A Fault-Tolerant Scheme for Multicast Communication Protocols

Bhed Bahadur Bista
Faculty of Software and Information Science,
Iwate Prefectural University,
Iwate-ken, JAPAN 020-0193

email: bbb@soft.iwate-pu.ac.jp

Abstract:- Since the multicast communication is the best technology to provide one to many communication, more and more service providers are using this technology to deliver the same service to multiple customers. As such, providing fault tolerance to multicast connections is gaining attention both in business and research communities because a single link or a node failure in the multicast communication delivery tree will affect a large number of customers. There are some existing schemes proposed for fault recovery in the multicast communication. They either calculate a new tree without using any node from the existing tree or calculate a path from affected node/tree to the unaffected tree when a fault occurs. In either case, they need the global view of the multicast communication tree. In this paper, we propose a fault tolerant scheme in which we do not need the global view of the multicast tree. We compute the shortest path from a node to the source of the multicast tree assuming that the node's link to its parent node in the multicast tree is broken. The shortest path information is sent hop-by-hop toward the source and is stored in the routers. When the assumed broken link really breaks the recovery message is sent toward the source and the previously stored fault recovery message at each node is used to make a multicast recovery tree.

Key-Words:- fault-tolerance, multicast, preplanned, on-demand, tree generation.

1 Introduction

Multicast communication is the only efficient and scalable technology available for one-to-many and many-to-may communication. Several protocols and algorithms such as DVMRP [1], PIM-DM [2], PIM-SM [3], CBT [4] and MOSPF [5] have been developed and implemented for multicast communications. These protocols and algorithms create so called multicast trees which are rooted at the source of the multicast data (or node which is selected as the root of the tree). The data packets are then forwarded from the source along the trees to the multiple receivers. Due to the improvement of computational power of computers and network devices multicast applications such as video-conference, white-board, web radio, stock online and so on are being developed and are in use also. However researchers and business communities are worried about the fault-tolerance of network running multicast applications because unlike a unicast communication, a link failure in a multicast communication affects numerous receivers at the same time.

There are a few works done in fault-tolerance of multicast communication. They can be divided into two categories; *preplanned* [6, 7, 8] and *on-demand* [9]. In preplanned, an alternative tree is computed corresponding to each node or link that is assumed to be failed. When a node or a link fails the corresponding pre-calculated tree is then activated for the multicast data flow and the data flows along the new tree to the receivers. In on-demand scheme, no alternative trees are calculated before hand. When a node or a link fails, the child/s and the grandfather node of the failed node reactivate their agents. The agent at the grandfather node encapsulates packets and sends to the agent in the child node/s which decapsulates the packets and forwards along the multicast tree. The agents bypass the failed nodes.

In the above preplanned schemes, a node monitors the whole network for faults. If it finds any fault, it activates the new tree for the multicast data packets. Monitoring whole network incurs delays and also consumes large network bandwidth as the monitoring packets should flow in whole networks periodically. Besides, how the activation of the new tree would match with the multicast protocol being used is not clearly mentioned. In the on-demand scheme, the encapsulation and decapsulation of a large number of packets at the beginning incurs large processing time.

In this paper, we propose a scheme in which each child router calculates the shortest path from itself to the source of the multicast data excluding the link to its parent node. It is assumed that a failure has occurred to the link to its parent node. Then it sends the tree generation message to the routers on the shortest path. The message travels hop-by-hop to the source in the worst case (i.e. when no routers on the shortest path are members of the active tree) and in the best case travels to the router which is already on the active tree (i.e. there are routers on the shortest path which are members of the active tree). Each router on the shortest path to the source of the multicast data keeps tree generation information in the cache entry. When a failure occurs, the child node of the failed node or link checks its cache entry and sends tree activate message toward the source. The message travels hop-by-hop and the routers on the way to the source generate tree from the information stored in the cache entry and multicast data will start flowing along the recovered tree.

The paper is organized as follows. In the section two, we give the assumption and the detail of how cache entry at each node are generated and how the cache entry is used to recover the fault. In the section three, we discuss the advantages and disadvantages of our proposed scheme and finally in section four, we conclude the paper.

2 Fault-Tolerant Planning and Fault Restoration

2.1 Assumptions

We assume that links in the network are bidirectional. We also assume that the multicast delivery tree can be either source based or shared based tree. Like in the most of the previous works, we assume that there is only one link failure in active multicast trees.

2.2 Fault Tolerant Planning

When a node becomes a member of a multicast tree it calculates the shortest path from itself to the root of the multicast delivery tree by excluding the link to its parent node because it is assumed that the link to its parent node is supposed to be failed. For shared tree multicast delivery tree the root will be the Rendezvous Point (RP) (For example for PIM-SM and CBT), and for source based tree it will be the source network of the multicast data. If the shortest path doesn't exist, then the network will be partitioned for which we can do nothing about it.

The node then sends *fault-tolerant message* in the form of **(group list, node list, source)** to the next node in the **node list**. The **group list** contains the multicast groups to which the **source** is sending data and the node is forwarding them. The **node list** contains the nodes in the shortest path. The node stores the message in the form of **(group list, incoming interface, outgoing interface list, source list)** in its fault-tolerant cache in order to generate reactive message for fault restoration and also to restore the fault. The **group list** is the list of multicast groups for which it is forwarding multicast packets down the tree. The **incoming interface** is the incoming interface of the multicast data for the groups in the **group list** when the fault is restored. The **outgoing interface list** is the outgoing interfaces of the multicast data. The **source list** is the list of multicast sources. Only the information that is not in the active tree is stored in the fault-tolerant cache

When the next node in the **node list** receives the fault-tolerant message, it checks from which interface it has received the message and puts the interface in its **outgoing interface list** of the fault-tolerant cache. It checks the next node (after removing itself) in the **node list** and the interface to the node and puts the interface in its **incoming interface** of the fault-tolerant cache. It puts the group list and source in its **(group list and source list)** respectively of the fault-tolerant cache. It removes itself from the node list and sends the message to the node which is on the top of the **node list**.

The fault-tolerant message will not be forwarded toward the source in either of the following cases:

case 1 The multicast groups, source of the multicast groups and incoming interface are already in the fault-tolerant cache entry.

case 2 The **node list** becomes Null after removing oneself from the **node list**.

The case 1 states that if nodes X, Y, Z are nodes in the active tree and X finds the shortest path to the source as X→Y→Z→ ... → S, Y does not need to send (calculate shortest path also) for the same group list and source list for which it has received the fault-tolerant message from X because it will not make any changes in fault-tolerant cache in node Y upward to S. The case 2 states that when the node removes itself from the **node list** and the **node list** comes NULL it knows there is no other node to send the message.

The message travels hop-by-hop toward the source and the fault-tolerant cache is built along the path to restore the fault.

2.3 Example of Fault-tolerant Cache Generation

Let us assume that the primary (i.e. active) tree for a multicast group G1 is as shown by thick arrows in Figure 1 where Sx and Sy are the sources for the group G1. The s0, s1, s2 and s3 in the figure are names of the interfaces of routers.

The router G calculates the shortest path to Sx without the link G-Sx and finds that the shortest path is G→D→Sx. G sends $fault-tolerant$ message ([G1],[D,Sx], Sx) to D and puts in its fault-tolerant cache ([G1], s1, [Null], [Sx]). The router D receives the fault-tolerant message from G at its interface s2. It removes itself from the node list and sends (G1, [Sx], Sx) to Sx and puts ([G1], s0, [s2], [Sx]) in its fault-tolerant cache. Sx receives the message from D at its interface s0. It removes itself from the node list. Since the node list becomes NULL, Sx doesn't need to send the message any further. Sx puts ([G1], Null, [s0], [Sx]) in its cache.

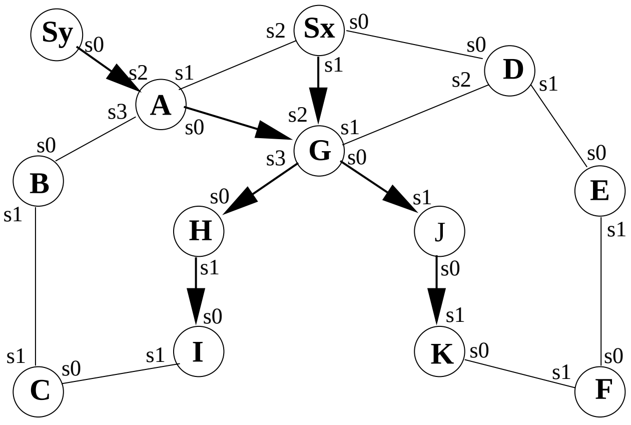

Figure 1: Fault-Tolerant Cache Generation

H calculates the shortest path to Sx without the link H-G and finds that the shortest path is I→C→B→A→Sx. H sends ([G1], [I,C,B,A,Sx], Sx) to I and puts ([G1], s1, [Null], [Sx]) in its fault-tolerant cache.

I receives the message from H at its interface s0. It removes itself from the node list and sends ([G1], [C,B,A,Sx], Sx) to C and puts ([G1],s1,[s0], [Sx]) in its fault-tolerant cache.

C receives the message from I at its s0 interface. It removes itself from the node list and sends ([G1], [B,A,Sx], Sx) to B and puts ([G1], s1, [s0], [Sx]) in its fault-tolerant cache.

B receives the message from C at its interface s1. It removes itself from the node list and sends ([G1], [A,Sx], Sx) to A and puts ([G1], s0, [s1], [Sx]) in its fault-tolerant cache.

A receives the message from B at its interface s3. It removes itself from the node list and sends ([G1], [Sx], Sx) to Sx and puts ([G1], s1, [s3], [Sx]) in its fault-tolerant cache.

Sx receives the message from A at its interface s2. It removes itself from the node list. Now the node list is NULL. Thus Sx doesn't need to forward the message. Sx updates its fault-tolerant cache to ([G1], Null, [s0,s2], [Sx]).

H calculates the shortest path to Sy also because it is forwarding the packets for the group G1 from the source Sy. The shortest path that is calculated is I→C→B→A→Sy. The fault-tolerant message travels hop-by-hop toward Sy. The fault-tolerant cache entry at nodes I, C, B and A is updated to:

Cache entry of I:	([G1], s1, [s0], [Sx,Sy])
Cache entry of C:	([G1], s0, [s1], [Sx,Sy])
Cache entry of B:	([G1], s0, [s1], [Sx,Sy])
Cache entry of A:	([G1], s1, [s3], [Sx]),
	([G1], s2, [s3], [Sy])

Note that there are two incoming interfaces thus two entry at A. Furthermore, A does not need to send any information to Sy because the link from Sy to A is already in the primary tree.

When the router I calculates the shortest path to Sx and Sy, it does not send fault-tolerant message up toward the sources because it has already got the Sx and Sy for the multicast group G1 in its fault-tolerant cache. Similarly, J will compute the shortest path to Sx and Sy and the fault-tolerant message will be sent and stored in each router on the shortest path. K does not need to compute the shortest path to Sx and Sy because K is already on the shortest path of J and no new information will be added to the cache entry.

2.4 Link Failure Detection and Activation of Recovery Tree

The first router to detect the link failure is the one which is attached to it. Whereas in centralize link detection, it will take sometimes to forward the link failure information to the central controller as in previous works.

When a child node detects the failure of the link to its parent node, it checks its fault-tolerant cache and sends tree activate message toward the source. The activate message is sent from the incoming interface stored in the fault-tolerant cache and contains the source and group lists. It transfers the message from the fault-tolerant cache to active tree's (group, source) states, incoming interface (ii) and outgoing interface list (oil).

When a router receives an activate message at one of its interface in the oil of the fault-tolerant cache, it puts that interface to its oil of the active tree. It sends the message up toward the source from its incoming interface in the fault-tolerant cache entry.

The process continues until it reaches the Source or the node which is already in the active tree. In worst case the activate message will travel all the way to the source and in the best case it will travel only to the next router.

When a node receives an activate message and it is already in the active tree, it does not send the activate message up toward the source. It puts the interface at which it received the message to the oil of its active tree.

2.5 Example of Recovery Tree Activation

Suppose in Figure 1, the router G detects the link failure between itself to the source Sx. G checks its fault-tolerant cache entry and finds for the source Sx and group G1, the incoming interface is s1 and sends activate message (i.e. (Sx,G1)) toward Sx via s1. It puts the s1 as the incoming interface for the (Sx, G1) in its active multicast tree (recovery tree). When D receives the activate message from G at its interface s2, it checks its fault-tolerant cache entry and finds that s2 in its oil for the group G1 and source Sx. It puts s2 in its oil, s0 as incoming interface list for the source Sx and the group G1, i.e. (Sx, G1). It sends the activate message to Sx from the interface s0. When Sx receives the message it checks its fault-tolerant cache entry and finds that its s0 is in oil for the group G1 and it is the source for it. It puts the s0 in its oil list for the group G1 in its active multicast tree. The new tree generated will be as shown in Figure 2 and the multicast data from Sx to G will flow via the router D after the fault recovery is completed.

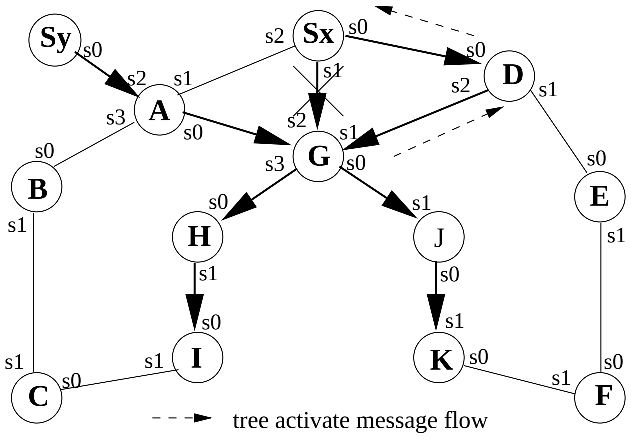

Figure 2: Fault Recovery Example

3 Discussion

In our proposed scheme, we do not pre-compute the recovery tree but keep information in fault-tolerant cache at the routers in the shortest path for creating recovery tree. As discussed in section 2.3, all the nodes in the active tree do not need to compute the shortest path to the sources. In fact, once a router calculates the shortest path to a source, all the nodes below it do not need to calculate the shortest path to the source if they are already included in its shortest path because no new information will be added in the fault-tolerant cache entry. Our scheme does not need the global view of the multicast tree as each node calculates the shortest path to the sources. The fault-tolerant message travels hop-by-hop to the first node in the active tree if there is such node in the shortest path to the source otherwise it will travel all the way to the source. Therefore, depending upon the network topology and the active tree, our scheme uses both existing tree or makes a new tree. The shortest path can be calculated using the underlying unicast routing protocols such as OSPF (Open Shortest Path First). Thus we don't need extra information about the topology of the network for this calculation.

Our scheme has some drawbacks too. Each router on the shortest path to the source have to put the following information per multicast packet incoming interface; **(group list, incoming interface, outgoing interface list, source list)**. If there are a large number of groups in the **group list** then the memory required will be proportional to the number of groups in the group list. As our scheme does not require the global view of the active multicast tree, the shortest path to the source might not be via a node in the active tree. A new path may be created all the way from the source instead of from the nearest node which is already in the active tree.

4 Conclusions

In this paper, we proposed a novel fault tolerant scheme for multicast communication. Unlike the previous works, the proposed scheme does not require the global view of the multicast tree to compute the fault recovery tree. A node computes the shortest path from itself to the source of the multicast data without the link to its parent node in the active multicast tree. Each node on the shortest path keeps the information for recovering from the fault, if the link to the parent node fails. If the link fails the node will send the fault recovery message (activate message) toward the source and the multicast data will continue to flow along the new path. The shortest path can be calculated using the existing unicast routing protocols such as OSPF.

References

[1] D. Waitzman, S. Deering, and C. Partridge, "Distance vector multicast routing protocol," in *RFC 1075*, Nov 1988.

[2] S. D. et al., "Protocol independent multicast-dense mode (pim-dm): Protocol specification," in *Internet draft*, June 1999.

[3] D. Estrin and D. F. et al., "Protocol independent multicast-sparse mode (pim-sm): Protocol specification," in *RFC 2362*, June 1998.

[4] T. Ballardie, "Core based trees (cbt version 2) multicast routing," in *RFC 2189*, Sept. 1997.

[5] J. May, "Multicast routing extensions for ospf," *Commun. ACM*, vol. 37, pp. 61–66, Aug. 1994.

[6] J.-H. Cui, M. Faloutsos, and M. Gerla, "An architecture for scalable, efficient, and fast fault-tolerant multicast provisioning," *IEEE Network*, vol. 18, pp. 26–34, March/April 2004.

[7] M. Medard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 641–652, October 1999.

[8] A. Fei, J. Cui, M. Gerla, and D. Cavendish, "A "dual-tree" scheme for fault-tolerant multicast," in *Proceeding of IEEE ICC 2001, Helsinki, Finland*, June 2001.

[9] 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, pp. 984–1000, October 1999.