Robust and Efficient Stream Delivery for Application Layer Multicasting in Heterogeneous Networks

Masahiro Kobayashi, *Student Member, IEEE*, Hidehisa Nakayama, *Member, IEEE*, Nirwan Ansari, *Fellow, IEEE*, and Nei Kato, *Senior Member, IEEE*

***Abstract*—Application Layer Multicast (ALM) is highly expected to replace IP multicasting as the new technological choice for content delivery. Depending on the streaming application, ALM nodes will construct a multicast tree and deliver the stream through this tree. However, if a node resides in the tree leaves, it cannot deliver the stream to its descendant nodes. In this case, Quality of Service (QoS) will be compromised dramatically. To overcome this problem, Topology-aware Hierarchical Arrangement Graph (THAG) was proposed. By employing Multiple Description Coding (MDC), THAG first splits the stream into a number of descriptions, and then uses Arrangement Graph (AG) to construct node-disjoint multicast trees for each description. However, using a constant AG size in THAG creates difficulty in delivering descriptions appropriately across a heterogeneous network. In this paper, we propose a method, referred to as Network-aware Hierarchical Arrangement Graph (NHAG), to change the AG size dynamically to enhance THAG performance, even in heterogeneous networks. Finally, we evaluate the proposed scheme by experiments using the network simulator ns-2. By comparing our proposed method to THAG and SplitStream, we show that our method provides better performance in terms of throughput and QoS. The results indicate that our approach is more reliable than other methods in heterogeneous networks.**

***Index Terms*—Application layer multicast, content delivery, multiple description coding, streaming contents.**

I. Introduction

RECENTLY, in response to the rapid growth in network speed and bandwidth, deployment of streaming content delivery systems has been increasing. Most streaming technologies currently in use in the Internet are based on unicast communication. However, since unicast based streaming communication increases the traffic load of the server and network, research has recently been redirected towards the multicast communication based streaming [1]. Most multicast communication based on IP multicast incurs a significant cost. Therefore, as an alternative to IP multicast, Application Layer Multicast (ALM) [2] has drawn much attention for its promise to benefit future Internet infrastructure.

ALM can be categorized into infrastructure-level ALM and end-system-level ALM [3]. In IP multicasting, the duplication and relay of packets are done at the router level. In contrast, infrastructure-level ALM does it in a similar way by using servers or proxies connected together via unicast, and end-system-level ALM uses the end-nodes for forwarding. If we apply ALM, we can virtually perform multicast communication at the application layer. In this paper, we focuses on designing an end-system-level ALM protocol. In end-system-level ALM systems, the multicast tree is rooted at the media server, and participating nodes join the tree as interior and leaf nodes. An interior node is responsible for forwarding data from its parent node to its children through unicast. Additionally, although IP multicasting requires specialized hardware, ALM does not; however, duplication and relay of packets performed by the end-nodes are generally less reliable than that performed by special routers. Therefore, end-system-level ALM needs to address the following issues.

First, since nodes are free to join and leave the service at any time, the number of "currently active" nodes is unpredictable. The departure of interior nodes in the multicast tree severely affects the descendant nodes causing the reliability of a multicast service to be greatly susceptible to node dynamics. Second, the propagation delay from media source to participating node may be excessive because the data is forwarded by a number of interior nodes along the multicast tree. Since end-nodes in ALM do not have the routing information available to routers, the multicast trees built in ALM suffer from the increase of propagation delay and the inefficient usage of bandwidth as compared to IP multicast.

In order to cope with these problems, multiple-tree multicast was proposed [4]–[7]. This method splits the original data stream into several descriptions with Multiple Description Coding (MDC) [8], [9] and delivers the descriptions by using multiple multicast trees in parallel. In MDC, we can playback the contents by receiving one of the descriptions. Higher quality can be achieved by obtaining more descriptions. MDC is emerging for practical use. Several MDC techniques [10], [11] have been proposed for the H.264/AVC video coding standard [12]. Recently, H.264/AVC has been very successfully deployed in many applications including television broadcasting,

Manuscript received June 04, 2008; revised September 15, 2008. First published December 16, 2008; current version published January 09, 2009. This work was supported in part through the strategic international cooperative program between the Japan Science and Technology Agency and the National Science Foundation Cyber Trust under Grant 0726549. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christophe De Vleeschouwer.

M. Kobayashi and N. Kato are with the Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan (e-mail: kobayasi@it.ecei.tohoku.ac.jp; kato@it.ecei.tohoku.ac.jp).

H. Nakayama is with the Department of Electronics and Intelligent Systems Faculty of Engineering, Tohoku Institute of Technology, Sendai 982-8577, Japan (e-mail: hidehisa@m.ieice.org).

N. Ansari is with the Advanced Networking Laboratory, Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: nirwan.ansari@njit.edu).

Digital Object Identifier 10.1109/TMM.2008.2008933

video streaming over packet networks, and digital media storage. In [7], the approach which uses a Topology-aware Hierarchical Arrangement Graph (THAG) has been proposed for the construction of multiple multicast trees. In THAG, all participating nodes are divided into a number of Arrangement Graphs (AGs) [13], [14], and several node-disjoint multicast trees are embedded in each AG. An AG is an undirected graph which possesses desired properties for overlay topology. "Node-disjoint" means that any node serves as an interior node in only one tree.

However, in THAG, it is difficult to deliver descriptions which can fit the various bandwidth constraints. The nodes and network infrastructure are heterogeneous in large scale ALM systems. The service capability of an interior node is subject to the available network bandwidth. Moreover, different network links exhibit different characteristics, such as bandwidth and delay, which greatly affect the Quality of Service (QoS). In THAG, the required node upload bandwidth needed for the minimum transmission is determined by the AG size and the streaming rate. Hence, if the actual upload bandwidth of a node which is trying to join an ALM is less than the required upload bandwidth, it will not be able to send all the descriptions. As a result, the QoS of the stream will be degraded depending on the amount of the descriptions which cannot be delivered. This problem is attributed to the characteristic that THAG uses a constant AG size.

Therefore, we propose a method to ensure the QoS of the received stream by dynamically changing the AG size (when a node joins or leaves the multicast) according to the available bandwidth and by preventing nodes from disabling participation in forwarding the stream. Furthermore, each node calculates the requested size based on the available bandwidth and searches for an existing AG with the appropriate size. By so doing, each node can receive the appropriate number of descriptions in heterogeneous networks. Our simulation results using network simulator ns-2 [15] demonstrate that our approach provides better performance in terms of throughput and QoS than those of the conventional THAG and SplitStream [6] approaches. The results indicate that our approach is more reliable in heterogeneous networks.

The rest of the paper is organized as follows. In Section II, we provide an overview of the conventional ALM tree construction method. Section III describes ALM which uses THAG. Our proposed ALM method is described in Section IV. In Section V, we present our simulation results and performance comparisons. Concluding remarks are given in Section VI.

II. Related Work

In end-system-level ALM, an overlay network is constructed at the application layer independently from the network layer by letting each end-node forward streaming data. A multicast tree is created by having the end-nodes (which are responsible for duplication of the received media stream) acted as a branch. The stream delivery by ALM flows through the multicast tree, and the root acts as the source node which owns and disseminates the media data. A significant amount of research efforts have been directed toward application layer multicast throughput and QoS improvement. Existing works can be roughly classified into single-tree multicast and multiple-tree multicast.

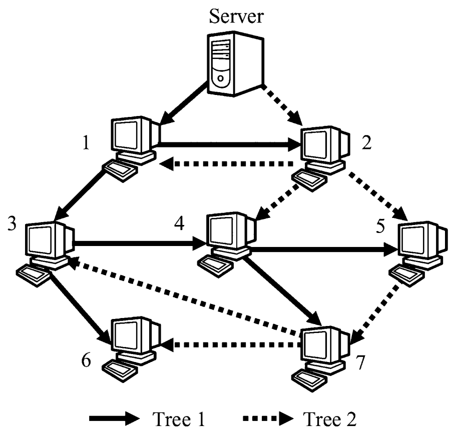

Fig. 1. Node-disjoint multicast trees.

A. Single-Tree Multicast

Many of the existing works have advocated on building a single-data distribution tree rooted at the media data originator (the sender). Therefore, each receiver has only one path from the sender along the tree. So far, Yoid [16], SpreadIt [17], ALMI [18], HBM [19], NICE [20], ZIGZAG [21], and Scribe [22] have been proposed. In Yoid [16] and SpreadIt [17], a Shortest Path Tree (SPT) constructs the minimum delay path from a source node to all its receivers. These protocols use SPT in order to construct a source-specific multicast tree (referred to as a "rooted tree" in Graph Theory). ALMI [18] and HBM [19] do not address the node bandwidth capacity issue, and instead try to only construct a low cost tree, also called a Minimum Spanning Tree (MST). Given a graph with a cost associated with each edge (usually delay), a MST is a tree with a minimum total cost spanning all the members. NICE [20] and ZIGZAG [21] construct a cluster of nodes that can be used to construct trees in order to better organize the overlay tree and reduce control message overhead. These protocols construct a hierarchical cluster of nodes with each cluster having a head at the higher layer. The advantages of a hierarchical clustering approach to multicast tree routing are the reduction in control overhead (nodes keep state information for only a subset of other nodes), and faster joining and management of the tree (at the cost of a suboptimal tree). The disadvantage is a lack of hard guarantees on the bandwidth capacity limitation of each node. Scribe [22] operates based on an existing peer-to-peer substrate (Pastry [23]) that serves as a mesh on top of which an overlay multicast tree can be constructed by using a reverse-path forwarding scheme. The advantage of this approach includes low control overhead and distributed management of the multicast tree, but it does not restrict the bandwidth capacity of each node, and is thus suboptimal.

B. Multiple-Tree Multicast

To utilize path diversity for improving reliability and QoS of streaming, multiple-tree multicast schemes have been proposed. Multiple-tree multicast constructs multiple paths between the

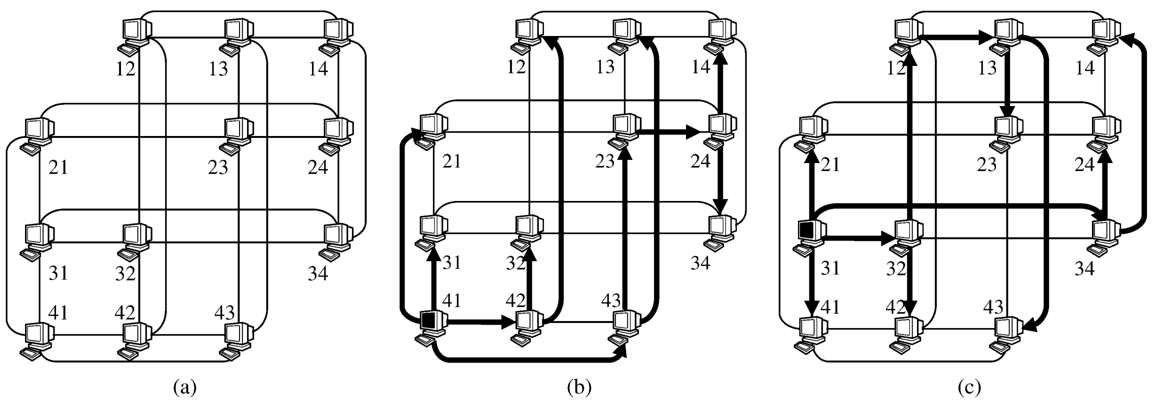

Fig. 2. Tree structure based on arrangement graph: (a) arrangement graph with size of 4, (b) multicast tree rooted at node 31, and (c) multicast tree rooted at node 41.

root and each receiver, and delivers descriptions by using Multiple Description Coding (MDC) [8]–[11]. MDC is able to split original streaming media into several descriptions. We can playback the contents by receiving one of the split descriptions, and higher quality can be achieved by obtaining more descriptions. So far, CoopNet [4], [5], Splitstream [6], and THAG [7] have been proposed.

CoopNet [4], [5] proposes a centralized algorithm to facilitate deployment of multiple-multicast trees from different sources, and does not have explicit mechanisms to maximize bandwidth. In contrast, SplitStream [6] has proposed a decentralized algorithm to construct a forest of multicast trees from a single source. SplitStream is based on Scribe [22], a tree-based multicast algorithm based on structured overlay networks. There are two fundamental differences between CoopNet and SplitStream. First, CoopNet uses a centralized algorithm (running on the server) to build the trees while SplitStream is completely decentralized. Second, CoopNet does not attempt to manage the bandwidth contribution of individual nodes. However, it is possible to add this capability to CoopNet [6].

Both CoopNet and SplitStream do not ensure the construction of node-disjoint multicast trees, implying a node can be an interior node in several multicast trees and its departure will prevent the descendant nodes from receiving descriptions [7]. In the THAG [7] scheme, node-disjoint multicast tree construction is ensured. Construction of node-disjoint multicast trees guarantees that the departure of any node will only affect data delivery in at most one multicast tree. However, THAG does not manage the bandwidth contribution of each node, and therefore it is difficult for THAG to deliver descriptions which can meet the various bandwidth constraints in heterogeneous networks. We have studied THAG in great details, and modified/tailored it for heterogeneous networks.

III. THAG

R. Tian *et al.* [7] proposed THAG to construct multiple node-disjoint multicast trees. The node-disjoint trees can be constructed by making a node which is a parent node in the specific tree into a leaf node on all other trees, as shown in Fig. 1. For example, node 1 is an interior node in tree 1 and leaf node in tree 2. By so doing, even when a node cannot receive the description due to the departure of a node in its upper position, the descendant node can still receive the descriptions from other trees. In THAG, participating nodes are grouped into a number of Arrangement Graphs (AGs) [13], [14]. In each AG, several node-disjoint multicast trees are embedded. Once embedded, THAG will assemble the AGs into a tree-like hierarchical structure.

A. *Node-Disjoint Trees in Hierarchical Arrangement Graph*

THAG uses an AG to construct node-disjoint trees. An AG is an undirected graph and has desired properties, such as symmetric vertex, symmetric edge, strong resilience, and maximal fault-tolerance [14]. An AG is denoted by $A_{s,k}$, and specified by integers s and $k (1 \leq k \leq s)$. Denote $\langle s \rangle = \{1, 2, \ldots, s\}$. There are k symbols denoted as $X = x_1 x_2 \ldots x_k; x_i$ refers to the ith element of X. $A_{s,k}$ defined in [13] is an undirected graph (V, E) defined as follows:

$$\begin{aligned} V &= \{X = x_1 x_2 \ldots x_k \mid x_i \in \langle s \rangle \text{ and } x_i \neq x_j \text{ for } i \neq j\}, \\ E &= \{(X, Y) \mid X, Y \in V \text{ and for some } i \in \langle k \rangle, \\ &\quad x_i \neq y_i \text{ and } x_j = y_j \text{ for } j \neq i\}. \end{aligned}$$

From this definition, X and Y differ in one position only. Therefore, an edge of $A_{s,k}$ connects neighboring nodes which differ in exactly one of their k positions from each other.

THAG constructs $s-2$ node-disjoint trees from $A_{s,2}$. In this paper, we call s the AG size. Generally, in an AG with size $s, s(s-1)$ number of nodes can participate. Fig. 2(a) shows an example when the AG size is 4, while Fig. 2(b) and (c) are examples of trees based on a size 4 AG. In these figures, the root of each tree is node $i1 (3 \leq i \leq s)$, and the two trees which have the root nodes 31 and 41 have been constructed. In these trees, we can see that the node which is the parent node in one tree is the leaf node in another tree. Therefore, these two trees are node-disjoint. Node 21 is a leaf node in all multicast trees, and so it will be selected as the AG entrance and will maintain the current states of all its AG members.

Furthermore, more nodes can join the AG in a hierarchical manner. When the number of nodes participating in an AG reaches the limitation, that AG is made into a parent-AG, which can spawn new child-AGs. As shown in Fig. 3, the parent-AG derives two child-AGs after it is filled. Nodes 32 and 42 in the parent-AG serve as the source nodes that forward

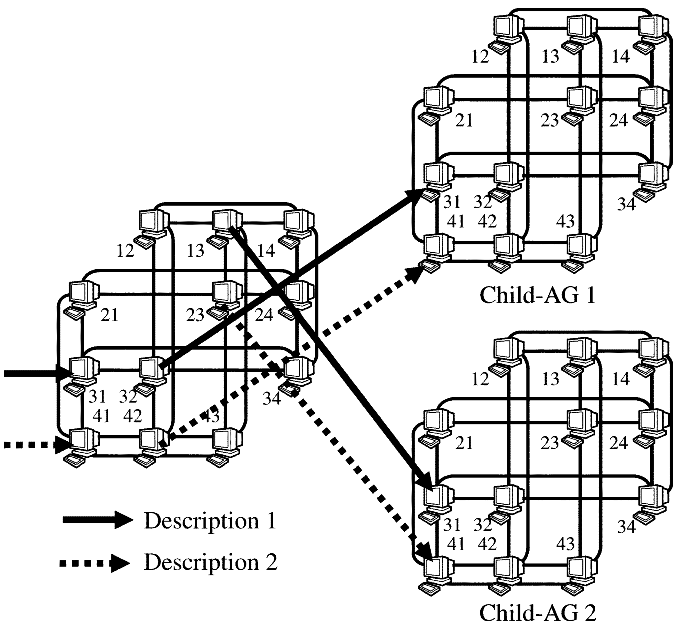

Fig. 3. Hierarchical AG in THAG.

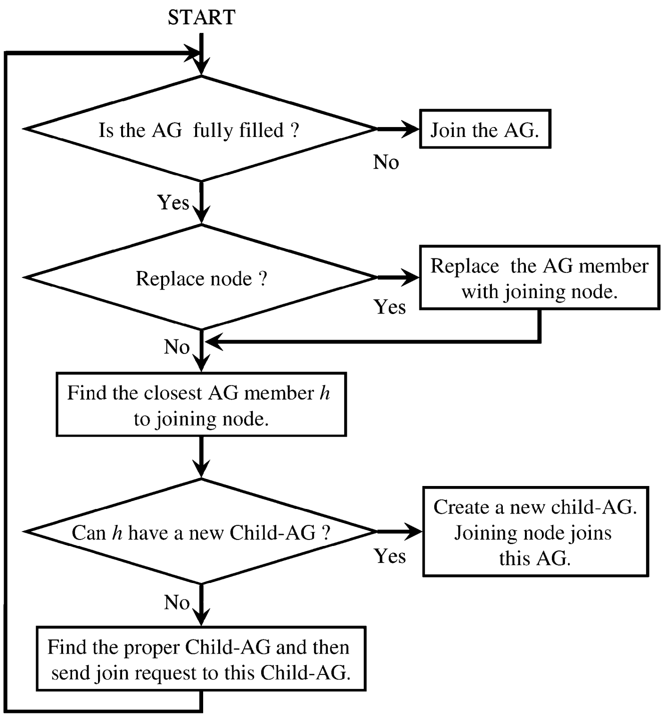
START
Is the AG fully filled ?
No
Join the AG.
Yes
Replace node ?
Yes
Replace the AG member with joining node.
No
Find the closest AG member *h* to joining node.
Can *h* have a new Child-AG ?
Yes
Create a new child-AG. Joining node joins this AG.
No
Find the proper Child-AG and then send join request to this Child-AG.

Fig. 4. Node joining procedure in THAG.

corresponding descriptions to child-AG 1. Similarly, nodes 13 and 43 in the parent-AG serve as the source nodes for child-AG 2. Suppose that a node acts as an interior node of one tree in a parent-AG, then it will also act as the source node for a child-AG. In this way, the node-disjointness of multicast trees is preserved. In general, nodes in each column deliver descriptions to child-AGs. A column of nodes in the parent-AG providing data to its child-AG is referred to as the AG source. The descriptions which are delivered to the parent-AG are also delivered to child-AG 1 and child-AG 2 as well. In other words, since the delivery of descriptions is performed based on the delivery tree constructed from the AG, we can easily achieve a large-scale delivery network in a hierarchical manner.

B. Node Joining Procedure

Here, we describe the process of joining nodes. With THAG, network coordination technology, such as Global Network Positioning (GNP) [24] and Vivaldi [25], is used in the node joining procedure. Network coordinates are obtained by mapping complex Internet topologies into simple geometric space.

At the beginning, the node that wishes to join, will first send a join message to the highest AG it can enter. If the AG is not fully filled, the node will join that AG. Otherwise, if the AG is fully filled, for each AG member e which already joins the AG, compute the function $G(e)$ which is the ratio of the sum of distances between node e and the AG sources to the sum of distances between the joining node and AG sources as follows:

$$G(e) = \frac{\sum_i d(e, s_i)}{\sum_i d(n, s_i)} \quad (1)$$

where n denotes the joining node, s_i denotes the ith AG source of each description, and $d(a, b)$ is the distance from node a to node b defined in network coordinates. Note that $G(e) > 1$ implies that the joining node is closer to the AG sources than node e. Therefore, node e with maximum $G(e)$ is replaced by the joining node n. Node e that is replaced will try to find a new child-AG. On the other hand, for all member nodes e, no replacement is performed if $G(e) \leq 1$. In this case, n will try to find a new child-AG. Next, the AG entrance that received the join message finds the closest AG member h to the joining node. The joining node contacts h and retrieves the information about all its child-AGs. If h has less child-AGs than it can serve, the joining node creates a new child-AG and joins the AG entrance. Otherwise, the joining node contacts all the child-AGs' entrances, and selects and joins the AG that has the smallest average distance between the joining node and the AG members.

In [7], this procedure is called the Locating Replacing Sinking (LRS) algorithm (Fig. 4). By repeating these procedures, the joining node eventually joins the closest child-AG.

C. Node Leaving Procedure

In this subsection, we describe the process of a node leaving a tree. If the leaving node is not at the AG entrance, its parent node in the same tree will undertake the position's tasks. If the leaving node is at the AG entrance, its parent node should collect information about the AG structure and takeover the function as the entrance. Moreover, if a node which has child-AGs leaves, its child-AGs will promote a non-root node in the child-AG to replace the leaving node. In the child-AG, similar maintenance can be performed afterward. Thus, the height of a THAG can be reduced as much as possible. In this way, if nodes leave from a tree, THAG can rapidly repair the structure and the media service can be quickly restored.

IV. The Proposed Method

A. *Shortcoming of THAG*

In THAG, we can create several node-disjoint multicast trees from an AG. The descriptions are delivered by using these created multicast trees in parallel. However, to deliver the descriptions stably, the node at the AG source can have no more than $2(s-2)$ child nodes at the maximum, and other nodes in the AG have to deliver descriptions to child nodes in the AG and the root nodes in child-AGs. Furthermore, there is a chance that every node may become an AG source. Therefore, the minimum bandwidth needed for streaming delivery is determined based on the AG size s and the streaming rate r, which is $2(s-2)r$. For example, assume the required bandwidth is 6 Mbps for an AG which has size 8 and streaming rate of 500 kbps. Nodes which connect with a bandwidth link less than this amount may not be able to send all descriptions. In a conventional THAG, the required bandwidth is not taken into account because it is assumed that the AG size is fixed and all descriptions consume the same amount of bandwidth in each link. However, in a real network, link bandwidth will vary with each user. In the current Internet environment, many broadband users have asymmetric connections $(\text{download} \gg \text{upload})$. Most DSL[1] hosts would easily be able to receive but not forward all received descriptions. In an academic or business environment, symmetric connections (e.g., FTTH[2]) are more common. Such hosts can often receive and forward several times more descriptions than asymmetric connections. In addition, node upload bandwidth is highly heterogeneous due to different link technologies and varying willingness to contribute. Thus, it is difficult for THAG to deliver descriptions which can meet the various bandwidth constraints in heterogeneous networks.

B. *Overview of the Proposed Method*

In this paper, we consider real network environments and propose a method that can be adaptive to the heterogeneity of link bandwidth by changing the size of an AG dynamically. We call the proposed method the Network-aware Hierarchical Arrangement Graph (NHAG).

In NHAG, we change the AG size dynamically for each AG (its properties are described in Section IV-F), and hence the AG size is different between a parent and child AG. If the AG size is s, the maximum number of descriptions that the AG can forward is $s-2$. If the AG size s becomes small, the number of descriptions that can be delivered in the AG decreases. Therefore, in this case, the parent-AG transfers $s-2$ descriptions to its child-AG with AG size s, as shown Fig. 5. For this reason, NHAG should locate the AG that has the largest size in the highest position and the AG that has the smallest size in the lowest position to improve stream delivery efficiency. To realize this AG structure, each node calculates the requested size based on the available bandwidth, and searches for the AG with the appropriate size based on this information. In Section IV-C, we discuss the required size. Additionally, we modify/tailor the node joining and leaving procedures, described in Sections IV-D and IV-E, respectively, to realize this. Finally, we discuss the problem when there is a sudden change in a node's available bandwidth along with the future research direction in Section IV-G.

[1]Digital Subscriber Line

[2]Fiber To The Home

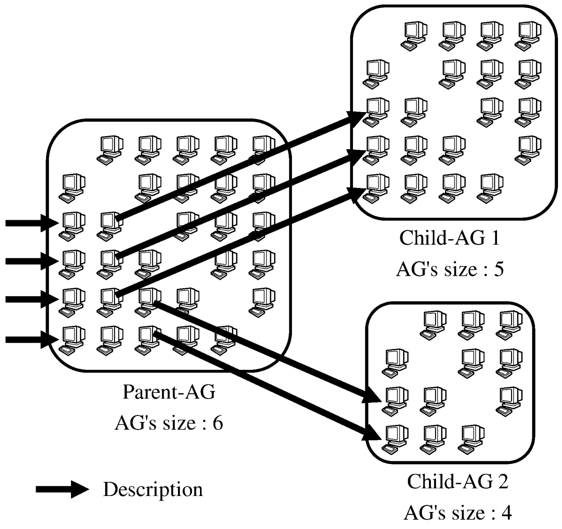

Fig. 5. Hierarchical AG in NHAG.

By following these procedures, each node can receive the appropriate number of descriptions based on its upload bandwidth in heterogeneous networks. Furthermore, NHAG does not increase control overhead as compared to THAG.

C. *Requested Size*

Node upload bandwidth is highly heterogeneous. Therefore, in NHAG, each node calculates the requested size s_R which is the maximum AG size required to stably deliver descriptions. The node joining and leaving processes operate based on this metric. In an AG with size s, the minimum bandwidth needed for streaming delivery is $2(s-2)r$. So, the requested size s_R satisfies (2).

$$2(s_R - 2)r = \mathrm{BW}. \tag{2}$$

Here, BW indicates the available upload bandwidth for each node. We can use bandwidth estimation technologies, such as Initial Gap Increasing (IGI) [26], Self-Loading Periodic Streams (SLoPS) [27], and JitterPath [28], to obtain BW. By solving this equation for s_R, we have:

$$s_R = \left\lfloor \frac{\mathrm{BW}}{2 \times r} + 2 \right\rfloor. \tag{3}$$

If s_R equals to 2, the receiver does not receive any descriptions. Therefore, when the node's requested size equals to 2, its requested size will be 3. However, if the nodes with requested size of 2 exist, NHAG is expected to outperform THAG, because NHAG can reduce the congestion at the nodes' upload links.

D. *The Node Joining Procedure*

In the joining process, each node searches for a joinable AG based on the requested size s_R. The AG entrance which receives the join request determines whether or not to let the joining node in by comparing the AG size s to the requested size s_R.

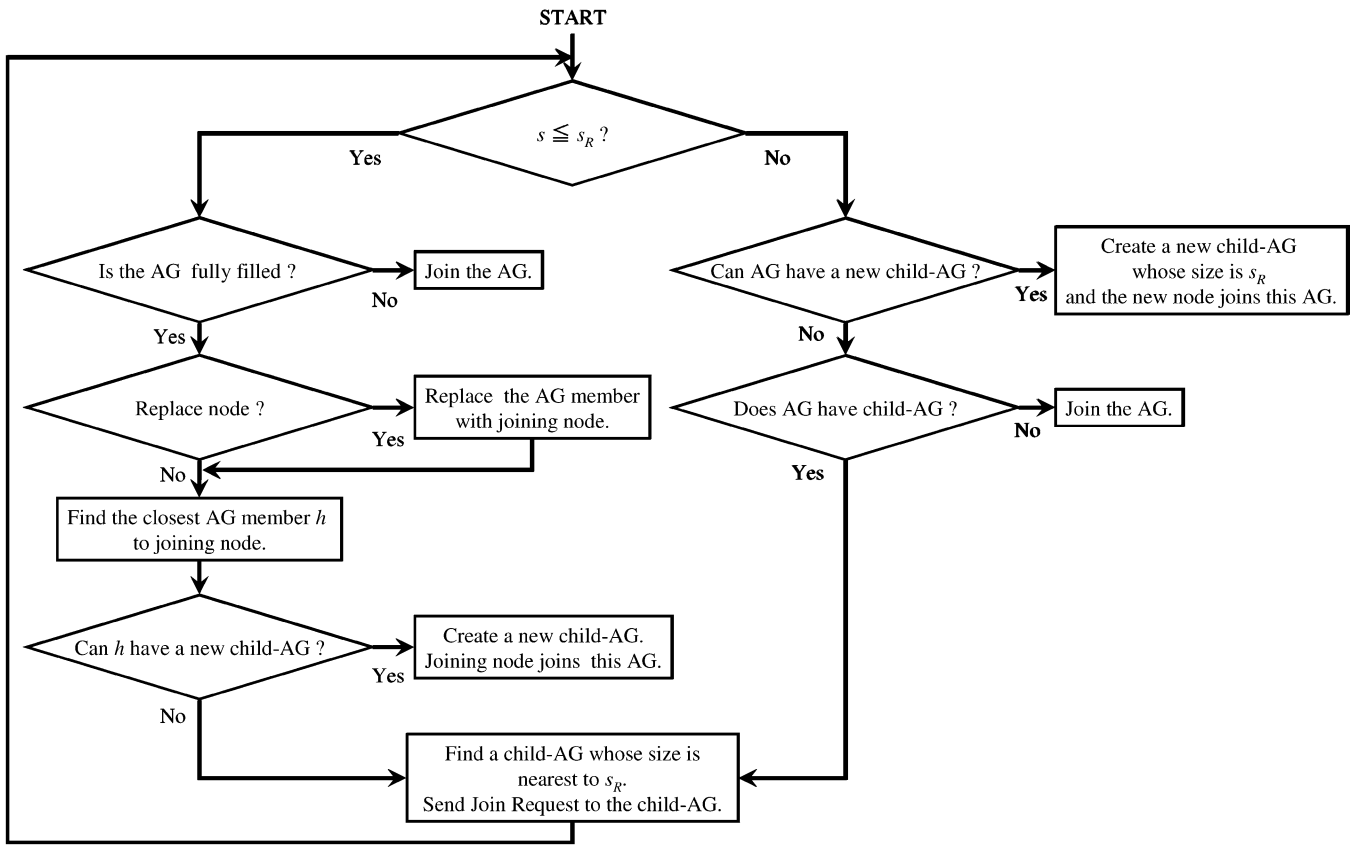
START
$s \leqq s_R$?
Yes
No
Is the AG fully filled ?
No
Join the AG.
Yes
Replace node ?
Yes
Replace the AG member with joining node.
No
Find the closest AG member h to joining node.
Can h have a new child-AG ?
Yes
Create a new child-AG. Joining node joins this AG.
No
Find a child-AG whose size is nearest to s_R. Send Join Request to the child-AG.
Can AG have a new child-AG ?
Yes
Create a new child-AG whose size is s_R and the new node joins this AG.
No
Does AG have child-AG ?
No
Join the AG.
Yes

Fig. 6. Node joining procedure in NHAG.

1) In the Case $s > s_R$: In this case, even when the joining node can join, the node will not be able to transfer all descriptions. Therefore, in the case where the AG can create a new child-AG, the AG entrance creates the new child-AG with size s_R and lets the joining node join this child-AG. On the other hand, in the case where the AG cannot create a new child-AG, if the AG already has child-AGs, the AG entrance sends notification to the joining node to join its child-AG whose size is the closest to s_R. The joining node, which receives the notification message, then sends a join request to that child-AG entrance. In the case where an AG does not have any child-AGs, because it does not have enough nodes, the joining node will be allowed to join the AG temporarily. We call this node "temporally joining node".

2) In the Case $s \leq s_R$: In this case, the joining process is similar to that of THAG as shown in Fig. 4. However, we use the requested size as the replacement metric instead of a distance metric. An AG entrance replaces nodes when the minimum value of the requested size $s_{\min}$ of all AG members is greater than s_R. If the number of nodes with $s_{\min}$ is numerous, we use the distance metric as THAG. By so doing, we can preferentially replace the temporally joining node with the node which has larger requested size. Furthermore, this procedure can promote the node with larger upload bandwidth to a higher tree position. However, NHAG creates trees with larger delay than those of THAG. Yet, NHAG is expected to achieve lower delay because of less congestion at the top of the trees.

By repeating the above process, a node can join the AG which can accommodate its requested size and receive the appropriate number of descriptions. These processes are summarized in Fig. 6.

E. The Node Leaving Procedure

In this subsection, we describe the node leaving process in NHAG. If a node that is leaving is an AG entrance, the process is the same as that of THAG. However, if the leaving node is not an AG entrance, its parent node in the same tree will takeover the AG entrance tasks. If the leaving node has child-AGs, one of its child-AGs promotes a node with the maximum requested size to replace the vacated position. To realize this procedure, each child-AG i entrance sends the maximum requested size s_i of the AG members to its parent-AG entrance periodically. The parent-AG entrance selects $s_{\max}$ which is the maximum number of s_i. When a node leaves, if $s \leq s_{\max}$, then the AG entrance to which the leaving node belongs sends a notification message to the child-AG entrance to which the node with $s_{\max}$ belongs. The child-AG which receives the notification message then promotes the node, which has the requested size $s_{\max}$, to its parent-AG. In the child-AG, similar maintenance can be performed afterward. On the other hand, if $s > s_{\max}$, the AG entrance does not send the notification message. Thus, we can promote a node from a Child-AG which has a requested size greater that or equal to s even when the node is leaving.

In this way, the height of NHAG can be reduced and the nodes with the largest upload bandwidth are promoted to a higher position, as much as possible.

F. Renewal of AG Size

Since nodes in an AG are frequently replaced due to their joining and leaving, the AG size must be dynamically renewed according to the network state. Therefore, the AG size must be recomputed whenever joining, leaving, or node replacement occurs. First, the AG entrance computes the average requested size s_{avg} of the N nodes joining the AG. In case the AG does not have a child-AG, the new AG size is updated as follows:

$$s \leftarrow \begin{cases} s, & \text{for } N > s_{\text{avg}}(s_{\text{avg}} - 1) \\ s_{\text{avg}}, & \text{for otherwise} \end{cases} \tag{4}$$

where s is the current AG size. In an AG with size $s, s(s-1)$ nodes can participate. So, in the case $N > s_{\text{avg}}(s_{\text{avg}}-1)$, if s is set to s_{avg}, some AG members cannot join the current AG. Thus, in the case $N > s_{\text{avg}}(s_{\text{avg}}-1)$, s is not changed. Otherwise, s is set to s_{avg}.

On the other hand, in the case where the AG has a child-AG, if s is changed drastically, the descriptions may not be delivered to all the child-AGs, and hence it is critical to change s slowly. Therefore, we change the new AG size based on the current AG size and the number of joining nodes as follows:

$$s \leftarrow \begin{cases} s+1, & \text{for } s_{\text{avg}} > s \text{ and } N = s(s-1) \\ s-1, & \text{for } N < (s-1)(s-2) \\ s, & \text{for otherwise.} \end{cases} \tag{5}$$

In NHAG when an AG has a size $s, s-2$ numbers of streams can be delivered to the nodes which join the AG. If we suddenly increase the AG size s, the nodes will have to deliver more descriptions to the nodes in the same AG. As a result, the number of child-AGs, to which the AG can have, will decrease because the node's upload bandwidth capacity is limited. Therefore, we propose to increase s in the case of $s_{\text{avg}} > s$, and limit the number of joining nodes in the AG by $N = s(s-1)$. Furthermore, in the case that the number of nodes decreases, even if the AG has a child-AG, since $s > s_{\max}$, the nodes will not be promoted from child-AGs. Therefore, in our approach, if the number of AG nodes has decreased to some extent $(N < (s-1)(s-2))$, we decrease s as well. Otherwise, s is not changed.

Through this procedure, we tailor the AG size to the joining node's network conditions.

G. Discussion

NHAG ensures QoS of the received streams in heterogeneous networks by dynamically changing the AG size. However, NHAG may encounter the following problem when a node's available bandwidth fluctuates.

AG size alleviates the effect of dynamic changes in network when the nodes join/leave. However, if join/leave events are sparse with highly fluctuating network dynamics, the nodes that initially (at join time) had large bandwidth and were assigned a large number of child nodes may struggle if there is congestion in their links. This problem can be solved by accurate measurements of the available bandwidth and recomputing requested size periodically at each node. By so doing, in NHAG, each AG can optimize the AG size with respect to the nodes' available bandwidth. This issue will be addressed in our future work.

V. Performance Evaluation

A. Simulation Setup

We evaluated the performance of the proposed method by using the network simulator ns-2 [15]. In our simulations, the transit-stub topology created by the GT-ITM tool [29] was used as the underlying network topology. The network topology consisted of 1010 routers and 4955 edges. The link delay was randomly set between 1 and 10 ms for each edge. We created end-nodes and randomly connected them to the routers chosen from the stub domain. One of end-nodes was selected as a media source. The number of end-nodes was varied from 100 to 500. The number of descriptions divided by the MDC was four, and each description delivery rate was 500 kbps. The number of multicast trees was four and THAG were restricted to an AG size of 6. Streaming is delivered in a scenario where each node joins the ALM one by one every 2 s, and after all nodes have joined, a node leaves one by one every 2 s. We compare NHAG to Split-Stream [6] and THAG [7] in the following two cases.

In case I, upload bandwidth for all nodes is set to 4 Mbps. The required upload bandwidth for THAG is 4 Mbps, and so the problem mentioned in Section IV does not occur. On the other hand, in case II, each node's upload bandwidth is randomly distributed between 2 and 5 Mbps. Therefore, some nodes cannot transfer all descriptions. Download for all nodes is set to 10 Mbps in both cases.

In order to evaluate the performance and QoS of each method, we use the following metrics: Total Throughput, Bandwidth Satisfaction Ratio (BSR), Relative Delay Penalty (RDP) [30], and Relative Delay Variation (RDV) [7].

1) Total Throughput: The total of all descriptions' throughput, as defined in (6).

$$\text{Total Throughput} = \sum_{i=1}^{M} \text{Throughput}(i). \tag{6}$$

Here, $\text{Throughput}(i)$ denotes the received ith description rate and M denotes the number of descriptions.

2) Bandwidth Satisfaction Rate (BSR): The ratio between the requested streaming rate and the received streaming rate, as defined in (7).

$$\text{BSR} = \frac{\sum_{i=1}^{M} \text{Throughput}(i)}{rnd \times r} \tag{7}$$

Here, rnd denotes the required number of descriptions for each node and r denotes the description delivery rate. In THAG and SpliteStream, each node's required number of descriptions is 4. In NHAG, the required number of descriptions for each node equals $s_R - 2$, where s_R is the requested size. If the received rate and the requested rate is almost the same, BSR will be nearly 1.

3) Relative Delay Penalty (RDP): The average ratio of propagation delay on the paths from the source to the receiver node in ALM trees over the end-to-end unicast latency between these nodes is defined in (8).

$$\text{RDP} = \frac{1}{M} \sum_{i=1}^{M} \frac{\text{mDelay}(i)}{\text{uDelay}}. \tag{8}$$

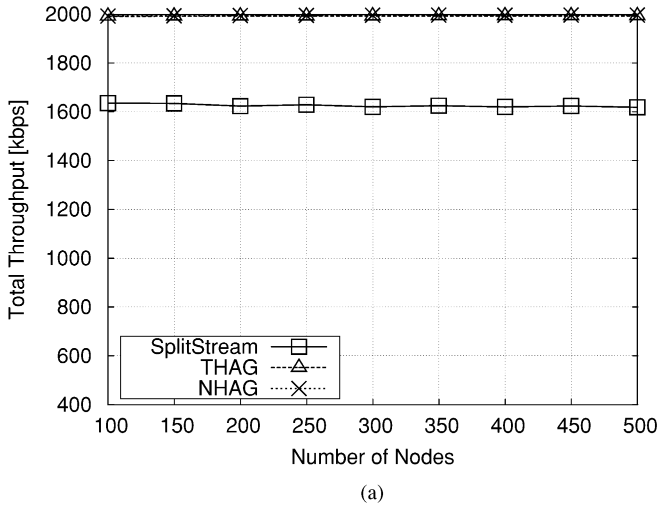

Fig. 7. Comparison of Total Throughput: (a) Case I and (b) Case II.

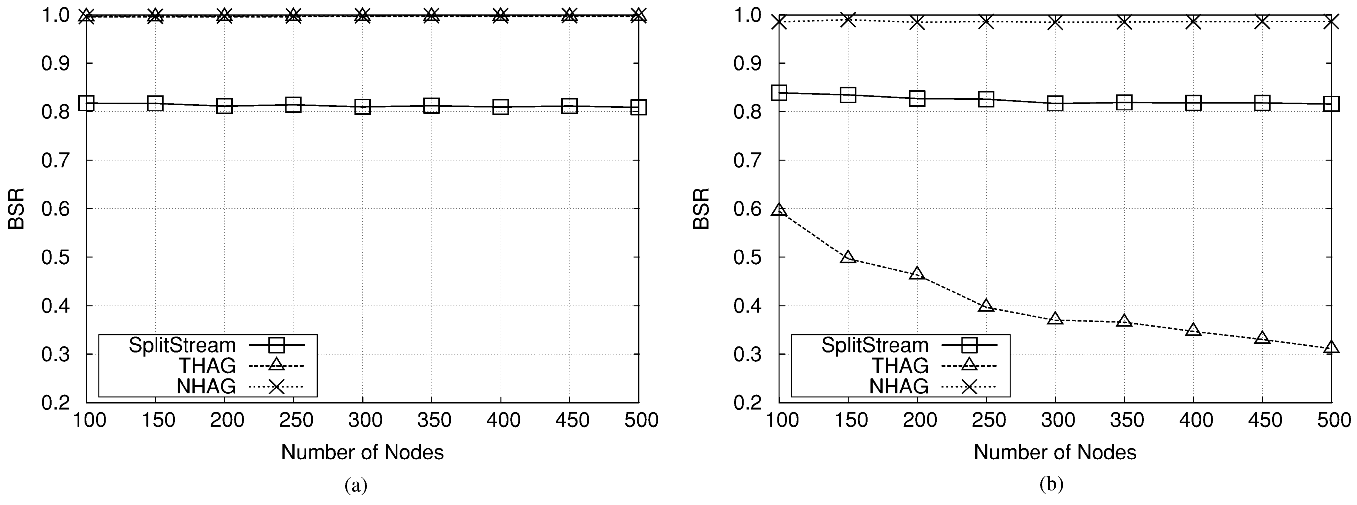

Fig. 8. Comparison of Bandwidth Satisfaction Rate (BSR): (a) Case I and (b) Case II.

Here, $\mathrm{mDelay}(i)$ denotes the multicast delay of the ith description and uDelay denotes the unicast delay. The RDP shows the relative increase in delay that a packet experiences in ALM as compared to IP multicast.

4) Relative Delay Variation (RDV): The average difference of delay in the paths from a source to a node in a different multicast tree is defined in (9).

$$\mathrm{RDV} = \frac{D_{\max} - D_{\min}}{D_{\min}}. \tag{9}$$

Here, $D_{\max}$ and $D_{\min}$ are the maximum and minimum propagation delays experienced by a node when receiving descriptions from different multicast trees. RDV is an important metric to measure the synchronization buffer and latency requirements for received media data descriptions of a node.

B. Simulation Results

1) Total Throughput: Fig. 7 shows the average total throughput of each method in case I and case II. In case I, we can see that our proposed NHAG provides equally high total throughput, the same as that of THAG. These methods receive all descriptions from the source. NHAG and THAG provide higher total throughput than that of SplitStream. On the other hand, in case II, NHAG achieves higher total throughput than that of THAG. However, SplitStream provides the highest total throughput of the three methods because it adapts to heterogeneous networks by using a spare capacity group [6]. The nodes that have less child nodes than they can serve are members of this group. With SplitStream, a node that cannot otherwise find a parent node can find a new parent in this group. SplitStream's stream QoS is, however, very low.

2) Bandwidth Satisfaction Rate (BSR): Next, we compare BSRs achievable by each method. We calculate the average value of BSRs in each node, as shown in Fig. 8. In case I, the BSRs of THAG and NHAG are almost 1 because this is an ideal case. The BSR of SplitStream is smaller than 1. In case II, NHAG achieves an average BSR of almost 1, close to the ideal case. However, BSRs achieved by SplitStream and THAG are smaller than 1. Especially, in the case of THAG, streams received by each node were less than half of the required quality. This result indicates that our proposed method delivers the required number of descriptions to each node.

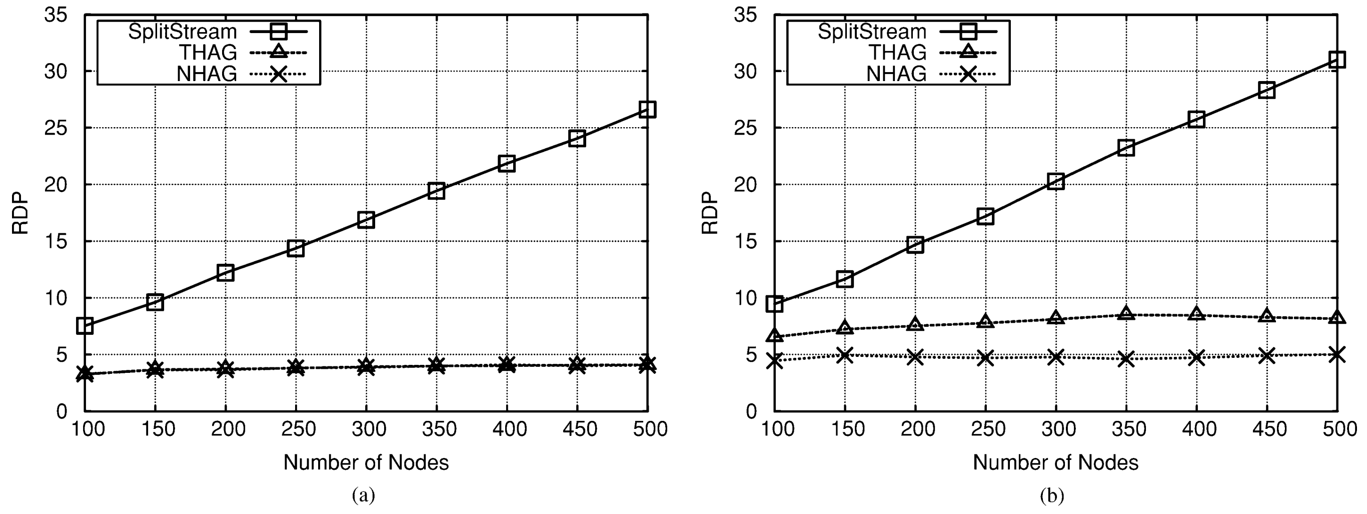

Fig. 9. Comparison of Relative Delay Penalty (RDP): (a) Case I and (b) Case II.

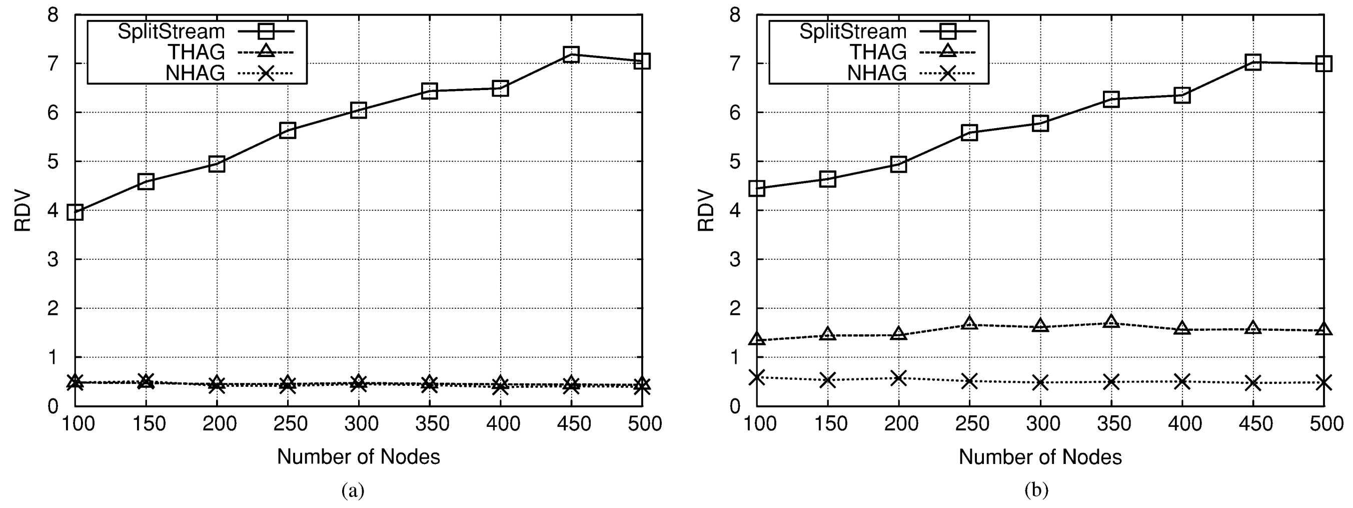

Fig. 10. Comparison of Relative Delay Variation (RDV): (a) Case I and (b) Case II.

3) Relative Delay Penalty (RDP): We study the average RDP in NHAG, THAG, and SplitStream. The results are shown in Fig. 9. In case I, both NHAG and THAG provide equally low RDP. In case II, NHAG achieves a lower RDP than that of THAG. This is because, in THAG, congestion occurs in the upload link of a node, and this queuing delay increases. On the other hand, SplitStream achieves a very high RDP in both cases. This result shows that THAG and NHAG achieve low delay while SplitStream's delay is high. No matter how large the system size is, the RDP in NHAG is very small in both cases. Since it is more difficult to optimize multiple multicast trees in a distributed network environment at the same time than a single multicast tree [6], the small RDP in NHAG is remarkable.

4) Relative Delay Variation (RDV): Fig. 10 indicates the average RDV achieved by each method. We can find that the proposed scheme and THAG provide low RDV below 1 in both cases. Hence, in THAG and NHAG, the difference between each description's delay is small. In SplitStream, the RDP is very high, and so the difference between each description's delay is large. When the number of nodes increases, the variation of delay with SplitStream is more drastic. Hence, the hosts in THAG can experience relatively consistent media delivery from different trees. From the RDP and RDV results, it can be seen that the proposed NHAG provides a high level stream QoS, while the SplitSteam's stream QoS remains low.

The simulation results show that if bandwidth is sufficient, THAG is able to perform stably. If a node's upload bandwidth is lower than the required bandwidth, the total throughput and BSR decrease drastically. SplitStream achieves high total throughput in heterogeneous networks, but the delay and the difference in delay for each description is large, and so the stream QoS still remains low. However, the proposed method provides high total throughput and high QoS in both cases. Therefore, NHAG can achieve stable content delivery in a real network environment.

VI. Conclusion

In this paper, we have studied and examined in details the THAG protocol, which splits a stream into several descriptions with MDC and delivers each stream along node-disjoint multicast trees constructed from AGs. However, because the AG size is constant, it is difficult to deliver descriptions appropriately across a heterogeneous network by THAG. Therefore, in this paper, we have proposed NHAG to change the AG size dynamically to enhance THAG performance even in heterogeneous networks. Our simulation results by using network simulator ns-2 have demonstrated the effectiveness of our proposal in terms of the throughput and QoS. These results indicate that NHAG is a reliable and efficient ALM scheme for streaming media services in heterogeneous networks.

REFERENCES

[1] A. M. Hamad and A. E. Kamal, "A survey of multicasting protocols for broadcast-and-select single-hop networks," *IEEE Netw.*, vol. 16, no. 4, pp. 36–48, Jul./Aug. 2002.
[2] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 3, pp. 58–74, 2007, Third Qtr..
[3] M. Hosseini and N. D. Georganas, "End system multicast protocol for collaborative virtual environments," *Presence: Teleop. Virtual Environ.*, vol. 13, no. 3, pp. 263–278, Jun. 2004.
[4] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. ACM Int. Workshop on Network and Operating Syst. Support for Digital Audio & Video (NOSSDAV)*, May 2002, pp. 177–186.
[5] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," in *Proc. IEEE Int. Conf. Network Protocol (ICNP)*, Nov. 2003, pp. 16–27.
[6] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in cooperative environments," in *Proc. Int. Workshop on Peer-to-Peer Syst. (IPTPS)*, Oct. 2003, pp. 298–313.
[7] R. Tian, Q. Zhang, Z. Xiang, Y. Xiong, X. Li, and W. Zhu, "Robust and efficient path diversity in application-layer multicast for video streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 8, pp. 961–972, Aug. 2005.
[8] M. Alasti, K. Sayrafian-Pour, A. phremides, and N. Farvardin, "Multiple description coding in networks with congestion problem," *IEEE Trans. Inf. Theory*, vol. 47, no. 3, pp. 891–902, Mar. 2001.
[9] V. K. Goyal, "Multiple description coding: Compression meets the network," *IEEE Signal Process. Mag.*, vol. 18, no. 5, pp. 74–93, Sept. 2001.
[10] O. Campana, A. Cattani, A. D. Giusti, S. Milani, N. Zandoná, and G. Calvagno, "Multiple description coding schemes for the H.264/AVC coder," in *Proc. Int. Conf. Wireless Reconfigurable Terminals and Protocols (WiRTeP)*, Apr. 2006, pp. 217–221.
[11] O. Campana, R. Contiero, and G. A. Mian, "An H.264/AVC video coder based on a multiple description scalar quantizer," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 2, pp. 268–272, Feb. 2008.
[12] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, July 2003.
[13] K. Day and A. Tripathi, Characterization of Node Disjoint Paths in Arrangement Graphs Computer Science Dept., Univ. Minnesota, Minneapolis, Tech. Rep. TR91-43, 1991.
[14] Y.-S. Chen, T.-Y. Juang, and E.-H. Tseng, "Congestion-free embedding of $2(n-k)$ spanning trees in an arrangement graph," *J. Syst. Archit.*, vol. 47, no. 1, pp. 73–86, Jan. 2001.
[15] *The Network Simulator-ns-2*, [Online]. Available: http://www.isi.edu/nsnam/ns/, [Online]. Available
[16] P. Francis, Yoid: Extending the Internet Multicast Architecture Apr. 2000 [Online]. Available: http://www.icir.org/yoid/, [Online]. Available
[17] H. Deshpande, M. Bawa, and H. Garcia-Molina, Streaming Live Media Over a Peer-To-Peer Network Stanford Univ., Stanford, CA, Tech. Rep. CS-2001-31, 2001.
[18] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *Proc. USENIX Symp. Int. Technol. and Syst. (USITS)*, Mar. 2001, pp. 49–61.
[19] V. Roca and A. El-Sayed, "A host-based multicast (HBM) solution for group communications," in *Proc. IEEE Int. Conf. Networking (ICN)*, July 2001, pp. 610–619.
[20] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM Special Int. Group on Data Commun. (SIGCOMM)*, Aug. 2002, pp. 205–217.
[21] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 1, pp. 121–133, Jan. 2004.
[22] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1489–1499, Oct. 2002.
[23] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in *Proc. IFIP/ACM Int. Conf. Distributed Syst. Platforms (Middleware)*, Nov. 2001, pp. 329–350.
[24] T. S. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, June 2002, pp. 170–179.
[25] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. ACM Special Int. Group on Data Commun. (SIGCOMM)*, Aug. 2004, pp. 15–26.
[26] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 6, pp. 879–894, Aug. 2003.
[27] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Trans. Netw.*, vol. 11, no. 4, pp. 537–549, Aug. 2003.
[28] Y.-C. Huang, C.-S. Lu, and H.-K. Wu, "JitterPath: Probing noise resilient one-way delay jitter-based available bandwidth estimation," *IEEE Trans. Multimedia*, vol. 9, no. 4, pp. 798–812, June 2007.
[29] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Mar. 1996, pp. 594–602.
[30] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.

Masahiro Kobayashi (S'08) received the B.E. degree in information engineering from Tohoku University, Sendai, Japan, in 2007. Currently, he is pursuing the M.S. degree in the Graduate School of Information Science (GSIS) at Tohoku University. His research interests are in the areas of application layer multicast and overlay networks.

Hidehisa Nakayama (M'06) received the B.E., M.S., and Ph.D. degrees in information sciences from Tohoku University, Sendai, japan, in 2000, 2002, and 2005, respectively.

He is a Senior Assistant Professor at the Tohoku Institute of Technology. He has been engaged in research on intelligent sensor technology, wireless mobile ad hoc network, computer networking, character string analysis, pattern recognition, and image processing.

Dr. Nakayama is a member of IEEE Communications Society, the Institute of Electronics, Information and Communication Engineers (IEICE), and the Information Processing Society of Japan (IPSJ). He received the Best Paper Award for Young Researcher of IPSJ Tohoku Chapter in 2000 and the Best Paper of Pattern Recognition Award in SCI 2003.

Nirwan Ansari (S'78–M'83–SM'94–F'09) received the B.S.E.E. degree (summa cum laude) from the New Jersey Institute of Technology (NJIT), Newark, in 1982, the M.S.E.E. degree from the University of Michigan, Ann Arbor, in 1983, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1988.

He joined NJIT's Department of Electrical and Computer Engineering as Assistant Professor in 1988, and has been Full Professor since 1997. He has also assumed various administrative positions at NJIT. He authored *Computational Intelligence for Optimization* (Springer, 1997, translated into Chinese in 2000) with E.S.H. Hou, and edited *Neural Networks in Telecommunications* (Springer, 1994) with B. Yuhas. His current research focuses on various aspects of broadband networks and multimedia communications. He has also contributed over 300 technical papers, over one third of which in refereed journals/magazines.

Dr. Ansari is a Senior Technical Editor of the *IEEE Communications Magazine*, and also serves on the Advisory Board of *Journal of Communications*, and on the editorial board of *Computer Communications*, the *ETRI Journal*, *Wireless Communications and Mobile Computing*, and the *Journal of Computing and Information Technology*. He was the founding general chair of the First IEEE International Conference on Information Technology: Research and Education (ITRE2003), was instrumental, while serving as its Chapter Chair, in rejuvenating the North Jersey Chapter of the IEEE Communications Society which received the 1996 Chapter of the Year Award and a 2003 Chapter Achievement Award, served as Chair of the IEEE North Jersey Section and in the IEEE Region 1 Board of Governors during 20012002, and has been serving in various IEEE committees such as Chair of IEEE COMSOC Technical Committee on Ad Hoc and Sensor Networks, and (TPC) Chair/Vice-chair of several conferences/symposia. He has been frequently invited to deliver keynote addresses, distinguished lectures, tutorials, and talks. His awards and recognitions include an IEEE Fellow (Communications Society), the NJIT Excellence Teaching Award in Graduate Instruction (1998), IEEE Region 1 Award (1999), IEEE Leadership Award (2007, from Central Jersey/Princeton Section), the NJIT Excellence in Teaching in Outstanding Professional Development (2008), IEEE MGA Leadership Award (2008), and designation as an IEEE Communications Society Distinguished Lecturer. 7.

Nei Kato (A'03–M'04–SM'05) received the M.S. and Ph.D. degrees from Tohoku University, Sendai, Japan, in 1988 and 1991, respectively.

He has been working with Tohoku University since then and is currently a full professor at the Graduate School of Information Sciences. He has been engaged in researches on computer networking, wireless mobile communications, image processing and neural networks. He has published more than 130 papers in journals and peer-reviewed conference proceedings.

Dr. Kato has served as a symposium co-chair at GLOBECOM'07 and ChinaCom'08, and a TPC member in a large number of IEEE international conferences, including ICC, GLOBECOM, WCNC and HPSR. He was a co-guest-editor for JCN special issue on Broadband convergence Network (BcN) in 2005. From 2006, he is a technical editor of IEEE Wireless Communications. From 2008, he is an editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He is a co-recipient of the 2005 Distinguished Contributions to Satellite Communications Award from the IEEE Communications Society, Satellite and Space Communications Technical Committee, the co-recipient of FUNAI information Science Award, 2007, and the co-recipient of 2008 TELCOM System Technology Award from Foundation for Electrical Communications diffusion. He is serving as an expert member of Telecommunications Council, Ministry of Internal Affairs and Communications, Japan. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE).