A Bandwidth and Latency-Aware Protocol for Streaming Multimedia Content

Meng Guo, Qi He
{mguo, qhe}@cc.gatech.edu
Networking and Telecommunication Group
College of Computing, Georgia Institute of Technology

September 16, 2003

Abstract

We design a bandwidth and latency aware application layer multicast protocol to stream multimedia content. This protocol considers network bandwidth heterogeneity, in which the cluster population is bounded by the capacity of the access point. The cluster range is limited by the end-to-end latency threshold. This protocol achieves high scalability by distributed and local administration. The application layer multicast tree is highly flexible in the sense that the host adjusts its location in the tree based on its packet receiving behavior. Through the simulation experiment, we demonstrate that our protocol achieves low latency, high reliability and high adaptivity, while incurring moderate control message overhead.

1 Introduction

Streaming media has been an active research area over the past decade. However, the scalability problem for deploying streaming service with acceptable quality is not yet appropriately tackled. Serving clients using unicast is clearly not scalable. Due to the bandwidth-intensive nature of video streams, the server side links can be easily saturated. IP multicast was proposed as a network layer solution for scalable group communication. *IP multicast* eliminates the duplication of packets on every link of the multicast tree, thus minimizing the incremental cost of serving a new client. IP multicast could be a good solution for scalable media streaming applications, but it is not widely deployed due to its own scalability problems, and its poor support for higher layer functionality.

Application layer multicast implements packet forwarding functions at end hosts [4, 5]. Hosts in the same multicast group form an overlay network, and data transmission is carried over the multicast tree out of this overlay. In previous researches on application layer multicast, the construction and optimization of the multicast tree are mainly guided by the goal of a "well-shaped" tree, evaluated often by the two metrics *stress* and *stretch* [1, 2, 9, 10]. *Stress* is defined by the number of identical packets over a link or node. *Stretch* is the ratio of the path length along the overlay to the length of the direct unicast path. Topology is the only information about the physical network that is used in those previous application

layer protocols. However, the ultimate network resource that enables data transfer is bandwidth. For some other bandwidth sensitive applications, e.g., media streaming, we have to take into account the bandwidth constraint in the design of an application layer multicast tree.

To be resilient to network bandwidth dynamics and node failures in the system, CoopNet [8] uses MDC encoding for media streaming, which is complemented by a bandwidth-aware algorithm to build the application layer multicast tree for each sub-stream. However, in building multicast trees, CoopNet does not consider the latency or proximity among nodes, which might result in multicast trees with large stretch. Furthermore, CoopNet employs centralized control, where the server maintains the global topology information, and is responsible for every node join/leave operation. In our protocol, we use distributed administration, each node in the tree is only responsible for local operations.

In this paper, we propose an application layer solution to streaming media in a bandwidth-constrained network, which also preserves the consideration for stretch. In our solution, the end hosts within the same multicast group are categorized into two classes: access points (AP) and leaf nodes. APs forward packets to other hosts in the group, while leaf nodes do not. APs at level i of the tree forward packets to a cluster of hosts at level $i+1$. Here, a *cluster* refers to as a group of end-hosts that are within the threshold T, in terms of latency, to their parent AP. Compared to previous application layer multicast protocols, our design has the following features:

- Consideration of peer bandwidth heterogeneity: in our protocol, the cluster population is bounded by the capacity of the APs. Previous application layer multicast protocols such as NICE [1, 2, 9] did not consider the bandwidth constraint and heterogeneity of peers.
- Scalability: our protocol is scalable in the sense that each node in the application layer multicast tree only maintains information about its cluster, its parent, and its children.
- Latency-aware clustering: a new node always joins the cluster whose parent AP is within the latency threshold, so that the cluster range in our protocol is limited by the end-to-end latency between adjacent nodes. Proximity of adjacent nodes reduces the stretch of the tree.
- Adaptivity: each host in the multicast group monitors its packet receiving behavior, and makes adjustments if a service quality degradation is discovered.

This paper is organized as follows. We describe our protocol design in Section 2. In Section 3, we show results from a set of simulation experiments. Finally, we conclude the paper and discuss our future research directions in Section 4.

2 Protocol Description

Figure 1 shows a typical application layer multicast tree for our protocol design. In each level of the tree, there are leaf nodes as well as AP nodes. All nodes at level i receive packets from the APs from level $i-1$. A new node joins the tree by connecting to the appropriate APs, and is assigned as AP or leaf node based

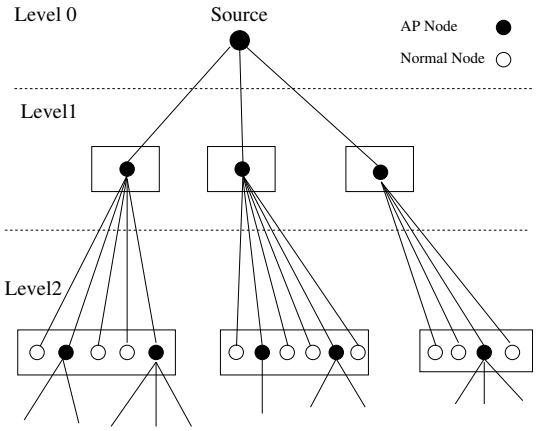

Figure 1: Tree Structure

on the characteristics of the node location and the tree structure. To make the multicast tree wide, all the nodes at level 1 are AP nodes.

The multicast tree in our protocol can be viewed as a two level hierarchy. First, the clusters form a tree structure. (In figure 1, each cluster is represented by a box.) Inside each cluster, the members are centered around the parent AP at the higher level. Every member inside the cluster shares the same parent. Two factors are considered in our protocol to determine the characteristic of the cluster: bandwidth and latency. The *cluster population* refers to the number of nodes in a cluster, is bounded by the available bandwidth of the parent AP. The *cluster range*, in terms of latency, is bounded by a threshold T. The latency from the nodes within the same cluster to their parent AP is smaller than T.

In this section, we first describe the process of a host joining the multicast tree. We then propose the node leave/failure handling and tree maintenance schemes. Along with the protocol description, we give some analysis regarding the major features of our protocol.

2.1 A node joins the tree

In our protocol, the media server S is the central contact point. It maintains the address of all the level 1 APs. When a new host C wants to join the multicast tree, it first gets the IP address of the server and level 1 APs. Then, it probes these addresses for end-to-end latency values. The probing results are stored in a set $L = \{(id, latency)\}$. L_{valid} is a subset of L whose latency value is below T. We further define l_{min} as the minimum latency value in L. The node join algorithm is shown in table 1.

Figure 2 shows several snapshots of the multicast tree. Section 1 is the original tree. New host in section 2 connects to the server since no AP is within the threshold. Section 3 and 4 shows that the client always joins deepest in the tree if possible.

2.2 Membership maintenance

A host in each cluster maintains three class of information: the information about its parent, its children, and its siblings.

```
1: Join (C, S) {
2:     Get the IP address of S and level 1 APs;
3:     Probe these IP addresses and store the results in L;
4:     if l_min > T, then
5:         Y = l_min.id
6:         Deep_Join(C, Y);
7:     else
8:         if l_min.id == S then
9:             if |L_valid| == 1, then
10:                C becomes a child of S and is an AP.
11:            else
12:                l_min = min(L_valid − S)
13:                Y = l_min.id
14:                Deep_Join (C, Y).
15:            endif
16:        else
17:            Y = l_min.id
18:            Deep_Join (C, Y).
19:        endif
20:    endif
21: }

1: Deep_Join(X, Y) {
2:     if Y has children APs within threshold then
3:         find the closest AP Z and Deep_Join(X, Z);
4:     else
5:         node X join Y and becomes a leaf node ;
6:     endif
7: }
```

Table 1: Node Join Algorithm

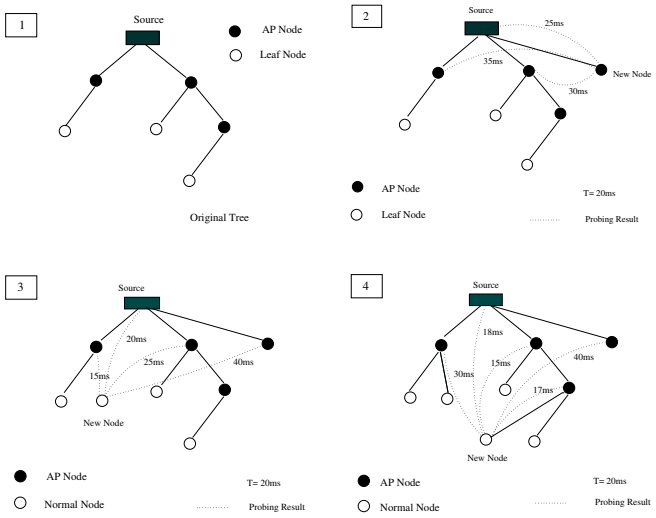

Figure 2: Node Join Process Example

Each node remembers its direct parent information obtained during the join phase. A node sends "keep alive" messages periodically to the parent AP to indicate its existence. The parent knows of all its children. Nodes in the same cluster learn of each other through the parent.

A node at layer i exchanges control messages only with nodes at three layers: $i-1, i, i+1$. At layer i, the intra-cluster control message exchange is performed through the parent AP. Nodes at layer i reports its status to its parent AP at layer $i-1$. AP nodes at layer $i-1$ informs all of its children at layer i of group membership status.

2.3 Node Leave/Failure

Clients may leave the multicast group at any time. There will also be node failure due to network outage, or system crash. When a node leaves the group, it will inform its parent node and the children nodes to adapt to such changes. As to node failure, it is detected by the membership maintenance protocol. The protocol handles node leave/failure in two cases:

- A leaf node leaves the group: the cluster members and its parent AP node update their membership information.
- An AP leaves the group: upon AP leaving the group, it will send a "LEAVE" message to its children. The nodes inside the cluster will elect a new AP point, which will be promoted to the upper layer. The node which has the largest bandwidth capacity, while close to the center of the cluster is promoted as an AP. We define an objective function:

$$w(i) = \frac{capacity(i)}{\sum_{\forall j \in cluster} latency(i,j)}.$$

Node i with the largest value is promoted. Other clients in the cluster will rejoin the newly promoted AP. Nodes that cannot rejoin through the newly promoted AP can rejoin the tree by contacting the server.

2.4 Adaptation to Network Dynamics

To provide high quality media streaming service, the multicast tree should be able to adjust its structure according to the network dynamics. In our protocol, the end host on the multicast tree monitors its packet receiving behavior, and if the packet loss rate is above some threshold value l, it triggers the *tree adaptation* signal. The tree adaptation signal works as follows:

- Single node within one cluster experiences packet loss: it first contacts its parent to see if its parent suffers packet loss. If so, the problem might happen on a higher level path, so it sends tree adaptation signal up to its parent and stands by. Otherwise, packet loss might be caused by congestion on the path from its parent AP to itself. Under this case, this host joins an appropriate sibling node, and this sibling node is promoted as an AP.
- Multiple nodes within one cluster experience packet loss: this situation is detected by the parent AP which receives several tree adaptation signals from its children and can happen in two cases. First, if the path from the server to their parent AP is congested, the tree adaptation signals are sent to the parent AP, and no further actions are taken. Second, if the parent AP is overloaded, those hosts experiencing losses have to rejoin the multicast tree by connecting to the appropriate siblings.

3 Performance Evaluation and Analysis

In this section, we describe the simulation environment and evaluate the performance of the protocol. Our experimentations focus on the following aspects: i. how do streaming packets and control messages relate to node capacity. ii. how is the path latency from the server to each node in our multicast tree compared to the latency of the unicast path between them. iii. how does our protocol perform in terms of join latency and convergence time after a node's leave.

3.1 Simulation Setup

We use ns-2 to evaluate our protocol. Our experimental topology is generated by the GT-ITM [3] transit-stub model. The generated network consists of 1418 nodes. Our bandwidth assignment algorithm follows the one used in [12]. A transit-transit link always has a capacity of 155Mbps. A transit-stub link has equal probability to be assigned a capacity of 155Mbps, 45Mbps and 1.5Mbps. Each of the stub-stub link is assigned a capacity of 10Mbps or 1.5Mbps with equal probability. In our experiment, each intra-stub node, i.e., a node that is attached to one and only one stub node, has an application client attached to it. The length of each client's stay in the group follows a Pareto distribution, and once out of the group, a

node stays outside for a period that follows another Pareto distribution and then rejoins the group from scratch. Throughout the trace, there is one fixed media server.

3.2 Node Degree vs. Node Capacity

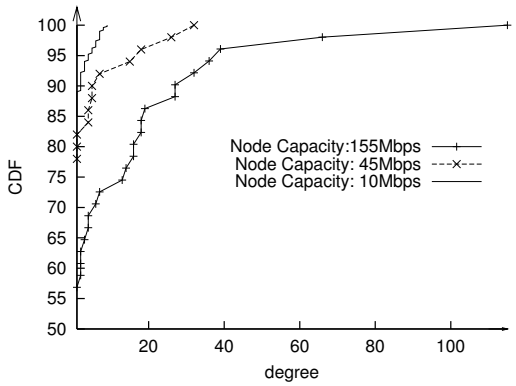

Figure 3: Node Degree Distribution

Node degree refers to the number of clients it is connected. Figure 3 shows the node degree distribution. We compare the result of nodes with $10, 45, 155 Mbps$ bandwidth respectively. As shown in the figure, the average node degree increases as the node capacity increases. And the maximum node degree is never beyond the capacity of each node. These results show that our protocol can effectively adapt the load of each node according to its capacity. We also find that most of the nodes are leaf nodes, which have node degree 1. Only a small portion of nodes are AP nodes, which have higher degrees. The fanout of high degree AP nodes is large, which in turn means that the tree is wide and short. We will see the benefits of the wide-short tree in terms of latency and control overhead in the following sections.

3.3 Control Message Overhead

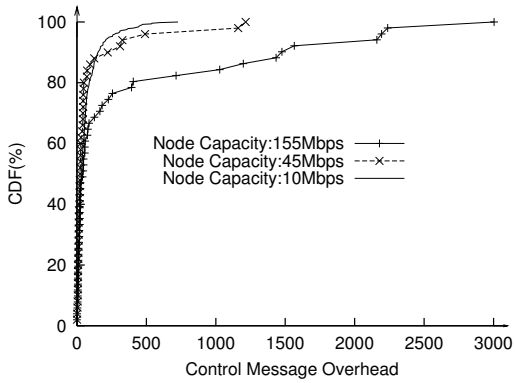

Figure 4: Control Message Overhead Distribution

Figure 4 shows the control message overhead distribution . Control message overhead for a node i in the tree is measured by $\sum_{\forall j \in C} hop(i,j) \times b_{control}$. Here C is the cluster that client i belongs to, $hop(i,j)$ is hop count between node i, j, and $b_{control}$ refers to the control message bandwidth. We can see that the control message overhead curve demonstrates very similar characteristics as figure 3. Most leaf nodes have very small control message overhead. A few AP nodes have to carry much more control message overhead

than average. That is because in our design, leaf nodes only exchange information with their parent APs, whereas the AP nodes exchange information with all the members in its cluster additionally. We also see that the maximum control message overhead corresponds well with the node capacity.

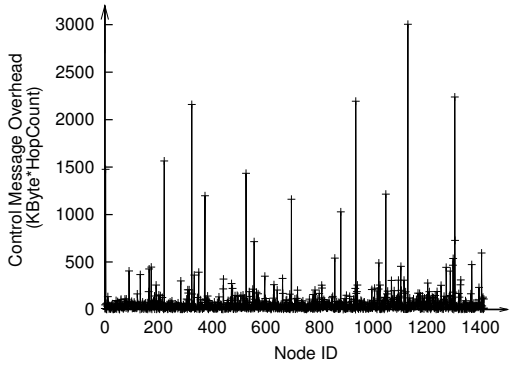

Figure 5: Control Message Overhead

To demonstrate the effects described above more clearly, Figure 5 shows the control message overhead across all the nodes in our multicast tree. We can see that most nodes have very a low control overhead. Whereas a few AP nodes have larger overheads, they also have larger capacities to handle the overhead.

3.4 Join Latency

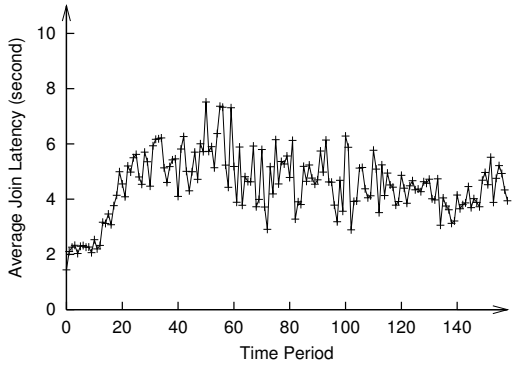

Figure 6: Average Join Latency Over Time

Figure 6 shows the node join latency over the simulation time. We calculate average join latency over every 10 second period. The result shows that the average join time is less than 5 seconds, which is acceptable for most of the multimedia applications. We also see that the join latency over time is relatively stable, which means the shape of the tree and distribution of the AP is well-maintained throughout the simulation.

Figure 7 shows the join latency distribution. It can be seen from the figure that, for more than 70% of the time, the join latency is smaller than 5 seconds. Only about 5% of clients will experience a join latency longer than 10 seconds.

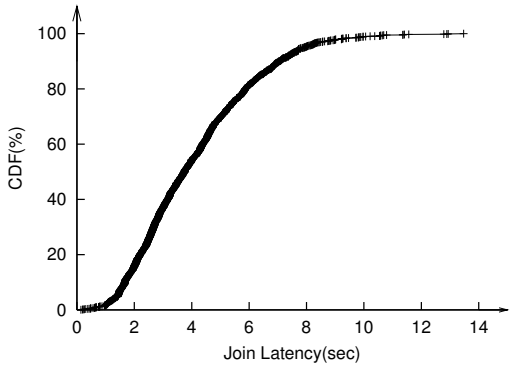

Figure 7: Distribution of Join Latency

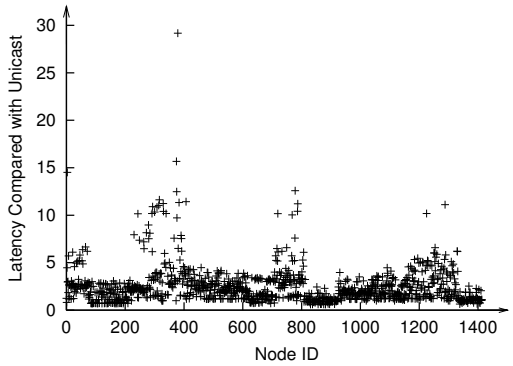

Figure 8: Relative Latency with Unicast

3.5 End-to-end latency compared to Unicast

We conduct simulation experiments to evaluate the performance of the overlay path on the multicast tree. For each client in the tree, we compare the end-to-end latency through the overlay path to the end-to-end latency through the unicast path. The result is shown in Figure 8. First, the average end-to-end latency is smaller than 2 times of the unicast case. Second, some of the nodes achieve smaller end-to-end latency than the unicast case. This is because that our node join algorithm tends to select the overlay path which have small end-to-end latency rather than hop count.

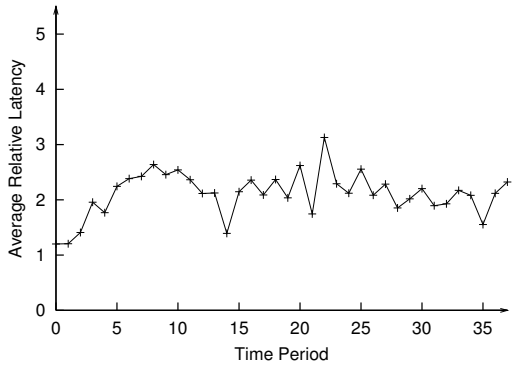

Figure 9: Relative Latency with Unicast over Time

Figure 9 shows the relative latency compared to the unicast over the time throughout the simulation. We find that the relative latency always remains at a stable and acceptably small value. This means that

our protocol is efficient over the time, and is highly adaptive under the dynamics of clients joining/leaving the group.

3.6 Node leave or failure

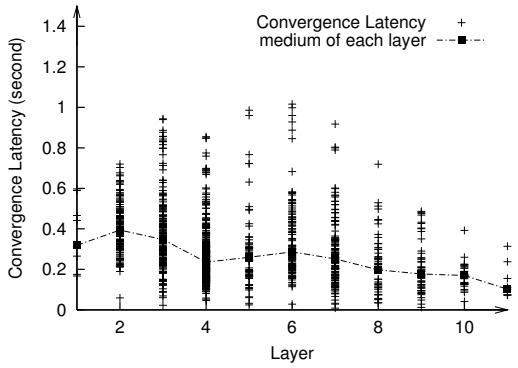

Figure 10: Converge Time after Node Leave/Failure

Since clients can leave the group at any time, it is critical for an application layer multicast tree to provide fast recovery upon clients leaving/failure. Figure 10 shows the convergence time of nodes in different levels of the multicast tree. The medium value of convergence time is generally less than 1/2 second. And the maximum convergence time seen is about 1 second. Our protocol achieves good convergence latency because of two reasons: first, the disconnected node knows the join target node; second, the rejoin process is performed most probability within the local cluster. We also discover that as the layer increases, the convergence time tends to decrease. This is because higher layer nodes tend to have more available capacity, thus the local rejoin success rate is higher.

4 Concluding Remarks

In this paper, we propose an application layer multicast protocol for media streaming. This protocol addresses the constraint and heterogeneity of end host bandwidth capacity. We design a hierarchical cluster-based multicast tree, where the cluster size is determined by capacity of the AP node, and varies from cluster to cluster. We design the node join algorithm such that the latency from any hosts to the AP node of its cluster is within certain latency threshold. Therefore, the protocol achieves low end-to-end latency, which is crucial to real time media streaming applications. To achieve high scalability, in our design, each node in the tree only maintains the state information of its parent, its children, and the peer nodes of its cluster. Through the local repair and adjustment mechanism, our protocol also achieves fast tree recovery, and low self-adjustment latency.

As to the future work, we think the following aspects are particularly interesting.

- AP re-election: sometimes the AP node of a cluster is not the best node to provide the forwarding functions. Members in the same cluster can probe each other to decide which node is the best as an AP node. The criteria is node location and capacity.

- Admission control: a host with low speed access link might hurt the performance of multicast group. A lightweight admission control scheme is needed to prevent such host from connecting into the multicast tree.

References

[1] S. Banerjee, B. Bhattacharjee, C. Kommareddy Scalable Application Layer Multicast In *Proceedings of ACM SIGCOMM*, August 2001.

[2] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, S.Khuller Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications To appear in *Proceedings of Infocom*, 2003

[3] K. Calvert, M. Doar, and E. Zegura. Modeling Internet topology. *IEEE Communications Magazine*, June 1997.

[4] Y.H. Chu, S.G. Rao and H. Zhang A Case For End System Multicast In *Proceedings of ACM SIGMETRICS*, June 2000,

[5] Y.H. Chu, S.G. Rao and H. Zhang Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture In *Proceedings of ACM SIGCOMM*, August 2001.

[6] D. Hrishikesh; B. Mayank; G. Hector Streaming Live Media over a Peer-to-Peer Network. Techniqual Report, Stanford, 2001

[7] M. S. Kim, S. S. Lam, D.Y. Lee Optimal Distribution Tree for Internet Streaming Media. *Technical Report* , U.T. Austin, 2002

[8] V. N. Padmanabhan, H. J. Wang, P. A. Chou, K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. In *Proceedings of NOSSDAV 2002*.

[9] D. A. Tran, K. A. Hua, T. T. Do ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming. To appear in *Proceedings of Infocom*, 2003

[10] D. A. Tran, K. A. Hua, T. T. Do Peer-to-Peer Streaming Using A Novel Hierarchical Clustering Approach To appear in *Proceedings of ICDCS*, 2003

[11] D.Y. Xu, M. Hefeeda, S. Hambrusch, B. Bhargava On Peer-to-Peer Media Streaming. In *Proceedings of ICDCS*, 2002

[12] L. Zou, E.W. Zegura, M.H. Ammar The Effect of Peer Selection and Buffering Strategies on the Performance of Peer-to-Peer File Sharing Systems. In*Proceedings of MASCOTS 2002*, October 2002.