World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:4, No:12, 2010

Improving Fault Resilience and Reconstruction of Overlay Multicast Tree Using Leaving Time of Participants

Bhed Bahadur Bista, *Member, IEEE,*

Abstract—Network layer multicast, i.e. IP multicast, even after many years of research, development and standardization, is not deployed in large scale due to both technical (e.g. upgrading of routers) and political (e.g. policy making and negotiation) issues. Researchers looked for alternatives and proposed application/overlay multicast where multicast functions are handled by end hosts, not network layer routers. Member hosts wishing to receive multicast data form a multicast delivery tree. The intermediate hosts in the tree act as routers also, i.e. they forward data to the lower hosts in the tree. Unlike IP multicast, where a router cannot leave the tree until all members below it leave, in overlay multicast any member can leave the tree at any time thus disjoining the tree and disrupting the data dissemination. All the disrupted hosts have to rejoin the tree. This characteristic of the overlay multicast causes multicast tree unstable, data loss and rejoin overhead. In this paper, we propose that each node sets its leaving time from the tree and sends join request to a number of nodes in the tree. The nodes in the tree will reject the request if their leaving time is earlier than the requesting node otherwise they will accept the request. The node can join at one of the accepting nodes. This makes the tree more stable as the nodes will join the tree according to their leaving time, earliest leaving time node being at the leaf of the tree. Some intermediate nodes may not follow their leaving time and leave earlier than their leaving time thus disrupting the tree. For this, we propose a proactive recovery mechanism so that disrupted nodes can rejoin the tree at predetermined nodes immediately. We have shown by simulation that there is less overhead when joining the multicast tree and the recovery time of the disrupted nodes is much less than the previous works. *Keywords*

Keywords—Network layer multicast, Fault Resilience, IP multicast

I. Introduction

Overlay, sometimes referred as application layer, multicast [1], [2], [3] can be deployed quickly over a large network for data dissemination. The network layer multicast (IP Multicast) [4] , [5] is not deployed in large scale due to various issues such as delay in standardization, implementation of protocols and upgrading of routers which are in various locations around the world [6]. In contrast in overlay multicast, end hosts handle the multicast functionality by themselves and communicate with each other using underling unicast. This usually requires more overall bandwidth and is slower compared to IP multicast because duplicate packets travel the same physical links multiple times. But it is inexpensive and easily deployable for point-to-multipoint communication. Although, overlay multicast brings flexibility in its use and deployment, it suffers from problem of service reliability because a member may leave multicast delivery tree abruptly causing data outages on all the downstream members.

B. B. Bista is with the Faculty of Software and Information Science, Iwate Prefectural University, Japan 020-0193, e-mail: bbb@soft.iwate-pu.ac.jp.

Various resilient overlay multicast protocols are broadly proposed for mesh-like topology and single tree-like topology. A good comparison of the resilient protocols for mesh-like topology can be found in [7]. The basic approach is to construct redundant multiple-tree or have several extra redundant cross links in an original single tree so that there is more than one path for data delivery to end hosts. In this paper, we are interested in tree-like topology for constructing resilient overlay multicast tree.

There are three main techniques in delivering data with minimum loss to all members in overlay multicast where a single tree for data delivery is constructed.

The first one is to construct a stable tree so that the impact of a node's departure from the tree is minimized. There are some researches in building a stable tree. In shortest tree, sometimes called fat tree, construction methods [8], [9], [10], [11] nodes with higher outgoing bandwidth, in other words, nodes which can add many children, are put or shifted higher up the tree, thus growing the tree side ways. So when a node leaves the tree, only a few nodes will be affected.

The second is the longest stayed, sometimes called heavy-tailed tree construction methods [12], [13] where nodes are rearranged so that the nodes which have stayed longer in the tree are moved up the tree expecting that they will stay longer in the network.

The third method is not for building a resilient tree but for recovery or reconstruction method of a tree when a node leaves the delivery tree. This method tries to minimize the time for affected nodes to rejoin the tree so that the data outage is minimized. There are two approaches; reactive [14] and proactive [15]. In reactive approach restoration of the tree starts after a node departs the tree whereas in proactive approach the restoration is planned beforehand so when a node departs the tree the affected nodes rejoin the tree without delay.

In this paper, we propose a method to build a tree which is fat, leaving time ordered and also offers a proactive approach to reconstruct the tree. In our approach, nodes are joined according to their leaving time from the network. A node continues to connect nodes as its children as long as it has available out-degree which will be explained in the next section. In other words, we also make subtree grow side ways, i.e. fat. Moreover, as soon as a node joins the network, it calculates at which node it will rejoin the tree if its parent node leaves the network.

The paper is organized as follows. In Section 2, we present the detail of our multicast delivery tree construction methods.

In Section 3, we present our proactive reconstruction method for multicast delivery tree when a node leaves the tree. In Section 4, we evaluate our proposals followed by the related works in Section 5. Finally we conclude the paper in Section 6.

II. Network Leaving Time Order Multicast Tree

In this section, we explain the leaving time and out-degree of a node.

A. Leaving Time of a Node

We propose that each node when sending join request to nodes which are already in the network also sends how long it intends to stay, i.e. its leaving time from the network. Using this parameter, we build a multicast tree so that nodes whose leaving time is later (i.e. staying longer) will be put higher up the tree. For example, when nodes 1 and 2 set their leaving time 13:05 and 13:10 respectively, they are planning to leave the network at 13:05 and 13:10 respectively. Here node 2 is planning to leave the network later than node 1 and will be put higher up in the multicast tree than node 1.

In order to have a single clock for setting leaving time, a node will set its leaving time in GMT. Here an approximation of leaving time is enough and clock synchronization is not required.

In order to prevent a node from putting a large leaving time and placing itself higher up the tree, we set an upper limit for leaving time. A node is not allowed to set a leaving time above the upper limit when it joins the network for the first time. Ye Tian et al. [16] have found by probing PPLive [17], a popular IPTV system based on overlay network, that nodes stayed in the streaming application for 40 minutes in average. Therefore, we put a threshold value of 40 minutes, in other words, a node sets its leaving time such that its staying time in the network will not exceed 40 minutes. However, a node wishing to stay longer after its leaving time approaches can extend it and continue to stay in the network which will be explained later.

B. Out-degree of a node

We also consider the out-degree constrain of a node. Total out-degree of a node depends upon the media streaming. If media play back rate is M_{rate} and bandwidth of a node is N_{BW}, then the total out-degree of the node will be $\lfloor N_{BW}/M_{rate} \rfloor$. How many children a node can add depends upon its total out-degree. If the out-degree of a node is zero it cannot add a child. In this paper, we assume that each node has total out-degree greater than or equal to 2. Though this is not a problem in broadband Internet environment, we make this assumption for proactively calculating at which node the children nodes will rejoin after their parent leaves the multicast tree. We will explain the details in Section 3.

C. Tree Construction

A multicast tree is constructed according to how join requests are handled. We propose multiple join requests and a single join request of a node to construct the tree.

1) Multiple Join Requests (MJR): If a node cannot join the multicast tree in its first join request, it sends the join request again to different nodes with the same leaving time or changes its leaving time and sends the join request to the same nodes again or both. The detail procedure is as shown below.

1) A node which wants to join the multicast tree sets its leaving time and sends requests to a number of nodes which are already on the tree.
2) When a node (which is already on the tree) receives the request, it checks its available out-degree and compares its leaving time with the leaving time of the requesting node. If the leaving time of the requesting node is earlier than its own leaving time and it has available out-degree which is greater than or equal to 2 it sends can join reply to the requesting node. Otherwise, it rejects the join request.
3) When the requesting node receives replies from requested nodes, it checks if there are any nodes which can joint/connect it as their child. If there is more than one node, it selects one which is nearest to it and sends join request and the node will be joined to the multicast tree at the requested node.
4) The parent node will then update its family list and forward it to its children which will update their family lists and forward to their children and so on.
5) If there are no nodes which can join it, the requesting node will do one of the followings: (i) select the same nodes and send join request with reduced leaving time, (ii) select different nodes and send join request with the same leaving time or reduced leaving time.

For example, if a node sets its leaving time 15:20 and sends join request to nodes 4, 7 and 8 in Figure 1(a), its request will be rejected as their leaving time is earlier than its own leaving time. Now if it sends join request again without lowering its leaving time, i.e. with the same leaving time, to other nodes e.g. nodes 2, 5 and 6 it will receive can join reply from nodes 2 and 5 because their leaving time is later than its own leaving time and they have available out-degree of 2. Finally the node can select either node 2 or 5 for its parent node, i.e. to be joined at.

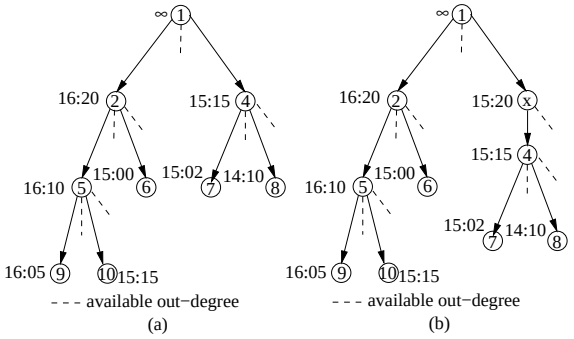

Fig. 1. Leaving time order tree

2) Single Join Request (SJR): A node sends join request only once and the request is forwarded up or down the tree to other nodes if the requested node cannot add it as its child

or replace one of its children because of leaving time or out-degree constrains. The detail procedure is as shown below.

1) As in method one, a node which wants to join the multicast tree sets its leaving time and chooses one of the nodes in the multicast tree and sends join request.
2) When a node in the tree receives the request, it checks the requesting node's leaving time. If it is earlier than its own leaving time and has available out-degree 2 or more, then it connects the requesting node as its child and sends join success to the requesting node.
3) If the leaving time is earlier than its own leaving time but does not have available out-degree to connect the node as its child, it checks the leaving time of its children. If there is a child whose leaving time is earlier than the leaving time of the requesting node, it will replace the child, i.e. the child will become the child of the requesting node and the requesting node will be the child of the requested node (i.e. the node which received the request for the first time). If there is more than one child whose leaving time is earlier than the requesting node then the child with the earliest leaving time will be replaced. If the leaving time of the requesting node is earlier than all the children's leaving time, the node will forward the join request to the child with the earliest leaving time. The child which receives the forwarded request will perform the same procedure.
4) If the requesting node's leaving time is later than the requested node's leaving time the requested node will forward the request to its parent node. The parent node will repeat the procedure in step 3, if its leaving time is later than the requesting node. If not it will forward the request to its parent.
5) Once the node is connected to the tree, its parent node will update its family list and forward it to its children which will update their family lists and forward to their children and so on.

Eventually the node will be connected to the tree. The node which connects the new node as its child will send the join success to the requesting node. For example in Figure 1(a), if a node x sets its leaving time 15:20 and sends join request to node 7, it will forward the request to its parent node, node 4, which will forward the request to its parent node, node 1, because the node x's leaving time is later than their leaving time. Finally the node 1 will replace node 4 by node x as shown in Figure 1(b), i.e. node x will become a child of node 1 and node 4 will become the child of node x.

The disadvantage of this method is that there may be join overhead as the joining procedure is handled by the nodes which are already on the tree.

D. Leaving Time Update

When a node's announced leaving time arrives and the node intends to stay longer, it will increment its leaving time by the difference of its initial leaving time and the time it joined the tree. For example, if its initial leaving time was 9:10 and it joined the multicast tree at 9:00 and if it does not leave the tree at 9:10, it will increment its leaving time by 10 minutes to 9:20. If it does not leave the tree at its following leaving times it will increment its leaving time by $10 + 2 \times n$ where n is the number of its previous leaving time it did not leave the tree. This is to prevent a node from climbing up the tree faster.

When the node increments its leaving time its leaving time may be later than its parent's leaving time. The node waits until its parent's leaving time arrives and checks whether the parent leaves the tree at its leaving time. If it does, the rejoin procedure of children will start. If it doesn't the parent will also increment its leaving time which will be later than its children nodes' leaving time. The tree will remain as it is.

III. Proactive Approach for Tree Recovery

In this section, we will explain a family list in which there are lists of nodes obtained from the parent node when a node joins/rejoins the multicast tree. Our proposed proactive approach for tree recovery uses the family list to find which node a node will rejoin at, when its parent leaves the network.

A. Family List

Each node maintains a children list, a brothers list, a parent, a parent's brothers list, a grandparents (includes grand.. grandparents) list and a grandparents' brothers list, referred as a family list in this paper. Each list contains a list of nodes. If a node's out-degree is zero it is removed from the list.

For example, the family list of node 9 in Figure 2 is as follows:

$ChildrenList_9$ = [17, 18]
$BrotherList_9$ = [10]
$Parent_9$ = [4]
$ParentBrotherList_9$ = [5, 6]
$GrandparentList_9$ = [2, 1]
$GrandparentBrotherList_9$ = [3]

When a node connects/reconnects a node as a child, it forwards its family list to the newly connected child. The child then transforms its parent's family list to its own family list. For example when node 9 connects node 19 as its child, it sends its family list to node 19 which uses it to make its own family list as shown below.

$ChildrenList_{19}$ = []
$BrotherList_{19}$ = $ChildrenList_9$ = [17, 18]
$Parent_{19}$ = [9]
$ParentBrotherList_{19}$ = $BrotherList_9$ = [10]
$GrandparentList_{19}$
= $GrandparentList_9$ + $Parent_9$
= [2, 1] + [4] = [2, 1, 4]
$GrandparentBrotherList_{19}$
= $GrandparentBrotherList_9$
+ $ParentBrotherList_9$
= [3] + [5, 6] = [3, 5, 6]

Node 9 forwards the new child (node 19) information to its other children, nodes 17 and 18, which update their brother

lists as shown below. Other lists are unchanged.

$BrotherList_{17}$ = [18, 19]
$BrotherList_{18}$ = [17, 19]

If nodes 17 and 18 have children they will forward the information about node 19 to their children which will forward to their children and so on all the way to leaf nodes and they will update their family lists.

Similarly, if a node leaves the tree, its parent will forward the information of the node that left the tree to its remaining children which will forward to their children and so on all the way to leaf nodes and they will update their family lists.

B. Family List Update

If a node commits certain bandwidth for multicast, the total out-degree of the node will remain the same. However, in reality the available bandwidth of a node dynamically changes as other network applications hold and release bandwidth. When a node's bandwidth changes, its out-degree also changes. If its available out-degree becomes to zero, it informs to its children and parent to remove it from the family list, because it cannot connect a node as its child. Its parent node and children nodes remove it from the family list and inform their children to do the same. All the nodes below its parents node will remove it from the family list. If a node's available out-degree changes from zero to one or more, it informs its children and parent about it which will then add it in their list and inform their children.

C. Rejoin Protocol

When a node leaves the tree its children need to rejoin the tree. We assume only the children, i.e. the root nodes of the disconnected subtrees, rejoin the tree. Other nodes of the disconnected subtree do not need to rejoin the multicast tree. We distinguish between a node's joining the multicast tree for the first time and its rejoining the tree after it is disconnected from it. In rejoining the tree a node does not change its leaving time. Similarly, a node treats a joining and a rejoining request of a node separately. When a node receives a rejoin request from a node and condition for leaving time is satisfied (i.e. rejoining node's leaving time is earlier than its own leaving time), it will add the node as a child until its out-degree becomes zero. Note that it will leave one out-degree available if the there is a join (not rejoin) request (see section 2.3).

As we have already said in section 2.3, each node leaves a out-degree available for rejoining nodes. A node knows leaving time and available out-degree of its children, brothers, parent, parent's brothers, grandparents' and grandparents' brothers. Since each node knows the above information, it can immediately rejoin at one of the above nodes if its parent leaves the tree. The procedure of which node a node will rejoin at is as shown below.

1) The child with the highest leaving time will take the place of its parent that has left the tree, i.e. rejoin at its grandparent. Since they know about their brothers, they will know which one will take the place of their parent. If the grandparent has more out-degree available other highest leaving time node will rejoin at the grandparent.
2) If the grandparent does not have any out-degree available, other children will rejoin at the brothers which have rejoined at the grandparent. If those brothers cannot rejoin any more brothers (nodes) because of unavailability of out-degree, remaining brothers will rejoin at the brother with the next highest leaving time and so on.
3) If a node cannot rejoin at its brothers it will rejoin at its parent's brother node. If that is not possible it will rejoin at its grand grandparent. The order of rejoining at nodes is grandparent, brothers, parent's brother, grand grandparent, grand grandparent's brother, grand grand grandparent, grand grand grandparent's brother and so on.

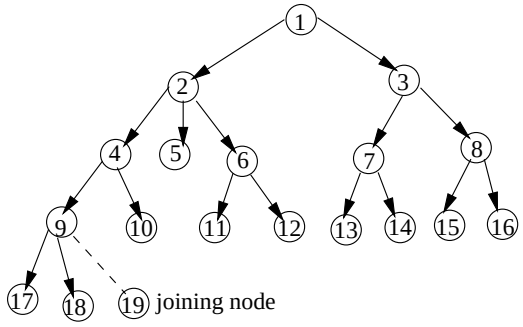

Fig. 2. Joining to tree

For example in Figure 3(a) if node 4 leaves the tree, node 7 (highest leaving time) will take its parent position, i.e. rejoin at node 2. Node 8 will rejoin at node 7, its highest leaving time brother. Node 9 will rejoin at node 5 which is parent's brother because node 7 and 8 (brothers) do not have any out-degree available to add a child.

The above procedure to rejoin the tree is predetermined (proactively calculated) from the family lists so the rejoining is performed as soon as a node finds that it is disconnected from the tree. The above order is arranged such that lower leaving time node will not be higher up the tree when they rejoin it. This maintains our original proposal of making a stable tree from the beginning based on the leaving time.

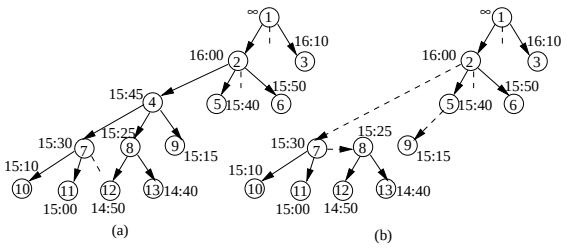

Fig. 3. Example of rejoining to tree

IV. Evaluation

We evaluate our proposal by simulation. A event-driven simulator was developed to study our proposal. We randomly

assign out-degree which ranges from 2 to 10 to each node. This means that for 500Kbps playback rate of media, outgoing bandwidth of nodes ranges from 1Mbps to 5Mbps. We assigned leaving time which ranges from 1 minute to 40 minutes to each node randomly. Nodes are joining the overlay multicast tree following the Poisson process.

The purpose of our simulation is to find the overhead of the proposed overlay multicast tree construction methods, maximum multicast tree depth and the time taken for a node to rejoin the multicast tree once its parent leaves the tree.

A. Multicast Tree Construction Overhead

In our proposed methods once nodes join the multicast tree, they are not rearranged again. However, while joining the multicast tree their join request may be forwarded number of times before they can successfully join the tree (single join request) or they may resend join request a number of times before they can successfully join the tree (multiple join request).

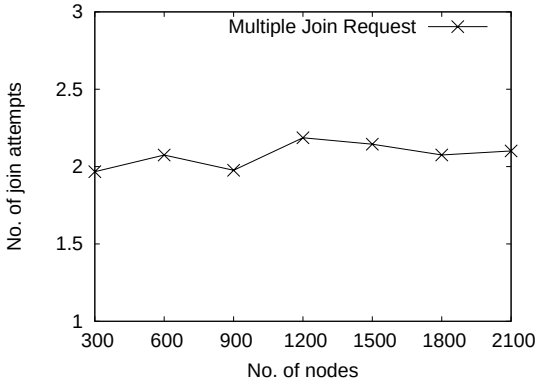

Fig. 4. Multiple join request overhead

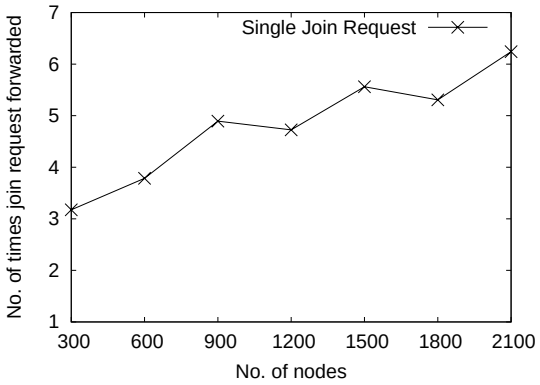

Fig. 5. Single join request overhead

As shown in Figure 4, In our simulation, we found that if a node cannot join the multicast tree in its first attempt it can join the tree in its second attempt in average. The number of join attempts does not change even if the number of participating nodes increases.

In single join request, if a node cannot join the tree at the requested node, we check the number of times its request were forwarded up or down the tree before it can join the tree successfully. We count as one forward if the node's request is forwarded one level to next level in the tree. As shown in Figure 5, the average number of request forwarded increases as the number of participating nodes increases. This may be because if a requested node is around the bottom/top of the tree and the new joining node's request has to be forwarded top/bottom of the tree due to its leaving time, then there will be more number of request forwarding.

B. Tree Depth

We measured the maximum and average depth of the multicast trees that are constructed using our proposed tree construction methods. Our measurement does not say that all leaf nodes are at the maximum depth but says at least one node is. As shown in Figure 6, the single join request has higher value than the multiple join request. This may be because in the single join request, a new joining node in some cases shifts a node which is already on the tree downward, i.e. becomes the parent of node which is already on the tree. This increases the depth of a tree, whereas in the multiple join request, this does not occur. However, the average depth of the both methods is almost the same.

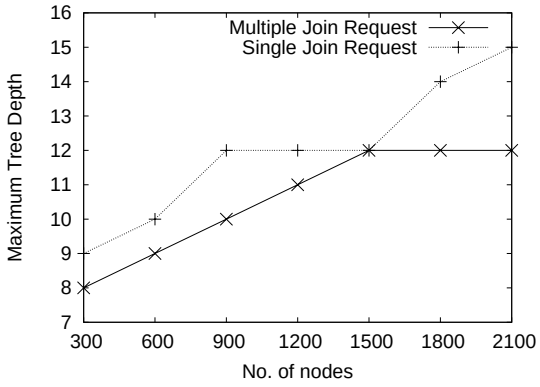

Fig. 6. Maximum tree depth

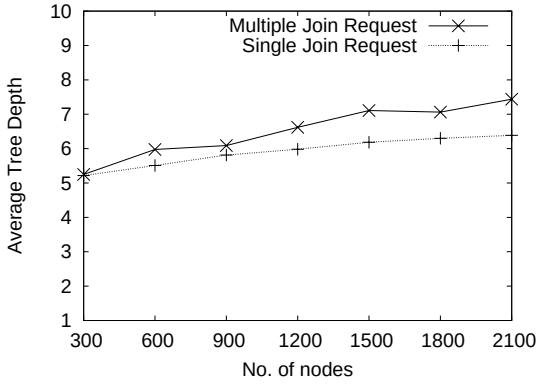

Fig. 7. Average tree depth

C. Recovery Time

The recovery time is measured as the time it takes for a node to rejoin the multicast tree after its parent left the tree. The average recovery time is the average of the recovery time of all children of the parent which has left the tree. As shown in Figure 8, the average recovery time of our proposal is significantly less than the best recovery time of the proactive method proposed in [18]. Their lowest average recovery time is 225ms, whereas ours highest is about 105ms. As the number of participating nodes in multicast increases, the average recovery time slightly decreases. One possible reason may be that as the number of participating nodes increases, the average distance between two nodes becomes shorter. So the affected children will have shorter distance to their grandparent, brothers, parent's brothers and so on.

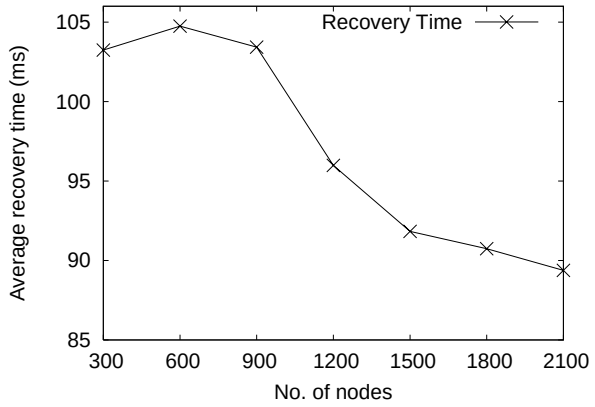

Fig. 8. Average recovery time

V. Related Works

There are various fault resilient overlay multicast protocols. Among them G. Tan et al. [19] have shown by simulation that their protocol outperforms all others. Their main idea is to calculate BTP (Bandwidth Time Product) which is defined as the product of a node's outbound bandwidth and its age (i.e. how long it has stayed in the network). Then they compare BTP of children and their parent node. If there is a child which has higher BTP than its parent then they switch them over. In other words, a node with higher BTP is moved slowly up the tree. The switch operation is performed after a node has joined and stayed for sometimes in the tree. While switching they either have to manage the data buffer or data is lost. Our approach is different from theirs. In our approach, there is joining overhead for a new joining node, such as number of join attempts (multiple join request method) or number of times a join request is forward (single join request method). However, once a node joins the tree there is no switching or rearrangement of nodes. Moreover, in our approach we do not need to manage the data buffer due to joining of new nodes.

In reconstructing mechanism of a overlay multicast tree, F. Zongming et al. [18] have shown by simulation that their proposal outperforms all other proposals. Their idea is to solve a Spanning tree algorithm with available degree of a node as degree constrain to find parents-to-be nodes for its children. Our recovery mechanism is much simpler and performs better than theirs. In our approach, using its family list, each node locally precalculates which node it will rejoin when its parent leaves the tree. As soon as it finds that its parent has left the tree it rejoins at the precalculated node with minimum delay. However, drawback of our mechanism is that when a node joins, rejoins or leaves the network this information should be forwarded to other downward nodes. This drawback is similar to the one in [18] where a node needs to know other nodes' available degree to find the parents-to-be nodes.

VI. Conclusion

In this paper, we try to improve the fault resilient and reconstruction of overlay multicast tree for media streaming using leaving time of participation nodes. Unlike previous works, the main idea of our proposal is that a node wishing to join the multicast tree sets its leaving time when sending join request to nodes in the tree. Using the leaving time of nodes, they are connected in the tree so that a child's leaving time is earlier than its parent's leaving time so there is no disruption if nodes leave the multicast tree when their leaving time arrives. We proposed two methods, single and multiple join requests, to construct the tree. Though the multiple join request has less overhead and maximum tree depth, it has to change its parameter after each unsuccessful attempt to join the tree where as single join request does have to do so. We also proposed a proactive reconstruction mechanism in which each node precalculates which node it will rejoin should its parent leaves the tree. Our simulation have shown that the overhead for constructing multicast tree is less and the recovery/rejoin time for disrupted children is far less than the previous works. Our future work is how to manage dishonest nodes, i.e. nodes which set high leaving time and do not stay in the tree until their leaving time arrives.

References

[1] L. Lao, J.-H. Cui, M. Gerla, and S. Chen, "A scalable overlay multicast architecture for large-scale applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 449–459, 2007.

[2] Y. Zhu, M.-Y. Wu, and W. Shu, "Comparison study and evaluation of overlay multicast networks," in *Proceedings of the 2003 International Conference on Multimedia and Expo (ICME '03)*, 2003, pp. 493–496.

[3] Y.-h. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast (keynote address)," in *SIGMETRICS '00: Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2000, pp. 1–12.

[4] B. Williamson, *Developing IP Multicast Networks Volume I*. Cisco Press, 2000.

[5] L. H. Sahasrabuddhe and B. Mukhergee, "Multicast routing algorithms and protocols: A tutorial," *IEEE Network*, vol. 14, no. 1, pp. 90–102, Jan/Feb 2000.

[6] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the ip multicast service and architecture," *IEEE Network*, vol. 14, no. 1, pp. 78–88, Jan/Feb 2000.

[7] S. Birrer and F. E. Bustamante, "A comparison of resilient overlay multicast approaches," *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 25, no. 9, pp. 1695–1705, Dec. 2007.

[8] M. Guo and M. H. Ammar, "Scalable live video streaming to cooperative clients using time shifting and video patching," in *Proc. IEEE INFOCOM 2004*. Hong Kong: IEEE, 11 2004, pp. 1501–1511.

[9] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao, and P. Dinda, "Fatnemo: Building a resilient multi-source multicast fat-tree," in *Proc. Ninth Int'l Workshop Web Content Caching and Distribution (WCW 2004)*. LNCS, Sept. 2004, pp. 182–196.

[10] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*. New York, NY, USA: ACM, 2002, pp. 177–186.
[11] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE J. Selected Areas in Comm. (JSAC)*, vol. 22, no. 1, pp. 121–133, Jan. 2004.
[12] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 41–54.
[13] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 107–120.
[14] M. Bawa, H. Deshpande, and H. Garcia-Molina, "Transience of peers & streaming media," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 107–112, 2003.
[15] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 102–113, 2003.
[16] Y. Tian, D. Wu, G. Sun, and K.-W. Ng, "Improving stability for peer-to-peer multicast overlays by active measurements," *J. Syst. Archit.*, vol. 54, no. 1-2, pp. 305–323, 2008.
[17] PPLive, <http://www.pplive.com>.
[18] Z. Fei and M. Yang, "A proactive tree recovery mechanism for resilient overlay multicast," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 173–186, 2007.
[19] G. Tan and S. A. Jarvis, "Improving the fault resilience of overlay multicast for media streaming," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 721–734, 2007.