CONTENT-AWARE P2P VIDEO STREAMING WITH LOW LATENCY

Pierpaolo Baccichet[*], *Jeonghun Noh*[†], *Eric Setton*[‡] *and Bernd Girod*[†]

[*] Max Planck Center for Visual Computing & Communication, Stanford, CA 94305
[†] Information Systems Laboratory, Stanford University, Stanford, CA 94305
[‡] Streaming Media Systems Group - Hewlett-Packard Labs, Palo Alto, CA 94306
{bacci,jhnoh,bgirod}@stanford.edu, eric.setton@hp.com

ABSTRACT

This paper describes the Stanford P2P Multicast (SPPM) streaming system that employs an overlay architecture specifically designed for low delay video applications. In order to provide interactivity to the user, this system has to keep the end-to-end delay as small as possible while guaranteeing a high video quality. A set of complimentary multicast trees is maintained to efficiently relay video traffic and a Congestion-Distortion Optimized (CoDiO) scheduler prioritizes more important video packets. Local retransmission is employed to mitigate packet loss. Real-time experiments performed on the Planet-Lab show the effectiveness of the system and the benefits of a content-aware scheduler in case of congestion or node failures.

1. INTRODUCTION

In recent years, video streaming over the Internet has become more and more popular due to the increasing availability of bandwidth and recent advances in video coding technologies. In several application scenarios, such as IP-TV, the same video content has to be transmitted to a large population of users. Content Delivery Networks (such as Akamai [1]) are often used to support large numbers of users but they require the deployment of special infrastructure. As an alternative, Peer-to-Peer (P2P) overlay networks have been considered for delivery of multimedia. The idea is that users who are interested in a video stream can act as relay points and forward the data to other users, thus allowing the system to scale with the number of nodes involved in the communication. P2P systems are widely used for video file sharing. For live multicasting, however, the problem is much more challenging, as the overlay network must guarantee high reliability of the connections and a constant flow of data, as well as low startup latencies.

Several different architectures have been proposed in the literature that can be grouped in two different categories. *Mesh-based* approaches (such as GridMedia [2]) aim to construct an overlay mesh whose connections are maintained through "gossiping". Even though these solutions provide good error resilience and network utilization performance, they are usually characterized by high startup delays. This is mostly due to the *Push-Pull* approach followed in the dissemination of the video data that requires the receivers to coordinate the download of different portions of information. On the other hand, *Tree-based* approaches simply *Push* video packets along routes that are determined by constructing one [3] or many [4, 5] multicast trees rooted at the media source. These solutions may lead to low latencies but they often assume the source signal to be encoded using Multiple Description Coding (MDC). With this approach independent descriptions can be sent over different network paths. The quality of the reconstruction at the decoder increases with the number of descriptions received. Alas, MDC introduces substantial redundancy relative to single-description coding. This redundancy unnecessarily degrades the rate-distortion performance, if no packet losses or node failures occur.

In this paper, we present a tree-based solution that does not require MDC but exploits local retransmission to mitigate packet loss. By maintaining multiple distribution trees, it is possible to direct a retransmission request for a missing packet from a particular tree to a parent in another distribution tree. Our approach integrates recent work done on Congestion-Distortion Optimized packet scheduling [6] to prioritize the delivery of video data according to the perceptual importance. An overview of the protocol is provided in Sec. 2 while the details of system architecture and implementation are reported in Sec. 3. Simulation results obtained on Planet-Lab are reported in Sec. 4.

2. OVERLAY MULTICAST PROTOCOL

The *Stanford P2P Multicast* (SPPM) protocol organizes peer nodes in an overlay that consists of a set of multicast trees. The source of the stream is the root of each distribution path and multiplexes video packets on different trees to ensure load balancing. Fig. 1 shows a simple topology with eight peers and two distribution trees as an example of the overlay created by *SPPM*. In the following, we describe how our solution maintains the overlay network and guarantees error resilience.

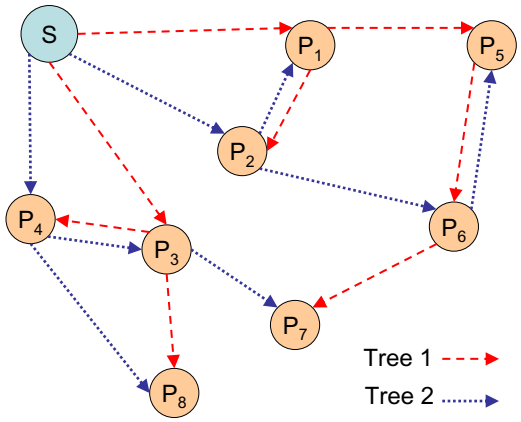

Fig. 1. An example overlay that consists of two multicast trees.

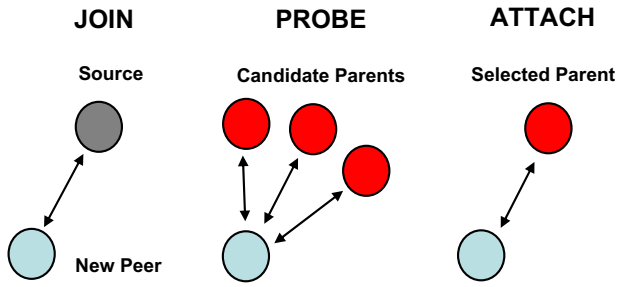

Fig. 2. *SPPM* implements a six-way handshake process to join peers to the session. Each new peer 1) contacts the source, 2) probes a set of candidate parents and 3) attaches to a selected parent.

2.1. Topology construction and management

A new node joining a multicast first contacts the source to obtain a set of nodes that are already connected to the system. These nodes are inquired for available resources to determine whether they can act as parents in the distribution trees. Since the source of the stream is continuously online, it can easily manage a database of these connected peers. The join process consists of a "six-way" handshake as described in the following and shown in Fig. 2.

- *JOIN*. The new peer contacts the source of the video stream that replies with some setup information, such as the number of multicast trees and the video bit rate. Also, it sends a set of currently connected peers that can be polled as candidate parents.
- *PROBE*. For each tree, the new peer probes each candidate parent to obtain their current state and determine the best parent to which to connect. Each candidate parent replies providing the available bandwidth and its height in the distribution tree.
- *ATTACH*. For each tree, one out of the set of candidate parents is selected and an actual connection is established. The parent is selected by minimizing the height of the distribution tree. Also, different parents are chosen as often as possible to exploit path diversity.

2.1.1. Monitoring the connection

Once the connection is established, each peer periodically sends `HELLO_REQ` messages to the parents in the distribution trees which in turn reply with a `HELLO_REP` message. Whenever a host leaves, it stops forwarding video packets and it is unresponsive to `HELLO_REQ` messages. Thus, peers can detect a parent disconnection and trigger the rejoin procedure for the affected tree. Similarly, parent peers record the arrival times of the `HELLO_REQ` messages received by each child, in order to detect children disconnections and release resources.

2.1.2. Local retransmission

Quality degradation due to packet loss and temporary disconnection can be mitigated using retransmission. Note that path diversity allows retransmission requests to be limited to local parents, thus avoiding feedback implosion at the source of the stream. See [7] for further details.

2.1.3. Loop prevention

During the reconnection process, a node may attach to one of its descendants. This loop would eventually starve the whole subtree. To prevent this problem, each peer keeps the list of its ancestors. In our implementation, a specific `ANCESTOR_UPDATE` packet allows updating of this list whenever a change in the topology occurs. Peers can easily reject attachment requests issued by an ancestor. This loop avoidance mechanism does not require global knowledge of the topology but only of a small subset of its nodes. Additional memory and processing power requirements are negligible.

2.2. Content-aware scheduling

Relaying traffic over the uplink of the peers may lead to congestion on the multi-hop path separating the source from any particular peer. Because the rate of a video stream often varies, a peer may sometimes lack the resources to forward all the data expected by its descendants. When a peer has to drop some packets to ensure timely delivery of the more significant portion of the streams, prioritized scheduling can help maintain video quality. In related work [7], we presented a Congestion-Distortion optimized *CoDiO* packet scheduler. This prioritization algorithm bases its decisions on the "importance" $D(n)$ of each enqueued packet. The metric reflects how decoding of a particular packet reduces distortion and it captures the dependencies among different packets as shown in Fig 3. Hence, the scheduler adapts its decisions to the video content by relaying more important packets first.

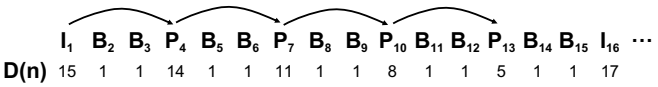

Fig. 3. Content-aware packet scheduling can be implemented by looking at the interdependencies among video packets.

3. SYSTEM ARCHITECTURE

We have implemented a real-time prototype of the *SPPM* protocol that uses the state-of-the-art H.264/AVC [8] to efficiently encode the source signal. Video frames are fragmented at the source to a maximum of 1300 bytes to reflect the MTU of a typical ethernet network thus avoiding packet fragmentation in lower layers of the protocol stack. We use UDP as a transport protocol to avoid the overhead involved in the usage of TCP. Two different UDP ports are used for the control and video traffic respectively. A list of the different packet types is reported in Table 1.

The software architecture of a *SPPM* peer is shown in Fig. 4. Several threads provide the services needed and the "*queue manager*" and the "*routing table*" monitors are used for synchronization. The basic modules are reported in the following.

- **Control Agent** is responsible for maintaining the connection to the system. It accepts connection requests from the children and updates the routing tables accordingly. It periodically monitors the status of parents by sending `HELLO_REQ` messages and, in case of a disconnection, it triggers the reconnection process.
- **Video Agent** receives the incoming video packets, retrieves from the routing table the list of children to which the packet has to be relayed and then stores it into the video queue.

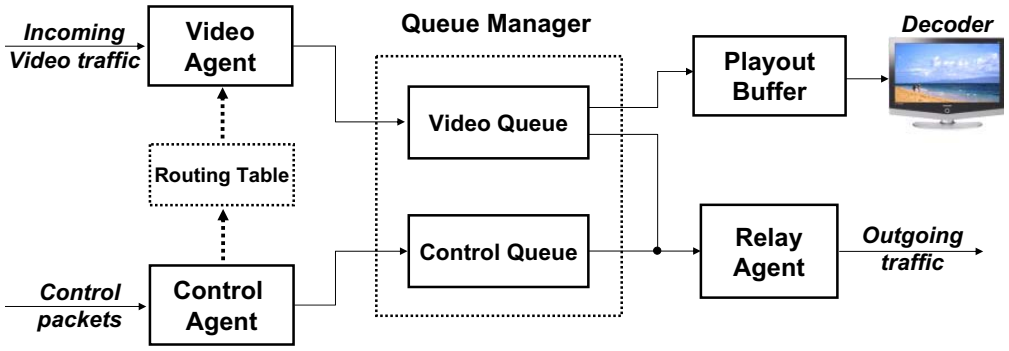

Fig. 4. The *SPPM* software architecture. Video and control packets received from the network are stored into a common buffer within a *Queue Manager* that performs prioritization and garbage collection. Packets are then relayed to children or displayed at regular intervals by appropriate playout and relay agents.

- **Routing table** stores the list of children for each tree.
- **Queue Manager** implements a common buffer to store packets that are to be relayed or displayed. A garbage collector periodically discards packets that are expired, i.e. that have passed their playout deadline.
- **Relay Agent** periodically extracts from the queue the next packet to be sent and forwards it to the appropriate child.
- **Playout Buffer** retrieves packets that are to be displayed from the video queue and passes them to the video decoder.

The source of the video stream implements two additional modules that are described in the following.

- **Video Packetizer** reads the H.264/AVC input video, either from a file or from a socket, and performs packet fragmentation whenever it is necessary. In our implementation, we set the maximum size of a video packet equal to 1300 bytes to reflect the MTU of an Ethernet.
- **Connected peers'database** maintains an approximate list of connected peers. Each entry in the list is refreshed with periodical information received from the peers via `PEER_UPDATE` packets. If the source fails to receive such packets from a registered peer for a certain amount of time, it considers the peer as disconnected from the system and frees the entry in the table.

3.1. NAT Support

A vast majority of residential users access the Internet via some kind of Network Address Translator (NAT) device that usually filters unsolicited incoming traffic directed to nodes of the local network. A NAT will forward incoming traffic from a specific IP address only if a packet has been sent to that IP address before. In order to support devices behind NATs, our "six-way" handshake procedure has to be slightly modified because the `PROBE_REQ` messages would be normally filtered as unsolicited traffic. We assume the source of the stream to be running on a computer with a static IP address. When the source receives a join request, it can detect the public address of the new peer and send this information to the set of potential candidate parents. The parents then transmit a dummy packet to the control port of the new peer. This creates a binding on the NAT device that will enable the forwarding of subsequent `PROBE_REQ` messages.

4. EXPERIMENTAL RESULTS

We tested our real-time *SPPM* prototype on Planet-Lab [9] and performed experiments using 100 nodes. As a test video, we concatenate several well-known sequences at CIF resolution at 30 frames per second. We encoded the video using H.264/AVC at 400kbps obtaining an average video quality of 34.5 dB of PSNR. In order to allow users to join at different times, we sent one intra coded frame every half second. The group of pictures (GOP) was similar to the one shown in Fig. 3. The playout deadline was set to 5 seconds and we used 8 different multicast trees.

Each session consisted of 30 minutes of video streaming. The peers were scheduled to randomly connect within the first minute and to stay connected to the system for the whole session. Note that each node on Planet-Lab shares both link and processing power with several other users. In this environment, conditions may vary quickly possibly causing temporary congestion to occur on the links.

4.1. Video performance

We show the results of three different experiments to demonstrate the effectiveness of the retransmission and prioritization scheme of our solution. We observed that several nodes fail to provide good video quality to the user when retransmission and prioritization are not used. In this case, the PSNR across all the peer is equal to 30.1 dB. A significant improvement is observed after introducing retransmission with an average quality of 31.8 dB while the best performance of 33.2 dB was obtained by combining retransmission with the content-aware packet scheduler described in Sec. 2.2. Fig. 5 shows the average video quality observed at selected peers.

Whenever a peer fails to receive data for a certain amount of time, the video decoder performs error concealment by displaying previously received frames in lieu of the lost ones. Long frame freezes can be considered an indication of a failure in the connection to the system and must be limited. We observed a significant decrease in the number of frame freezes by using retransmission and prioritization, as reported in Table 2.

4.2. Startup delay

The main design goal of our solution is to provide high video quality with very low latency. To validate the effectiveness of the implementation, we recorded the minimum startup delay necessary to receive the first decodable picture at each peer. We observed an average

Type	Size	Description
JOIN_REQ	40	Join request is sent by the new peer to the source.
JOIN_REP	200+	Join reply contains setup information and list of candidate parents.
PROBE_REQ	32	Probe request is sent to every candidate parent to collect information about the current status.
PROBE_REP	40	Probe reply contains information regarding the current status of the candidate parent.
ATTACH_REQ	40	Attach request is used to establish the actual connection with the selected parent.
ATTACH_REP	40	Attach reply confirms the established connection.
HELLO_REQ	36	Hello request sent to periodically monitor the state of the parent.
HELLO_REP	40	Hello reply is used to disseminate information about the end-to-end delay from the source.
PEER_UPDATE	60	Status information periodically sent to the source to update the list of connected peers.
ANCESTOR_UPDATE	60+	Propagates the list of ancestors on each tree to avoid creation of loops.

Table 1. Summary of the packet types used by *SPPM*. Sizes are expressed in bytes and include the overhead due to UDP/IP.

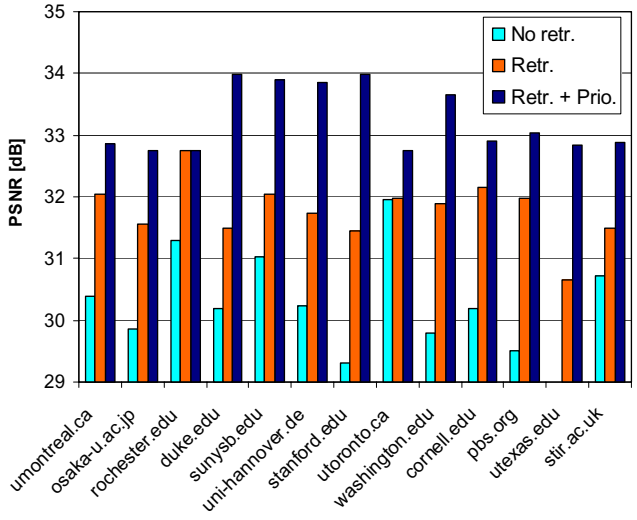

Fig. 5. Average PSNR measured over 30 minutes of video on Planet-Lab for 13 sample nodes.

Scheme	0.5 to 1 sec.	1 to 2 sec.	More than 2 sec.
No retransmission	4262	1784	457
Retransmission	742	283	127
Retrans. & Prioritization	364	143	43

Table 2. Number and duration of the observed frame freezes for the three considered scenarios over 100 different peers.

startup delay of one second. All the peers connected to the system within two seconds from the join request.

4.3. Protocol overhead

The *SPPM* protocol introduces a very small control overhead to manage the overlay. We recorded the size and number of packets exchanged, including the overhead due to UDP/IP encapsulation. We observed control packets are no more than 4% of the total traffic. If we consider the additional header information introduced for each video packet, i.e., sequence numbers and timestamps, the overhead of the protocol is smaller than 8% of the total traffic.

5. CONCLUSIONS

In this paper, we reported the design and implementation of the *Stanford Peer-to-Peer Multicast - SPPM* protocol that allows the delivery of a video stream from a source to a population of users with very low delay. Experimental results obtained on the Planet-Lab confirm the effectiveness of the protocol. Startup delay was measured on the order of one second (on average). Video quality was improved by 2dB on average by introducing a Congestion-Distortion Optimized packet scheduler and local retransmission.

6. REFERENCES

[1] "Akamai," *http://www.akamai.com*.

[2] M. Zhang, J. Luo, L. Zhao, and S. Yang, "A peer-to-peer network for live media streaming using a push-pull approach," *Proc. of the 13th annual ACM international conference on Multimedia*, pp. 287–290, 2005.

[3] Y. Chu, S. Gao, and H. Zhang, "A case for end system multicast," *Proc. ACM Sigmetrics, Santa Clara USA*, June 2000.

[4] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-bandwidth content distribution in a cooperative environment," *Proc. IPTPS'03, Berkeley, CA*, Feb. 2003.

[5] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," *Proc. ACM NOSSDAV, Miami Beach, FL*, May 2002.

[6] E. Setton, "Congestion-Aware Video Streaming over Peer-to-Peer Networks," *Ph.D. Thesis, Stanford University*.

[7] E. Setton, J. Noh, and B. Girod, "Rate-Distortion Optimized Video Peer-to-Peer Multicast Streaming," *Workshop on Advances in Peer-to-Peer Multimedia Streaming at ACM Multimedia, Singapore*, pp. 39–48, Nov. 2005.

[8] ITU-T and ISO/IEC JTC 1, *Advanced Video Coding for Generic Audiovisual services, ITU-T Recommendation H.264 - ISO/IEC 14496-10(AVC)*, 2003.

[9] "Planet-Lab," *http://www.planet-lab.org*.