An Architecture for Scalable, Efficient and Fast Fault-Tolerant Multicast Provisioning

Jun-Hong Cui[1], Michalis Faloutsos[2], Mario Gerla[3]
[1] Computer Science & Engineering, University of Connecticut, Storrs, CT 06269
[2] Computer Science & Engineering, University of California, Riverside, CA 92521
[3] Computer Science Department, University of California, Los Angeles, CA 90095

Abstract— Providing fault-tolerance to multicast connections is an important and challenging requirement of future networks. The existing techniques for fault-tolerant multicast can be grouped into on-demand and pre-planned approaches. On-demand approaches can have long recovery latency. For faster recovery, pre-planned approaches have been developed. However, in this type of approaches, the overhead cost is generally very high, especially when there is a large number of simultaneous groups in the network. In this article, we first provide an overview of the current multicast fault-tolerance methods. In addition, we propose a novel architecture called **Aggregated MPLS-based Fault-Tolerant Multicast** (**AMFM**) for scalable, efficient and fast fault-tolerant multicast provisioning. AMFM falls in the category of pre-planned approaches. Using the concept of *aggregated multicast* [5], AMFM facilitates fault-tolerance in a very elegant way: it reduces the protection cost significantly; it is scalable to large numbers of groups; and it can also recover failure in a very fast manner. This article describes the architecture of AMFM and provides a feasibility check from an implementation point of view. We also conduct experiments to evaluate the performance of AMFM and show it can provide fault-tolerance in a scalable fashion.

Keywords— **Fault-Tolerance, Fast Recovery, Multicast Provisioning, Aggregated Trees, MPLS (Multiple Protocols Label Switching)**

I. Introduction

Fault-tolerant networking has become prominent concern of the research and business community. Fault-tolerant group communications have received relatively little attention so far. Group communications or multicast involves the communication of multiple entities at the same time. Fault-tolerance in multicast communications is more demanding than unicast, since one link failure affects many receivers of the same group, and multicast routing is a more involved procedure. Several emerging group applications will need fault-tolerance to ensure high levels of quality, such as video conferencing, distributed network games, telemedicine, remote robot steering, and distance lectures with student participation. We consider that users will expect and be willing to pay for service robustness, which is a trend that we observe so far with network services (i.e. switching from dial-up modem lines to the more expensive cable or DSL).

Multicasting involves the dissemination of information to multiple receivers at the same time from one or more sources. The data is distributed with the use of a tree structure, which we call multicast tree. The establishment and maintenance of this tree makes multicast routing more challenging than unicast routing. First, the creation of the tree requires the establishment of routing state. For network level multicasting, this state has to be installed in the participating routers[1]. As a result, the routing state is proportional to the number of active multicast groups. Second, reliability and fault-tolerance become exponentially more challenging due to the multiplicity of receivers. For example, in contrast to unicast communications, any packet loss or link failure can lead to a large number of "complaining" nodes. Thus, a simple closed feedback protocol from unicast communications cannot be applied here.

The focus of this article is the efficiency of fault-tolerant multicast schemes. The primary aspect of the efficiency is fast recovery: we want to minimize the disruption perceived by the user. Other equally important efficiency aspects are the state scalability and communication overhead, which will make the scheme feasible in practice. Finally, we want a scheme that will facilitate the management of the network and provide control to the network administrator. Ideally, we would like our scheme to integrate well with the ability to provide guaranteed services. We assume a target network with a large number of groups and the possibility of failing links and nodes. We focus on a network that is under an administrative authority, who is interested in maximizing the network efficiency and performance.

Several fault-tolerant schemes for multicast exist, which satisfy some of our efficiency requirements, but there is not a scheme that satisfies all of them. The existing schemes can be grouped in two categories: pre-planned and on-demand. On-demand approaches do not need to compute backup routes beforehand, so the computational and maintenance cost is low. However, these schemes usually experience longer recovery latency, since the whole rerouting procedure is triggered on-demand. In contrast, pre-planned failure restoration predefines the backup routes, as introduces a large amount of computational and maintenance cost when there are large numbers of groups ongoing in the network. The big advantage of this type of fault-tolerance is the much shorter recovery latency, which is desired by many real-time communications and some other time-critical applications (such as coordination among multiple sites during NASA satellite launches).

In this article, we provide an overview of fault-tolerant schemes for multicast communications. We discuss the relative advantages and disadvantages of the schemes. We identify a major trade-off between high overhead and fast recovery. To this effect, we propose a novel architecture which we call

[1]The state requirements can vary between multicast routing protocols. Lately, application layer multicast has been proposed to alleviate this problem, but provides suboptimal resource utilization. In this article, we focus on network level multicasting.

Aggregated **M**PLS-based **F**ault-Tolerant **M**ulticast (**AMFM**). Our scheme falls in the category of pre-planned approaches, and it is based on the concept of aggregated multicast [5]. We show that our scheme can provide fast recovery while minimizing the required overhead. To achieve this, our scheme brings bandwidth efficiency as a third element in the trade-off in a tunable way: we can define the amount of bandwidth overhead that we are willing to incur in order to achieve fast recovery in a scalable way. Finally, we provide details of how our approach would be implemented in an MPLS environment[2].

The key idea of our approach is the separation of the tree from the multicast group: the tree becomes a routing abstraction, while the multicast group is a communication abstraction. Groups are mapped to trees on a temporary basis and with an explicit mapping at the ingress and egress points of the network. In case of routing failures, the group is re-mapped to a new tree. This re-mapping can be done quickly and efficiently, since we basically need to change the mapping at the boundaries of the network. In addition, we can map multiple groups to one tree, which reduces the required routing state inside the network. Aggregated multicast was initially designed as a state-reduction scheme, but here, it becomes a powerful tool to facilitate fault-tolerance. AMFM can reduce the protection cost significantly, and it scales well to large numbers of groups.

The rest of this article is organized as follows. We first review related work, and then outline the motivation and key concepts of our design. We then present our proposed architecture, AMFM, in detail, and examine the performance of the approach. We also discuss the issues of extending AMFM to QoS-aware network environments.

II. Background and Related Work

In this section, we briefly review some previous work on pre-planned restoration for multicast. We also discuss Multiple Protocol Label Switching (MPLS), and its use in fault-tolerance in multicasting. As we already mentioned, the implementation of our approach could be greatly facilitated by MPLS, and we describe the principles of such an implementation in the subsequent sections.

A. Pre-planned Restoration for Multicast

Multicast traffic is usually distributed over a tree structure. A single failure will affect all members at the downstream of the failure point. A couple of pre-planned fault-tolerant multicast schemes have been previously studied [14, 8, 4]. Some of them use link protection [14]; some employ path protection [14]; and some others utilize redundant tree [8] or "dual" tree [4]. Fig. 1 gives an illustration of four main pre-planned multicast fault-tolerance schemes. In the figure, S is the source node, and shaded nodes are destinations. Solid thick lines represent links used in the primary tree and dashed lines represent links in the backup paths. Different types of arrows represent the directions of the links used in the primary tree or the activated backup paths, and all links are assumed to be bidirectional.

Wu et al. [14] study the link protection and path protection approaches. In link protection, for each link, a backup route is set up between the two end nodes. In path-protection, for each destination, a path vertex-disjoint with the path in the multicast tree from the source to that destination is set up as backup. These two types of protection schemes are illustrated in Fig. 1(a) and Fig. 1(b) separately. In Fig. 1(a), to protect the link $(1 \rightarrow 4)$, a backup path $(1 \rightarrow 3 \rightarrow 6 \rightarrow 4)$ is established. While in path protection (shown in Fig. 1(b)), a backup path $(S \rightarrow 9 \rightarrow 6)$ is activated when link $(1 \rightarrow 4)$ is down, since the primary path from S to 6 (that is path $(S \rightarrow 1 \rightarrow 4 \rightarrow 6)$ is broken. A modified link protection is also proposed in [14], in which a backup path that protects link $(u \rightarrow v)$ can originate from u's ancestor nodes or sibling nodes. For example, in Fig. 1(a), when link $(1 \rightarrow 4)$ is down, a backup path $(3 \rightarrow 6 \rightarrow 4)$ is activated instead of the path $(1 \rightarrow 3 \rightarrow 6 \rightarrow 4)$, since 3 is a sibling node of 4, and group data reaching 3 can be delivered directly to 4 through path $(3 \rightarrow 6 \rightarrow 4)$. In [14], the authors also studied backup capacity sharing strategies to reduce backup capacity requirement and save network resource usage. It should be noted that, in the link/path protection schemes, usually a single link failure is assumed.

In a more recent approach, Fei et al. [4] propose a "dual tree" scheme. Instead of protecting each link or member individually in the multicast tree, a secondary tree is built among a subset of multicast members for the purpose of fault-tolerance. Dual-tree scheme requires the underlying network topology is a bi-connected graph, in which there are at least two vertex-disjoint or link-disjoint paths between any two nodes. If the secondary tree is only link-disjoint with the primary tree, then it is a link-disjoint dual-tree, otherwise, it is called a node-disjoint dual tree. An example of node-disjoint dual-tree structure is shown in Fig. 1(c). If there is a failure, for example, assuming link $(2 \rightarrow 5)$ is down, one of affected primary leaves (node 7 and 8 in our example) needs to be connected to a non-affected node (node 6) through some path on the dual tree ($(6 \rightarrow 9 \rightarrow 7)$ in our example). Once the reconfiguration message (sent from node 5 to 6) reaches the unaffected node (node 6), it activates the backup path on the dual tree and starts delivering packets to the affected leaves. Depending on the properties of the underlying network topology, dual-tree scheme can protect against both single link and node failures. However, to do so it requires node-disjoint dual-tree which has stronger requirement on the connectivity of the graph than link-disjoint dual-tree. It is also worth pointing out that, compared with link and path protection approaches, dual-tree scheme does not require per-link or per-path fault-tolerance management.

Fig. 1(d) shows a simple example of redundant tree protection scheme. A node-disjoint redundant tree is created to protect any link/node failure in the primary tree. [8] proposed

[2]Although MPLS is a natural match for our approach, it could be implemented by other mechanisms. In the Internet at large, it could be implemented by packet encapsulation and the existing multicast routing algorithms [6].

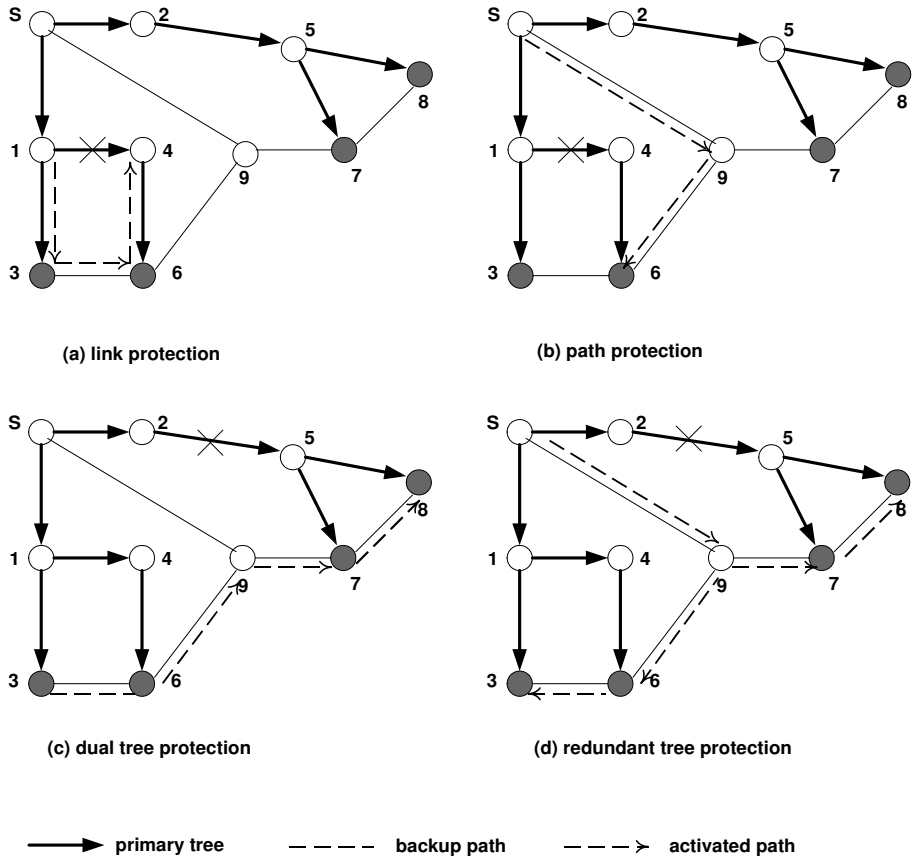

Fig. 1. Illustration of four main pre-planned multicast fault-tolerance schemes. (a) link protection; (b) path protection; (c) dual-tree protection; (d) redundant tree protection.

algorithms to create redundant trees for pre-planned recovery in arbitrary vertex-redundant or edge-redundant graphs. In [8], the proposed algorithm can compute two trees in such a fashion that the elimination of any vertex (edge) in the graph leaves each destination vertex to the source by at least one of the directed tree. Note that redundant tree protection scheme can protect more than one failure, however, it does require more connectivity of the network graph than dual-tree and link/path protection schemes.

Table I gives a high-level comparison of the four main protection schemes we discussed above. In paper [4], Fei et al. conducted a performance comparison in terms of failure restoration time and tree cost increase after failure restoration between link/path protections and dual-tree scheme. The simulated results in random network graphs show that dual-tree scheme performs better than other schemes.

Qualitative comparison of overhead. Methods that require per link or per path protection incur a high overhead and will not scale easily to large networks or many groups. In Table I, one can notice that, for fault tolerance management, all the above pre-planned schemes we have discussed are at least per group based. In other words, they compute backup routes for links, paths, or trees for each individual groups. When network failure is detected, the backup routes need to be activated for individual links, paths, or trees too. Thus when there are a large number of simultaneous groups in the network, the cost of restoration becomes high since this includes backup route computation cost and recovery overhead (introduced by message exchange during backup route activation). For link and path protection schemes, the case is even worse, since in a group, the backup routes of each links or paths are computed and maintained individually. This scalability issue is one of the main concerns of this work.

B. Best Effort versus MPLS Fault-Tolerance

Following the best-effort mentality, the Internet typically follows on-demand approach to failure recovery. The restoration is achieved through routing table update. Pre-planned restoration involves setting up backup paths and activating backup paths when a failure is detected, however, inherently, IP network does not provide mechanisms to support these procedures efficiently. Then, a natural question arises: how can we provide pre-planned multicast fault-tolerance in the Internet? The concept of virtual circuit packet switching with IP has appeared as the answer.

Virtual circuit packet switching technologies that have been used in the Internet backbone are Asynchronous Transfer Mode (ATM) [1], and, more recently, Multiple Protocol La-

TABLE I

A COMPARISON OF PRE-PLANNED MULTICAST FAULT-TOLERANCE SCHEMES

Scheme Name	*#. of Failures to Protect*	*Network Connectivity Requirement*	*Fault Tolerance Management*
Link Protection	single link failure	no strong requirement	per-link per group
Path Protection	single link failure	no strong requirement	per-path per group
Dual-tree Protection	single link/node failure	bi-connected graph	per group
Redundant-tree Protection	multiple link/node failures	vertex/edge-redundant graph	per group

bel Switching (MPLS) [13]. ATM is a technology which was standardized in the late 1980s. However, sending IP traffic over ATM proves to be very complex. On the other hand, MPLS, compared with ATM, has been devised specifically to interface better with IP. Thus, more and more Internet backbones employ MPLS technology.

MPLS combines the advantages of datagram packet switching and virtual circuit switching. In an MPLS domain, when a stream of data traverses a common path, a Label Switched Path (LSP) can be established using MPLS signaling protocols. At the ingress Label Switch Router (LSR), each IP packet is assigned a label (by inserting a "shim" MPLS header) and is transmitted downstream. At each LSR along the LSP, the label is used to forward the packet to the next hop. At the egress LSR, each packet pops out the label, and continues to be distributed into IP networks. To deliver multicast traffic, MPLS trees (that is, "labelled" multicast trees) need to be established. MPLS trees can be either mapped directly from level 3, namely IP level [10], or established by explicit routing [11][3].

MPLS multicast fault-tolerance has also attracted much attention recently. A seminal project, called MPLS Multicast Fast Reroute, has been conducted in the Multimedia Networks Group, University of Virginia [12]. The pre-planned scheme used in this project can be categorized into link-protection approaches, while the backup path selection algorithm is different from previous link-protection schemes. For a protected link (assuming bidirectional), a backup path is chosen among the shortest paths between any two nodes on the multicast tree so that the number of group members dropped upon the link failure is minimized. A protocol extending (unicast) MPLS fast reroute [7] is also designed and implemented.

Due to employing virtual circuit switching techniques, MPLS facilitates rerouting significantly. For this reason, we show how we would develop our architecture in network environments that support MPLS.

III. KEY CONCEPTS OF OUR APPROACH

Considering our design goals (that is, scalability, efficiency, and fast recovery) for fault-tolerant multicast, we are inspired by the idea of aggregated multicast [5]. We first briefly present the concept of aggregated multicast, and then high-light how it facilitates multicast fault-tolerance.

[3]Note that, later in this article, AMFM is illustrated with explicit MPLS multicast routing.

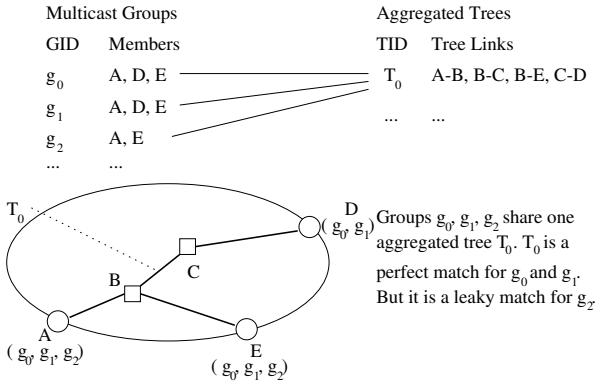

Fig. 2. Illustration of aggregated multicast

A. Aggregated Multicast

Aggregated multicast is targeted as an intra-domain multicast provisioning mechanism. The key idea is to "force" several multicast groups to share a single distribution tree. Data packets from different groups are multiplexed on the same distribution tree, which we refer to as **aggregated tree**. As a result, routers in the middle of the network, which we call **core** routers, need to keep state only per aggregated tree. Thus, there is a smaller number of trees, for which we need to maintain state in the core network.

The "mapping" of groups to trees happens at the ingress and egress routers. The data packets of each group are labelled or encapsulated so that they will travel on the same tree. MPLS lends itself naturally to this task. The edge routers of the network need to maintain sufficient information to multiplex and de-multiplex groups to and from aggregated trees. Fig. 2 illustrates the basic idea of aggregated multicast. Multicast groups g_0, g_1 and g_2 use the same tree. Nodes A, E, and D connect to members of group g_0 and g_1. Group g_2 has group members only at nodes A and E, but g_2 packets reach node D inevitably.

The group-to-tree matching problem hides several subtleties. How do we match a group to a tree? A group specifies a set of member nodes, and these nodes must be reached by the tree. In this case, we say that the tree **covers** the group. Furthermore, a match is called **perfect** for a group, if all the tree leaves correspond to group members. In the previous example, group g_0 and g_1 are perfect matches for the tree they use. In contrast, a match is called **leaky**, if there are leaves of the tree that do not have group members. Group g_2 is a leaky match for its current aggregated tree. Clearly, a leaky match wastes

bandwidth by delivering unwanted packets to some parts of the network, towards node D in our example. The advantage of the leaky match is that it increases our ability to aggregate groups. With leaky matches, we can trade off bandwidth utilization for higher aggregation.

To control this trade off, we define a threshold of bandwidth waste that any group is allowed to cause. More specifically, we define the **bandwidth waste threshold** (denoted as *bt* for short) as the ratio of the additional bandwidth that the aggregation uses over the bandwidth that the group would have used without the aggregation.

B. How Aggregated Multicast Facilitates Fault-Tolerance

Aggregated multicast reduces the number of trees we need to set up and maintain in the core of the network: the number of aggregated trees is significantly less than the number of multicast groups. As a result, the backup tree computational and maintenance cost is greatly reduced. In addition, there is less communication overhead associated with a failure, since the recovery is related to the number of aggregated trees rather than individual multicast groups. Note that the aggregated multicast reduces the required routing state in the core routers of the network, which makes the packet lookup faster. Another big advantage is that we map groups to trees on a temporary basis, with explicit mapping at the ingress and egress points of the network. Once a failure occurs, the affected groups are re-mapped to their backup trees, and this re-mapping can be done very quickly and efficiently since we only need to change the mapping entries at the edge of the network. The bottom line is that the aggregated multicast improves the scalability, efficiency, and latency of the failure recovery.

Our proposed architecture AMFM employs the concept of aggregated multicast. Aggregated multicast was initially designed as a state-reduction scheme, but here, it becomes a powerful tool to facilitate fault-tolerance.

IV. Structure and Functions of AMFM

We present in more detail the architecture of Aggregated MPLS-based Fault-Tolerant Multicast or AMFM. AMFM targets multicast provisioning in a single MPLS-enabled domain, particularly backbone domains. MPLS is the mechanism which enables us to "multiplex" packets of different groups on the same aggregated tree.

In a nutshell, AMFM maintains MPLS aggregated trees and their corresponding backup trees. A multicast group is assigned to an aggregated tree after examining how well the tree matches the group. For each aggregated tree, a backup tree is computed in case of failure. The backup tree has no common edges with the primary tree[4]. If the primary tree fails, all its groups are switched to the backup tree.

[4]Depending on the network, we can not always find edge disjoint trees. In such cases, we compromise and choose the tree that has the least number of common edges with the primary tree.

A. Implementing AMFM

Let us take a closer look at the entities and functions of our scheme.

We introduce a logical entity, called *tree manager*, which is responsible for mapping groups to aggregated trees, managing the aggregate trees and their backups. The tree manager needs to have information of the network topology, the group membership, the aggregate trees, backup trees, group-tree matching table, and backup mapping table. Note two important things. First, the tree manager needs to keep information only for the groups and trees it manages. Second, it can be implemented in either centralized or a distributed way [5]. For simplicity of presentation, we can think of the tree manager as a single node.

The tree manager provides the following functions that we consider as separate modules: group-tree matching, routing, failure recovery, and policy control. The routing module peers with routers to obtain the topology information of the network domain, and is responsible for establishing new aggregated trees, computing backup trees, and tearing down obsolete aggregated trees. The group-tree matching module matches a multicast group to the appropriate existing tree or requests a new tree. The failure recovery module is responsible for switching groups from a failed tree to a backup tree. The policy control module enforces additional policy issues such as call admission, and QoS considerations. Due to space limitation, this article will focus on the routing, group-tree matching, and failure recovery modules for AMFM.

Before explaining design issues in more detail, we provide an overview of the AMFM scheme. An illustration is depicted in Fig. 3, where A, D, and E are edge routers, and B, C, F, G and H are core routers.

In Fig. 3(a), given a new multicast group g with members in edge routers A, E, and D (assuming A is the source, and E and D are the receivers), AMFM calls the group-tree matching module of the tree manager and tries to find a match with an established aggregated tree. If no such tree exists, the tree manager computes a new multicast tree according to membership through the routing module. At the same time, it computes a backup tree of the new multicast tree and inserts it into the backup tree table. Fig. 3(a) shows that after the tree manager is consulted, a primary tree for group g is computed (with links of A-B, B-E, B-C, C-D, which are marked with thick lines) and a backup tree is computed (with links of A-F, F-G, G-D, A-H, H-E, which are marked as thin lines). Once a new multicast tree is computed, the corresponding MPLS tree is established through a Label Distribution Protocol (LDP). After a multicast tree is found or established, the tree manager distributes the corresponding group-tree matching information to the edge routers of the tree, as is illustrated in Fig. 3(b). Then data transmission can be started: the ingress edge routers encapsulate the group packets that arrive, and the egress edge routers decapsulate the packets that leave the network appropriately (as shown in Fig. 3(c)). When a failure occurs (link B-C is down in the example of Fig. 3(d)), the tree manager first detects which aggregated trees are affected (the primary tree with links of A-B,

B-E, B-C, and C-D has fault), and then it invokes the failure recovery module which activates the corresponding backup trees (the backup tree with links of A-F, F-G, G-D, A-H and H-E in our example) and switches the affected groups (g)to the backup trees.

AMFM and Network Information. AMFM is a fault-tolerance scheme that attempts to optimize resource utilization and reliability. Note that AMFM is not a routing or monitoring protocol. AMFM interacts with the existing protocols of the network such as the unicast and multicast routing protocols, the membership information, and any available statistical information on utilization and performance. The more accurate information a network can provide, the better AMFM can perform. Let us re-iterate that AMFM targets the networks which are managed to obtain such information in order to optimize resource utilization.

Depending on the nature of the network, AMFM can have different levels of information. For example, in a link state network, the tree manager has the view of the whole topology and its capabilities are increased. In addition, group membership can be sent directly to the tree manager or be piggybacked on link-state packets if unicast routing uses a link state approach. Naturally, if we can not collect the necessary information, AMFM may operate sub-optimally or not at all.

In the following, we discuss in more detail some of the functions of AMFM.

A.1 Multicast Routing

The routing module can employ the available multicast routing algorithms to compute an aggregated MPLS tree. This can be done pro-actively, or in response to a new multicast group which cannot be supported by an existing aggregated tree. AMFM can use almost any multicast routing algorithm, such as shortest path tree algorithm (as in MOSPF [9]) and core-based tree algorithm (as in CBT [2] or PIM-SM [3]). AMFM can be based on either unidirectional or bidirectional trees. However, we find that bidirectional trees are a better choice, since this way we can support groups where all receivers can be senders as well for applications such as teleconferencing. This can help reduce the number of required aggregate trees: the same tree can support groups that have the same members even if in each group the source is different. By contrast, in unidirectional trees, a group and an aggregate tree must match in both the members and the source, i.e., in the direction in which the information will flow. It should be noted that no full-fledged multicast routing protocols are needed in AMFM, since a new aggregated tree is computed first in the routing module, then an explicit MPLS multicast routing protocol is used to establish the corresponding MPLS tree (as will be detailed more in MPLS tree management section).

A.2 Backup Tree Computation

Once a new aggregated tree is computed and established, the tree manager computes a backup tree in the routing module. The backup tree computation can use any redundant tree fault-tolerance scheme, such as the algorithm described in [8]. However, this redundant tree algorithm results in unidirectional trees upon a link or vertex failure. A simple way to compute a bidirectional backup tree is to use existing multicast routing algorithms to find a new tree based on the members (that is, source and destination nodes) of the primary tree, while avoiding the links of the primary aggregated tree, if possible.

A.3 Group-Tree Matching Algorithm

To match a group to an aggregated tree, the tree manager needs to maintain tables for establishing aggregated trees, active multicast groups, and group-tree matching entries.

First, we introduce some notation. Let us denote as A the available multicast routing algorithm. Given a group g, the algorithm A would compute a tree that we denote by $T_A(g)$. Using AMFM, the group g may be routed on the aggregated tree $T(g)$. As mentioned in the introduction of aggregated multicast, it is possible that $T(g)$ does not have a perfect match with group g, which means that some of the leaf nodes of $T(g)$ are not member nodes of g, and then packets reach some destinations that are not interested in receiving them. Recall that bt is the bandwidth overhead threshold, whose value is set by the network manager. We will denote as $cost(T)$, the cost of the tree T in terms of bandwidth.

Given a new multicast group g, the tree manager invokes the group-tree matching module, which does the following. If it can find an aggregated tree that can support the new group **with less** bandwidth waste than the threshold, it will use this tree. If there are multiple such trees, it will pick the one with the minimum bandwidth overhead. If no such tree can be found, a new aggregated tree is established for the new group.

In more detail, the process of finding a new tree has the following steps.

(1) Compute a multicast tree $T_A(g)$ for g (without considering aggregation) and calculate its cost;

(2) For each established aggregated tree T, if T covers g, compute the bandwidth overhead. If the bandwidth overhead is less than given threshold, that is, $1 - cost(T)/cost(T_A(g)) < bt$, then tree T is considered as a candidate to cover g;

(3) Among all candidate trees, choose the one with minimum bandwidth overhead T_m and use it cover g;

(4) If no candidate tree is found in step (2), use the $T_A(g)$ tree to cover g.

A.4 Failure Recovery

When a failure occurs, the tree manager invokes the failure recovery module, which first detects which aggregated trees are affected. The recovery module retrieves the backup trees of the failed trees, and switches the related multicast groups to the backup trees. Note that this tree-switching can be done very quickly and efficiently since we only need to change the group-tree matching entries at the edge of the network.

The tree manager can detect the failure in different ways depending on the network. First, the tree manager can cooperate with any available network monitoring facilities which

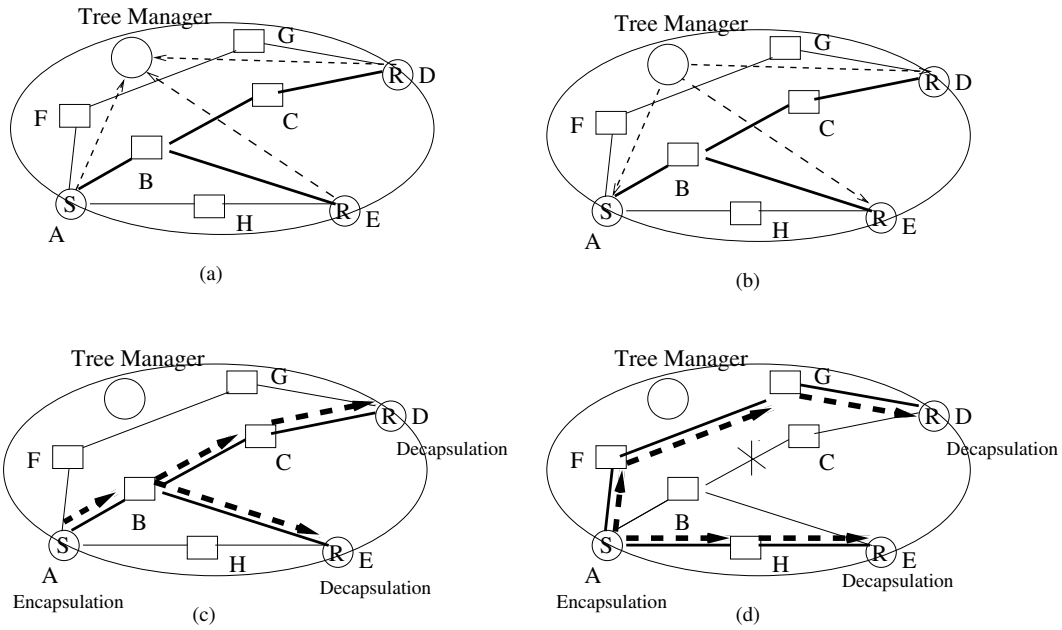

Fig. 3. The overview of AMFM: (a) Membership collection; (b) Group-tree matching entry distribution; (c) Multicast packets transmission on aggregated tree; (d) Failure recovery.

most commercial networks have. Second, it can actively or passively monitor the network itself. Third, it can consult the edge routers and use tools such as Real-Time Control Protocol (RTCP) to monitor the health of the multicast tree within its network.

The backup trees can exist in two ways: they can be established (by explicit MPLS routing protocol) or they can just be computed. If they are established, the routers have the related entries in their routing tables. This way, when the failure occurs we only need to switch the labelling of the incoming packets at the edge routers. This, however, means that routers maintain the extra routing state even when the tree is not used. Alternatively, the backup trees may have been computed at the tree manager, but they are set up only (after tree switching is conducted) when the failure occurs. This introduces some delay in the recovery.

A.5 MPLS Tree Management

After a new multicast tree is computed, its corresponding MPLS tree needs to be established. As we mentioned earlier, AMFM uses explicit MPLS routing. In the literature, there exist solutions to distribute labels for multicast trees, such as [11]. However, this protocol is designed for unidirectional tree. Note that AMFM suggests bidirectional trees. Thus, we need to design a new MPLS routing protocol, that is, an LDP (Label Distribution Protocol) for establishing bidirectional multicast trees.

MPLS tree setup. We have two kinds of solutions for bi-directional MPLS tree setup: one is centralized, and the other is distributed. In the centralized solution, the tree manager generates all the MPLS labels for the bi-directional tree and then distributes them to the corresponding routers directly. In this approach, the label space for aggregated bi-directional multicast trees should be separated from the space for other type of labels (e.g. unicast paths), because all the labels associated with different FECs (Forwarding Equivalency Classes) should be unique among all the interfaces in the entire LSR (Label Switch Router). In addition, the tree manager has to keep all the assigned labels for existing aggregated bi-directional multicast trees. After a multicast tree is established, the tree manager needs to update the labelling database for all related label switching edge routers.

Alternatively, we could use a distributed approach. We can extend the existing unidirectional MPLS tree setup schemes [11]: root-initiated or leaf-initiated. A bi-directional tree can be viewed as a combination of n unidirectional trees, where n is the number of the leaf routers in the bi-directional tree. Each unidirectional tree has a leaf router of the bi-directional tree as its "root". Since the whole bi-directional tree is available, it is not difficult to create unidirectional tree objects. Thus, the tree manager can send the n unidirectional tree objects to the corresponding "root" routers. Then each "root" router uses root-initiated unidirectional MPLS tree setup approach. Same as in centralized method, each edge router should be configured as an ingress/egress LSR. The leaf-initiated approach can be used similarly. More details about the root-initiated approach and the leaf-initiated approach can be found in [11].

Tearing down aggregated trees. When an MPLS tree becomes obsolete, the tree manager will tear it down. Depending on what kind of approach is used for MPLS tree setup, the tree manager sends label withdraw messages to all the in-tree routers of the aggregated multicast tree if the centralized approach is employed; or, if we adopt the distributed ap-

proach, the tree manager only notifies the leaf routers of the bi-directional multicast tree, and each leaf router invokes a tear down procedure.

V. Performance Study

We conduct a series of experiments to study the performance of AMFM. We examine to what extent the performance of multicast fault tolerance is improved in terms of computational and maintenance cost and recovery overhead.

In our experiments, we compare AMFM to MPLS multicast with redundant tree (which will be referred to redundant tree MPLS multicast, or **R-MPLS** for short). In R-MPLS, each group is routed on a separate MPLS multicast tree, and each tree has its own redundant tree as a backup. This is a very straightforward way to do fault-tolerance for MPLS multicast, which we use as a reference point. Note that we use the same multicast routing algorithm and backup tree computation approach for R-MPLS and AMFM in order to make the comparison fair. We define the following two metrics.

Backup-Tree Reduction Ratio (BTRR). The backup tree computational and maintenance cost can be measured by the number of backup trees. Hence, we define BTRR as follows:

$$BTRR = 1 - \frac{\#\ of\ multicast\ trees\ of\ AMFM}{\#\ of\ multicast\ trees\ of\ \text{R-MPLS}}. \quad (1)$$

Recovery Overhead Reduction Ratio (RORR) The number of recovery messages (we count a backup tree activation message traversed on one link as one recovery message) is a measurement of failure recovery overhead. Then, RORR is defined as follows:

$$RORR = 1 - \frac{\#\ of\ recovery\ messages\ of\ AMFM}{\#\ of\ recovery\ messages\ of\ \text{R-MPLS}}. \quad (2)$$

It should be noted that state reduction is another important feature of AMFM, since the routing state in the core routers of the network is reduced, as in turn makes the packet lookup faster. State reduction is already studied in our previous work [5], thus we omit the results of state reduction in this article.

A. Simulation Environment

In our simulations, we use a network abstracted from a real network topology, the Abilene backbone, which is one of the core networks of Internet-2. This abstracted network has 12 core routers, and each is attached to an edge router.

In the target network, we assign nodes with different weights, which represent their probabilities to participate in the multicast group. Core routers are assigned weight 0, since they will not be members for any multicast group. Any other edge router is assigned a weight 0.2 to 0.8 according to the real-time traffic on its links connected to the corresponding core routers. The rationale behind this is, for a node, more traffic means more participation in the network communication, it has higher probability to join a multicast group. We also assume multicast group requests arrive as a Poisson process with arrival rate λ, and groups' life time has an exponential distribution with average μ^{-1}. Then, at steady state, the average number of groups is $\bar{N} = \lambda/\mu$. In our simulation experiments, we fix the group average life time as $100s$, and change the group arrival rate in order to get different number of groups. We run the simulations for $1000s$, and collect performance data after steady state is reached (after $400s$ in our scenario). In our experiments, due to the connectivity of the target network, backup trees (for both AMFM and R-MPLS) are edge-redundant trees instead of node-redundant tree to the primary trees. When we measure RORR, we generate link failures uniformly distributed in the set of network links.

B. Experiments and Results

In our experiments, we vary the bandwidth waste threshold (denoted as *bt*) from 0 to 0.3 for AMFM. In each network scenario, i.e., R-MPLS or AMFM with different bandwidth waste threshold, we change the average number of multicast groups and measure the proposed two metrics: BTRR and RORR.

AMFM reduces significantly the number of backup trees. Fig. 4 shows the results for BTRR (Backup Tree Reduction Ratio) vs the average number of concurrent active groups. For all the curves in Fig. 4, we can see that the backup tree reduction ratio increases when the number of groups grows: the more groups become active, the more backup tree reduction we expect. For example, for the curve with bandwidth waste threshold *bt* as 0, when the number of groups grows from 500 to 3000, the backup tree reduction ratio is increased from 0.06 to 0.15. This agrees with our intuition: as more groups are pumped into the network, more groups can share one aggregated tree, and thus smaller number of aggregated trees are needed, and correspondingly, smaller number of backup trees are computed. From Fig. 4, we also observe that, when the bandwidth waste threshold *bt* is increased, more backup tree reduction is achieved, as verifies that the trade-off between the gain of aggregated multicast and the bandwidth waste: when we are willing to waste more bandwidth, we can have more aggregation (i.e., more groups share one aggregated tree, and thus much smaller number of aggregated trees and backup trees need to be computed and maintained). In AMFM, even a very small amount of bandwidth waste, say, $bt = 0.05$, can result in very high backup tree reduction ratio, 0.65 when there are 3000 groups.

AMFM reduces significantly the recovery overhead. Fig. 5 plots the results for RORR (Recovery Overhead Reduction Ratio) vs the average number of concurrent active groups. It is interesting to notice that this figure looks very similar to Fig. 4. At first, this seems surprising, since the metrics refer to different quantities. However, this can be explained, since this metric is closely related to the number of backup trees: the number of links traversed by recovery messages is proportional to the number of backup trees times the average number of nodes in a tree.

In summary, our experiments have shown that AMFM can improve the performance of multicast fault tolerance signif-

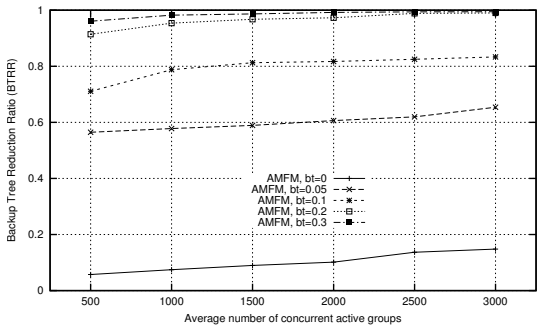

Fig. 4. Backup Tree Reduction Ratio (BTRR) vs average number of groups. *bt* denotes bandwidth waste threshold.

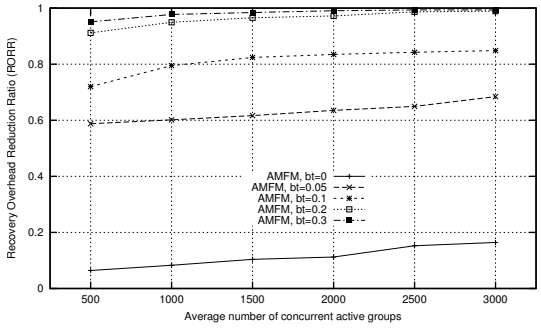

Fig. 5. Recovery Overhead Reduction Ratio (RORR) vs average number of groups. *bt* denotes bandwidth waste threshold

icantly in terms of computational and maintenance cost and recovery overhead (measured by the proposed two metrics and state reduction as is studied in our earlier work). In addition, a trade-off between the gain and overhead is also demonstrated, which can be one of the important policy concerns of AMFM network manager.

VI. Discussion

We discuss how we can extend AMFM to support QoS and perform load balancing. So far, we have made the implicit assumption that aggregating as many trees as possible is good. In other words, we do not consider issues arising from tree-overloading. Our scheme can easily be extended to incorporate multiple QoS and load balancing criteria in the group-tree matching.

Tree capacity and application requirements. If we assume that we know the tree capacity and the traffic profile of the multicast group, we can do more performance aware routing. The tree capacity or the tree bottleneck can be obtained from interaction with the routing protocol and routing database or it can be estimated with monitoring and probing methods.

Load balancing. By aggregating the groups on the same tree, we minimize the routing state, but we overload the links of the tree. The simplest solution to this issue is to have an upper bound on the number of groups that can share a tree. However, this does not consider the possibly different traffic intensity of each group. If we have tree and group specific information we can make intelligent decisions to reach an acceptable level of traffic on a tree. This can be added as an additional filter in selecting trees for a given group: *a group is matched to a tree only if the expected group load does not exceed the capabilities of the tree.* Furthermore, given the choice, we can assign the new group to the least congested of the candidate trees.

QoS aware tree selection. Assuming information on the tree and application, we can pick trees in a way that conforms to QoS requirements. These requirements can reflect bandwidth, loss expectations and end-to-end performance metrics such as end-to-end delay. Based on the information of QoS requirement of the group and resource utilization in the network, group-tree matching algorithm can be devised to select a optimal tree for the group.

VII. Conclusions

In this article, we provide an overview of the most prominent schemes for fault-tolerant multicasting. We identify the weaknesses of the previous schemes and propose a novel architecture, AMFM, for efficient and fast fault-tolerant multicast provisioning. The idea is based on the aggregated multicast concept and it is naturally suited for an MPLS environment.

In AMFM, we separate the concept of the tree from the multicast group: the tree becomes a routing abstraction, while the multicast group is a communication abstraction. Groups are mapped to trees on a temporary basis and with an explicit mapping at the ingress and egress points of the network. In case of routing failures, the group is re-mapped to a new tree. This re-mapping can be done quickly and efficiently, since we basically need to change the mapping at the boundaries of the network. In addition, we can map multiple groups to one tree, which reduces the required routing state inside the network, and also reduces the number of backup trees needed to set up and maintain.

Using simulations, we see that AMFM reduces the protection cost and recovery overhead significantly, and it scales well to large numbers of groups. We observe that a small amount of wasted bandwidth in leaky matching can lead to significant gains in reduction of multicasting overhead. In our simulations on a real topology, we observe that 10% of wasted bandwidth can lead to almost 80% reduction in the number of backup trees we need to maintain.

In conclusion, the proposed approach shows great promise in providing fast and scalable fault-tolerance for multicasting. Furthermore, our approach has the potential to become the foundation for a scalable multicast management architecture.

For the future, we want to integrate QoS and load balancing considerations in our fault-tolerance scheme to provide a comprehensive tree management architecture. We believe that such an architecture will be necessary to meet the requirements of future networks for high performance and reliability.

References

[1] The ATM forum. *http://www.atmforum.com/*.
[2] A. Ballardie. Core Based Trees (CBT version 2) multicast routing: protocol specification. *IETF RFC 2189*, Sept. 1997.

[3] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. *IETF RFC 2362*, June 1998.

[4] A. Fei, J.-H. Cui, M. Gerla, and D. Cavendish. A "Dual-Tree" Scheme for Fault-Tolerant Multicast. In *Proceedings of IEEE ICC'01*, Helsinki, Finland, June 2001.

[5] A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast: an approach to reduce multicast state. *Proceedings of Sixth Global Internet Symposium(GI2001)*, Nov. 2001.

[6] A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast with inter-group tree sharing. *Proceedings of NGC2001*, Nov. 2001.

[7] D. Haskin and R. Krishnan. A method for setting an alternative label switched paths to handle fast reroute. *Internet Draft*, Nov. 2000.

[8] M. Medard, S. Finn, R. Barry, and R. Gallager. Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. *IEEE/ACM Transactions on Networking*, 7(5):641 –652, Oct. 1999.

[9] J. Moy. Multicast routing extensions to OSPF. *RFC 1584*, Mar. 1994.

[10] D. Ooms and et al. Framework for IP-multicast in MPLS. *Internet draft: draft-ietf-mpls-multicast-05.txt*, 2001.

[11] D. Ooms, R. Hoebeke, P. Cheval, and L. Wu. MPLS multicast traffic engineering. *Internet draft: draft-ooms-mpls-multicast-te-00.txt*, 2001.

[12] Y. Pointurier. Link Failure Recovery for MPLS Networks with Multicasting. *Master's Thesis, University of Virginia, Department of Computer Science*, Aug. 2002.

[13] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching architecture. *IETF RFC 3031*, 2001.

[14] C. Wu, W. Lee, and Y. Hou. Back-up VP preplanning strategies for survivable multicast ATM networks. In *Proceedings of IEEE ICC'97*, pages 267–271, June 1997.