Edge Device Multi-unicasting for Video Streaming

T. Lavian[1], P. Wang[1], R. Durairaj[1], D. Hoang[2], F. Travostino[1]
[1]*Advanced Technology, Nortel Networks, CA, USA*
[2]*Faculty of Information Technology, University of Technology, Sydney*
{tlavian, pywang, radurai, travos}@nortelnetworks.com {dhoang}@it.uts.edu.au

Abstract

After a decade of research and development, IP multicast has still not been deployed widely in the global Internet due to many open technical issues: lack of admission control, poorly scaled with large number of groups, and requiring substantial infrastructure modifications. To provide the benefits of IP multicast without requiring direct router support or the presence of a physical broadcast medium, various Application Level Multicast (ALM) models have been attempted. However, there are still several problems with ALM: unnecessary coupling between an application and its multicasting supports, bottleneck problem at network access links and considerable processing power required at the end nodes to support ALM mechanisms. This paper proposes an architecture to address these problems by delegating application-multicasting support mechanisms to smart edge devices associated with the application end nodes. The architecture gives rise to an interesting Edge Device Any-casting technology that lies between the IP-multicasting and the Application Layer Multicasting and enjoys the benefits of both. Furthermore, the architecture may provide sufficient cost-benefit for adoption by service providers. The paper presents initial results obtained from the implementation of a video streaming application over the testbed that implements the proposed architecture.

1. Introduction

Streaming media is becoming the de facto global media broadcasting and distribution standard, incorporating other media, including radio, television, and film. The low cost, convenient, worldwide reach, and remarkably simple but powerful technical innovation allows a web site visitor to view a sporting event, a tradeshow keynote, or listen to CD-quality music at a click of a button.

Streaming works by first compressing a digital audio or video file and then breaking it into small packets, which are sent, one after another, over the Internet. When the packets arrive at their destination (the requesting user), they are decompressed and reassembled into a form that can be played by the user's system. To maintain the illusion of seamless play, a small buffer space is created on the user's computer, and data start downloading into it. As soon as the buffer is full, the player starts to play. As the player plays, it uses up information in the buffer, but while it is playing, more data is being downloaded. As long as the data can be downloaded as fast as it is used up in playback, the player will play smoothly. However, when the stream of packets gets too slow due to network congestion, the client player has to wait since there is nothing to play, and one may get the familiar drop-out phenomenon.

IP multicasting allows multiple users of a network to listen to a single live stream, making efficient use of network resources. Multicasting avoids delivering numerous connections by broadcasting one stream to a certain point in the network where other users are requesting the same file. IP multicast would have been an ideal technique to broadcast data stream from a source to multiple destinations if it wasn't for several critical factors. First, the business model does not adequately cover the cost of replication of data at each intermediate router. Hence there is no incentive for Internet Service Providers (ISPs) to deploy the service model. Second, multicasting adds software complexity and requires support inside the network in terms of elaborate control support from IP routers, membership managements (IGMP), and multicast routing protocols. Third, enterprises do not want to run multicast for fear of degrading the performance of other mission critical applications. Fourth, WAN connections are extremely expensive. Enterprises are not willing to pay the additional charges incurred from content streaming.

Application level multicast mode (ALM) is attractive since it does not require explicit support from the network. With scalable self-organizing protocol and effective bandwidth monitoring and adaptation [1], ALM may well be an alternative service model to IP multicasting. Recent attempts have demonstrated that sizable cluster of consumers can be served with an ALM service model with adequate quality of service [2].

However, there are still several problems with ALM. Firstly, to deploy a multicast-application successfully, the application has to carry the burden of integrating itself with a particular ALM scheme. This means that multicasting and its associate supports become an integral part of the application unnecessarily. It is desirable to decouple an application from its associated multicasting mechanisms for easy deployment. Secondly, ALM applications often encounter a bottleneck problem at network access links

0-7803-7661-7/03/$17.00

where bandwidth is always limited. It is desirable to avoid the bottleneck at the access link by shifting it to the enterprise's network where LAN bandwidth is abundant and inexpensive; or by shifting it to the network core where bandwidth is more readily available. Thirdly, considerable processing power is required of the end nodes to support ALM mechanisms. This paper proposes an architecture to eliminate these problems by delegating application-multicasting support mechanisms to smart edge devices associated with the application end nodes. The application only needs to unicast a necessary stream to edge devices associated with the destination ends where multicast clients are clustered; hence access links are not overloaded with multiple traffic streams. The destination end edge devices in turn unicast (or multicast or even broadcast) to each individual end application, where bandwidths are readily available within an enterprise. With this architecture, application layer associated multicasting mechanisms are handled by the edge devices and the application is decoupled and released from the underlying multicast support. The paper describes a testbed that implements the proposed architecture for video streaming. The paper also presents some preliminary performance results of the video streaming application using Real Video server and Real Video players.

The paper is organized as follows. Section 2 reviews the application level multicast model and discusses related work. Section 3 describes application level multi-unicast architecture. Section 4 describes a testbed for application level multi-unicast application. Section 5 presents initial experiments and their measured performance and discussion of the results. Section 6 concludes the paper with suggestions for future research.

2. Application Layer Multicast and Related Work

If network support (for multicasting) is already in place, multicasting applications can be developed on-demand and deployed rapidly. In reality, extensive network support is not always readily available. As experiences indicated with IP-multicasting, deployment of applications that rely on IP-multicasting, is slow, even after a decade of beta-operation. On the other hand, if an application does not required the support of the network, it to be deployed immediately. The price to pay for the rapid deployment is that the application has to incorporate many additional elements, otherwise supported by the network, among participating application entities across the network to provide necessary framework for the application.

Application Layer Multicast model has been studied as an alternative model for IP multicasting. In order to multicast data stream from a server to multiple clients at the application level, an overlay network structure must be constructed at the application layer to connect participating end systems. Furthermore, mechanisms for adapting the overlay structure are necessary to provide and maintain adequate level of QoS of the application. Only until recently, ALM has gained momentum when it is apparent that problems with IP multicast will not disappear. Efforts have been made over the last few years to demonstrate that ALM can provide benefits of IP multicasting, yet can be flexibly deployed in the global Internet. Most ALM systems are in the experimental stages. A brief review of these related works is in order.

Yoid [3] is a generic architecture for overlay networks for content distribution. Yoid addresses diverse application such as netnews, streaming broadcast, and bulk mail distributions. Yoid uses disk space to "time-shift multicast distribution and automatic tree configuration.

Overcast [4] possesses similar characteristics to Yoid but focusing on scalable single-source multicast. It implements a new protocol for tracking the global status of a changing distribution tree.

End System Multicast [1] is an overlay network that provides small-scale multicast groups for teleconferencing applications. A new protocol (Narada) is designed for multi-source multicast.

ALMI [5] is an ALM infrastructure for multi-sender multicast that scales to a large number of groups with small number of members. ALMI adopts a centralized approach to overlay multicast where algorithms for group management and overlay optimization are coordinated at a central point.

This paper proposes an architecture that supports an overall effect similar to application layer multicasting. However, the end application does not perform multicasting. Rather, the edge devices associated with the end application perform stream replication, multiple unicasting and management of the clients.

3. Application Layer Multi-Unicast

In this section, we propose an architecture that helps to achieve the effect of application layer multicasting, but the application does not really perform any multicasting across the access link, therefore the bottleneck problem at the access link is avoided. The architecture assumes that an end application can always be associated with an edge router. On behalf of an end application, an edge router can be dynamically configured to perform multi-unicast to other edge devices, if it is associated with the stream server; or to end clients if it is associated with the stream clients. Effort is also made to simplify the support structure. Our general architecture is depicted in figure 1. However, in this paper, we only implement part of the architecture: the client edge device and client end application. That is the application server is directly connected to the destination edge device through the access link, and the edge device provides necessary support for video streaming once it receives the

data stream across the access link. The main objective is to demonstrate that the architecture provides good DVD quality streaming, yet scalable in terms of serving large number of clients, and avoiding access link bottleneck.

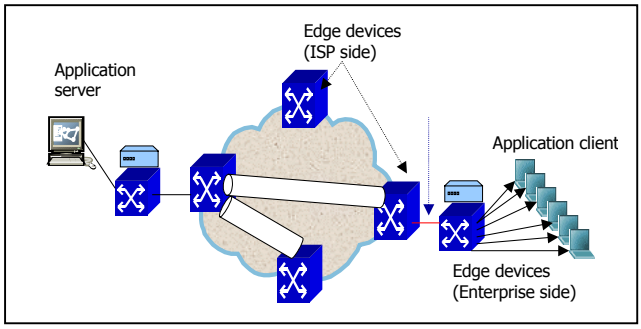


Figure 1. General Application Layer Multi-Unicast from Edge Device Architecture

The main ideas are as follows. Only a copy of the data stream needs to be sent to the edge node associated with the source end application (server) and hence the bottleneck bandwidth problem at the access link at the source is reduced significantly. The approach here is not considered a general approach to multicasting; the emphasis is on deploying processing power of an edge device to provide the required service, yet overcome the limitations of access link bandwidth.

A cluster of clients is always associated with a far-end (destination) edge node. Only one copy of the stream needs to be sent across the network from the source edge to a destination edge. As far as the overlay structure is concerned, it involves only edge nodes. The resulting overlay is simple with each edge node in the overlay structure being just one hop away from the source edge.

A source edge only needs to replicate and unicast the source data stream to a number of destination edge devices. A destination edge only has to replicate and unicast the received data stream to the clients in its cluster. The architecture relies on intelligent edge devices that are capable of filtering and intercepting stream requests, and that have adequate processing power to keep track of various clusters and destination edge nodes participating in the streaming session. Although the edge devices can handle multicasting to various clients, in this paper we only focus on the capability for replication and multiple-unicasting data streams with adequate quality of service. Until our Edge Device Multicasting is fully developed, we assume

- No multicast protocol in the network, and
- No need to have multicast support on the end hosts.

With this architecture, the overlay structure utilizes only edge devices of a network domain: for example, a DiffServ region, an ISP network, or an optical domain. This property is particularly useful since it allows the overlay structure to be adapted easily. An interesting application is to dynamically construct an overlay structure over an optical domain. The source edge device can intelligently discover a light path through the network.

We are particularly interested in the application scenario as depicted in figure 2.

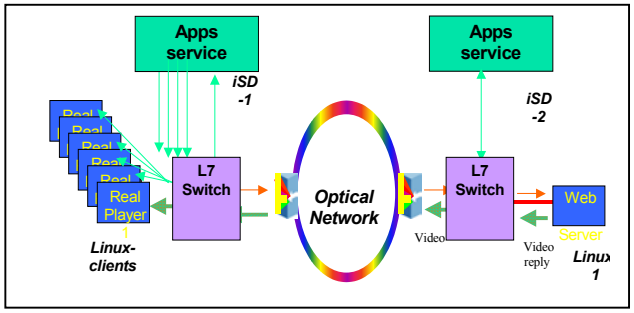


Figure 2. Video Streaming across an Optical domain

However, constructing overlay structures is not considered in this paper. We only explore the replication and management of streaming clients at the destination end in our testbed. Tunneling across a network domain will be explored in the future. It should also be noted that the Nortel Openet Alteon-Based Active Nets platform [6] is used as the edge device inside the enterprise, not on the service provider side in our experiment. The scenario depicted in figure 2 is planned for the next step of our work.

4. Application Layer Multi-Unicast Testbed

We attach a powerful programmable device to a commercial L2-L7 switch. We employ an Alteon web switch as an edge device. The Alteon is a L3-L7 switch that has been used successfully for web-servers load balancing. Here it is used to implement protocol and support for the streaming media application. Using the filtering capability of the Alteon, the switch intelligently intercepts requests, and then performs all the tasks associated with replicating the stream to multiple clients. It leaves the application server the simple task of sending a single data stream to the edge.

Another feature of the Alteon is its ability to redirect a flow of interest, through an extremely fast tunnel, to a more powerful Programmable iSD (Integrated Service Director, a Linux device with a programmable execution environment) for data or control processing in real time [7].

Each iSD runs Linux, and in this set up, performs all tasks associated with streaming. It should be noted that the iSD-Alteon combination constitutes a programmable platform in terms of functionality. The platform allows services to be created and introduced dynamically on demand.

4.1. Experiment setup

We have constructed a simple testbed for demonstrating the proof of concept. The main components of the concept demo are shown in Figure 3. The Alteon iSD runs customized Linux kernel and loaded with NAAP (Nortel Appliance Acceleration Protocol) driver that supports tunnel communications between the Alteon iSD and the Alteon 184. This tunnel is used by the NAAP to package and deliver all those packets that are redirected from the Alteon switch to the iSD by a filter.

The combination of iSD and Alteon 184 switch forms the host platform for the streaming media demo application. The application consists of two parts, 1). Module primarily developed over NAAP APIs that communicates and controls functions like filter creation and, packets capture redirection & packet forwarding on the switch, and 2). Module that does the packet replication function and Real Video stream management.

Real Server and Real player are installed and operated on Red Hat Linux 6.2 based systems. The Real Video server is enabled with three protocols, 1. HTTP [port 8080]. 2. RTSP [Port 554]. 3. PNA. The server is connected directly to the Alteon 184 switch and configured with 10Mbps speed. Real Video player clients are also Red Hat Linux 6.2 systems and all of them share a 10Mbps single link to the Alteon switch. This setup of 10Mbps is done intentionally for making a comparative performance analysis easier.

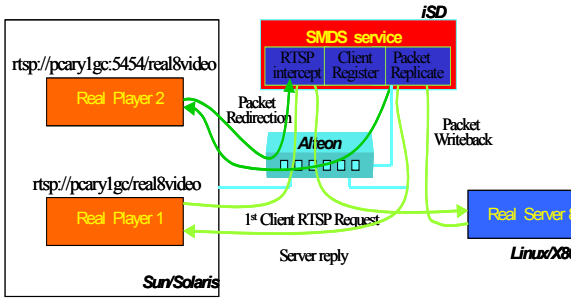


Figure 3. A streaming media demo

In our demo setup, the Streaming Media Distribution Service (SMDS) service that resides on the iSD intercepts all the requests that are made from a Real Player to a Real Server over RSTP and responds to the Real Player as if it were the Real Player. It separately makes a connection to Real Video Server and requests the media content. It then delivers the data stream to the first waiting Real Video Player. Though this resembles a Real Proxy server function, the similarity ends here. For the next request that come from another player for the same live broadcast content, instead of contacting the real player for another separate stream of content delivery, the application replicates the packets that it is already getting for the first session and sends a unicast copy back to the second Real Video player. The same thing is repeated for third request, fourth request and so on.

Operation Details. The SMDS service on the destination Alteon/iSD platform operates as follows:

1. Once the SMDS is downloaded and enabled on the destination iSD, it sets up two filters on the Alteon switch via API calls. One filter is to trap any packets with the Destination IP address of the Real server on Destination port 554 and redirect the traffic to the SMDS on the iSD. The other filter is to trap all the content packets from the Source IP address of the Real server and Source port 554 and redirect them to the SMDS on the iSD.
2. The SMDS software application then binds itself to the end of each of these tunnels (resulting from each filter) and suspends waiting for the filters to trigger.
3. When the first client attempts to connect to the Real server, the first filter is hit and its request packets arrive at the SMDS. The SMDS registers the clients onto a local database using API calls.
4. The SMDS software then forwards the client connection requests on to the Real server and awaits the response.
5. When the Real server starts streaming video content back to the client, the second filter on the destination Alteon is triggered and the content packets are redirected to the SMDS.
6. The SMDS software simply forwards the traffic onto the client that originally requested the content.
7. On successive connection requests from other clients for the same content as in step 3, the SMDS intercepts the request packets but does not forward them to the Real server.
8. The SMDS completes handshaking with each requesting client (acting as a proxy for the Real server) to complete the connection.
9. The SMDS on the iSD then starts duplicating the video content that is being sent by the Real server (to the first client) and forwards the duplicated traffic to each of the requesting clients.
10. On successful completion of the streaming media content, SMDS closes all the client and server connections and waits for a new request.

The core of the SMDS software, that generates the duplicate packets, (as in step 9) involves creating a new header with the appropriate destination IP address and port (of the client), filling the packet with payload from the original content packet and computing the new IP checksum before sending the packet to the physical link.

5. Performance measurement

In order to test the ability of the architecture to eliminate bottleneck at the access link, we measure the performance of streaming video session using our described scheme against the situation where the server has to repeatedly unicast the stream to all clients. The performance is measured in terms of the average number of bits received by each client.

5.1. Bottleneck problem across the access link.

The experiment is conducted as follows. We begin by streaming the video file to one client only and observing the throughput performance. In this experiment the client obtains its streaming data directly from the server without the Alteon in between. The file is encoded at 1.5 Mbps. This simulates T-1 connection and NTSC video quality. Figure 4 shows the throughput of the streaming session with just one client. (The vertical axis measures throughput in BYTES per second for all figures).

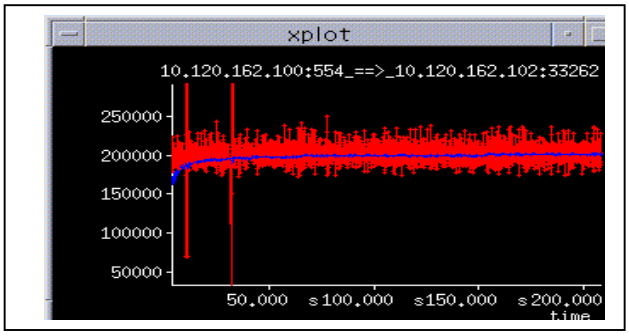
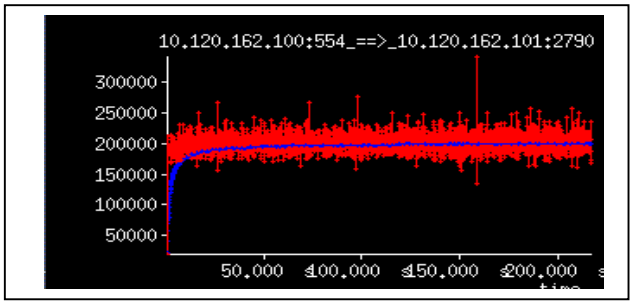


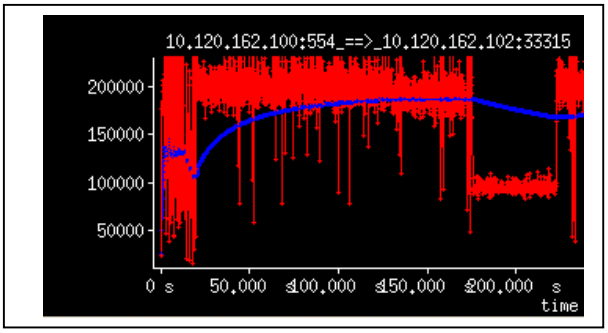
Figure 4 – Streaming with one client

It seems that the streaming progresses as follows. Initially, the server sends data stream as fast as it can across the connection until the client buffer is filled, while the player plays the data stream is continually sent from the server. The average throughput in this case is about 1.5 Mbps, which is the rate of the encoded data stream. No bottleneck occurs with just one client.

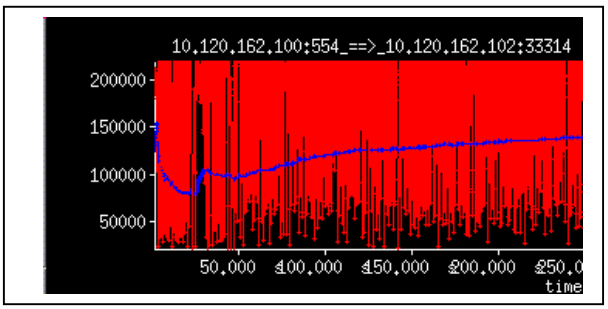
Figure 5 depicts the situation when all clients request streaming at about the same time. The server has to serve 4 clients simultaneously; again streaming data goes directly to each of the clients without being intercepted by the Alteon.



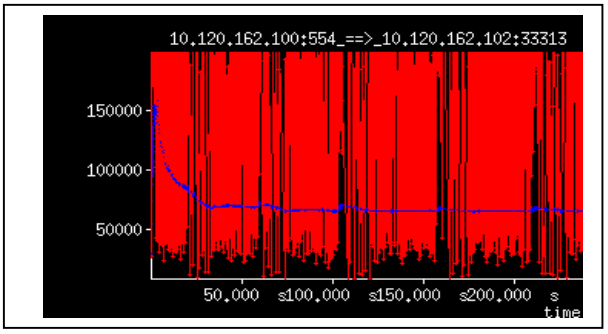
(a)



(b)



(c)



(d)

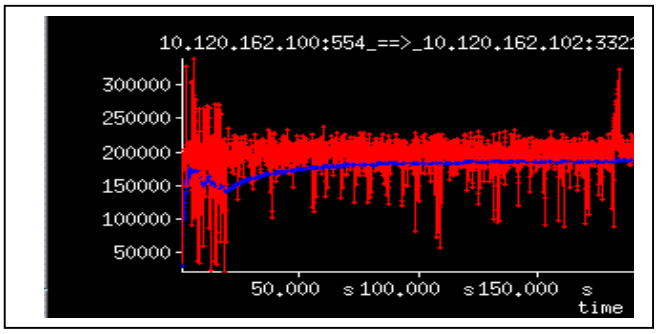
Figure 5. Throughput for individual streams – 4 Real Players request simultaneously

The results imply several interesting phenomena. Firstly, the throughput is distributed unfairly across the four streams. Stream 1 (Fig. 5a) receives the maximum throughput of about 1.5 Mbps, stream 2 (Fig. 5b) receives about 1.3 Mbps, stream 3 (Fig.5c) about 1 Mbps, and stream 4 (Fig. 5d) about 500 Kbps. Secondly, the bottleneck occurs at the access link. The maximum obtained bandwidth is about 4.3 Mbps and this amount of bandwidth has to be distributed across all streams. This means that the quality of the streaming video is degraded at the player when more connections share the access link.

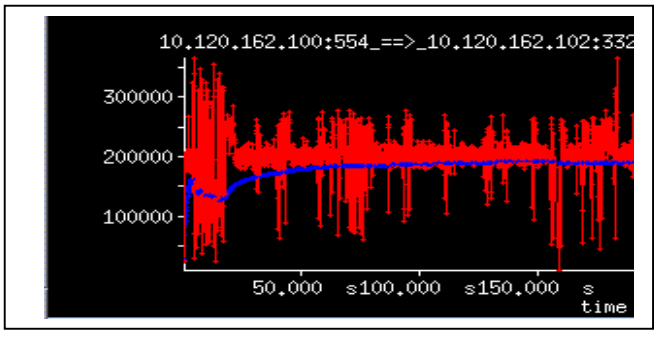
The unfairness may come partially from the unfairness property of the TCP protocol which seems to favor earliest requests, partially come from the Real Video protocol itself which reduces the sending rate to a player when bandwidth is limited or when the player can only receive at a lower encoded rate.

5.2. Video streaming with multi-unicast from the edge device.

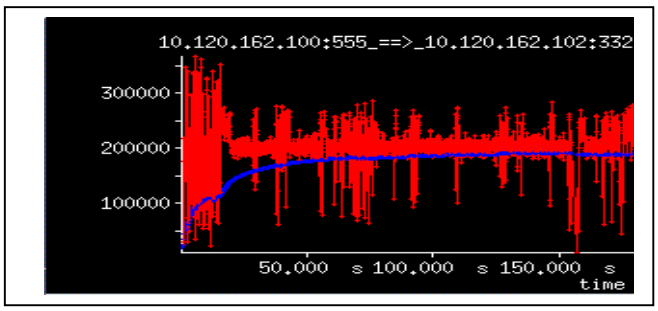
In this experiment, a video streaming is performed following the procedure described in section 4.



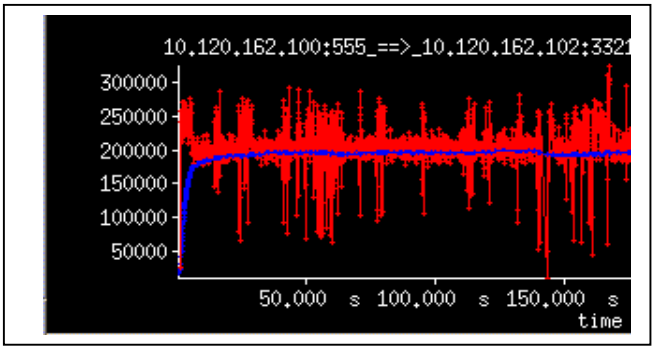
(a)



(b)



(c)



(d)

Figure 6 - Edge Device Multi-unicasting

The aim is demonstrate that the proposed multi-unicast architecture can provide quality of service (in terms of bandwidth) to each of the clients by avoiding the bottleneck problem at the access link. The Real server sends only one stream to the iSD across the access link. The iSD replicates the stream as necessary and sends them to each Real player across separate connections. It is clear from Figure 6 that each client receives the maximum throughput of about 1.5 Mbps. Comparing with the results of Figure 5, the "multi-unicast from the edge device" scheme serves its purpose of avoiding the bottleneck at the access link. The scheme works by basically shifting the bottleneck to the enterprise end of the access link where bandwidth is more readily available.

6. Conclusion

In this paper an architecture that supports the application layer multicast model, but delegates any multicasting tasks to the edge devices, is proposed. The architecture allows the decoupling of an application from its underlying multicast mechanisms, hence permits rapid deployments of the application. By using intelligent edge devices with appropriate active services the bottleneck problem at the access link often encountered by ALM applications can be avoided. The working of the architecture is demonstrated through a real-life video streaming application. Our

streaming media distribution service is simple, yet it demonstrates several important points. Firstly, Streaming Media is just one convenient service that can be deployed on the architecture. The edge devices can be programmed to handle necessary protocols and tasks such as QoS monitoring, adapting self-organizing overlays, and data replications for any ALM schemes. Secondly, the architecture implies that there exists a technology that lies between the IP-multicasting and the Application Layer Multicasting: Edge Device Any-casting (Multi, Uni, or Broadcast). The Edge Device Any-casting is attractive when one considers streaming media across an optical domain where no replication or interception is allowed within the optical core. Furthermore, the architecture will be most effective in that it allows simple overlay multicast structure.

7. References

[1] Chu, Y. H., Rao, S. G., and Zhang, H.: A case for end system multicast. In Proc. ACM SIG-METRICS Conference (SIGMETRICS '00), June 2000.

[2] Chu, Y. H., Rao, S. G., Seshan, S., and Zhang, H. Enabling Conference Applications on the Internet using an Overlay Multicast Architecture. SIGCOMM'01, San Diego, 2001.

[3] Francis, Paul: Yoid: Your Own Internet Distribution. Technical Report, ACIRI, April 2000. www.aciri.org/yoid.

[4] Jannotti, J., Gifford, D., K., Johnson, K., L., Kaashoek, M., F., and O'Toole, J., W. Overcast: Reliable Multicasting with an Overlay Network. In Proc. Of the Fourth Symposium on Operating System Design and Implementation, October 2000.

[5] Pendarakis, D., Shi, S., Verma, D., and Waldvogel, M. ALMI: An Application Level Multicast Infrastructure. In Proc. Of the 3rd Usernix Symposium on Internet Technology and Systems, March 2001.

[6] Lavian, T, Hoang, D., Travostino F., Wang, P., Subramanian, S., "Openet Alteon-Based Commercial Active Nets Platform," submitted for publication, May 2002.

[7] Subramanian, S., Wang, P., Durairaj R., Rasimas J., Travostino F., Lavian, T., and Hoang. D, "Practical Active Network Services within Content-aware Gateways," to appear in DARPA Active Networks Conference and Exposition (DANCE), May 29-31, 2002, San Francisco