Strategies towards Robust and Stable Application Layer Multicast

Tetsuya Kusumoto, Su Zhou, Jiro Katto and Sakae Okubo
Dept. of Computer Science, Waseda University, Tokyo, Japan

***Abstract*— The purpose of this study is to construct a robust and stable overlay tree of ALM (Application Layer Multicast) for real-time video transmission. Firstly, we propose a proactive route maintenance which enables fast recovery of the overlay tree against node departures and failures. By forcing free node degrees for route backup, signaling overheads for route maintenance can be reduced in a scalable manner. Secondly, we improve performance of the proposed method by introducing layered video coding. Smooth layer management instead of coarse degree management contributes to reduction of the depth of overlay tree (i.e. delay) as well as to efficient bandwidth utilization. Thirdly, we introduce session records of each node into the overlay tree construction. Long-term history of user access records is expected to bring more robust overlay trees. Finally, we carried out extensive evaluation experiments. Simulations and implementations demonstrate that our methods lead to fast recovery of the overlay tree against node departures and reduction of the signaling overheads. Furthermore, introduction of layered video coding is proved to be efficient to reduce delays, to improve bandwidth utilization and to avoid severe degradation of picture quality. Final experiments show that incorporation of node stability reduces the number of nodes affected by parent node departures by promoting adequate nodes to upper layer of the ALM tree.**

I. INTRODUCTION

ALM (Application Layer Multicast) implements the multicast functionally at end-hosts. Different from IP multicasting [1], which unrealistically needs global deployment of routers with IP multicasting capability, ALM needs only installation of application software and requires no change in the current network infrastructure. In addition, it provides flexibility in routing such as multipath packet transfer and load balancing. The most active research area in ALM is design of routing protocols [2]-[15]. There are several measures to evaluate the effectiveness of the routing protocols as the following: (a) quality of the data delivery path, that is measured by stress, stretch and node degree parameters of overlay multicast tree, (b) robustness of the overlay, that is measured by the recovery time to reconstruct a packet delivery tree after sudden end host failures, and (c) control overhead, that represents protocol scalability for a large number of receivers.

In the ALM session, each end host is a member of the delivery tree, and it leaves freely and may fail sometimes. This is not a problem in IP multicast, because the non-leaf nodes in the delivery tree are routers and do not leave the multicast tree without notification. We therefore have to consider selfish node behaviors in ALM delivery tree management. Our objective of this paper is to numerate possible node behaviors which affect ALM delivery performance and to propose mechanisms to construct robust and stable ALM delivery trees. In this paper, we consider two properties of ALM participant nodes, (1) node departures and failures and (2) session access history. We also consider introduction of layered video coding to cope with the first issue in a flexible manner.

Node departures and failures:

When a node departure happens, the time of disconnecting is important for multicast applications such as live media streaming, because all the child nodes can not receive packets for the time. Quick reconstruction of the overlay trees and the tree structure which is less affected by node departures are therefore quite important to maintain the media quality, but little attention has been given to this problem.

One of the proactive route maintenance approaches against this problem was proposed by Yang et al [13], which we call Yang's method in this paper. Yang's method enables nodes to recover from node departures by ensuring backup route nodes. However, the way of searching backup route nodes is not scalable. As the number of nodes increases in the session, the search area becomes large. This generates extra data overheads. Volume of control traffic can be significant for some overlay multicast applications.

We therefore propose a new proactive approach in order to avoid heavy overheads. Our backup route search is carried out completely locally by forcing free degree to each node. This approach achieves fast recovery in low overhead because backup route search does not depend on the session size. A problem of this approach is clearly increase of depth of the delivery tree due to vacant degree.

Layer based management:

We then consider introduction of layered video coding to reduce depth of the delivery tree in our proactive approach and to increase bandwidth utilization efficiency. By integrating layered video coding into our proactive route maintenance, more flexible streaming can be enabled and better performance is expected. Experiments will show that our combined approach solves the tree depth problem as well as avoids quality degradation of streaming media.

Session access history:

To build more stable ALM delivery trees, long-term session history of participants should be introduced into the tree construction procedure. In addition to performance measures such as delay and bandwidth, we pay attention to session records about how long they have stayed in the past sessions. In general, user behavior is quite different in ALM session; some nodes immediately leave the session soon after joining the session, but some nodes stay for a long time. We utilize long-term characteristics of user behavior to avoid short living nodes to be inserted into higher layer positions of the ALM tree.

We then propose a tree construction method using node stability. We define node stability as a function of session records of the node in which how long the node stays in the overlay tree was reflected. Experiments will show that the number of nodes affected by node departures is reduced against the classical methods which do not consider about node stability.

In this paper, we propose three methods to construct robust and stable ALM overlay trees. The rest of the paper is structured as follows. Section II provides an overview and problem descriptions. Section III describes our proposals in detail. Section IV presents experimental results. Finally, Section V concludes the paper.

II. ALM Overview and Problem Descriptions

A. ALM Overview

Most ALM protocols had focused on how to construct an efficient multicast tree. ALMI [2] employs a centralized solution. In this scheme, a central controller computes and instructs the construction of the delivery tree based on performance metrics provided by the end hosts, hence the load of the central controller becomes heavy. In decentralized solutions, end hosts exchange the metric information each other and constructs an overlay network in a distributed manner. They are categorized into mesh-first protocols and tree-first protocols. Narada [3], Scattercast [4] and CoolStreaming [5] are examples of mesh-first protocols. Each host keeps many connections to keep the session stable, but the overhead to maintain the mesh topology becomes large. In contrast, Yoid [6], Overcast [7] and Peercast [8] are tree-first protocols. In these approaches, the constructed tree is rooted at a single node, which is generally a data source of the session. The overhead to maintain the tree topology is less than that of the mesh-first protocols, but the sessions are less stable because the connections of each node become fewer. OMNI [9] defines a periodical local transformation of the overlay tree to minimize average latency among entire hosts. ZIGZAG [10] and NICE [11] try to achieve low control overhead by building an overlay of hierarchical clusters.

Node departures in overlay network have been recognized in more recent works. Tree reconstruction methods are mainly divided into reactive approaches and proactive approaches. Finding a next parent is done after node departure in the reactive approach, but it is done beforehand in the proactive approach. PRM (Probabilistic Resilient Multicast) [12] uses a proactive approach for the overlay multicast. It introduces a method called randomized forwarding, which generates redundant packets constantly to handle node departures. Another proactive approach [13] prepares a backup parent, which is decided beforehand according to the residual degrees of nodes over the ALM tree. Split Stream [14] utilizes a Multiple Description Coding (MDC) to split media stream into multiple stripes and then provides a separate multicast tree for each stripe. Even if affected nodes cannot receive one stripe after a node departure happens, they can receive media stream of other stripes.

B. Node Departure Problems

Most of existing ALM methods employ a reactive approach, in which tree recovery is initiated after node departure. In this reactive approach, a node which leaves the overlay tree sends a message to inform other nodes to be affected by its leaving such as its parent and child nodes. The child nodes cannot receive data temporarily until they connect to new parent nodes. When a node suddenly fails, it cannot send a message to affected nodes, and they will not notice the failure for a while. Heartbeat mechanism helps the affected node to notice the failure. However, the child nodes need a timeout period to recognize the failure.

In proactive approaches, each host has a backup route to recover quickly from the parent node departure. In Yang's method [13], each non-leaf node decides a backup parent for its child nodes. This backup route is ensured by using residual degree of nodes in the overlay tree. Each host uses Eq.(1) to figure out if its all children nodes can form their backup routes:

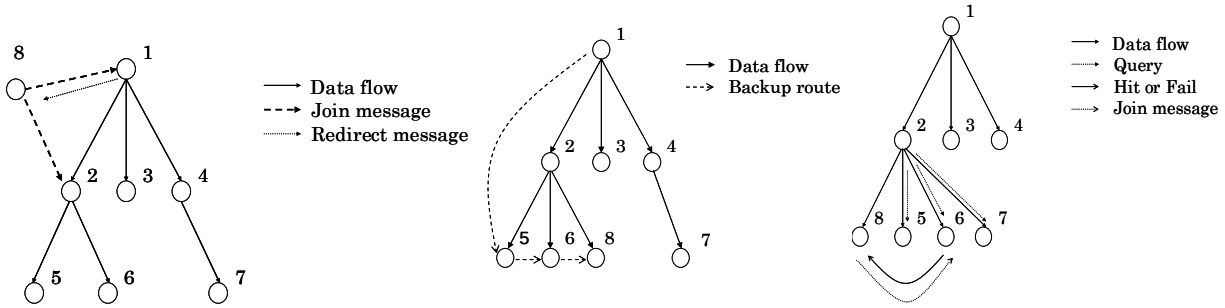
$$\sum_{j=0}^{n-1} d(C_j) \geq n-1 \qquad (1)$$

where a node has n child nodes $\{C_0, C_1, \cdots, C_{n-1}\}$, $d(C_j)$ is the residual degree of the child C_j, and $\sum_{j=0}^{n-1} d(C_j)$ denotes sum of the residual degrees of the child nodes. Firstly, a parent node calculates residual degrees of its child nodes. If the total residual degree of the child nodes is not less than n-1, all the child nodes can form their backup routes. If not, the child nodes cannot and the parent node calculates the total residual degree including the residual degree of descendant nodes of the child nodes. Secondly, the parent node selects the child node that has the smallest latency from the grandparent node of the child node. The child node holds the backup route to the grandparent. The subtree of the child node which holds the backup route supplies a backup route to the other child nodes. Then, the child node and its descendant nodes measure the latencies to the other child nodes, and the smallest edge is selected. This operation is repeated until all the child nodes hold their backup routes. A critical problem of this approach is that, when this operation continues in lower layers, it tends to generate heavy overhead packets.

C. Node Promotion Problems

Different from stable IP-multicasting, every node in ALM trees behaves selfishly. Especially, node departures in upper level of the tree affect the large number of its descendant nodes. Therefore, to maintain stable overlay trees, we have to take care of long-term characteristics of participant nodes to maintain a stable overlay tree. In classical approaches such as PeerCast, new participant nodes connect to the overlay tree as leaf nodes in every join process. This rule is reasonable for newcomers, but for nodes which already contributed to the session but failed accidentally, they have to wait for a long time to climb again to upper level of the tree. In Bandwidth-Ordered (BO) tree, nodes are placed in order of bandwidth (i.e. degree). A new participant node having large degree can achieve higher position in the tree immediately after joining the session. However, when the node having large degree leaves the session immediately, the number of nodes affected by the node's departure also becomes large.

Bandwidth-Time Product (BTP) tree [15] utilizes more stable operation to optimize the overlay tree. The metric is defined as product of node's outbound bandwidth and its age, that is elapsed time from the joining time of a node to



(a) New node participation (b) Finding a backup route (c) Tree Reconstruction

Fig. 1: Proactive Route Maintenance I

the session. This metric moves nodes with large BTPs to higher position in the tree so that better service quality (less stream disruptions and smaller service delay) can be offered to participant nodes. For every interval of a certain time, each node compares its own BTP with its parent and switches their parent-child relationship when the BDP value exceeds that of the parent. In this method, introduction of long time characteristics of participant nodes reduces occurrence of stream disruptions. However, it is clear that single session history is not sufficient. Short time staying nodes having large outgoing degree can achieve higher position in the tree, and nodes other than always-on nodes would be affected by the node departures.

III. PROPOSALS

A. Proactive Route Maintenance by Node Degree

Firstly, we propose a proactive route maintenance method in which nodes can hold their backup route with low overhead, which is slight extension of our own approaches [16][17] with analytical support. In our approach, we construct an overlay tree without each node exhausting its degree. Each node constantly has residual degrees not less than 1. The child nodes of each node can ensure their backup routes between the grandparent node and themselves by using their residual degree.

1) Tree Construction

We show the process of node joining the overlay tree in Fig. 1(a). It is assumed that maximum degree of each node is equal to 4. We then limit the active degree of each node to 3 and reserve 1 degree for backup route maintenance. In ordinary tree, when new node 8 requests to connect to node 1, node 1 accepts node 8 to join as its child node, because its degree is not exhausted. However, in our proposal, node 1 refuses the request because the residual degree of node 1 is only 1. Node 8 sends a join request to node 2 after receiving a redirect message from node 1. As a result, node 8 becomes a child node of node 2.

2) Backup Route Construction

Next, we show how to decide the backup route of each node in our proposal in Fig. 1(b). Grandparent node 1 measures RTT to child nodes 5, 6 and 8. A node having the smallest round trip time holds a backup route to the grandparent. The second node has a backup route to the smallest RTT node, and the third node has a backup route to the second node. A node other than the smallest RTT node has the backup route to the next smaller RTT node than itself. In Fig. 1(b), if the ascending order of the nodes in round trip time is node 5, 6, 8, the smallest RTT node 5 has the backup route to node 1. The second node 6 has the backup route to node 5. The largest RTT node 8 has the backup route to the second node 6. We show the pseudo code of backup route calculation of our proposal in Fig. 2. This backup route calculation is carried out whenever a node joins, leaves and fails. Note that the backup route calculation is required only at the child layer of the departure node. It never goes down to the lower layers unlike in Yang's method.

Backup routes created in our proactive approach are certainly efficient as long as each node does not exhaust its degree. However it is possible that a node exhausts its degree by accepting a node rejoining in the backup route procedure. When this happens, a tree reconstruction procedure is invoked by the node itself in order to keep each node not to exhaust its degree. We show the procedure in Fig. 1(c). Node 2 uses up its degree because node 8 joined node 2 as its backup route. Node 2 sends a query to other child nodes, which are nodes 5, 6 and 7, and they reply hit or fail messages to node 8. The hit message means it can accept joining of new hosts. The fail message means it cannot accept. Node 8 moves to the node which has sent the hit message first. In Fig. 1(c), node 6 sends a hit message to node 8, and node 8 joins node 6. If all messages of the child nodes are failed, the newly connected node joins the node which it has received a message first from. Then, it receives a redirection message from the first node. We show the pseudo code of tree reconstruction of our proposal in Fig. 3.

Tree Reconstruction

```
1. Connect (BackupRoute)
2. if BackupRoute.Degree ==
   BackupRoute.MaximumDegree then
3.    Query (SiblingNodes)
4.    if Reply == HIT then
5.       JoinMessage (FirstHitNode)
6.    else
7.       JoinMessage(FirstReplyNode)
```

Fig. 2: Pseudo code of tree reconstruction.

Backup Route Calculation

```
// S is set of grandparent and the child nodes
1. SortedS ← Sort S in ascending order of RTT from
grandparent node
//{ SortedS[0] = grandparent node }
2. for i ← 1 to N do
3.  SortedS[i].BackupRoute ← SortedS[i-1]
4. end for
```

Fig. 3: Pseudo code of backup route calculation.

B. *Proactive Route Maintenance by Layered Coding*

Nodes in our proactive approach can hold their backup route nodes by forcing at least one degree reserved. However, this approach clearly causes increase of the tree depth and inefficiency of bandwidth utilization because the node degree is not always exhausted. To solve the problem, we adopt layered video coding as integration work of our own previous approaches [17][18].

In layered video coding, streaming data is divided into a base layer and enhancement layers. The media can be played by receiving the base layer only. Receiving the enhancement layers improves the media quality. Furthermore, we can design more flexible ALM protocols than the single rate case.

When using layered video coding, we need to extend the definition of node degree. We first assume that each stream is encoded into M layers and the rate of the m'th layer (m=1,…, M) is r_m. Then we can set the rates of multiple streams to be $\{r_1, r_2,\ldots, r_m,\ldots, r_M\}$, where r_1 is the rate of the base layer. Let B_i denote the outbound bandwidth of node i, the degree of the node i will be

$$D_i = \frac{B_i}{r_1} \tag{2}$$

The ratio of accumulative rates from layer 1 to m can be obtained by

$$R_m = \frac{\sum_{q=1}^{m} r_q}{r_1} \tag{3}$$

We assume that a node has n child nodes, and the child k receives layers 1 to m. Then the number of used degree of node i is

$$U_i = \frac{\sum_{k=0}^{n-1}\sum_{q=1}^{m} r_q}{r_1} \tag{4}$$

Nodes can receive the layers which they are willing to by using layered coding, but in this paper, nodes receive the full layers ordinarily and the layers are adjusted in tree reconstruction.

This proactive approach combined with layered video coding allows nodes to exhaust their degree as shown in Fig. 4. When a rate which is equivalent to degree 1 of the single rate case is 300kbps, it is divided into 100 kbps base layer and two 100 kbps enhancement layers. In Fig. 4 (a), the node has three child nodes and exhausts its degree. In Fig. 4(b), another node affected by a node departure tries to connect to the node in Fig. 4(a). In this transient period, the requested node then drops enhancement layers temporarily, and the connecting node can receive streaming data to play the media continuously. Later, the connecting node finds a new parent node and connects to it. The way of finding a new parent is the same as the case of Fig. 1(c). Until the finding process is completed, the connecting node can continue receiving streaming data.

Such a smooth transition is impossible by our previous degree-based approach because its decision is carried out according to whether the node can accept a request or not. Furthermore, this smooth transition enables to eliminate the tree depth problem and leads to reducing delivery delay of the overlay tree. In this way, layered video coding can indeed improves our proactive approach, in which nodes can hold backup routes with low overhead and utilize bandwidth efficiently.

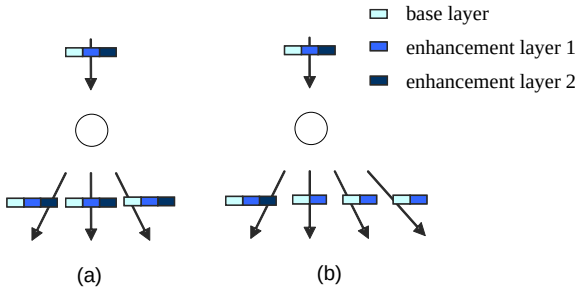


Fig. 4: Tree reconstruction using layered video coding.

C. *Tree Construction by Node Stability*

Our proactive route maintenance with layered video coding enables fast recovery of the overlay tree against node departures but does not contribute to stable overlay tree construction in which adequate node differentiation should be carried out. Therefore, in this subsection, we describe a tree construction method based on node stability, which brings long term stability to the overlay tree. Our basic idea is to place the nodes which stay the session for a long time at the upper layer of the overlay tree.

1) *Node Stability*

We define node stability as how long a node is expected to stay in an overlay tree in one session. This metric ranges from 0.00 to 1.00. Let S be the node stability of a specific node managed by the node itself, N be the number of session access records, T_k be the k-th staying time (from login to logout) to the session, W_k be the pre-assigned weight of the k-th session access record, and α be the constant number corresponding to 1.00 of the node stability. The calculating formula of node stability is

$$S = \sum_{k=1}^{N} \frac{W_k T_k}{\alpha} \tag{5}$$

where

$$\sum_{k=1}^{N} W_k = 1 \tag{6}$$

is satisfied.

2) *Rank*

The node stability is divided into some ranges, and each range is assigned a rank in order. If we have 4 ranges, the lowest range of node stability is rank 0, and the highest is rank 3. A node has its own node stability and corresponding rank. Then, the overlay tree is composed in decreasing order of the rank. We can construct a stable overlay tree by placing higher rank nodes at the upper layer of the tree and lower rank nodes at the lower layer.

3) *Configuration in Rank*

We construct an overlay tree in decreasing order of the degree like BO tree in case of the same rank nodes. This approach contributes to decreasing the tree depth. It decreases the number of nodes affected by node departures and the delay from a source node to client nodes. Exceptionally in the area of rank 0, however, we force nodes of rank 0 to connect to the overlay tree as leaf nodes when joining the session. These nodes are not

expected to stay much time in the session because some of them might join the session and leave immediately. On the contrary, compared with conventional methods, nodes of rank 0 can stay the session stably by avoiding the influence of departures of the nodes. We show the tree constructed by the rank and degree of node in Fig. 5, and the pseudo code for tree construction in Fig. 6.

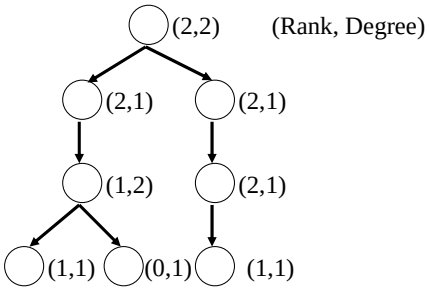


Fig. 5: Tree structure by rank and degree.

Tree Construction with Node Stability

1. If the degree is not exhausted then
2. accept the new node as a child node
3. else if the rank of a new node == 0 then
4. redirect the new node to the child nodes
5. else
6. // compare rank
7. if the rank of the new node > one of those of the child nodes then
8. switch from the child node of lowest rank to the new node
9. else if the rank == the rank of new node then
10. // compare degree
11. if the degree of the new node > one of those of the child nodes then
12. switch from the child node of lowest rank to the new node
13. else redirect it to the child nodes
14. else
15. redirect it to the child nodes

Fig. 6: Pseudo code of tree construction with node stability.

IV. PERFORMANCE EVALUATIONS

We evaluate the performance of our proactive methods by simulations and software implementations. We also evaluate performance of the tree construction with node stability by simulations.

A. Simulation Results

1) Proactive Route Maintenance by Node Degree

We firstly carried out simulations of our first proactive method, the reactive method and Yang's method by ns-2.26 [19]. Fig. 7 shows simulation topology, in which there are 24 routers, of which four routers are domain-to-domain routers and the others are set up at random between clients. The delay and bandwidth between the domain routers are 100ms and 100Mbps. The delay between the routers in a domain varies from 10 to 50 ms and the bandwidth is 100Mbps. The delay and bandwidth between a router and clients are 10ms and 10Mbps. The clients are randomly connected to one of the 20 routers except the four inter domain routers. The maximum degree of each node varies from 1 to 6. For the experiment, we fixed the degree of each node at a particular value. The total simulation time is 300 sec. In the beginning, all nodes join the tree, and after that each node randomly chooses to stay, leave, fail or re-join per 15 sec intervals.

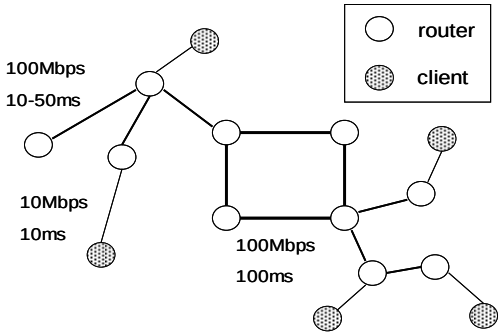


Fig. 7: Simulation topology.

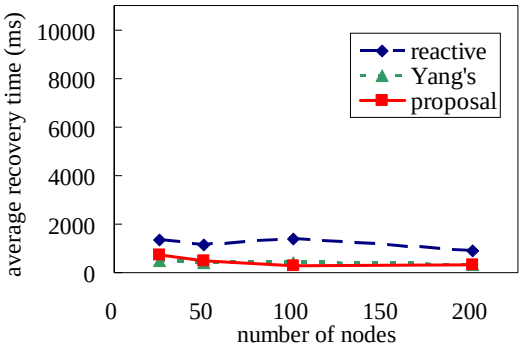


Fig. 8: Average recovery time of node leave case.

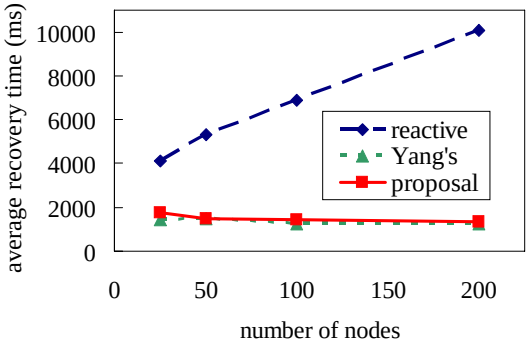


Fig. 9: Average recovery time of node failure case.

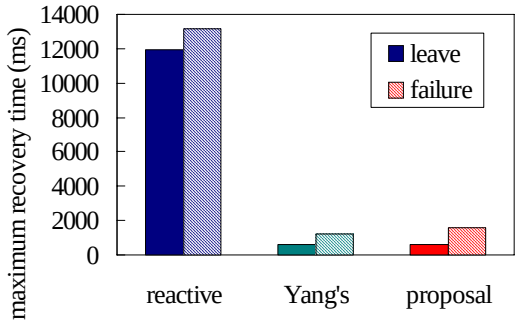


Fig. 10: Maximum recovery time of 200 node case.

a) Comparison of Recovery Time

Figs.8 and 9 plot the average recovery time of leave and failure when changing the number of participant nodes into the ALM tree, and Fig.10 shows the maximum recovery time of leave and failure cases when the number of nodes is 200.

In Fig.8, the average recovery time against node leaving in the reactive method is larger than those in proactive methods (our proposal and Yang's method) in each number of nodes. In Fig.9, as the number of nodes increases, the average recovery time of the reactive method becomes larger. The average recovery time of proactive methods are suppressed despite the failure case. In Fig.10, we can see the trend notably. The maximum recovery times of the reactive method are much larger than those of our proposal and Yang's method. The time of leave case of the reactive method is as large as that of failure case. This is because, when the node close to the root node leaves the session, many affected nodes are redirected from the root node to the edge node of the tree. On the contrary, the maximum recovery times of proactive methods keep small against those of the reactive method. This is because proactive methods can connect to the backup route node immediately both in leave and failure cases.

These facts can be quantitatively explained as follows. Assume that tree depth is *n*, round trip time between two nodes is *RTT* and detection time of leaving or failure (by heartbeat message) is Δt. In case of the reactive method, it is possible that *n* time redirections from the root happen. Therefore, maximum recovery time of the reactive method is expected to be $\Delta t + n \cdot RTT$ ms. In contrast, nodes of the proactive method can connect to the backup route nodes immediately, and the recovery time is about $\Delta t + RTT$ ms. This tendency especially holds for the maximum recovery time case and is observed in Fig.10, where *n* can be estimated by $n = \log_d N$ where *d* is the degree and *N* is the number of nodes.

b) Comparison of Control Overheads

Figs.11 and 12 show comparison of overheads among three methods, and Fig.13 shows ratio of the control packets. Overhead is the total amounts of bytes of all control packets to maintain the overlay tree. Control packets are composed of Join, Redirect, Leave, and Backup messages. Join is used for join request, Redirect is used in redirect procedure, Leave is used in noticing the node departure, and Backup is used to search backup routes. We assume the size of each packet is 128 bytes.

For the reactive method, control overheads come from the control messages exchanged by the affected nodes to find new parents. For the proactive method, the control messages consist of two parts. 1) Similar to the reactive method, control messages are exchanged by the child nodes of departure nodes to find their new parents, though the proactive method needs fewer steps. 2) In addition, every non-leaf node exchanges information for deciding a backup route.

We experimented with two redirection methods; a "round robin" method and a "round trip time" method. In the round robin method, when a node whose degree is exhausted receives a join message from a newly joining node, the node redirects the message to each of its child nodes in pre-determined order. In the round trip time method, when a node whose degree is exhausted receives a join message from a newly joining node, the node sends its children list to the newly joining node. Then the newly joining node measures the round trip time between each node on the list and itself, and sends a join message to the smallest round trip time node. It is expected that the round trip time method uses more packets than the round robin method, but the overlay tree is optimized to be low latency.

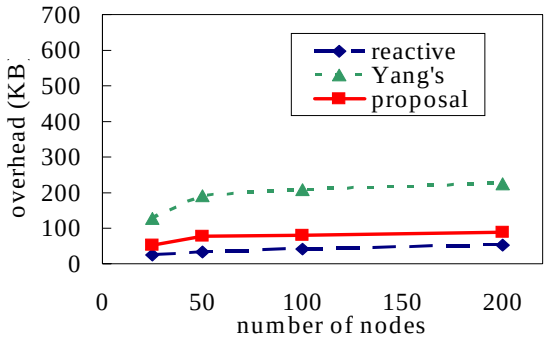


Fig. 11: Control overheads of the round robin method.

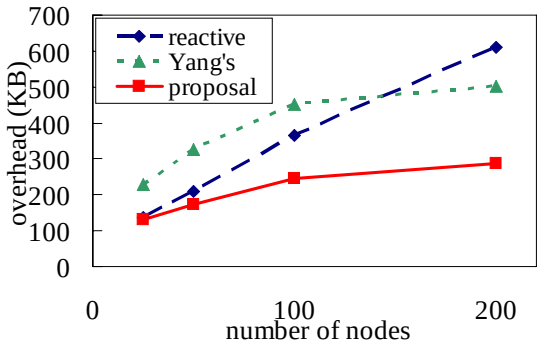


Fig. 12: Control overheads of the round trip time method.

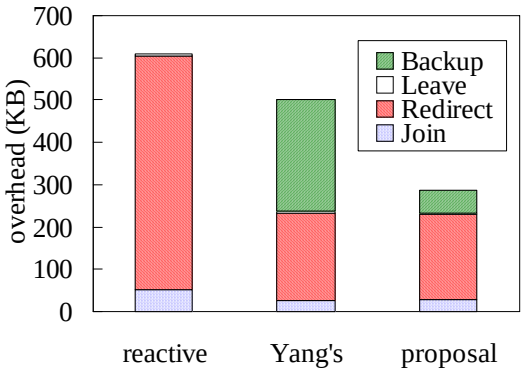


Fig. 13: Ratio of the control packets.

Fig.11 compares overheads of the round robin method for redirection, and Fig.12 compares the overheads of the round trip time method for redirection. As shown in these figures, our proactive method does not generate as many control packets as Yang's method for holding backup routes. This is because backup route search of our method is carried out locally. On the other hand, overheads of Yang's method and our proposal are larger than that of the reactive method in the round robin method, but the reactive method generates more packets than the proactive

methods in the round trip time method. This is because the reactive method using round trip time based redirection needs more overheads and is not scale to the session size. Fig.13 shows ratios of the control packets of the round trip time method for redirection of 200 nodes case. We can see that the reactive method generates many redirection messages, Yang's method generates many backup route messages but our approach can suppresses both of them.

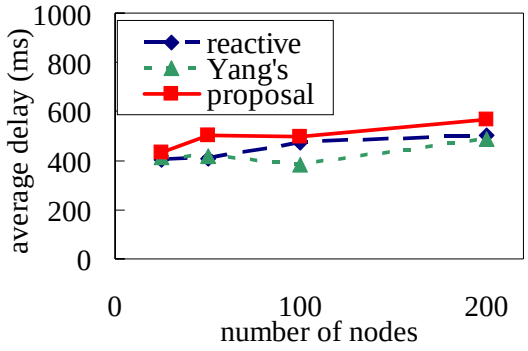


Fig. 14: Average delivery delay.

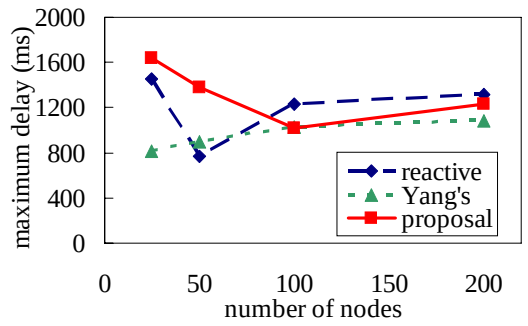


Fig. 15: Maximum delivery delay.

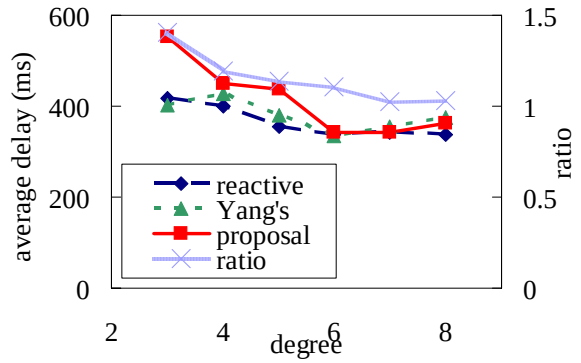


Fig. 16: Average delivery delay according to node degree.

c) *Comparison of Delivery Delays*

As expected, our proactive method simplifies the backup route search and successively contributes to overhead reduction. However, its structure causes depth of the overlay tree to be larger and possibly leads to overall delay increase. Accordingly, Fig.14 shows the average delay when the number of nodes in the tree changes, Fig.15 shows the maximum delay of the same case, and Fig.16 shows the average delay per node which have the same maximum degree in 200 nodes case. We apply the round trip time method for redirection in this experiment.

In Fig.14, we can see that the average delivery delay of our proposal is slightly larger than other methods. This is because our proposal does not exhaust node degree (for fast recovery against node departure). In Fig.15, we can see the same tendency for the maximum delay. However, when the numbers of nodes becomes larger (100 and 200), the maximum delay of our proposal becomes smaller than that of the reactive method. This is because the tree is constructed effectively enough by round trip time metric. In the reactive method, when node departure happens, the affected nodes send join messages to their grandparent nodes and redirection happens. When the redirected nodes have to connect to the leaf node of the tree due to degree exhaust, the maximum delay of the redirected node becomes large.

In Fig.16, we append analytical ratio of the delays which is derived as follows. Firstly, we assumed all the peer-to-peer delays are same and the tree is balanced. Then, analytical sum of the delay of arbitrary overlay tree is estimated by

$$\sum_{i=0}^{M} N_i \times i \qquad (7)$$

where N_i, M and i represent the number of nodes which stay at the i-th layer (where the root is 0 layer) of the tree, the maximum depth of the tree, and the delay of the nodes of the i-th layer. The ratio is the sum of the delay of our overlay tree divided by that of the ordinary overlay tree. Secondly, let k be the degree of each node. It is clear that the maximum number of nodes of our proactive method in the i-th layer is $(k-1)^i$, and that of Yang's method (or ordinary overlay tree) is k^i. Then, the analytical delay ratio is given by

$$\frac{(k-1)^i}{k^i} = \left(1-\frac{1}{k}\right)^i, \qquad (8)$$

which clearly approaches to unity as k (degree of each node) increases. This means that the difference of average delays between our proposal and other methods become small when the degree increases.

2) *Proactive Route Maintenance by Layered Coding*

We carried out simulations to verify performance improvement achieved by our second proactive method which is enhanced by layered video coding. The simulation topology is the same as in Fig.7. The maximum degree of each node varies from 1 to 3. In the beginning of simulation, all nodes join the tree, and after that each node leaves the tree every 5 sec randomly. Streaming rate is 300kbps total, which is divided into 100 kbps base layer and 2 enhancement layers of 100 kbps each. Later in this paper, we call our first approach without layered video coding by "Method I" (proposal) and the second approach with layered video coding by "Method II" (improved proposal). We show the simulation results in Figs.17 and 18.

a) *Comparison of Delivery Delays*

We show the average delivery delays of our two proposals in Fig.17. The delay of Method II is smaller than that of Method I. This is because layered video

coding enables our proposal to exhaust node's degree and contributes to reducing the depth of overlay tree. Also, the difference becomes large as the number of nodes increases. In ALM, it is probable that a large number of nodes join sessions to watch a popular content and this result is preferable especially in large sessions.

b) Comparison of Bandwidth Efficiency

We then show the bandwidth utilizations of our two proposals in Fig.18. The bandwidth utilizations are measured at intermediate nodes in the overlay trees. Layered video coding enables us to use bandwidth more efficiently than Method I. This is because nodes of Method II can exhaust their degree and control the rate efficiently by selecting adequate layers. Note that, in Method I, the case that nodes exhaust their degree happens only in tree reconstruction.

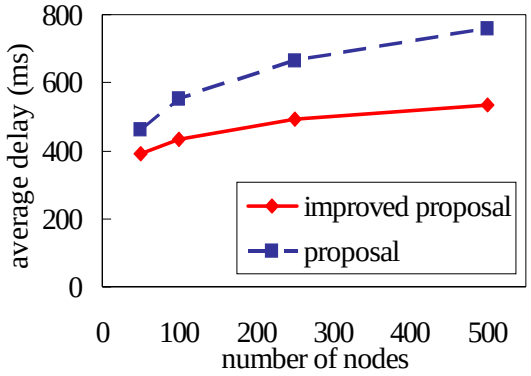


Fig. 17: Comparison of average delivery delays.

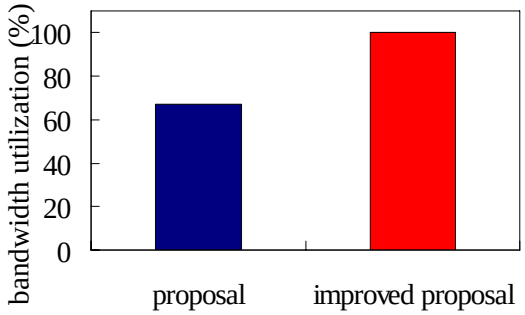


Fig. 18: Comparison of bandwidth utilizations.

3) Tree Construction by Node Stability

We carried out final simulations of the tree construction method using node stability. We also compare performances of PeerCast, BO tree and BTP tree methods by using ns-2.28 [19]. We show the simulation result in Fig.19. We used bounded Pareto distribution to assign degrees to nodes. The upper bound, the lower bound and a shape of the distribution are 6, 3 and 1.2, respectively. Each node is assigned a fixed probability at random as a rate parameter of session staying time, which is used to decide whether or not it leaves the overlay tree at the decision interval. The decision interval is from 400 to 500 sec. Each node joins the session at random before 2000 sec. A node decides whether to leave the session every decision interval after it joins the session. A node which left the session joins the session again 1000 sec after. We divide the node stability to four ranks, 0, 1, 2 and 3, which correspond to 0.00 to less than 0.33, 0.33 to less than 0.66, 0.66 to less than 0.99 and 1, respectively. We compared the number of nodes which are affected by node departures. In this comparison, there are 400 nodes in the session and, around 10000 sec, new 400 nodes join the session intensively (i.e. flash crowds). Around 13600 sec, the nodes stay in the session out of the new 400 nodes leave the session. They never return to the session after they leave. In our proposal, we set parameters of subsection III.C to N=3, $\alpha = 3600$ and $W_k = 1/N$ according to auxiliary experiments, and stability update is done per 1200 sec by each node. The reason we set α at 3600 sec (1 hour) is that the nodes which have the rate parameters in rank 2 range are expected to stay for about 3000 sec in average in our auxiliary experiments. We also evaluated the number of nodes affected by node departures when α is 3600, 7200 and 10800. When α is 3600, the result is best and the number of rank 2 nodes affected by node departures is the fewest.

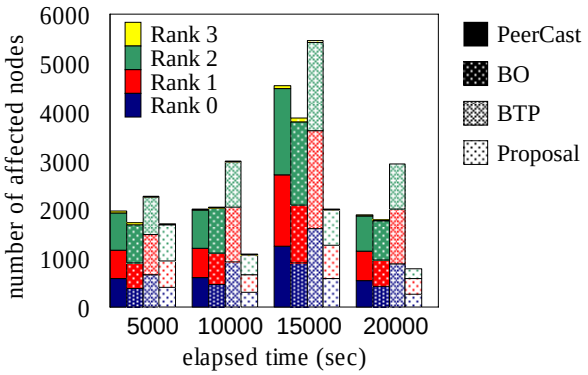


Fig. 19: Number of affected nodes by node departures.

a) Comparison of Number of Nodes Affected by Node Departures

We show the number of nodes affected by node departures in Fig.19. The numbers are total of the affected nodes every 5000 sec. Affected nodes are nodes in the subtree of the departure nodes. Nodes are marked with each rank by their rate of staying assigned at random. We also show the ratio of affected nodes with each rank in Fig.19. From 0 to 5000 sec, the number of affected nodes of each method is similar. This is because all nodes' rank are 0 in their first join in our proposal. From 5000 to 10000 sec, the number of affected nodes of our proposal becomes smaller. From 10000 to 15000 sec, the affected nodes of the methods other than our proposal increase. They are affected seriously by the bursty joins of new participant nodes around 10000 sec. In our proposal, already connected nodes are not affected severely because rank of new participant nodes is 0 and they can connect to the overlay tree as leaf nodes only. Besides, since the high rank nodes already connect to the nodes at upper layer, they are not connected to by the new nodes.

In this figure, BTP tree shows the largest affected nodes in compared methods. This is because nodes of BTP tree leave the session after going to the upper layer of the overlay tree according to the BTP switching procedure. In every entering the session, nodes connect to the tree as leaf nodes, and long time staying nodes are affected many times before they climb to the upper layer of the overlay tree. BO tree also shows larger affected nodes than our

proposal. This is because short time staying nodes which have large degree connect to the intermediate of the overlay tree. The influence is so large that we do not see the contribution of tree depth reduction in the result. PeerCast also shows larger affected nodes than our proposal. This is because long time staying nodes are affected many times before they stay at upper layer of the overlay tree. By this result, we can confirm that our proposal constructs the most stable overlay tree which is furthermore robust against bursty joins.

B. *Implementation Results*

In addition to the simulations above, we implemented software prototypes for proactive route maintenance and compared three methods; the reactive method and our two proactive methods (Methods I and II), over actual networks. All the prototypes are written by C++ and operated on Windows XP. We use ITU-T H263+ video codec. We then show several implementation results below.

1) *Proactive Route Maintenance by Node Degree*

We compare the reactive method and Method I. Maximum degree of each node is fixed at 3. Total 25 nodes are deployed over three different networks and each network connects to the backbone in Japan. We can expect the backbone to have high bandwidth. Firstly, all nodes join the ALM session, and then each node joins or leaves randomly for 30 minutes.

a) *Comparison of Recovery Time*

Fig.20 shows average and maximum recovery times of the case of 25 nodes. Recovery time of our proposal is less than half of the reactive method value. This point is the same as in simulations. As compared to the reactive method, we can also confirm that the media playback subjective quality of our proposal was much better than that of the reactive method when node departures happen. In the reactive method, playback was felt like “freeze frame” for a moment, but in our proposal, decoded pictures had continued to be played smoothly.

b) *Comparison of Control Overheads*

Fig.21 represent the overheads when the number of nodes is 15 and 25 in the implementations. Control packets are composed of the same messages of the simulations and the size of each message is 50 bytes. In this experiment, we used the round trip time method for redirection. As the number of nodes increases, overhead of the reactive method increases. This trend is the same as the most-left result in Fig.12.

c) *Comparison of Delivery Delays*

Fig.22 shows the average and maximum delivery delays when the number of nodes in session is 25. The delay is more in our proposal than the reactive method. However, in media playback, we do not feel any difference between our proposal and the reactive method. We think that current difference is not so critical and within the human perception limitation.

2) *Proactive Route Maintenance by Layered Coding*

We used JGN Ⅱ of Japan, which is a network backbone testbed for research and development. In the network, there are two local networks. The delay is about 30 ms from one network to the other. We used I pictures as base layer and P pictures as enhancement layer to design layered video coding. The number of nodes is 10. There are 5 nodes in each network. The degree of each node is 3.

a) *Comparison of Delivery Delays*

Fig.23 shows the delivery delays of our two proposals. The delay of Method II is smaller than that of Method I as expected from the simulation results. Furthermore, we implemented visualization software by Java which visualizes overlay tree structure. Using this software, we confirmed that the tree depth of Method II was smaller than that of Method I.

b) *Comparison of Bandwidth Efficiency*

Fig.24 shows the bandwidth utilizations of our two proposals. The bandwidth utilizations are measured at intermediate nodes in the overlay trees. Method II can use bandwidth more efficiently than Method I as expected from the simulation results.

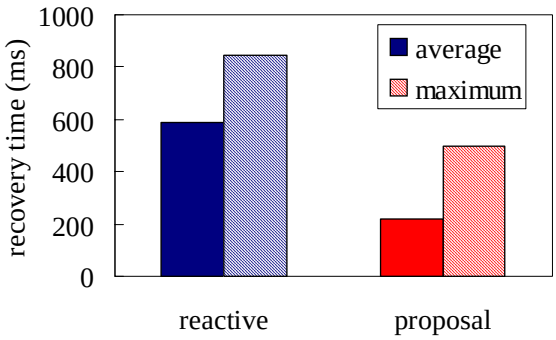


Fig. 20: Comparison of recovery delays.

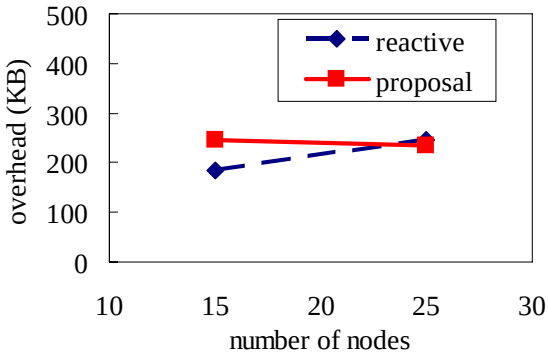


Fig. 21: Comparison of control overheads.

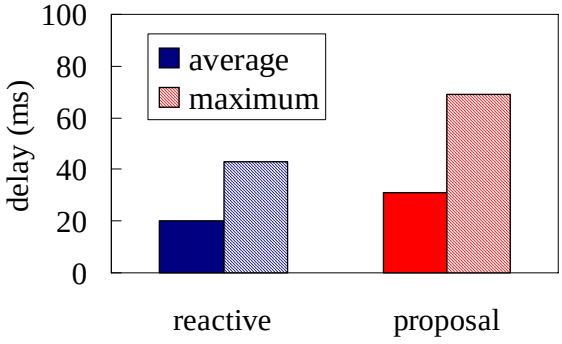


Fig. 22: Comparison of delivery delays.

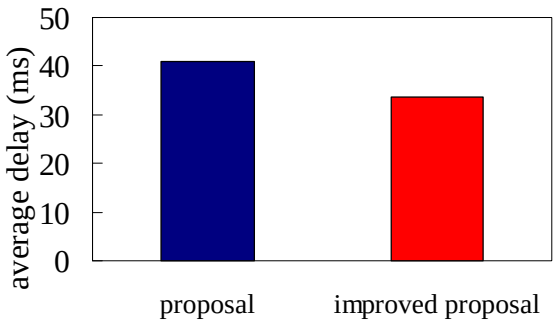


Fig. 23: Comparison of average delivery delays.

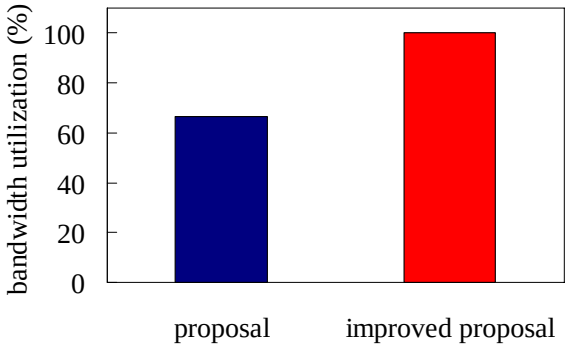


Fig. 24: Comparison of bandwidth utilizations.

V. Concluding Remarks

In this paper, we proposed three kinds of methods to improve robustness and stability of the ALM overlay tree. Firstly, we presented a novel method of proactive route maintenance exploiting node degrees. It enables fast recovery from node departures and reduction of control overheads. In comparison with conventional methods, our proposal demonstrates much faster recovery than the reactive method, and almost same as Yang's proactive method. Control overhead of our proposal is less than Yang's method and, in the specific case, it is less than the reactive method. In our proposal, backup route construction is done in the local area, and the control overhead does not heavily depend on session size. We also realized that media playback quality of our proposal was much better than that of the reactive method when node departures happen. Secondly, we improved our proactive route maintenance by introducing layered video coding. Layered video coding enables our proposal to exhaust node's degree. This contributes to reducing depth of the overlay tree or delivery delay, and to efficient bandwidth utilization. These expectations are confirmed by experiments. Thirdly, we proposed a tree construction method using node stability, which is analogous to incentives [20]. The number of nodes affected by node departures can be smaller than those of the methods which do not consider node staying time to sessions. We then confirm that our approach constructs a stable and robust overlay tree.

As future work, implementation evaluation of the node stability should be carried out, and more global performance evaluations should be conducted over the world-wide testbed such as PlanetLab or the public Internet. Furthermore, since there are a lot of proposals to improve resilience and efficiency of the P2P streaming such as mesh overlays like swarming, FEC and network coding as in [21], extensions of our work to cover recent innovations should be considered.

References

[1] S. Deering, "Host Extension for IP Multicasting," RFC 1112, August 1989.

[2] D. Pendarakis, S. Shi, D. Verma, M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," Proc. USENIX USITS 2001, 99.49-60, March 2001.

[3] Y. Chu, S. G. Rao, H. Zhang, "A Case for End System Multicast," Proc. ACM SIGMETRICS 2000, pp.1-12, June 2000.

[4] Y. Chawathe, S. McCanne, E. Brewer, "Scattercast: An adaptable broadcast distribution framework," ACM Multimedia Systems Journal, vol.9, no.1, pp.104-118, July 2003.

[5] X.Zhang, J.Liu, B.Li and T.Yum, "DONet/CoolStreaming: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming",. Proc. IEEE INFOCOM vol.3, pp.2102-2111, Apr.2005.

[6] P. Francis, "Yoid: Extending the Internet Multicast Architectuire," http://www.icir.org/yoid/, April 2000.

[7] J. Jannotti, D. Gifford, K. Johonson, M. Kaashoek, J. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network," Proc. 4th Symposium on Operating Systems Design & Implementation, October 2000

[8] H. Deshpande, M. Bawa, H. Garcia-Molina, "Streaming Live Media over Peers," Technical Report 2002-21, Stanford University, March 2002

[9] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications," Proc. IEEE INFOCOM 2003, vol.2, pp.1521-1531, April 2003.

[10] D. Tran, K. Hua, T. Do, "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming," Proc. IEEE INFOCOM 2003, pp. 1283-1292, April 2003.

[11] S. Banerjee, B. Bhattacharjee, and C kommareddy, "Scalable application layer multicast," Proc. ACM SIGCOMM 2002, pp.205-217, Aug 2002.

[12] S. Banerjee, S. Lee, B. Bhattacharjee, A. Srinivasan, "Resilient multicast using overlays," Proc. ACM SIGMETRICS 2003, June 2003.

[13] M. Yang, Z. Fei, "A Proactive Approach to Reconstructing Overlay Multicast Trees," Proc. IEEE INFOCOM 2004, March 2004.

[14] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high-bandwidth multicast in cooperative environments" Proc. ACM SOSP 2003, October 2003.

[15] G. Tan, A. Jarvis, P. Spooner, "Improving the Fault Resilience of Overlay Multicast for Media Streaming," Proc. DSN 2006, June 2006

[16] Y.Kunichika, T.Kusumoto, J.Katto and S.Okubo, "Application Layer Multicast with Proactive Route Maintenance over Redundant Overlay Trees", Proc. Packet Video 2004, Dec.2004.

[17] T.Kusumoto, J.Katto and S.Okubo, "Tree-Based Application Layer Multicast using Proactive Route Maintenance and Its Implementation", Proc. ACM P2PMMS 2005, Nov.2005.

[18] Y.Okada, M.Oguro, J.Katto and S.Okubo, "A New Approach for the Construction of ALM Trees using Layered Video Coding", Proc. ACM P2PMMS 2005, Nov.2005.

[19] The Network Simulator ns-2, http://www.isi.edu/nsnam/ns

[20] Bram Cohen, "Incentives Build Robustness in BitTorrent" 2003. http://bittorrent.com/bittorrentecon.pdf.

[21] V.Fodor and G.Dan: "Resilience in Live Peer-to-Peer Streaming", IEEE Commun. Mag., Vol.45, No.6, pp.116-123, June 2007.