Design and Implementation of Overlay Multicast Protocol for Multimedia Streaming

Thilmee M. Baduge Akihito Hiromori Hirozumi Yamaguchi Teruo Higashino
Graduate School of Information Science and Technology
Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, JAPAN
{thilmee, hiromori, h-yamagu, higashino}@ist.osaka-u.ac.jp

Abstract

In this paper, we propose a new protocol called Shared Tree Streaming (or STS in short) protocol that is designed for interactive multimedia streaming applications. STS is a decentralized protocol that constructs a shared tree called s-DBMDT (sender-dependent Degree-Bounded Minimum Diameter Tree) as an overlay network that involves all the participants of the application. For a given set of nodes where some of them are senders, s-DBMDT is a spanning tree where the maximum delay on the tree from those senders is minimized and the degree constraint on each node is held. We believe that this is the first approach that defines s-DBMDT construction problem and presents a distributed protocol for the purpose. Our performance evaluation is based on experiments in both simulated networks and real networks that strongly shows the efficiency and usefulness of STS protocol.

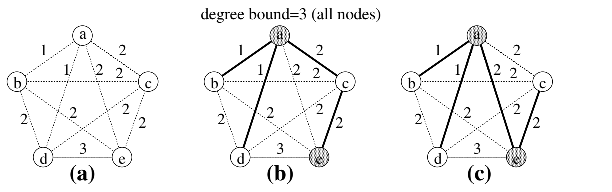
1 Introduction

Recent innovation of the Internet has brought us several *interactive group communication* models. Especially recent applications may require multimedia-based group communication methods such as whiteboard, audio and streaming video. It is a common consensus that we need multicast solutions for group communication and IP multicast is not suitable for such a purpose because it is designed for large-scale content distribution. Instead, overlay multicast solutions have had a lot of attentions where application nodes (end systems, referred to as simply *nodes* in this paper) are connected by unicast channels and consequently form tree-like virtual networks (overlay networks) among them.

A lot of research efforts have been dedicated so far to deploy overlay multicast. They are classified into the following three categories based on their policies to generate overlay topologies, (i) mesh-first approaches like [1, 2], (ii) tree-first approaches like [3, 4, 5, 6, 7] and (iii) other approaches like [8]. In Ref. [7], we have taken a tree-based approach and have presented a protocol called MODE (Minimum-delay Overlay tree construction by DEcentralized operation). MODE aims at minimizing maximum delay (referred to as *diameter*) between any pair of nodes under the degree-constraints given by the nodes, assuming interactive applications where every node can be a potential sender. Some techniques for constructing Degree-Bounded Minimum Delay (or Diameter) Trees (DBMDTs) have been proposed [4, 5, 7], considering the maximum delay of overlay trees and the bandwidth constraints around nodes. Unfortunately none of those methods has considered the following several important features of interactive *multimedia* applications such as video conferencing.

First, such an application may have several sources. These sources are subject to change, but are not changed so frequently. For example, in video-conferencing, pictures of some primary persons should be continuously delivered to the other audience. In such an application, we would like to efficiently build a tree where the maximum delay *from the current senders* is minimized satisfying the degree bounds of nodes. Hereafter, for a given set of senders, such a tree is called *sender-dependent DBMDT* and denoted as *s-DBMDT*. Fig. 1 shows an example that explains the difference between DBMDT and s-DBMDT. For a given complete graph that represents an overlay network in Fig. 1(a), DBMDT is a spanning tree which involves all the nodes of the graph and has a minimum diameter, as shown in Fig. 1(b). In this case, the diameter path is *b*-*a*-*c*-*e* (or *d*-*a*-*c*-*e*) of delay 5. Here, let us suppose that currently only nodes a and e, which are shown by the meshed circles in the figures are senders. In this case, considering the fact that only these nodes send data and others are receivers, s-DBMDT in Fig. 1(c) achieves smaller maximum delay 4 (*a*-*e*-*c*) from those senders, while 5 (*e*-*c*-*a*-*b* or *e*-*c*-*a*-*d*) in DBMDT of Fig. 1(b).

Secondly, considering practical aspects of interactive multimedia applications, we need to identify incapable hosts and prevent them from staying in the center of s-DBMDT, since



(a) An Overlay Network (b) DBMDT (dia.=5,s-dia.=5)
(c) s-DBMDT (dia.=5,**s-dia.=4**)

Figure 1. DBMDT and s-DBMDT on Overlay Networks

those hosts may delay or drop packets due to limitation of network bandwidth or processing power to forward packets, or instability of hosts. Similarly, we should provide a reasonable way to allow each host to determine an appropriate degree bound, since the capability overflow of such a host may also cause packet delay or dropping at the host.

In this paper, we propose a new protocol called *Shared Tree Streaming (or STS in short)* protocol that constructs s-DBMDT adaptively in a decentralized heuristic manner. We also design and implement a Java middleware based on STS protocol. Compared with the existing literatures, our contribution can be summarized in to the following two points. First, we define a new problem that is well-suited to multimedia interactive applications and design a new decentralized protocol for the problem. Secondly, we have designed and implemented an adaptation mechanism that is needed for multimedia streaming on overlay shared trees. Our performance evaluation is based on experiments in both simulated networks and real networks that strongly shows the efficiency and usefulness of our protocol.

2 Shared Tree Streaming (STS) Protocol

First, we give the definition of a Degree-Bounded Minimum Diameter Tree (DBMDT)[7]. Let $G = (V, E)$ denote a given undirected complete graph where V denotes a set of nodes and E denotes a set of potential overlay links which are unicast connections between nodes. Also let $d_{max}(v)$ denote a degree bound of each node $v \in V$ (the maximum number of overlay links attached to v), and let $h(i, j)$ denote the delay of each overlay link $(i, j) \in E$. DBMDT is a spanning tree T of G where the *diameter* of T (the maximum delay on T) is minimum and the degree of each node $v \in V$ (denoted as $d(v)$) does not exceed $d_{max}(v)$.

Based on the above, we define a sender-dependent DBMDT (s-DBMDT) introduced in this paper as follows. For a given $G = (V, E)$ and a given set $S \subseteq V$ of senders, s-DBMDT is a spanning tree T of G where the maximum delay from the senders in S is minimum and $d(v) \leq d_{max}(v)$ where $v \in V$. The maximum delay from the senders is called *sender-dependent diameter* and denoted by *s-diameter*.

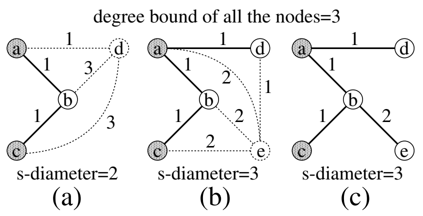


Figure 2. Join Procedure

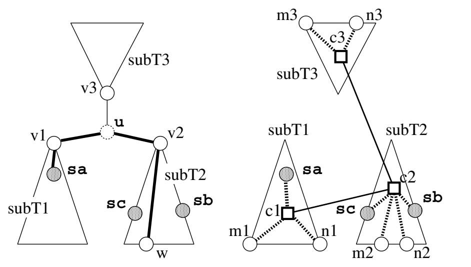
The DBMDT construction problem has been proved to be NP-hard [4]. The DBMDT construction problem is a special case of our s-DBMDT construction problem where $S = V$. Therefore, we need an efficient heuristic algorithm for the problem.

2.1 Minimizing Maximum Delay from Senders

The proposing protocol, *Shared Tree Streaming (STS) protocol*, consists of two main procedures, join and repair. The join procedure makes a new joining node connect to a node which positions the joining node “closest to the senders” of the current tree in order to prevent the new node from making the s-diameter longer. The repair procedure is activated when a node on the current tree leaves (or suddenly disappears) and it connects appropriate intermediate nodes in the isolated sub-trees to make a new tree with a shorter s-diameter. To execute the above procedures in decentralized manner, each node in our STS protocol autonomously collects the information about the current sender nodes and diameter paths of the sub-trees that will appear by neighboring node’s leaving. This information collection is executed periodically to keep up with the status changes of the tree (e.g. location of sender nodes and diameter paths of the sub-trees). This will be explained in Section 3.3. In this subsection, we explain how the two procedures keep the s-diameter as small as possible, satisfying the given degree bounds of nodes.

[Join Procedure Outline]: For a new joining node u, the join procedure never changes the current form of the tree, but lets the new node u connect to such a node (say v) where the maximum delay from the senders to the new joining node u (*i.e.* a candidate for the s-diameter of the consequent tree) is minimum. Note that node v must be such a node that has at least one residual degree.

Fig. 2 shows an example where senders are denoted by meshed circles. We assume that the degree bounds of all the nodes are 3. In Fig. 2(a), we have a tree involving three nodes a, b and c where a and c are senders. Also the s-diameter of the tree is 2. Let us assume that a node d wants to join the tree, and the dotted lines in Fig. 2(a) represent the measured delay between the new node d and the existing nodes on the tree. Consequently, node d is connected to node a since the maximum delay from senders a and c becomes 3 (Fig. 2(b)) and it is the minimum in all the possible connecting positions.



(a) T (b) T'

	path	type
1.	longest path from s_a in $subT_1$	(i)
2.	s_a-c_1-c_2-m_2	(ii)
3.	s_a-c_1-c_2-c_3-m_3	(ii)
4.	longest path from s_b in $subT_2$	(i)
5.	s_b-c_2-c_1-m_1	(ii)
6.	s_b-c_2-c_3-m_3	(ii)
7.	longest path from s_c in $subT_2$	(i)
8.	s_c-c_2-c_1-m_1	(ii)
9.	s_c-c_2-c_3-m_3	(ii)

(c) Candidates for s-diameter path of T'

Figure 3. Repair Procedure

And the subsequent joining nodes (*i.e.* node e in Fig. 2(c)) also follow the same method.

[Repair Procedure Outline]: Whenever a node's disappearance occurs, the disconnected sub-trees are repaired by the repair procedure. At that time, we try to shorten the s-diameter of the repaired tree, by connecting the *core nodes* of the isolated (disconnected) sub-trees. Here, let m and n be the both end nodes of the diameter path (*not s-diameter path*) of a tree t. The *core node* of t is a node whose maximum delay from the nodes m and n is minimum in all the nodes of t. This means that the core node is located on the "center" of the diameter path. Here, we will explain why such a procedure can make the s-diameter of the repaired tree shorter. An example is shown in Fig. 3 where the senders are s_a, s_b and s_c and represented by meshed circles. In tree T of Fig. 3(a), let us assume that its s-diameter path is s_a-v_1-u-v_2-w and a new node u leaves the tree. By the leave of node u, the tree T is partitioned into three sub-trees $subT_1$, $subT_2$ and $subT_3$, and they are connected through their core nodes c_1, c_2 and c_3 (denoted by squares) and reorganized into a new tree T' as shown in Fig. 3(b) (there are some possibilities to connect among the core nodes and this figure shows one of them). In this case, the s-diameter of T' is either of (i) the maximum delay from a sender to a node *within the same sub-tree*, or (ii) the maximum delay from a sender to a node on a *different sub-tree*. We enumerate all the candidates for the new diameter path in Fig. 3(c) along with their classification of the above type (i) or (ii). We note that m_i and n_i are both ends of the diameter path of $subT_i$ and without loss of generality we assume that delay of path c_i-m_i on the tree, denoted by $L(c_i, m_i)$, is always equal to or larger than $L(c_i, n_i)$.

Obviously, if a path of type (i) (either path 1, 4 or 7 in Fig. 3(c)) becomes the s-diameter path of T', the s-diameter is equal to or smaller than that of T. Otherwise, one of the paths of type (ii) becomes the s-diameter path of T'. Here we can say that for any sub-tree $subT_i$, the following inequality, $L(s_i, c_i) \leq L(c_i, n_i) \leq L(c_i, m_i)$, always holds where L is the delay of the path between two nodes on the tree[1]. The above inequality suggests that for any path of type (ii), the delay from a sender to the core node on the same sub-tree is not larger than the half of the diameter of the sub-tree. Also on another sub-tree, the delay from its core node to an end of the diameter path on the sub-tree is the half of the diameter of the sub-tree. Consequently, the diameter of T' may be equal to or less than the sum of halves of the diameters of the two sub-trees plus the delay between the core nodes. Here, the half of the diameter of a sub-tree of T is always smaller than the half of the diameter of the original tree T. Therefore, this may shorten the s-diameter compared with T in many cases even though it depends on the delay between the core nodes. To make it easy to understand, let us compare our strategy with the simple one where we simply connect the neighboring nodes of u (*i.e.* v_1, v_2 and v_3). This procedure makes the new s-diameter almost equal to that of T with a high possibility, since the maximum delay paths from the neighboring nodes v_1, v_2 and v_3 remain as they are, and they may again be a part of the s-diameter path of the repaired tree.

We note that the above description is valid for the case that the leaving node u is on the s-diameter path of T. For the case that node u is not on the s-diameter path of T, the s-diameter of T' is not smaller than that of T. This is because the s-diameter path of T is preserved in an isolated sub-tree as it is, and thus it remains in T'. However, we think that the s-diameter of T' is not changed from T in most cases.

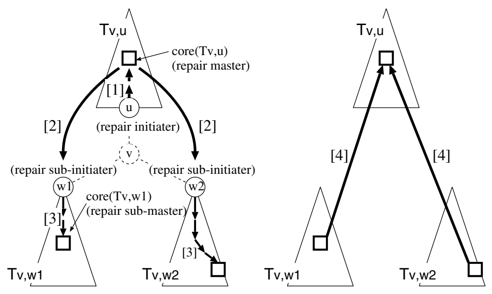
As a whole, we can expect the s-diameter to be smaller when nodes leave.

3 Design of STS Protocol

3.1 Join Procedure Design

A new node which wants to join the current tree first sends a *query message* to a well-known node on the tree to ask the address of the *center node* of the tree. The *center node* of a tree is a node whose maximum delay from the senders is the minimum. Intuitively, the accepter-node that makes the maximum delay from the senders to the new node minimum seems to be located near the center node of the current tree. Therefore, we start searching the accepter-node from the center node. To do this, in our STS protocol, we assume that each node can know the center node of the current tree (thus any node can be a well-known node). We also assume that each node knows the maximum delays from all the senders. The way of this information collection will be explained later in Section 3.3.

[1]This is obvious since $L(c_i, n_i) + L(c_i, m_i)$ is the diameter (the maximum delay) of $subT_i$.



(a) phases (1)-(3) (b) phase (4)

Figure 4. Repair Procedure

Once the joining node receives the reply from the well-known node, it sends a *connection request message* to the center node. Then, the center node sends a *connection permission message* to the joining node only if its residual degree is not zero. At the same time, it broadcasts the connection request message to its neighboring nodes of the tree. In response to the reception of the connection request message, each neighboring node acts in the same way as the center node. Therefore the connection request message is delivered on the tree to all the nodes. Here, to prevent the joining node from receiving a large number of connection permission messages, a maximum hop count from the center node can be assigned to the connection request message.

The new joining node can know the maximum delay from the senders to each responded accepter-node, by making the connection permission message contain those information. Then, by knowing the delay to each accepter-node (this can be measured using *ping* for instance), the joining node can choose the accepter-node that minimizes the maximum delay from the senders to itself.

3.2 Repair Procedure Design

Hereafter, for a pair of two adjacent nodes v and w on the tree T, let $T_{v,w}$ denote the sub-tree rooted at node w and isolated by node v's disappearance. Suppose that the *sub-tree information* of $T_{v,w}$ contains the IDs and network addresses of (i) w, (ii) w's neighbors except v and (iii) the core node of $T_{v,w}$ (denoted as $core(T_{v,w})$). We assume that each node, say u, has the sub-tree information of $T_{v,w}$ for every pair of v and w where v is a neighboring node of u and w is a neighboring node of v including u. We also assume that for adjacent two nodes, parent-child relationship on the tree is pre-defined. Each node autonomously collects necessary sub-tree information by the information collection procedure described in Section 3.3.

Under the assumptions above, we present the design of the repair procedure as follows. For simplicity of discussion, we explain the repair procedure for a single node's disappearance without considering other nodes' disappearances.

The procedure repairs the tree for node v's disappearance in the following four phases (Fig. 4). In Fig. 4, we assume that node u is the parent node of node v.

(1) The parent node u of node v activates the repair procedure and sends all the sub-tree information of T_{v,w_j} (w_j is one of the child nodes of v, $1 \leq j \leq d(v) - 1$) to the core node $core(T_{v,u})$ of $T_{v,u}$. This information passing is done along the tree.

(2) $core(T_{v,u})$ sends its address to each w_j directly.

(3) Then each w_j sends the address of $core(T_{v,u})$ to its core node $core(T_{v,w_j})$ (*i.e.* sub-tree T_{v,w_j}'s core node). This information passing is done along the tree.

(4) Now each $core(T_{v,w_j})$ knows the address of $core(T_{v,u})$, so it connects itself to $core(T_{v,u})$. If $core(T_{v,u})$ or $core(T_{v,w_j})$ has no residual degree, it delegates its role to its closest node with some residual degree. After this phase (4), the core nodes exchange their sub-tree information with their neighboring nodes to prepare for future node disappearance.

By this procedure, the core nodes or their neighboring nodes get connected to each other. One may think that this procedure contains redundant operations (for example, we can directly send sub-tree information between the neighboring nodes of v and the core nodes in phases (1) and (3)), and that the sub-tree information contains unused data such as the neighboring nodes' information of the root node. This redundancy is necessary to prepare for another disappearance occurs during the procedure, but due to limitation of space, we omit the discussion of validation in case of simultaneous occurrence of multiple disappearances. Readers may refer to Ref. [7] to see analogous idea for the validation.

3.3 Information Collection

In STS protocol, all the nodes periodically collect the information required to execute join and repair procedures by message exchange along the current tree. We assume that there exists a node that never disappears, for example, the first member node or a well-known node. Such a node is called the *root node* [2]. The root node starts the information collection in the *collection phase* for every (regular) interval by broadcasting *synchronization messages* on the current tree. Obviously, the number of synchronization messages is $n - 1$ where n is the number of nodes on the current tree. A non-leaf node enters the collection phase if it has received a synchronization message from its parent and has sent synchronization messages to all its children. A leaf node does not send synchronization messages. Instead, when it receives a synchronization message, it enters the collection phase and replies a *collection*

[2]This is not an essential assumption. By giving a unique ID to each node, a new root node can be found autonomously by using a distributed algorithm for the leader election when the current root node has disappeared.

message to its parent. Each non-leaf node in the collection phase acts as follows. Whenever it receives collection messages from all the neighboring nodes except one neighboring node (say x), it sends a collection message to node x. Each node u leaves the collection phase if node u has received a collection message from every its neighboring node v and has sent a collection message to v. This means that in the collection phase, $2(n-1)$ collection messages are exchanged on the tree. So, totally, the number of messages required for the collection of the current status is only $3(n-1)$, that is, only three messages are exchanged on each link of the tree.

Hereafter we will explain how the information required by join and repair procedures are collected by collection messages. The information that each node x must hold are, (i) The ID and address of the center node of the current tree , (ii) The maximum delay from the senders to node x and (iii) The sub-tree information of $T_{y,z}$ for every pair of y and z where y is a neighboring node of x and z is a neighboring node of y . Note that the sub-tree information of $T_{y,z}$ contains the IDs and network addresses of z, z's neighboring nodes and the core node of $T_{y,z}$ ($core(T_{y,z})$).

And the following auxiliary information are needed to calculate the sub-tree information of $T_{x,y}$. (i). $dia(T_{x,y})$: the diameter of $T_{x,y}$, (ii). $L(T_{x,y})$: the maximum delay of $T_{x,y}$ from y, (iii). $H(T_{x,y})$: the maximum delay path of $T_{x,y}$ from y. The delay of each link on the path is also included and (iv). $S(s, T_{x,y})$: the path from a sender s in $T_{x,y}$ to y. The delay of each link on the path is also included.

We let each node (say v) be responsible for calculating the sub-tree information of $T_{u,v}$ for every its neighbor u. For this purpose, we let the collection message sent from v to u have the following information. (i). $dia(T_{u,v})$, $L(T_{u,v})$, $H(T_{u,v})$ and $S(s, T_{u,v})$ (for each sender s in $T_{u,v}$), (ii). the sub-tree information of $T_{u,v}$ and (iii). the sub-tree information of $T_{v,w}$ for each w (except u) of the neighboring node of v. Now we show that node v can calculate the above information if it receives the collection messages from all the neighboring nodes except u (let W denote the set of neighboring nodes of v except u). First, the parameters $dia(T_{u,v})$, $L(T_{u,v})$, $H(T_{u,v})$ and $S(s, T_{u,v})$ (for each sender s in $T_{u,v}$) can be defined as follows.

$$\begin{aligned} dia(T_{u,v}) &= \max_{w,x,y \in W} \{dia(T_{v,w}),\ L(T_{v,x}) + \\ &\qquad h(v,x) + L(T_{v,y}) + h(v,y)\} \\ L(T_{u,v}) &= \max_{w \in W} \{L(T_{v,w}) + h(v,w)\} \\ H(T_{u,v}) &= [v]@H(T_{v,w_o}) \quad (w_o \in W \text{ where } w_o \\ &\qquad \text{maximizes } L(T_{u,v})) \\ S(s, T_{u,v}) &= [v]@S(s, T_{v,w}) \quad (\forall w \in W \text{ where} \\ &\qquad S(s, T_{v,w}) \text{ is not empty}) \end{aligned}$$

Note that $h(x,y)$ is the link delay on the tree. The equation for $dia(T_{u,v})$ comes from the definition of the diameter. The diameter of $T_{u,v}$ is the maximum value of (i) the diameters of its sub-trees and (ii) the sum of the two longest depths from node v. The others are straightforward.

Secondly, regarding the sub-tree information of $T_{u,v}$, the IDs and network addresses of v and v's neighbors are known by v. Therefore, $core(T_{u,v})$ can be defined as follows.

$$core(T_{u,v}) = \begin{cases} core(T_{v,w}) \quad (\text{if } dia(T_{u,v}) = dia(T_{v,w}) \\ \qquad\qquad \text{where } w \in W) \\ \\ \text{center of } rev(H(T_{v,x}))@[v]@H(T_{v,y}) \\ \quad (\text{if } dia(T_{u,v}) = L(T_{v,x}) + h(v,x) + \\ \quad L(T_{v,y}) + h(v,y) \text{where } x, y \in W) \end{cases}$$

Here "rev" is the reverse function of a given path.

Thirdly, the sub-tree information of $T_{v,w}$ is included in the collection message from w.

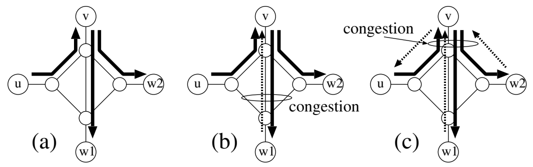
From the above, we have proved that each node v can calculate the content of the collection message to be sent to node u. Assuming those information, we show that any node, say x, can calculate the information that node x must hold, listed previously in this section. The center node and maximum delay from the senders can be calculated by all the $S(s, (T_{x,*}))$ included in the received collection messages. Also the sub-tree information of $T_{y,z}$ is directly included in the collection message from y to x. Consequently, we have proved that every node can obtain the required information after the collection phase.

4 Implementation of STS Protocol as Java Middleware

We have implemented our protocol as a Java based middleware called STS/J. Due to limitation of space, we only present our degree bound adaptation mechanism implemented in STS/J, which is an important functionality for media streaming on overlay networks.

Theoretically, we have shown importance of s-DBMDT for interactive multimedia applications in previous sections. Here, we focus on practical aspects of multimedia streaming on a tree. Practically, it is difficult to determine an appropriate degree for each node. It is theoretically simple, since usually an end host has only one network interface, and all the overlay links attached to the host uses this interfaces. Therefore, the upper bound of the degree bound $d_{max}(v)$ of node v is determined by $d_{max}(v) \leq \frac{N}{\sum_{s \in S} B_s}$ where N is the bandwidth of the network interface and B_s is the bitrate transmitted from a sender s. However, the actual bandwidth of network interface, especially wireless network interface, changes from time to time. Therefore, the degree bound should be adapted according to the network status.

Here, we adopt the following scheme for each node v. We denote the set of the neighboring nodes of v by W and the



(a) Physical network and overlay tree.
(b) Congestion occurs on a remote link of node v.
(c) Congestion occurs on the local link of node v.

Figure 5. Adaptation Mechanism

neighboring node which sends a stream to v by u. Thus node v relays the stream to the nodes in $W - \{u\}$ (Fig. 5(a)). Our implementation uses RTP and RTCP, and if a node detects loss or jitter of packets, it sends receiver reports to its upstream. In Fig. 5, streams are represented by thick arrows, while RTCP reports are represented by dotted arrows. In the figure, the underlying network is shown where small circles represent physical routers and big ones represent end hosts.

- If node v receives an RTCP receiver report from only one node (or some nodes) $w \in W - \{u\}$, node v determines that network congestion happens not on the local link (*i.e.* network interface) but on a remote link on the unicast path between v and w (Fig. 5(b)). In this case, node v sends *compulsory leave message* to node w to let it leave and rejoin the tree.

- If node v receives an RTCP receiver report from each node in $w \in W - \{u\}$ and also node v sends an RTCP receiver report to node u, node v determines that network congestion happens on the local link (Fig. 5(c)). In this case, node v ignores the compulsory leave message from node u, and sends compulsory leave messages to some nodes in $w \in W - \{u\}$ to let them leave and rejoin the tree. This is done to dissolve the congestion on the local link of node v by decreasing its current degree. After that, node v sets its degree bound $d_{max}(v)$ to the adjusted degree to prevent itself from accepting other neighbors.

- If node v continues stable states for a while, it increments its degree bound.

5 Experiments

5.1 Simulation Experiments

We have implemented our STS protocol on ns-2 to evaluate the enhancement against our previous work MODE[7]. In our experiments, networks with 400 physical nodes have been generated and used as underlying networks. We have selected 200 nodes, including both wireless and wired nodes, as overlay participant nodes. In the simulation, we have set the initial end-to-end delay (overlay link delay) to vary between 10ms to 200ms for both wired and wireless nodes, while setting the wireless nodes to change their end-to-end delay up to 300ms during the simulation.

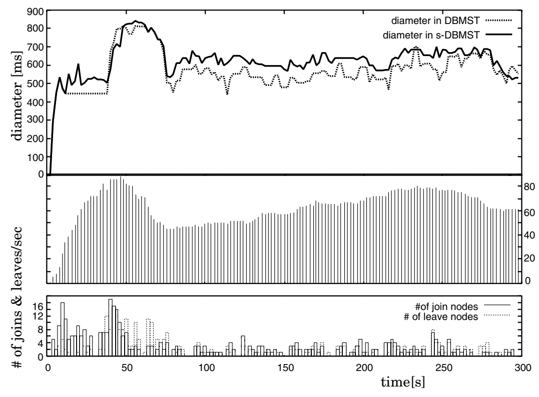
Considering practical situations, we have prepared the following scenario that simulates a real-time session in collaborative applications such as a video-based meeting or groupware. Note that we set the interval between collection phases to 60 seconds. The initial degree bound was set to 5 for all the nodes. The scenario is as follows. (i) The session period is 300 seconds. (ii) Each of 200 nodes joins the session only once and eventually leaves the session. (iii) Within the first 30 seconds, about 60 nodes join the session. (iv) From 30 seconds to 270 seconds, additional joins are processed. Also some existing nodes leave the session. The collection phases starts at 30, 90, 150, 210 and 270 seconds successively. (v) After 270 seconds, no node joins and about 40 nodes leave the session.

[Diameter and Sender-Dependent Diameter]: We have measured (a) the diameters and (b) sender-dependent diameters (s-diameters) at every one second for STS and MODE. According to the goals of those two protocols, we can expect that STS could achieve smaller s-diameters, but a bit larger diameters than MODE. Fig. 6 shows the results. We can figure out that the s-diameter of STS is smaller (Fig. 6(b)). On the other hand the diameter of STS remains larger than MODE according to Fig. 6(a). Note that the diameter is not dominant on the streaming as long as one of the sender nodes play the role of streaming source.

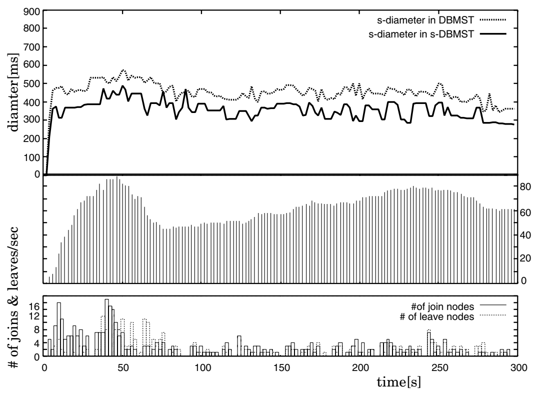
The average diameters and s-diameters were 651, 364 successively for STS and 585, 480 successively for MODE in milliseconds. They were measured performing simulations for 10 different sessions, where each followed the above scenario. According to those results, we can again say that STS can support multimedia applications better than MODE.

[Control Traffic]: The control traffic is shown in Fig.7. The highest traffic amount has been generated around 30 seconds (around the first collection phase) as the number of nodes has reached to top. Even taking this peak value (350kbit) together with the number of nodes in the session at that time (90 nodes approximately), the average traffic amount on a single node can be calculated as 4kbit/sec. We can say this value (4Kbps/node as maximum) is small enough for streaming applications which usually consume several hundreds of Kbps.

[Join/Repair Procedure Overhead]: The time required for join and repair procedures explained in Section 3 was 930 and 790 milliseconds successively. According to these results, the time required for restoration of isolated trees (repair time) remains less than 1 second, which can be considered small enough for multimedia streaming. The time required for the join procedure (join time) here is larger than the repair time. Here we have set the connection permission message timeout (the time each joining node waits to receive connection permission messages before it selects the best position to connect) large enough (0.6[s]) to receive as more as connection permission messages. So the join time holds a larger value



(a) Diameter



(b) Sender-Dependent Diameter (S-diameter)

Figure 6. Dynamics of (a) Diameters and (b) S-diameters

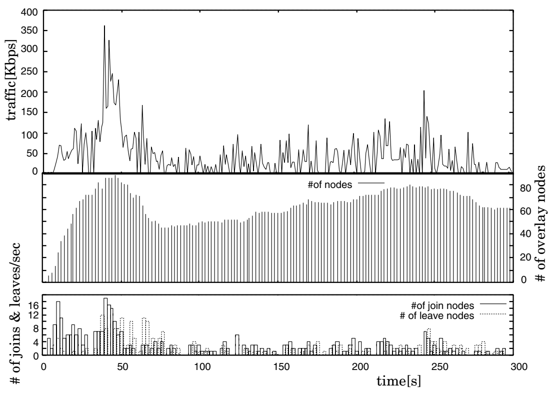


Figure 7. Control Traffic (Total kbits on tree per each second)

(0.93[s]), but this value is considered reasonable enough as time required for bootstrapping.

5.2 Experiments on Real Networks

We have used 8 machines placed in the same LAN (100Base-TX) and run 4-6 processes on each machine to emulate 40 user nodes. Here, we made each process keep the packets for a certain time period before forwarding, if the forwarding target is located in the same machine where that process runs. In this way, we could prevent the link-delay between the nodes located in the same machine from being too small. The scenario is as follows. All nodes join the tree before streaming video so that they can be ready to receive the initial frame of the stream. It includes some information to decode and play video such as its resolution and frame rate. Throughout the session, the streaming-source changes it's position through the sender nodes. During the streaming, 3-5 nodes are set to leave in every interval between collection phases. The period is of 60 seconds. Here the link delays vary from 10[ms] to 100[ms] and the initial degree bound for each node is 4. Each node's degree bound is dynamically changed to enhance the media

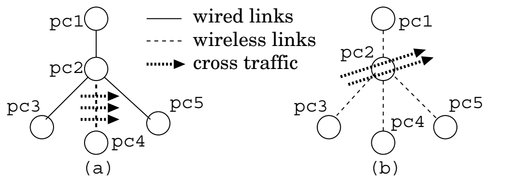


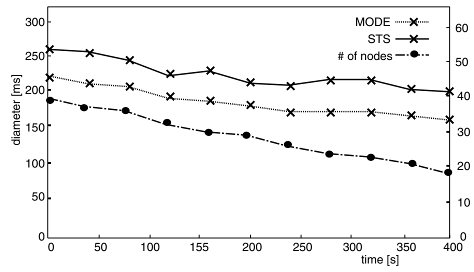
Figure 9. Degree Adaptation Experiment

delivery performance on the tree as described in Section 4. And also, we set STS to select its sender nodes increasingly with the total node count in a manner to make the senders, which includes the initial node as well, occupy 20% of the entire nodes.

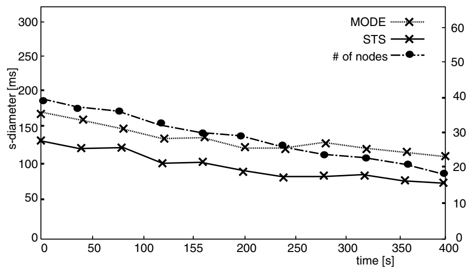
[Diameters and S-Diameters]: We have measured the dynamics of diameters and s-diameters. We have used the same scenario for 10 cases each of which has different locations for sender nodes, and have measured the average for every 25 seconds. The measured variance is shown in Fig. 8. We can see that the diameter of STS is held higher than that of MODE (Fig. 8-(a)). But s-diameter, which counts more in multimedia streaming, is held smaller in STS according to the results shown in Fig. 8-(b).

[Degree Adaptation]: We have checked whether the degree adjustment strategy described in Section 4 works well. For that we have set the 2 scenarios shown in Fig. 9 which illustrates a part of the overlay nodes (machines) we used and the 802.11b wireless links. Fig. 9-(a) represents a case, where a congestion occurs on a unicast link between 2 nodes, and Fig. 9-(b) is another case, where a congestion occurs on the physical network link connected to a node. For each of these cases we have used a 256 Kbps media stream as the multicast media and a wireless node to make the congestions. And also the lower threshold of the bitrate, below which a node detects that a network congestion has occurred, has been set to -20% (205 Kbps) of the stream's real bitrate.

In the first case, we have generated some additional traffic across the link between $pc2$ and $pc4$. Then we could see node $pc2$ has detected the congestion on the down link to $pc4$,



(a) Diameters



(b) S-Diameters

Figure 8. Dynamics of (a) Diameters and (b)S-Daimeters

where it has sent a compulsory-leave message to $pc4$. Then $pc4$ has successfully left and rejoined the session. In the second case, we have generated a separate process which requires some additional traffic, which is enough to make a congestion on the physical network link, on $pc2$. Here we have found that nodes $pc1$ and $pc2$ have detected that downward links are congested, after receiving RTCP reports from their child nodes. Then $pc2$ has sent a compulsory-leave message to $pc3$ and $pc4$ (these are randomly selected to occupy around 50% of the total connection count, in our experiments). Here, the forcibly disconnected nodes, which are subjected to follow the ordinary join procedure, has rejoined to the tree after 1.28 second average value. And the isolated sub-tree, which was located under the forcibly disconnected node, has reconnected to the tree after average 0.43 milliseconds following the ordinary repair procedure.

[Time Required for Join/Repair Procedures]: We have measured the time required to complete the join and repair procedures. The average and maximum values of them were 1212 and 1415 milliseconds for join and 318 and 734 milliseconds for repair successively. We can confirm that the time required for a repair procedure is small even in the worst case. Considering the fact that the repair procedure completes less than in one second, we do not have serious distortion in playback of received video. Here, the time for join procedures remains higher. This is because we have made each joining node wait at least 1 second before making a link to the tree. It allows a node to receive as many permission messages as possible (see Section 3.1). This contributes to let the joining node connect to more closer node to the center node.

Currently STS nodes convey each multimedia data packet to the corresponding neighbor nodes without using the cache (the ring buffer) to support the real-time video conferences. But, readers may understand that STS can be simply modified to adapt applications requiring a higher quality playback (*i.e.*lower jitter, less stream loss) by using the ring buffer, where the jitter can be made lower and the data loss in subtree restoration process can be get to zero or to a negligible value.

6 Concluding Remarks

In this paper, we have stated design and implementation of an overlay multicast protocol for interactive multimedia applications including media streaming. The protocol is called Shared Tree Streaming (STS) protocol that constructs a shared tree called *s-DBMDT* (sender-dependent Degree-Bounded Minimum Diameter Tree) as an overlay network that involves all the participants of the application.

Evaluating STS protocol on large-scale, real environments such as PlanetLab is part of our future work.

References

[1] Y. H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proc. of ACM SIGMETRICS*, 2000.

[2] Y. Nakamura, H. Yamaguchi, A. Hiromori, K. Yasumoto, T. Higashino, and K. Taniguchi. On designing end-user multicast for multiple video sources. In *Proc. of 2003 IEEE Int. Conf. on Multimedia and Expo (ICME2003)*, pages III497–500, 2003.

[3] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proc. of 3rd Usenix Symp. on Internet Technologies and Systems*, 2001.

[4] S.Y. Shi, J.S. Turner, and M. Waldvogel. Dimensioning server access bandwidth and multicast routing in overlay networks. In *Proc. of ACM NOSSDAV 2001*, 2001.

[5] S. Banerjee, C. Kommareddy, K. Kar, S. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proc. of IEEE INFOCOM 2003*, 2003.

[6] R. Cohen and G. Kaempfer. A unicast-based approach for streaming multicast. In *Proc. of IEEE INFOCOM 2001*, 2001.

[7] H. Yamaguchi, A. Hiromori, T. Higashino, and K. Taniguchi. An autonomous and decentralized protocol for delay sensitive overlay multicast tree. In *Proc. of 24th IEEE Int. Conf. on Distributed Computing Systems (ICDCS2004)*, pages p 662–669, 2004.

[8] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. of ACM SIGCOMM 2002*, pages 205–217, 2002.