Path Diversity and Bandwidth Allocation for Multimedia Streaming

Thinh Nguyen, Puneet Mehra, and Avideh Zakhor
Department of Electrical Engineering and Computer Sciences,
University of California, Berkeley, CA 94720
{thinhq, pmehra, avz}@eecs.berkeley.edu

Abstract—

The recent advent of widely available broadband Internet access has resulted in an explosive growth of new video streaming applications and research into methods to efficiently support such applications over today's Internet. Many approaches, including source and channel coding techniques, have been proposed to deal with the delay, loss, and time-varying characteristics of best-effort packet-switched networks. In this paper we present two class of techniques for such networks: the first set of techniques is designed for streaming to receivers with bandwidth-limited last mile ***connections to the Internet, while the second set explores techniques relying on path diversity to handle situations in which the path to the video source, and not the access link itself, causes degradation in the quality of the video stream. We show results of simulations and Internet experiments, demonstrating the effectiveness of the two techniques.***

I. Introduction

From news clips on sites such as CNN.com, to video on demand from MovieFlix [1], the current Internet offers a richer multimedia experience than the past. The challenges of video streaming over best-effort networks, including delay, packet loss, and varying bandwidth, have led to several different approaches to efficiently support multimedia streaming applications. The source coding community has proposed layered and error-resilient codecs [2], [3] to deal with the problems of varying bandwidth, and packet loss respectively. Forward Error Correction(FEC) [4], a channel coding technique, has been proposed to reduce retransmission delay at the expense of increased bandwidth utilization.

In this paper we present two sets of techniques designed to deal with the problems mentioned above. We first focus on streaming to receivers with bandwidth-limited access links, running multiple concurrent networking applications. These applications tend to compete for access link bandwidth and may therefore interfere with streaming performance. Traditionally, streaming applications have used TCP-Friendly UDP protocols, which must maintain fairness with all flows, including these competing applications. However, by streaming over TCP, it is possible to allow a user to break fairness, locally, among her own connections, and to reserve additional bandwidth for high-priority streaming applications. In prior work [5] we outlined a receiver-based bandwidth sharing system (BWSS) for allocating the capacity of last mile bottlenecks among different TCP connections, and demonstrated its utility for video streaming [6]. We demonstrate that such a system can actually provide better streaming performance than congestion-aware UDP protocols.

Since the BWSS does not provide any benefits if the path to the video source is a bottleneck, we discuss another class of techniques which may be used to effectively handle such situations. In prior work, we have demonstrated the benefits of path diversity for multimedia streaming over the Internet. Specifically, we have shown that streaming portions of a stream from multiple concurrent sources offers benefits over streaming from a single host [7], and that this method may be combined with FEC for additional benefits [8]. We have also demonstrated that a single source may be combined with multiple relay nodes to emulate multiple sources [9]. We provide a more detailed discussion of these contributions in this paper.

The rest of the paper is organized as follows. In Section II we discuss the BWSS and the benefits provided to receivers with bandwidth-limited access links. In Section III we discuss the different path diversity related techniques which may be used to deal with undesirable network conditions on the path from source to receiver. We conclude this paper in Section IV.

This work was supported by NSF grant ANI-9905799 and AFOSR contract F49620-00-1-0327.

II. Case 1: Bandwidth-Limited Access Links

Standard TCP is unsuitable for streaming to bandwidth-limited access links, since TCP shares the link capacity of bottleneck links according to connection round-trip time (RTT). Hence, since TCP is unaware of application or user preferences, a user's low priority FTP connection may get much more bandwidth than their high-priority video streaming application. Our previously proposed BWSS addresses this shortcoming of TCP by allowing receivers to allocate link capacity according to application priorities as specifed by the user. It functions by adjusting the flow-control window advertised to the sender, i.e. by controlling the receiver advertised window for each connection. The essential idea behind the system is to constrain the throughput of certain low-priority applications in order to provide additional bandwidth, if possible, for higher-priority flows as specified by the user's preferences. A user is allowed to set a minimal rate, a weight and a priority for each connection, and the link capacity is allocated as follows: first the minimal rate is provided to each connection in decreasing order of priority, and then any remaining bandwidth is shared according to application weight. A more detailed description of the BWSS is provided in [5], [6].

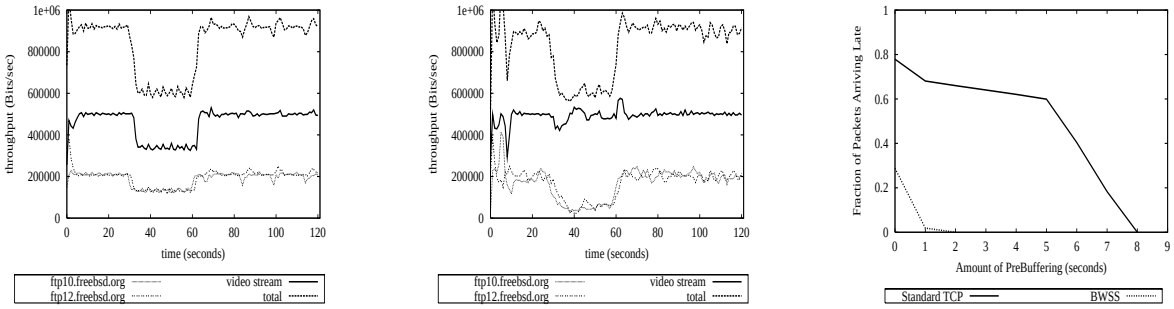
A. Internet Experiments

In what follows, we will describe experimental results demonstrating the effectiveness of BWSS in video streaming applications over bandwidth-limited access links.

A.1 Experimental Setup

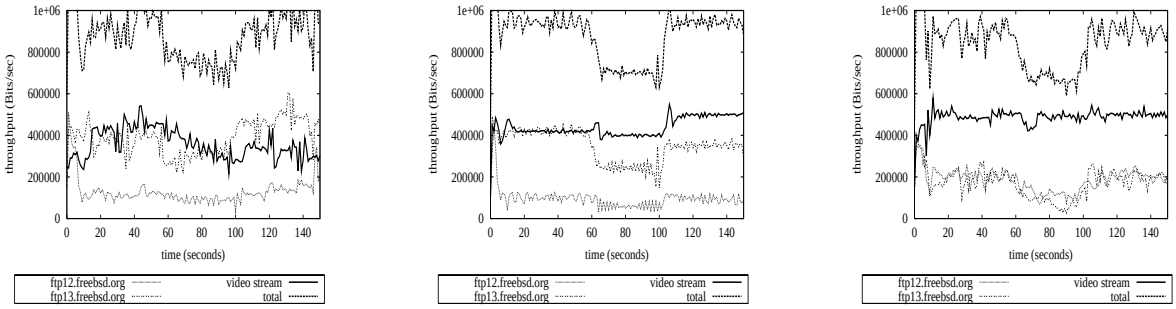
In all of our experiments, **vonnegut**, a machine located in the eecs.berkeley.edu domain, serves as the receiving host. We use the NIST Net [10] network emulation package to emulate a slower Broadband connection, and limit the incoming link to 960Kbps with an additional delay of 30ms, thus modeling a 1Mbps broadband access link.

All experiments involve a video source and two FTP sources sending data to **vonnegut**. At some point during each experiment, cross-traffic from another host in the eecs.berkeley.edu domain is sent to **vonnegut**, creating congestion on the access-link. The interfering cross traffic is a constant bit-rate (CBR) UDP stream generated using the RUDE software package [11].



(a) (b) (c)

Fig. 1. Comparison of video streaming using TCP and BWSS; (a) TCP bandwidth partition; (b) BWSS bandwidth partition; (c) Fraction of late packets. For graphs (a) and (b), the top line is the total bandwidth, the dark solid line represents BWSS throughput, while the light dashed lines represent ftp throughput.



(a) (b) (c)

Fig. 2. Comparison of SureStream streaming using (a) TCP, (b) UDP, and (c) TCP with BWSS; The top line in the graphs is the total bandwidth, the dark solid line represents BWSS throughput, while the light dashed lines represent ftp throughput.

The throughput received by applications is measured by intercepting *read()* system calls, and recording the timestamp and size of the received packet. We perform 3 trials of each given experiment and graph the average of these runs.

A.2 Results

The first experiment compares standard TCP against TCP using the BWSS. We send video packets at a rate of 496Kbps in the form of 62 one kilobyte packets per second. There are 2 concurrent ftp connections from **vonnegut** to ftp10.freebsd.org and ftp12.freebsd.org along with the streaming application. In addition, from t=30s to t=60s we generate a 320Kbps interfering UDP cross-traffic stream. We assign the following parameters for the BWSS to reflect user's preferences: the video stream has the highest priority, a minimal rate of 496Kbps and a weight of 0, while the ftp streams share the remaining link capacity equally. As shown in Figure 1(a), when congestion occurs, the video stream bandwidth is reduced in the case of standard TCP. Meanwhile, as Figure 2(b) demonstrates, the BWSS is able to reduce the bandwidth given to the ftp connections to ensure the desired minimal rate for the video stream. Finally, in Figure 2(c) we plot the fraction of packets arriving past their playback deadline at the receiver as a function of the number of seconds of pre-buffering. As seen, the fraction of late packets is considerably smaller in BWSS than in regular TCP. This experiment demonstrates that standard TCP alone is not good enough for efficient streaming.

We now demonstrate that streaming with TCP and the BWSS can offer better performance than a congestion-adaptive UDP protocol. We stream a trailer for the movie "The Lion King" encoded in RealNetworks's SureStream[1] format, which supports encoding a video stream at multiple rates and selects the appropriate streaming rate based on client feedback about network conditions. The trailer is encoded at 450Kbps, 350Kbps, 262Kbps, and 60Kbps. The TCP-friendly, or congestion-adaptive in the case of SureStream, streaming protocols must react to congestion by reducing the sending rate regardless of whether congestion occurs within the network, or at the user's access link. Meanwhile, the BWSS is aware of the other TCP connections on the user's machine, and is able to break the standard TCP fairness among these flows, without adversely affecting external flows, in order to provide additional bandwidth for high-priority streaming applications.

The detail of the experiment are as follows: the video client used is the RealOne player for Linux, while the video stream is generated from a host in the eecs.berkeley.edu domain running a basic version of the Helix Universal Server. In addition, two concurrent ftp sessions to ftp12.freebsd.org and ftp13.freebsd.org are initiated on the receiving host. A 240Kbps cross-traffic UDP stream is generated from t=60s to t=100s in the experiment. The following parameters are used for the BWSS to reflect user's preferences: a minimum rate of 520Kbps for the video stream, and the remaining bandwidth is shared equally between the ftp connections. As shown in Figure 2(a), standard TCP experiences bandwidth fluctuations which prevent consistent streaming at 450Kbps. Furthermore, the SureStream technology does not select a lower-rate video encoding, resulting in many dropped frames and an overall poor video experience for the user. Meanwhile, as Figure 2(b) demonstrates, UDP streaming results in streaming at 350Kbps until after congestion subsides, at which point the 450Kbps encoding is chosen for streaming by the server. Finally, as Figure 2(c) shows, using the BWSS results in streaming the 450Kbps encoding throughout the experiment. The BWSS allows the user to break fairness among applications in order to stream the highest quality video encoding possible.

[1] SureStream, Helix Producer, Helix Universal Server and RealOne Player area all trademarks of RealNetworks Inc.

III. Case 2: Large Bandwidth Access Link

Traditionally, media streaming over the Internet has been accomplished using a single fixed route between the sender and the receiver throughout the session. If the network is congested along that route, video streaming could suffer from high loss rate and jitter. In this section, we describe a new class of techniques based on path diversity to combat packet loss and insufficient bandwidth for media streaming over the Internet for situations where the bottleneck is not at the last mile.

In the path diversity framework, either multiple senders cooperate with each other to stream the media simultaneously on separate routes at appropriate rates to the receiver, or a single sender streams the media to the receiver using multiple routes created via relay node. If the bandwidth bottleneck is not at the last mile, multiple disjoint routes could potentially increase the streaming throughput. It is also a diversification scheme in that it combats unpredictability of congestion in the Internet. If a particular route experiences congestion, the receiver can redistribute streaming rates among other routes, and use FEC to recover lost packets on one route from the received packets on the other routes.

A. Path Diversity Protocol for Multiple Senders

Our transport protocol is a receiver-driven one in which, the receiver coordinates transmissions from multiple senders based on the information received from the senders. Each sender estimates and sends its round trip time to the receiver. The receiver uses the estimated round trip times and its own estimates of senders' loss rates to calculate the optimal sending rate for each sender in such a way as to minimize the probability of packet loss and satisfy the TCP-friendly constraint [12]. The receiver monitors variations in route conditions of each sender in order to re-adjust rate distribution among senders. When the receiver decides to change sending rates, it sends an identical control packet to each sender. The control packet contains the synchronization sequence number and the optimal sending rates as calculated by the receiver for all senders. Using the specified sending rates and synchronization sequence number, each sender runs a distributed packet partition algorithm, to be described shortly, to determine the sender for the next packet; this is done in such a way that every packet is guaranteed to be sent by one and only one sender, and to minimize the probability of late packets.

A.1 Rate Allocation Algorithm

The rate allocation algorithm is run at the receiver in order to determine the optimal streaming rate for each sender. It is based on two following observations. First, in bursty loss environments, sending L/M bps on each of M channels results in less bursty loss than sending all L bps on a single channel. Second, FEC is more effective in uniform random loss environments than in bursty ones. Since Internet packet loss behavior has been characterized as "bursty"', intuitively, FEC is more effective when packets are sent on multiple routes than on one route to the receiver. Using two-state continuous time Markov model for the Internet bursty packet loss behavior [13], our rate allocation algorithm determines the sending rate on each route in order to (a) minimize the irrecoverable loss probability, and (b) satisfy the TCP-friendly bandwidth constraints [12]. The irrecoverable loss probability is the probability that FEC is not able to recover the lost packets completely. For Reed-Solomon code, this probability equals to the probability of more than $N-K$ per N packets are lost. The details of the rate allocation algorithm can be found in [8].

A.2 Packet Partition Algorithm

After receiving the control packet from the receiver, each sender immediately determines the next packet in the video stream to be sent, using the packet partition algorithm. All the senders simultaneously run this algorithm to ensure that no two or more senders send the same video packet, all packets are sent, and also to minimize the probability of packets arriving late at the receiver. The algorithm can be described as follows. Each sender receives control packet from the receiver through a reliable protocol whenever the receiver determines there should be a change in any of the sending rates. The control packet contains the necessary information for every sender to compute $A(i,k)$, the time difference between the estimated receive and playback time for k^{th} packet. The basic idea in our packet partition algorithm is that among all senders $i = \{1...N\}$, the one that maximizes the time difference $A(i,k)$ is chosen to send k^{th} packet. Hence, maximizing $A(i,k)$ is equivalent to minimizing the probability that the k^{th} packet is late. The details on estimating $A(i,k)$ and other practical issues are described in [7].

A.3 Internet Experiments

We have implemented an actual multiple sender path diversity system that employs the rate allocation and packet partition algorithms. We now demonstrate the effectiveness of the optimal rate allocation scheme in reducing the number of lost packets by performing three following experiments. In experiment one, a sender at Purdue university streams a H.263 encoded video to a receiver at U.C. Berkeley at the rate of 200 packets per second. In experiment two, the same video is also streamed at 200 packets per second from a sender at Sweden to a receiver at U.C. Berkeley. In experiment three, both senders at Sweden and Purdue universities simultaneously stream the video to a receiver at U.C. Berkeley with the pre-computed optimal rates of 96 and 104 packets per second, respectively. In all three experiments, the streamed H.263 video is 720$kbps$ and is packetized into 500 bytes packets protected by $RS(100,90)$ code.

Figure 3 plots the number of lost packets per 100 for experiments one, two, and three denoted with squares, circles, and crosses, respectively. The points above horizontal line represent irrecoverable loss events. Since we are using $RS(100,90)$, irrecoverable loss happens when there are more than 10 lost packets per 100 sent packets. As seen, there are 5 instances of irrecoverable loss for experiments one and two where only one sender is used to stream video to receiver. On the other hand, in experiment three where both senders at Sweden and Purdue university stream video simultaneously to the receiver at U.C. Berkeley, all the lost packets are successfully recovered by FEC.

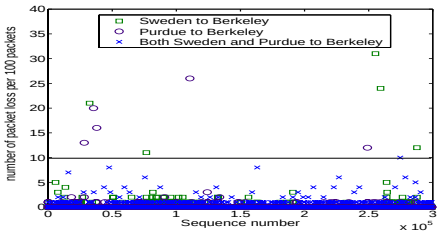


Fig. 3. Actual Internet experiments showing the benefits of distributed video streaming over conventional approach.

B. Path Diversity for Single Sender Case

Since multiple sender architecture cannot be used in interactive and live streaming applications, in this section, we describe the path diversity (PD) system in which multiple paths are created via relay nodes [14]. The system consists a set of participating nodes. At any instance, a node can act simultaneously as a receiver, a sender, or a relay node. A sender can send video packets to the receiver using the default Internet path or via a relay node which then forwards the video packets to the receiver. By choosing an appropriate relay node, the packets traverse an underlying physical path that is different from the one used by default Internet path.

B.1 Redundant Path Selection

Our approach to redundant path selection is to use a simple, but suboptimal technique for selecting redundant path. In particular, we argue that finding two paths with absolute lowest loss rates for the proposed PD system may not be needed in practice due to two reasons: (a) an active monitoring of probing packets and maintaining the link state information associated with all the paths [14] increase complexity and hence is not scalable; (b) sending packets on two paths with the absolute lowest loss rates may not be necessary to achieve reasonable performance in a PD system with appropriate FEC protection level [9]. Using router's names and round-trip link delay between two nodes provided by *Traceroute*[15], the path selection algorithm first finds a set of redundant paths that are as disjoint as possible from the default path. Within this set of redundant paths, it then selects the one that results in minimum latency.

B.2 Simulation Results

For all the simulations, we use the Internet topology generator software Brite [16] to generate various flat and hierarchal *Albert-Barabasi* topologies with the numbers of nodes and edges shown in Table I. *Albert-Barabasi* model has been shown to approximate the Internet topology reasonably well [17][16].

Models	No. Nodes	No. Edges
Flat Albert-Barabasi	1500	2967
H-Albert-Barabasi I	1500	2997
H-Albert-Barabasi II	1500	4377

TABLE I
Information for various topologies

We observe that with 10% participating nodes, the probabilities that the redundant and default paths sharing two or fewer shared links for *Flat Albert-Barabasi*, *Albert-Barabasi II*, and *Albert-Barabasi I* are roughly 100%, 90%, and 85%, respectively. This indicates that a redundant path with few shared links can be found with high probability, and hence PD system can be deployed effectively.

We now use NS[18] to characterize the packet loss reduction as a function of number of shared links between redundant and default paths. In particular, the number of links for default and redundant paths are set to 11 and 18 respectively. Each link speed is 2Mbs with propagation delay of 4 *milliseconds*. To simulate bursty packet loss, random exponential traffic is generated at each link with the peak rate of 1.8Mbs, average idle period of 8 seconds, and the burst period of 40 *milliseconds*. The streaming rate is 800 kbps on a single route experiment and 400 kbps for each route in two-route experiments. All *500-byte* packets are protected using RS(30,27).

Figure 4 shows the ratio of the effective packet loss rate of uni-path scheme to that of dual path scheme as a function of number of shared links between them. The effective packet loss rate is the ratio between the number of irrecoverable lost packets to the total sent packets. As seen, the effective loss rate for the single path scheme is more than 7 times that of the path diversity scheme with completely disjoint redundant and default paths. The performance of using path diversity decreases as the number of shared links between the two paths increases.

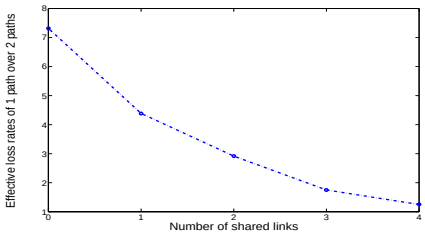


Fig. 4. The ratio of average loss rates using default path to that of using both redundant and default paths with various number of shared links between them.

IV. Conclusion

In this paper we have presented two techniques for efficient streaming over best-effort packet switched networks, such as the Internet, for situations when either the access link, or the path to the video source, impede the quality of video streaming.

References

[1] "MovieFlix. http://www.movieflix.com," .

[2] W. Tan and A. Zakhor, "Real-time internet video using error resilient scalable compression and tcp-friendly transport protocol," *IEEE Transactions on Multimedia*, vol. 1, pp. 172–186, june 1999.

[3] G. De Los Reyes, A. Reibman, S. Chang, and J. Chuang, "Error-resilient transcoding for video over wireless channels," *IEEE Transactions on Multimedia*, vol. 18, pp. 1063–1074, june 2000.

[4] H. Ma and M. Zarki, "Broadcast/multicast mpeg-2 video over wireless channels using header redundancy fec strategies," in *Proceedings of The International Society for Optical Engineering*, November 1998, vol. 3528, pp. 69–80.

[5] Puneet Mehra, Christophe De Vleeschouwer and Avideh Zakhor, "Receiver-Driven Bandwidth Sharing for TCP," in *IEEE INFOCOM*, 2003.

[6] Puneet Mehra and Avideh Zakhor, "Efficient Video Streaming using TCP," in *Under Submission*.

[7] T. Nguyen and A. Zakhor, "Distributed video streaming," in *Proceedings of the SPIE - The International Society for Optical Engineering, Multimedia Computing and Networking*, San Jose, CA, January 2002, vol. 4673, pp. 186–95.

[8] T. Nguyen and A. Zakhor, "Distributed video streaming with forward error correction," in *Packet Video Workshop*, Pittsburg, PA, April 2002.

[9] T. Nguyen and A. Zakhor, "Path diversity with forward error corection (pdf) system for packet switched networks," in *INFOCOM*, San Francisco, CA, April 2003.

[10] "NIST Net network emulator. http://snad.ncsl.nist.gov/itg/nistnet/," .

[11] "Real-Time UDP Data Emitter (RUDE) http://cvs.atm.tut.fi/rude/," .

[12] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast application," in *Architectures and Protocols for Computer Communication*, October 2000, pp. 43–56.

[13] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *INFOCOM*, 1999, vol. 1, pp. 345–52.

[14] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R. Morris, "The case for resilient overlay networks," in *Proceeding of HotOS VIII*, May 2001.

[15] *Traceroute*, http://www.tracert.com.

[16] *Internet topology generator*, http://www.cs.bu.edu/brite.

[17] A.L. Barabasi and R. Albert, "Emergence of scaling in random networks," *Science*, pp. 509–512, October 1999.

[18] Information Sciences Institute, http://www.isi.edu/nsnam/ns, *Network simulator*.