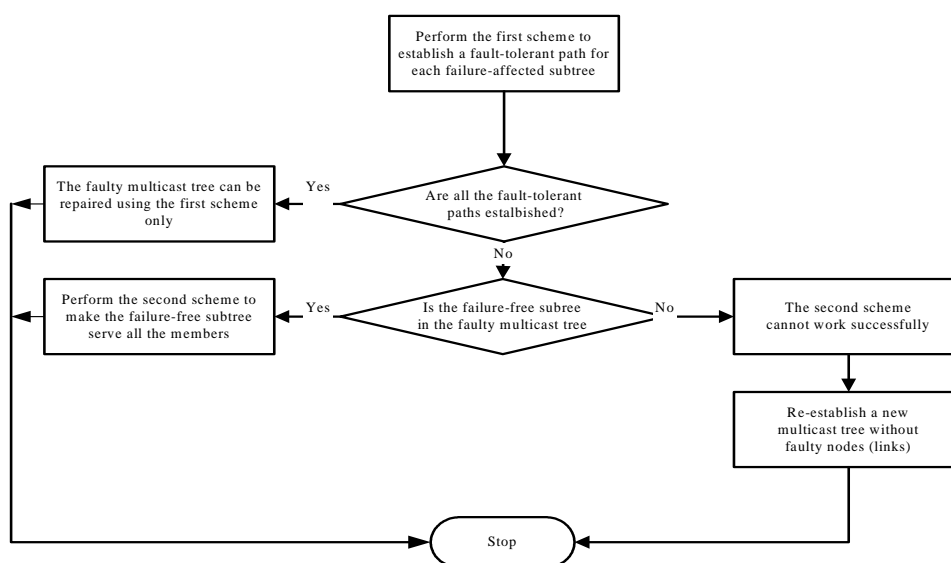


Fig. 5. Integration of the two proposed schemes.

3.3 Integration

The above two proposed schemes can be integrated to tolerate more failures. The integration is shown in Fig. 5. In the beginning, the first scheme utilizes the non-tree nodes (links) of the mobile network to establish fault-recovery paths. If any failure-affected subtree cannot find its fault-recovery path, the second scheme is performed. The second scheme first checks whether a failure-free subtree exists in the faulty multicast tree. As the assumption made in section 2.3, if the failure-free subtree exists, it is extended by restoring preserved multicast paths on it. Then, the serving FAs of all the failure-affected members are changed to the FAs in the extended failure-free subtree. Next, the extended failure-free subtree can deliver multicast packets to all the members. Conversely, if the assumption is not true, the failure-free subtree does not exist. The pro-

posed approach will fail in this failure case. In such case, a new multicast tree is re-established for all the members. In section 6.2, we will perform simulations to observe the fault-tolerant capability of the proposed approach.

4. IMPLEMENTATION ALGORITHMS AND CORRECTNESS

In this section, we give the algorithms for implementing the two proposed schemes. The correctness of each of the two schemes is also proved.

4.1 The First Scheme

The detailed algorithm for implementing the first scheme is given in Fig. 6. For convenience, the fault-tolerant operations are stated from the node point of view. These fault-tolerant operations execute the following three events.

- Notifying failure occurrence: When failures occur in a multicast tree, the root of each failure-affected subtree can know this failure event (see section 2.3). Then, the root delivers a *failure-notified message* along its corresponding failure-affected subtree. Upon receiving the failure-notified message, each node makes a mark on it to prevent involving it on a fault-recovery path. This is to avoid generating loops while repairing the faulty multicast tree.
- Finding possible fault-recovery paths: The root of each failure-affected subtree attempts to establish a fault-recovery path for reconnecting it with the core without passing through its descendent nodes. Based on the topology support of a mobile network, there may be several possible fault-recovery paths between the root and the core. There is a common characteristic for all the possible fault-recovery paths that each of fault-recovery paths must pass through one neighboring node of the root. For finding all the possible fault-recovery paths, the root disseminates *path-found* messages from all its communication interfaces towards the core, and asks the core to acknowledge the messages within a pre-determined time interval. Note that the dissemination of path-found messages is based on the source demand routing [20]. When the core receives a path-found message, the path information about the intermediate hops from the root to the core has been traced on the message. The path information is then extracted and piggybacked on the corresponding acknowledgment. The delivery delay of the traced path (e.g. the number of intermediate hops) is also attached on the corresponding acknowledgment. To further shorten the search time of possible fault-recovery paths, if an on-tree node in the failure-free subtree receives the path-found message, it, instead of the core, stops forwarding the message and immediately acknowledges the message since the path segment from it to the core has been already established in the multicast tree (see lines 3-7 of Fig. 6 (b)). If the path-found message is received by an on-tree node in the failure-affected subtree, the path-found message is dropped (see lines 3-7 of Fig. 6 (b)). In addition, for constraining the delivery delay of a fault-recovery path, the corresponding acknowledgment is asked to be replied within a pre-determined time interval. When the pre-determined time interval is time-out, if the acknowledgment is still not received, it represents that the corresponding fault-recovery path does

not exist or its delivery delay is too large. If more than one acknowledgment is received, two or more feasible fault-recovery paths exist between the root and the core. The root will select the one with the least delivery delay.

<p><u>The root of a failure-affected subtree</u></p> <ol style="list-style-type: none"> 1. Deliver a failure-notified message along its subtree 2. Disseminate path-found messages from all its communication interfaces to the core 3. Set a pre-determined time interval 4. When the time is up 5. If (one or more acknowledgements of the path-found message are received) 6. Extract the path information from each acknowledgement 7. Select the path with the least delivery delay to be the specified fault-recovery path 8. Send a path-established message to establish the determined fault-recovery path 9. Else 10. Not find the fault-recovery path 11. End

(a)

<p><u>The on-tree node (including the core)</u></p> <ol style="list-style-type: none"> 1. If (the path-found message is received) 2. Stop forwarding the message again 3. If (the on-tree node is in the failure-free subtree) 4. Respond to an acknowledgement for the message 5. Else /* the on-tree node is in a failure-affected subtree */ 6. Drop the message 7. End 8. End 9. If (the path-established message is received) 10. Add a multicast routing entry corresponding to the node sending the message 11. Stop forwarding the message again 12. Respond to an acknowledgement for the message 13. End
--

(b)

<p><u>The non-tree node</u></p> <ol style="list-style-type: none"> 1. If (the path-found message is received) 2. Forward the message towards the core 3. End 4. If (the path-established message is received) 5. If (the non-tree node has not received this type message from other nodes) 6. Add a multicast routing entry corresponding to the node sending the message 7. Make the non-tree node as a new on-tree node 8. Follow the route indicated on the message to forward the message to the next hop 9. Else /*The path segment between this node and the core is being established*/ 10. Wait for the arrival of the acknowledgement of the previous path-established message 11. Send the acknowledgement to the corresponding root 12. End 13. End
--

(c)

Fig. 6. The algorithm for the first scheme. (a) The operation of a root node. (b) The operation of an on-tree node. (c) The operation of a non-tree node.

- Establishing the determined fault-recovery path: After each failure-affected subtree determines its fault-recovery path, the path information (the identities of the intermediate hops) is put on a *path-established* message and then its root sends the message using the source demand routing. Upon receiving the path-established message, each node makes itself as a new on-tree node, adds a multicast routing entry corresponding to the node forwarding the message, and forwards the message to the next hop indicated on the message. Finally, when the destination of the path-established message (the core or an already on-tree node) receives the message, the fault-recovery path between the root and the core is established.

The above scenario is to establish a fault-recovery path for reconnecting the core with a failure-affected subtree. In a faulty multicast tree, there may be more than one failure-affected subtree. Other failure-affected subtrees can also simultaneously perform the above three steps to establish their respective fault-recovery paths. It is required to avoid generating loops among the fault-recovery paths. Once a node becomes a new on-tree node, if it receives a path-established message again from another failure-affected subtree, it will not forward this message. The reason is that the fault-recovery path segment between the new on-tree node and the core is being established by the previous path-established message. If the forwarding of the new path-established message is not prohibited, a loop may be generated since a different fault-recovery path segment may be set between the new on-tree node and the core.

4.2 The Second Scheme

With the second scheme, there are four procedures to implement it. The first procedure is executed during the failure-free period. The last three procedures are invoked when detecting failures in a multicast tree, as shown in Fig. 7.

```

/* The function determines which FAs incur the failure affection */
Find_Failure-Affected_FAs( $r_i$ )
1. Detect a failure event from node  $r_i$ 's upstream direction
   /* node  $r_i$  is a root node of a failure-affected subtree */
2. Send a failure-affected message along all the multicast paths beneath node  $r_i$ 
3. Set a failure-affected flag in each node receiving the message
4. Regard the FAs receiving the failure-affected message as the failure-affected FAs
   /* Note that the leaf nodes of a failure-affected subtree are FAs */
End
/* The function determines which FAs do not incur the failure affection */
Find_Failure-Free_FAs( $l_i$ )
1. Detect a failure event from node  $l_i$  downstream direction
   /* node  $l_i$  is a leaf node of the failure-free subtree */
2. Send a failure-immune message to the core from node  $l_i$ 
3. Deliver the message along the multicast delivery paths beneath the core
4. Regard the FAs receiving the failure-immune message as the failure-free FAs
   /* Note that the leaf nodes of a failure-free subtree are also FAs */
End

```

(a)

Fig. 7. The algorithm for the second scheme. (a) Finding the failure-affected and failure-free multicast FAs. (b) Activating past multicast paths. (c) Performing the changes of serving FAs.

```

/* The function restores the past paths on the failure-free subtree for extending it */
For each node  $n_i$  in the mobile network to add a "visit" field ( $n_i.visit$ ) in its data structure
 $n_i.visit \leftarrow$  "False" (Initially set a non-visited mark to each node)
End
Activate_Past_Paths( $n_i$ )
1. Active_Mark  $\leftarrow$  "False" /* First assume that node  $n_i$  will not involved in any restored paths */
2. If (node  $n_i$  is not a leaf node (FA))
3.   IF ( $n_i.visit =$  "False")
4.      $n_i.visit \leftarrow$  "True" (Reset a visit mark to  $n_i$ )
5.     For each "R" mark multicast entry ( $ME_i$ ) in  $n_i$ 
6.        $DR \leftarrow$  the recorded downstream multicast router in  $ME_j$ 
7.       Active_Mark  $\leftarrow$  Activate_Past_Paths( $DR$ )
8.       If (Active_Mark = "True")
9.         Active  $ME_j$  during the failure period
10.      End
11.    End
12.  Else
13.    Return (Active_Mark)
14.  End
15. Else
16.   Active_Mark  $\leftarrow$  "True"
17. End
18. Return (Active_Mark)
End

```

(b)

```

/* The function makes the FA serving changes of the failure-affected members */
Change_Serving-FAs()
1. For each failure-affected RAN
2.   Calculate the number of the failure-affected members located within the radio
   coverage area of the RAN
3.   Select an extended failure-free multicast FA with low load to be the new serving FA
   of the failure-affected members
4.   Add the identifier of the failure-affected RAN to the managing RAN record
   of the selected FA
5.   Update the loading status of the selected FA
6. End
End

```

(c)

Fig. 7. (Cont'd) The algorithm for the second scheme. (a) Finding the failure-affected and failure-free multicast FAs. (b) Activating past multicast paths. (c) Performing the changes of serving FAs.

- The first procedure is to trace past path segments during the failure-free period. As described in section 2.2, if a member leaves the serving area of an FA, a portion segment of the member's multicast path may not be used again. The multicast routers in the unused path segment will be notified to remove the multicast entries corresponding to the left member when receiving a *Quit-Notification* message. In the first procedure, when a multicast router receives a *Quit-Notification* message, it does not really remove the corresponding multicast entry. The multicast router only makes a special "R" (reserve) mark on the multicast entry. During the failure-free period, the special multicast entries

are not used to send multicast packets. They are preserved for extending the failure-free part of a faulty multicast tree. The overhead of preserving past paths will be discussed in section 5.1.

- The second procedure (Fig. 7 (a)) is to divide the FAs in the original multicast tree into two types: *failure-free multicast FAs* and *failure-affected multicast FAs*. As stated in section 2.3, if an on-tree node fails, the failure event will be notified to the root of each failure-affected subtree and one termination node of the failure-free subtree. Then, the root of each failure-affected subtree disseminates a *failure-affected* message along all the multicast paths beneath it. The dissemination will be finally terminated on some termination nodes (FAs) of the faulty multicast tree. These FAs are the failure-affected multicast FAs. Similarly, the termination node of the failure-free subtree sends a *failure-immune* message towards the core for asking it to find all the failure-free multicast FAs. The core delivers the failure-immune message along its multicast tree. Since there is a failure in the multicast tree, the failure-immune message is actually delivered along the failure-free subtree. The message is finally terminated on the FAs in the failure-free subtree. These FAs are the failure-free multicast FAs.
- The third procedure (Fig. 7 (b)) is to extend the failure-free subtree by restoring the past path segments in the subtree. There are two main steps in this procedure. The first step is to find past path segments based on the top-down way. The root of the failure-free subtree (the core) delivers an *Extension-Request* message along the subtree. Upon receiving the message, each node first checks whether it has been received the message. If the message is duplicated, the received node returns a negative acknowledgment to inform the sent node to abort the path restoration since the received node has been involved in a restored path (see lines 3-14). This is to avoid generating loops due to involving the same node in multiple restored paths. If the node first receives the *Extension-Request* message, it looks up its multicast routing table to find the entries with the “R” mark, and then forwards the message to the downstream nodes recorded on the found multicast entries. The downstream nodes continue to forward the message using the above same way. Finally, the *Extension-Request* message will be sent to some past termination nodes (pre-visited FAs) of the faulty multicast tree. Next, the second step begins to restore the found past path segments based on the bottom-up way. For a past termination node (n_p), if it receives the *Extension-Request* message, it will return a positive acknowledgment to the node sending the *Extension-Request* message. Then, the node receiving the positive acknowledge activates the “R” mark multicast entry corresponding to n_p . Other upstream nodes in the forwarding path of the *Extension-Request* message also recursively activate their corresponding “R” mark multicast entries (see lines 7-10). The activation of the “R” mark multicast entries will graft the past path segments in the failure-free subtree. Some of the pre-visited FAs are correspondingly added in the failure-free subtree. These FAs are certainly not affected by failure. If they are incurred by the failure, they cannot receive the *Extension-Request* messages from the core. In addition, the delivery delays of the restored paths can be guaranteed since these paths were previously established by the used multicast routing protocol.
- The fourth procedure (Fig. 7 (c)) is to change the serving FAs of the failure-affected members to the *extended failure-free multicast FAs*. This can be also imagined that the failure-affected members are virtually moved to the serving areas of the extended fail-

ure-free subtrees. Here, the extended failure-free multicast FAs indicate the FAs in the extended failure-free subtree. From section 3.1, we know that the FA serving changes of the failure-affected members can be easily achieved by modifying the managing RAN records of the extended failure-free multicast FAs to include the identifiers of failure-affected RANs (the RANs now managed by the faulty multicast FAs). In Fig. 7(c), each failure-affected RAN determines the number of failure-affected members within its coverage area. Then, a suitable extended failure-free multicast FA is selected to be the new serving FA of the failure-affected members. Next, the managing RAN record of the selected FA is modified to add the identifier of the failure-affected RAN. The further details about changing the serving FAs can refer to [19] (see sections 3.1 and 4.1).

4.3 Correctness

The correctness of a fault-tolerant multicast scheme can be verified by making sure that no loops are generated in the established fault-tolerant multicast tree [8, 21].

Lemma 1 The first scheme can repair the faulty multicast tree to form a healthy multicast tree without loops.

Proof: In the first scheme, it attempts to establish a fault-recovery path for each failure-affected subtree. As described before, the fault-recovery path is found by disseminating several path-found messages towards the core (see section 4.1). It is not given by the underlying unicast routing protocol after routing update. The found fault-recovery path does not involve any descendant nodes in the corresponding failure-affected subtree; therefore, the loop is impossible to be generated within a failure-affected subtree. In addition, the loops are neither generated within the failure-free subtree. While establishing the fault-recovery path, if the path-found message reaches a node of the failure-free subtree (an on-tree node), the on-tree node, instead of the core, immediately acknowledges the path-found message (see lines 1-8 of Fig. 6 (b)). The fault-recovery path cannot introduce a new alternative path for any two nodes in the failure-free subtree. Last, it is also ensured that loops are not formed among the failure-affected subtrees and the failure-free subtree (among the new on-tree nodes). Once an original non-tree node becomes a new on-tree node, it will suppress all later incoming path-established messages from other failure-affected subtrees (see lines 9-12 of Fig. 6 (c)). This suppression excludes the possibility that the new on-tree node introduces a loop (see section 4.1). In conclusion, if the first scheme can establish a fault-recovery path for each failure-affected subtree, loops are impossible to be generated in the repaired multicast tree.

Lemma 2 The second scheme can extend the failure-free part of the faulty multicast tree to form a new loop-free multicast tree to serve all the members.

Proof: In the second scheme, the failure-free part of the faulty multicast tree is extracted to form an initial fault-tolerant multicast tree. The loop is impossible to exist in the fault-tolerant multicast tree. To extend the fault-tolerant multicast tree, the past path seg-

segments without affecting by the failure are restored on the tree (see Fig. 7 (b)). For a past path segment, its nodes and links are not contained in the initial fault-tolerant multicast tree. While extending the fault-tolerant multicast tree, if one node in a past path segment has been added in the tree, other past path segments with the same node cannot be restored (see lines 3-14 in Fig. 7 (b)). This ensures that each newly added node has only one multicast path to the core. Obviously, loops cannot be generated in the extended fault-tolerant multicast tree.

5. EVALUATION

This section evaluates the failure-free and fault-tolerant overheads of the proposed approach.

5.1 Failure-free Overhead

During the failure-free period, only the second proposed scheme incurs the overhead of preserving past paths. The path-preservation operation is accompanied with the leave operation by making an “*R*” mark on the related multicast entries. The additional overhead of the mark operation is very trivial. In addition, the multicast entries corresponding to a past path are not preserved forever. The preservation of the past paths does not affect resources offering for multicast members. If a multicast node (router) has no available multicast entries for a new arriving member, the least recently used (LRU) multicast entry will be released. The victim multicast entry must be one preserved multicast entry (one “*R*” mark multicast entry) since it has not been used for current working multicast paths. The past path with the released multicast entry will not be used to extend the failure-free subtree.

5.2 Fault-tolerant Overhead

With the fault-tolerant overhead, the message and time complexities for each of the two proposed schemes are concerned. The two complexities are in terms of the number of control messages issues and the execution time taken, respectively.

5.2.1 The first scheme

For the message complexity of the first scheme, each failure-affected subtree issues four types of control messages to notify a failure event, find all the possible fault-recovery paths, acknowledge the found fault-recovery paths, and establish the determined fault-recovery path (see section 4.1). If there are $n_{affected}$ failure-affected subtrees, the message complexity of the first scheme can be represented as:

$$O(n_{affected} \times (1 + Int_{root} + Ack_{root} + 1)) = O(n_{affected} \times Int_{root}) \quad (1)$$

where the first-type control message is the failure-notified message. In the second-type control message, several path-found messages are sent from the root of a failure-affected

subtree. The number of path-found messages is dependent on the number Int_{root} of communication interfaces of the root. The third-type control message includes the acknowledgments of the path-found messages. The number Ack_{root} of the acknowledgments is at most Int_{root} . The final control message is the path-established message.

For the time complexity of the first scheme, time t_{notify} is first taken to deliver the failure-notified message along each failure-affected subtree. Then, a pre-determined time interval $t_{constrain}$ is set to constrain the delivery delay of a used fault-recovery path. Since the root of each failure-affected subtree can simultaneously disseminate the path-found messages, each failure-affected subtree can find all its possible fault-recovery paths within $O(t_{constrain})$. Next, each failure-affected subtree takes time t_{select} to determine the best one among all the possible fault-recovery paths and time $t_{establish}$ to establish the determined fault-recovery path. The time complexity of the first scheme is:

$$O(t_{notify} + t_{constrain} + t_{select} + t_{establish}) = O(h_{failure-affected} + l_{pre-length}) \quad (2)$$

where t_{notify} is dependent on the height $h_{failure-affected}$ of a failure-affected subtree. $t_{constrain}$ can be transformed to a pre-determined length $l_{pre-length}$ (number of hops). The time t_{select} is taken for comparing the costs of all the possible fault-recovery paths, and it is very trivial. The best fault-recovery path is found within $t_{constrain}$; therefore, $t_{establish}$ is proportional to $t_{constrain}$.

5.2.2 The second scheme

Based on the procedures presented on section 4.2, the control messages of the second scheme are issued to perform the following items:

- Classifying all the termination nodes (FAs) of the faulty multicast tree into failure-affected multicast FAs and failure-free multicast FAs.
- Extending the failure-free subtree.
- Finding all the failure-affected members.
- Changing the serving FAs of the failure-affected members.

For the first item, each on-tree node neighboring the faulty node (link) issues a failure-affected or failure-immune message. Then, each FA in the faulty multicast tree is categorized as the failure-affected or failure-free multicast FA according to its received message (failure-affected or failure-immune message). The message complexity of the first item is bound to $O(N_{neighbors})$, where $N_{neighbors}$ is the number of on-tree nodes neighboring the faulty node (link). The second item is done by issuing an Extension-Request message along the failure-free subtree to restore the past path segments in the subtree. Only one control message is issued in the second item, and its message complexity is $O(1)$. The third item is achieved by issuing a command message to each failure-affected multicast FAs to search its visitor list. The message complexity of the third item is bounded to $O(N_{FAs-affected})$, where $N_{FAs-affected}$ is the number of failure-affected multicast FAs. The final item is performed by issuing a command message to each failure-free multicast FA to modify its managing RAN record for including the identities of

some failure-affected RANs. The message complexity of the fourth item is bounded to $O(N_{FAs-free})$, where $N_{FAs-free}$ is the number of failure-free multicast FAs. Finally, the message complexity of the second scheme can be represented as:

$$O(N_{neighbors}) + O(1) + O(N_{FAs-affected}) + O(N_{FAs-free}) = O(N_{FA}) \quad (3)$$

where N_{FA} is the number of FAs in the faulty multicast tree, and $N_{neighbors}$ must be less than N_{FA} since the fault-affected and failure-immune messages issued from the faulty neighboring nodes are finally delivered to the FAs in the faulty multicast trees. For the term $O(N_{FAs-affected}) + O(N_{FAs-free})$, it can be simplified as $O(N_{FA})$ since N_{FA} is the sum of $N_{FAs-affected}$ and $N_{FAs-free}$.

With the time complexity, the first item takes maximum time $\{t_{failure-affected}, t_{failure-immune}\}$ to make each FA in the faulty multicast tree know its working status: failure-affected or failure-free, where $t_{failure-affected}$ is the maximum delay for delivering the failure-affected message from the root of a failure-affected subtree to one of its termination nodes, and $t_{failure-immune}$ is the maximum delay for delivering the failure-immune message from the core to one of termination nodes in the failure-free subtree. $t_{failure-affected}$ and $t_{failure-immune}$ are dependent on the height $h_{failure-affected}$ of a failure-affected subtree and that $h_{failure-free}$ of the failure-free subtree, respectively. The second item needs to take time $t_{extension}$ for delivering the Extension_Request message along the failure-free subtree to make each node in the subtree restore the past path segments. Then, the past path segments can be recursively restored by taking time $t_{restoration}$. The times $t_{extension}$ and $t_{restoration}$ are dependent on $h_{failure-free}$, which are bounded to $O(h_{failure-free})$. For the time complexity of the third item, it is dominated by searching the visitor list of an FA. As to the final item, its time complexity is determined by modifying the managing RAN record of failure-free multicast FAs. The above search and modification operations take small time ε . The details have been analyzed in our previous paper [19] (see section 5.2 pp. 214-215). Finally, the time complexity of the second scheme can be represented as

$$\begin{aligned} & \max\{O(h_{failure-affected}), O(h_{failure-free})\} + O(h_{failure-free}) + \varepsilon \\ & = \max\{O(h_{failure-affected}), O(h_{failure-free})\} \end{aligned} \quad (4)$$

5.2.3 Increasing transmission order

In the second scheme, all the failure-affected members are served by the extended failure-free multicast FAs. As described in section 2.2, the FA sends multicast packets to its serving members using the unicast transmission way. For a normal member, its transmission order in its serving FA may be preempted by some failure-affected members. The second scheme incurs the following additional overhead:

- The increasing transmission order T_{Inc_Order} that causes a member to postpone its transmission order in a FA in comparison to per-failure.

To derive T_{Inc_Order} , the multicast traffic behavior of an FA as well as the following probabilities and expected numbers should be obtained first.

- The probability P_{M_k} that there are k members served by an FA.
- The expected number N_{B_Member} of members served by an FA before a failure.
- The expected number N_{A_Member} of members served by a failure-free multicast FA after a failure.

We assume that the member arrival to an FA follows a Poisson distribution with the mean arrival rate λ_a . The member handoff to an FA is also a Poisson process with the mean handoff rate λ_h . The distribution function of a multicast session time is a general function with the mean session time $\frac{1}{\mu_s}$. Similarly, the residence time of a member in an FA's serving area is also not dependent on any specific distribution. The mean residence time is $\frac{1}{\mu_r}$. Under these assumptions, the multicast traffic behavior of an FA can be modeled as the $M/G/c/c$ queuing model [22]. In [23, 24], they also use the similar queuing modes to model the multicast traffic. Note that M denotes a Poisson process for the arrivals of members, G denotes an independent distribution for the service (residence) time, the first c represents the number of resources in an FA for arriving members, and the second c indicates the limit for the maximum number of members simultaneously served by an FA.

After modeling the multicast traffic, P_{M_k} can be easily obtained from [22], expressed as follows:

$$P_{M_k} = \frac{\frac{(\frac{\lambda}{\mu})^k}{k!}}{\sum_{i=0}^c \frac{(\frac{\lambda}{\mu})^i}{i!}} \quad (5)$$

where $\lambda = \lambda_a + \lambda_h$ and $\mu = \mu_r + \mu_s$

For $N_{B_Members}$, it can be expressed as:

$$N_{B_Member} = \sum_{k=0}^c k \times P_{M_k} \quad (6)$$

Form Eq. (5), Eq. (6) can be further rewritten as

$$N_{B_Member} = \sum_{k=0}^c k \times P_{M_0} \times \frac{(\frac{\lambda}{\mu})^k}{k!} = \frac{\sum_{k=1}^c \frac{(\frac{\lambda}{\mu})^k}{(k-1)!}}{\sum_{i=0}^c \frac{(\frac{\lambda}{\mu})^i}{i!}} \quad (7)$$

where P_{M_0} is the probability that there is no members in the serving area of an FA.

As for N_{A_Member} , it is the sum of N_{B_Member} and the number of failure-affected members served by an extended failure-free multicast FAs, which is expressed as:

$$N_{A_Member} = N_{B_Member} + ((N_{FA} \times p_f) \times N_{B_Member}) \times r \quad (8)$$

where p_f is the probability of an FA incurring the failure affection, N_{FA} is the number of FAs in the faulty multicast tree, the term $(N_{FA} \times p_f) \times N_{B_Member}$ represents the average number of all the failure-affected members, and r is the ratio of the number of failure-affected members served by a extended failure-free multicast FA over the total number of failure-affected members. If the failure-affected members are evenly served by all the extended failure-free multicast FAs, r is $\frac{1}{(N_{FAs} + N_{D_FAs}) \times (1 - p_f)}$. N_{D_FA} is the number of the pre-visited FAs. Due to space limitation, the derivations of N_{FA} and N_{D_FA} are presented in [25] (see Appendix A).

The maximum value of T_{Inc_Order} can be obtained by subtracting Eq. (7) from Eq. (8), as follows:

$$T_{Inc_Order} = \left((N_{FA} \times p_f) \times \left(\frac{1}{\sum_{i=0}^c \frac{(\frac{\lambda}{\mu})^i}{i!}} \times \sum_{k=1}^c \frac{(\frac{\lambda}{\mu})^k}{(k-1)!} \right) \right) \times r \quad (9)$$

6. COMPARISON AND SIMULATION

6.1 Comparison

Table 1 makes the comparisons between the previous approaches (the ODFT and PFR methods) and the proposed approach with respect to the network topology support, failure-free overhead, fault-tolerant capability, fault-tolerant overhead, and increasing delay.

- Network topology support: It is the main idea for the ODFT and PFR methods since the two methods utilizes the existing redundant paths and the predefined backup paths to achieve fault tolerance. In contrast, the proposed approach does not fully need the network topology support. If some failure-affected subtrees cannot be reconnected by using the existing redundant paths, the proposed approach extracts the failure-free part of the faulty multicast tree to form a fault-tolerant multicast tree.
- Fault-free overhead: The ODFT and PFR methods do no take any actions against failures during the failure-free period. Only the second scheme of the proposed approach needs to perform the path preservation operation. However, this operation can be accompanied with the leave operation and its cost is very trivial (see section 5.1).
- Fault-tolerant capability: For the ODFT method, its fault-tolerant capability is dependent on the available redundant paths in the network. If the reconnection path of a failure-affected subtree cannot be found from the existing redundant paths, the ODFT method will fail. For the PFR method, if m simultaneous failures are desired to be tolerated, m disjointed backup paths must be predefined from an on-tree node to its m

ancestors in the multicast tree [8]. The fault-tolerant capability is strongly dependent on the number of the backup paths predefined for a node. However, if many backup paths are predefined, the management and maintenance of the network will become very complex. From Fig. 3 (d), we also know that the ODFT and PPFR methods do not allow failures to occur in the termination nodes of a multicast tree if the faulty termination nodes are not equipped with the backup nodes. Compared to the ODFT and PPFR methods, the fault-tolerant capability of the proposed approach is best. If some failure-affected subtrees cannot be reconnected by using the existing redundant paths, the failure-affected members are able to receive multicast packets by changing their serving FAs.

- **Fault-tolerant overhead:** For the ODFT method, its fault-tolerant overhead includes the finding and connection of suitable redundant paths as well as the loop elimination. To find a suitable redundant path for a failure-affected subtree, the root of the subtree sends a *Rejoin_Request* message towards the core [9] (see section 3 pp. 4-5). The time complexity of finding the suitable redundant path and that of connecting the found path are bounded to $O(h_{tree})$, where h_{tree} is the height of the faulty multicast tree. The connection of the redundant path may generate a loop in the repaired multicast tree. To eliminate the loop, the structure of the failure-affected subtree is adjusted by exchanging the positions of some nodes in the subtree [9] (see section 4 pp. 5-8). The structure adjustment may perform several times until the loop is wholly removed. The time complexity of the loop elimination is bounded to $O(h_{failure-affected} \times t_{adjust})$, where t_{adjust} is the times of structure adjustment. However, in the worst case, the loop cannot be wholly removed by using the structure adjustment. In such case, the FAs in the failure-affected subtree are asked to rebuild their respective new failure-free multicast paths, and the time complexity of this loop elimination becomes $O(h_{tree} \times n_{FA-affected})$, where $n_{FA-affected}$ is the maximum number of FAs in a failure-affected subtree. For the PPFR method, it needs to perform the loop prevention while activating the backup paths. The loop prevention is done by not allowing the activated backup paths to intersect with each other [8] (see section 4.3.3 pp. 990-992). If one of the backup paths intersects with another, the tail segment of the backup path is trimmed from its intersection point. The time complexity of the loop prevention is bounded to $O(N_{path} \times P_{node})$, where N_{path} is the number of activated backup paths, P_{node} is the number of nodes in the backup path with the largest length. After trimming the backup paths, time $O(P_{node})$ is taken for activating all the used backup paths. For the proposed approach, the fault-tolerant overheads of its two schemes have been analyzed as $O(h_{failure-affected} + l_{pre-length})$ and $\max\{O(h_{failure-affected}), O(h_{failure-free})\}$, respectively (see section 5.2). Note that only the first proposed scheme needs to perform the loop avoidance. The loop avoidance is incorporated into the establishment of fault-recovery paths by delivering a failure-notified message along each failure-affected subtree, which time complexity is bounded to $O(h_{failure-affected})$.
- **Increasing delivery delay:** In the ODFT method, the increasing delay of a fault-tolerant multicast tree is dependent on the greatest number $n_{non-tree1}$ of non-tree nodes in a used redundant path. If a used redundant path has many non-tree nodes, the delivery delay of the fault-tolerant multicast tree is obviously larger than that of the original multicast tree. Similarly, in the PPFR method, the paths in the fault-tolerant multicast tree consist of original paths and activated backup paths. The increasing delivery delay of the

PPFR method is dependent on the most number $n_{non-tree2}$ of non-tree nodes in an activated backup path. For the proposed approach, the first scheme pre-determines a length $l_{pre-length}$ to constrain the delays of all the fault-recovery paths. The increasing delay of the fault-tolerant multicast tree can be controllable and is less than $l_{pre-length}$. As for the second scheme, the fault-tolerant multicast tree is extracted from the faulty multicast tree and its extension is done by restoring past multicast paths. The delivery delay of the fault-tolerant multicast tree may be larger or less than that of the original faulty multicast tree.

Table 1. Comparisons between the previous approaches and the proposed approach.

Comparing Metrics	ODFT	PPFR	Proposed approach	
			First scheme	Second scheme
Network topology support	Yes	Yes	Yes	No
Fault-tolerant capability	Dependent on redundant paths	Dependent on backup paths	Dependent on redundant paths	Not dependent on network topology
Fault-tolerant overhead	Loop elimination $O(h_{tree} \times n_{FA-affected})$	Loop prevention $O(N_{paths} \times P_{nodes})$	Loop avoidance $O(h_{failure-affected})$	Extension of failure-free subtree and virtual move $\max\{O(h_{failure-affected}), O(h_{failure-free})\}$
	Connection of redundant paths $O(h_{tree})$	Activation of backup paths $O(p_{nodes})$	Establishment of fault-recovery paths $O(l_{pre-length})$	
Failure-free overhead	No	No	No	Path reserve
Increasing delay	Dependent on the number of non-tree nodes on a redundant path $n_{non-tree}$	Dependent on the number of non-tree nodes on a backup path $n_{non-tree}$	Dependent on the predetermined length $l_{pre-length}$	Not sure

From Table 1, we can obviously see that the proposed approach outperforms the PPFR and ODFT methods. Although the network topology used for the PPFR method is more complex than those used for the ODFT method and the proposed approach, the PPFR method does not have less fault-tolerant overhead in comparison with the proposed approach since the loop prevention is performed in the PPFR method and it may take a considerable time. Strictly speaking, if two fault-tolerant multicast approaches are not based on the same network topology, it cannot make a fair comparison between them since they are from different viewpoints to treat the incurred overheads. It seems to be more reasonable for comparing the ODFT method and the proposed approach. Extensive simulation experiments were performed to quantify the effectiveness of the proposed approach over the ODFT method.

6.2 Simulation

The simulation model used in this paper is similar to those used in [26, 27]. Based on the simulation model, 20 random graphs are generated from a $25 * 25$ coordinate grid to be the topologies of the Mobile IP network, respectively. The random graph possesses characteristics of a real network [27]. There are 100 nodes (multicast routers) in each generated random graph, and the link for a pair of nodes (u, v) is according to the edge probability $P_e(u, v) = \frac{k\bar{e}}{N} \beta e^{-\frac{d(u,v)}{L\alpha}}$ [27]. For the 100 nodes, 20 nodes are specified to be the FAs of the Mobile IP network. Then, 20 different multicast trees are built from a generated random graph using the CBT algorithm. For each built multicast tree, it contains at most 20 FAs to be its termination nodes. The number of members in the serving area of an FA is varied between 0 and 50. The maximum number of members served by an FA is 50. Next, each simulation run is based on a built multicast tree to perform join, leave, and handoff operations for 1000 units of time. To understand the effect of different multicast traffic on the ODFT method and that on the proposed approach, the operations (join, leave, and handoff) in a simulation run are appropriately arranged to generate 4 different traffic intensities (Erlangs): 10, 25, 50, and 100, respectively. Here, the traffic intensity is defined as the average ratio of member arrival rate over the member service rate at a termination node of a multicast tree (at a multicast FA). Failures are randomly generated on the on-tree nodes (links) during the execution of each simulation run.

Based on the above description, the number of simulation runs is $20 * 20 * 4 = 1600$. In the 1600 simulation runs, 22505 failures occur. The ODFT approach can tolerate 4799 (21.32%) failures. In contrast to the proposed approach, 22036 (97.92%) failures can be tolerated. Among the 22036 failures, 19.36% can be successfully recovered by the first proposed scheme only, but 80.64% must be recovered via the second proposed scheme. These simulation runs measure the three overhead metrics: average number of control messages, average number of time units, and average increasing transmission order mentioned in section 4. In addition, for understanding the delivery delay of a fault-tolerant multicast tree, the average maximum path length is also measured in the simulations.

Fig. 8 illustrates the above four simulation metrics, respectively. For conveniently measuring the second simulation metric, the average units of time taken is measured as the average number of hop count while establishing the fault-tolerant multicast tree. The number of hop count is also used to represent the multicast traffic in [28, 29]. From Figs. 8 (a) and (b), we can see that the first scheme has obviously lower overhead than the ODFT method. Even if the faulty multicast trees need to be recovered by the second scheme, the incurred overhead is still less than the ODFT method. In the ODFT method, it needs to perform the topology-change routing update for getting the information about the suitable redundant paths. However, the cost of the routing update is not included in the simulation results of the ODFT method. Moreover, as mentioned before, the ODFT method may introduce a loop in its established fault-tolerant multicast tree. Table 2 shows the average ratio of loop generation by the ODFT method is at least 0.59. If a fault-tolerant multicast tree is established, the probability that there is a loop in the tree is not small. When $\frac{\lambda}{\mu}$ is 50, the probability is large as 0.92. If the loops in a fault-tolerant multicast tree cannot be eliminated, the tree cannot be used to deliver multicast packets.

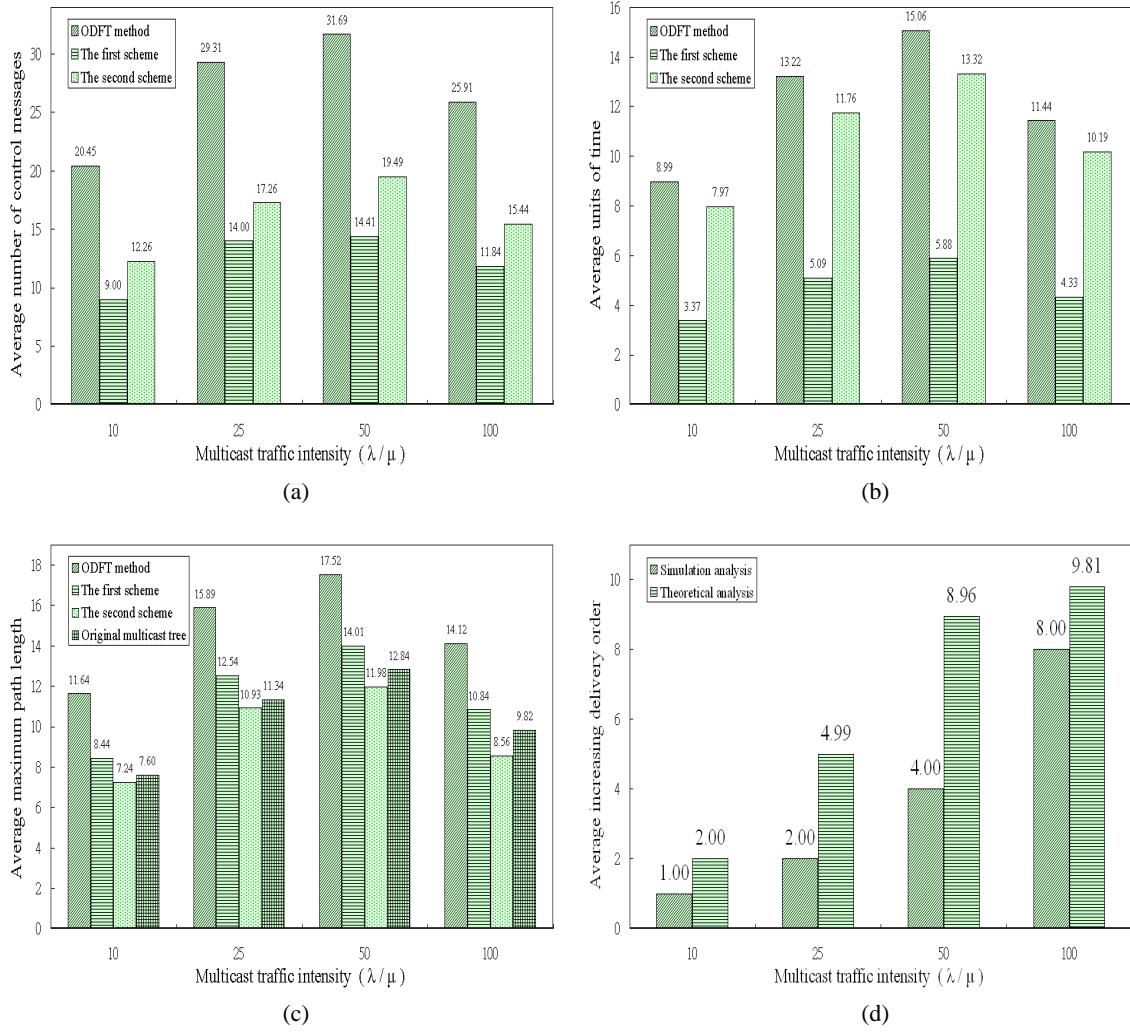


Fig. 8. The fault-tolerant overheads for the ODFT method and the proposed approach. (a) Average number of control messages. (b) Average units of time. (c) Average maximum path length. (d) Average increasing delivery order.

Table 2. Average ratio of possible loop generation.

Multicast traffic intensity	Average ratio of possible loop generation
10	0.59
25	0.81
50	0.92
100	0.73

Also shown in Fig. 8 (c), the two proposed schemes have better performance on the average maximum path length than the ODFT method. Especially, the second proposed scheme nearly does not increase the average maximum path length in its established fault-tolerant multicast tree. It is due to the fact that the fault-tolerant multicast tree in the second scheme is extracted from the original faulty multicast tree. Last, Fig. 8 (d) shows that the increasing transmission order is not large. (Note that this simulation metric is relevant only for the second scheme of the proposed approach). When the multicast traffic intensity is not large (e.g. 10), the increasing transmission order is small. Even when the multicast intensity increases from 50 to 100, the increasing transmission order only increases from 8.96 to 9.81. This also means that the second scheme does not introduce significant performance degradation on a failure-free multicast FA even when the multicast traffic intensity is very large. In addition, Fig. 8 (d) also shows that the theoretical analysis (Eq. (9)) of the increasing delivery order is close to the simulation analysis.

As for the effect of the multicast traffic intensity, the first three simulation metrics do not increase as the multicast traffic intensity increases. The reason is explained as follows. From section 5, we know that the structure of a faulty multicast tree and the topology of the mobile network are two important factors for affecting the simulation metrics. The variance in the structure of a multicast tree is mainly determined on the execution pattern of the multicast operations, not the increase of the multicast traffic intensity. In the last simulation metric, the increasing transmission order slowly increases as the multicast traffic intensity increases.

7. CONCLUSIONS

This paper has proposed an efficient approach to tolerating failures for mobile multicast. There are two schemes in the proposed approach. The first proposed scheme uses the redundant resources of a mobile network to repair a faulty multicast tree. If the faulty multicast tree cannot be wholly repaired, the second proposed scheme extracts a failure-free subtree from the faulty multicast tree to continuously deliver multicast packets to all members. To avoid introducing a large overload on the failure-free subtree, the subtree extension is also presented. Compared to the previous approaches, the proposed approach possesses the following advantages:

- Not fully needing the network topology support.
- Not incurring the potentially large fault-tolerant overhead
- Constraining the maximum delivery delay of the fault-tolerant multicast tree.

The detailed comparisons were also made in Table 1. Although the PFR method uses a complex network topology, its fault-tolerant overhead is not always small. In some cases, the PFR method needs to take a non-trivial cost ($O(N_{paths} \times P_{nodes})$) to trim the interaction of backup paths for avoiding loop generation. Extensive simulations were also performed. The results show that the proposed approach outperforms the ODFT method in the fault-tolerant capability and various performance overheads.

ACKNOWLEDGMENT

This research was supported by the Nation Science Council, Taiwan, R.O.C., under Grant NSC 92-2213-E-030-004 and 93-2475-E-030-005-URD.

REFERENCES

1. C. Perkins, "IP mobility support," Technical Report IETF RFC 3344, 2002.
2. S. Paul, K. K. Sabnani, J. C. H. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (RMTP)," *IEEE Journal on Selected Areas in Communications*, Vol. 15, 1997, pp. 407-421.
3. W. J. Liao, C. A. Ke, and J. R. Lai, "Reliable multicast with host mobility," *GLOBECOM '00, IEEE Global Telecommunications Conference*, Vol. 3, 2000, pp. 1692-1696.
4. B. Whetten and G. Taskale, "An overview of reliable multicast transport protocol II," *IEEE Network*, Vol. 14, 2000, pp. 37-47.
5. G. Anastasi, A. Bartoli, and F. Spadoni, "A reliable multicast protocol for distributed mobile systems: design and evaluation," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, 2001, pp. 1009-1022.
6. A. M. Kermarrec, L. Massoulié, and A. J. Ganesh, "Probabilistic reliable dissemination in large-scale systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, 2003, pp. 248-258.
7. A. J. Ballardie, P. F. Francis, and J. Crowcroft, "Core based trees," in *Proceedings of ACM SIGCOM*, 1993, pp. 85-95.
8. W. Jia, W. Zhao, D. Xuan, and G. Xu, "An efficient fault-tolerant multicast routing protocol with core-based tree techniques," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, 1999, pp. 894-1000.
9. L. Schwiebert and R. Chintalapati, "Improved fault recovery in core based trees," *Computer Communications*, Vol. 23, 2000, pp. 816-824.
10. A. Fei, J. Cui, M. Gerla, and D. Cavendish, "A dual-tree scheme for fault-tolerant multicast," *IEEE International Conference on Communications*, Vol. 3, 2001, pp. 690-694.
11. B. Sarikaya, "Packet mode in wireless networks: overview of transition to third generation," *IEEE Communications Magazine*, Vol. 38, 2000, pp. 164-172.
12. M. Hauge and Ø. Kure, "Multicast in 3G networks: employment of existing IP multicast protocols in UMTS," *ACM Sigmobile*, 2002, pp. 96-103.
13. D. Waitzman, C. Patridge, and S. Deering, "Distance vector multicast routing protocol," Technical Report IETF RFC 1075, 1988.
14. J. Moy, "Multicast extensions to OSPF," Technical Report IETF RFC 1584, 1994.
15. S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei "The PIM architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, Vol. 4, 1996, pp. 153-162.
16. B. Cain *et al.*, "Internet group management protocol, version 3," Technical Report IETF RFC 3376, 2002.
17. S. Deering, "ICMP router discovery messages," Technical Report IETF RFC 1257,

- 1991.
18. X. Wang, C. Yu, H. Schulzrinne, P. Stirpe, and W. Wu, "IP multicast fault recovery in PIM over OSPF," *IEEE International Conference on Network Protocols*, 2000, pp. 116-125.
 19. J. W. Lin and J. Arul, "An efficient fault-tolerant approach for mobile IP in wireless systems," *IEEE Transactions on Mobile Computing*, Vol. 2, 2003, pp. 207-220.
 20. J. Postel, "Internet protocol," Technical Report IETF RFC 791, 1981.
 21. G. N. Rouskas and I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," *IEEE Journal on Selected Areas in Communications*, Vol. 15, 1997, pp. 346-356.
 22. D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*, John Wiley & Sons, Inc. Publication, 1985.
 23. C. C. Cheung, H. K. Tsang, and S. Gupta, "Dynamic multicast routing based on mean number of new calls," *IEEE/ACM Transactions on Networking*, Vol. 8, 2000, pp. 679-688.
 24. N. Banerjee and S. K. Das, "Analysis of mobile multicasting in IP-Based wireless cellular networks," *IEEE International Conference on Communication*, Vol. 5, 2002, pp. 3388-3392.
 25. J. W. Lin, "An integrated fault-tolerant multicast approach for wireless mobile systems," Technical Report of Fu Jen Catholic University, Taiwan, available at <ftp://jwlin1.csie.fju.edu.tw/> (file: supplement of fault-tolerant multicast in the public directory), 2003.
 26. B. M. Waxman, "Routing of multipoint connections," *IEEE Journal on Selected Areas in Communications*, Vol. 6, 1988, pp. 1617-1622.
 27. H. C. Lin and S. C. Lai, "VTDM-A dynamic multicast routing algorithm," *IEEE INFOCOM '98*, in *Proceedings of 17th Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, 1998, pp. 1426-1432.
 28. Y. J. Suh, H. S. Shin, and D. H. Kwon, "An efficient multicast routing protocol in wireless mobile networks," *ACM Wireless Networks*, Vol. 7, 2001, pp. 443-453.
 29. H. S. Shin and Y. J. Suh, "Multicast routing protocol in mobile networks," *IEEE International Conference on Communications*, Vol. 3, 2000, pp. 1416-1420.



Jenn-Wei Lin (林振緯) received the M.S. degree in Computer and Information Science from National Chiao Tung University, Hsinchu, Taiwan, in 1993, and the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, in 1999. He is currently an Assistant Professor in the Department of Computer Science and Information Engineering, Fu Jen Catholic University, Taiwan. He was a researcher at Chunghwa Telecom Co., Ltd., Taoyuan, Taiwan from 1993 to 2001. His current research interests are fault-tolerant computing, mobile computing and networks, distributed systems, and broadband networks.