Improving Wireless Access of Video Over the Internet

Jesse Steinberg
Dept. of Computer Science and Engineering
University of California, San Diego
jsteinbe@cs.ucsd.edu

Joseph Pasquale
Dept. of Computer Science and Engineering
University of California, San Diego
pasquale@cs.ucsd.edu

Abstract

Network flow buffering is the use of a simple remote flow-regulating buffer that is dynamically deployed between a Web client and server to improve the performance of HTTP-based access and playback of video. We show that HTTP enhanced with network flow buffering performs well, especially under high packet loss and highly variable bandwidth conditions, when compared with using either straight HTTP or streaming. The benefits of network flow buffering are numerous, and include reducing interruptions to playback, maintaining high video-image quality, and decreasing client buffering requirements. Network flow buffering can be easily implemented using existing Web mechanisms.

1. Introduction

Video continues to be an increasingly larger fraction of Internet traffic [21] [5]. For a powerful desktop client accessing video from a high bandwidth server over a broadband Internet connection without congestion, smooth video streaming is already a reality today. However, when lower-performance elements are involved, such as accessing video using resource-limited clients over wireless links, the user will often experience extended startup delays and frequent interruptions in playback.

While this problem can be addressed using specialized video streaming protocols, HTTP-based video streaming is becoming increasingly popular for a number of reasons. One is the use of HTTP proxies from behind firewalls that allow access to the outside world. Another is that it is easier for Internet users to place video clips on Web sites provided by their ISPs as they may not have direct access to a video server, or they may lack the expertise to install and administer their own server. In the former case, the video can be streamed directly via HTTP, or a streaming protocol can be tunneled through HTTP. However, in the latter case, the server dictates that HTTP be used directly.

In the paper, we describe the performance benefits of *network flow buffering* (NFB), a simple client-oriented approach to improving video streaming over HTTP to make it a more viable alternative to specialized streaming protocols. With NFB, an application-layer flow-regulating buffer is dynamically deployed between a Web client and server. These buffers act on a per-session basis. For example, multiple buffers can be simultaneously deployed to different locations to handle video sessions originating at different servers. Furthermore, these buffers do not need to store video data beyond the duration of a session, and in many usage scenarios will store only a small portion of the video data at any time. Dynamic deployment does require a suitable host for NFB between the client and server; however, the assumption that such a location exists is not unreasonable, considering that the location need not belong to a third party. For example, the user's own PC can be used, as in the common scenarios of the user with a thin wireless client that is close to the user's home PC when at home, or office PC when at work. Finally, we use the Web Stream Customizer Architecture (WSCA) [15, 16] as a platform for practical support of NFB on the Web, as it supports dynamic deployment and relocation of software intermediaries between Web clients and servers.

The remainder of the paper is organized as follows. In Section 2, we introduce two common usage scenarios that are the focus of our investigation. In Section 3, we analyze the performance of NFB-enhanced HTTP by comparing it to direct HTTP and streaming given these two usage scenarios. We review related work in Section 4, and present conclusions in Section 5.

2. Usage Scenarios

The optimal location for network flow buffering depends on network conditions. Ideally, a network flow buffer is located just beyond a "problem hop," generally a boundary point where there is a significant change in network performance or reliability characteristics. This location will often be at the LAN/WAN gateway, such as when the client accesses the Internet via a wireless connection. Or, it may be more practically located beyond this point; perhaps the user's home or office PC as mentioned earlier may provide such a convenient location.

During a video session, the network bandwidth seen by the user may change as the result of changing network or server conditions. Reasons for this include changes in packet loss rate in, say, a wireless connection as the user roams, router congestion, or the server becoming overloaded and being forced to reduce the transmission rate. In this paper, smoothing refers to using NFB to mask (from the client) dynamic changes in *relative* bandwidth between the WAN and LAN that can cause the effective bandwidth seen by the user to be lower than the video playback rate. Thus, smoothing occurs when the WAN and LAN bandwidths are highly variable relative to each other.

Fig. 1 shows a scenario of LAN and WAN bandwidths during a video session. Each line shows the potential amount of data transferred over time, and hence the *slope*

represents the bandwidth. Although the LAN and WAN have the same average bandwidth, the WAN bandwidth is bursty while that of the LAN is constant. As shown, the effective bandwidth (i.e., minimum slope) seen by the client has a lower average.

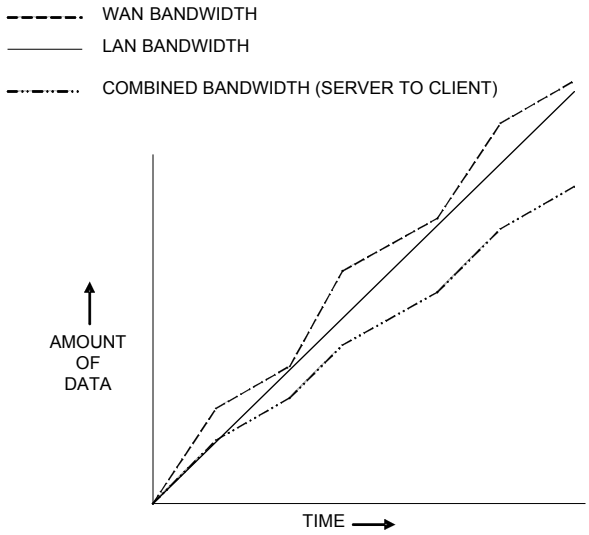

Figure 1. Smoothing scenario bandwidths.

In Fig. 2 and 3, video playback is depicted with and without NFB. A specialized video streaming protocol is expected to perform poorly under these conditions of high relative variability for two reasons. The effective bandwidth is considerably below the video playback rate, so the video application will be forced to either delay the video start time to buffer a large amount at the client, reduce video quality, or interrupt playback in order to buffer. Furthermore, the unpredictable nature of the bandwidth will make it more difficult for the video application to make correct protocol and buffering decisions on the fly.

When NFB is used, a smoothing effect allows the video to play smoothly. When the WAN bandwidth is higher than the LAN bandwidth, the buffer will fill up. At a later time, when the LAN bandwidth is higher than the WAN bandwidth, data will be available at the buffer in order to maintain higher throughput. In the scenario depicted in Fig. 3, the client will see the bandwidth as higher than the video playback rate, and will therefore be able to maintain uninterrupted, full-quality playback. Even if the smoothed playback rate is less than the video playback rate, the frequency and duration of interruptions will still be reduced.

Finally, NFB can reduce the path over which retransmitted packets travel, lessening the performance penalty incurred by packet loss. For a wireless client, if the wireless access network is more prone to packet loss than the rest of the wired WAN, retransmissions will typically only have to travel between the client and an intermediary supporting NFB (assuming it is beyond the wireless portion) and not suffer the delay of the entire network.

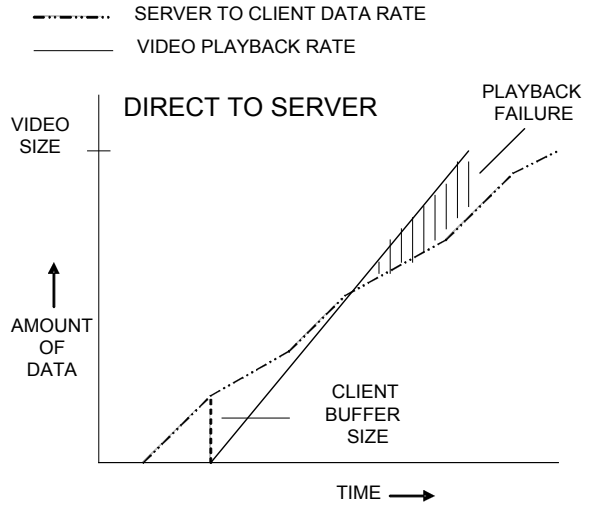

Figure 2. Failure when end-to-end data rate drops below video playback rate.

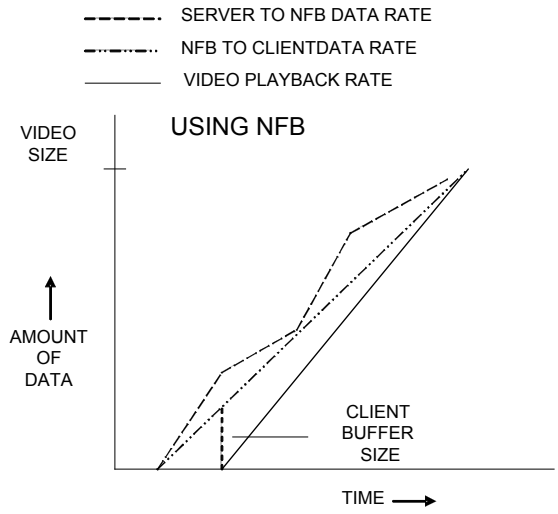

Figure 3. Benefits of smoothing when using NFB.

3. Performance

We now compare NFB-enhanced HTTP to direct HTTP and to "Real streaming" under the two usage scenarios described in Section 2. By "Real streaming," we mean streaming as implemented by Real™ when using their commonly available RealOne™ Player accessing video from their servers, and essentially treating it as a black box, measuring end-to-end performance as observed by the output at the player.

3.1 Experimental Setup

The experimental setup client, video server, and intermediate "gateway" machine used for NFB. For the client, we used a notebook computer with a 500MHz Intel Pentium III processor running the RealOne™ Player on Windows 98. The server was a P3 933MHz PC running Windows 2000. The intermediate machine was a Pentium II 450MHz PC running FreeBSD. To simulate a network with a given bandwidth, we used DummyNet in FreeBSD

(IP Firewall kernel module), which supports the creation of pipes to control bandwidth, delay, and packet loss between two communication endpoints. Four pipes are used in total, one pair for symmetric bidirectional LAN control and another for symmetric bidirectional WAN control.

3.2 Bandwidth Variation and Smoothing

In this experiment, we show how NFB can provide smooth playback that would otherwise be interrupted multiple times. In addition to performance improvement, using the NFB requires no more application-layer buffering than is required when streaming directly from server to client. In fact, in circumstances where smoothing works well, it is normally expected to use less. The video clip used for this experiment was 202 seconds (3.4 minutes) in length, with a bit rate of 38.5 KBps (308 Kbps).

We used DummyNet to create cyclical client/gateway "LAN" and gateway/server "WAN" bandwidths as shown in Fig. 4. The averages for the WAN and LAN for each cycle are 41.75 KBps, which is above the average video playback rate of 38.5 KBps. Without smoothing, the effective bandwidth seen by the client is the minimum at any point in time of the bandwidth cycles, averaging 25 KBps, and we expect the video player to have to interrupt playback so that the network can catch up to the video.

Fig. 5 shows the video playback as amount of data played over time for direct HTTP, Real streaming, and NFB-enhanced HTTP. With NFB-enhanced HTTP, the player buffers for 17 seconds before playback begins. Once playback begins, the video plays smoothly and at full quality for its entirety without any additional buffering by the player. This is due to the smoothing effect of NFB, which allows a higher average bandwidth to be sustained to the client. For direct HTTP (i.e., without NFB), the player buffers for 28 seconds before playback begins. During playback, the video plays at full quality, but the player interrupts the video three times to refill its buffer, for a total of 124 seconds of buffering after playback has started, and 152 seconds of total buffering, as compared to just 17 seconds with NFB-enhanced HTTP.

When Real streaming is used, there is an initial 5 seconds of buffering before playback begins. The player quickly recognizes that bandwidth is inadequate and attempts to adapt accordingly. In doing so, it reduces the frame rate and picture quality, and ends up only retrieving 1.97MB (1966954 bytes), or about 25%, of the video. This results in a choppy, "slide show" resulting in very poor-quality playback. Despite this adaptation, the video playback is still interrupted 7 times for a total of 165 seconds of additional buffering.

The extra buffering that occurs when there is no smoothing is required because the player's buffer is suffering from underflow, as the average bandwidth is lower than the video bit rate. Note that even if the player had perfect knowledge of the future, it would have to delay the start of the video by 152 seconds to ensure smooth playback under direct HTTP. Not only would this frustrate the user, it would also require a buffer size of nearly 6MB (152 x 38.5 KBps). Given that the player cannot anticipate network traffic bursts, or that it may not be desirable to delay the start of the video for so long and force the user to wait, or to reduce memory consumption, the player is forced to interrupt the video for two periods of half a minute or more to complete playback.

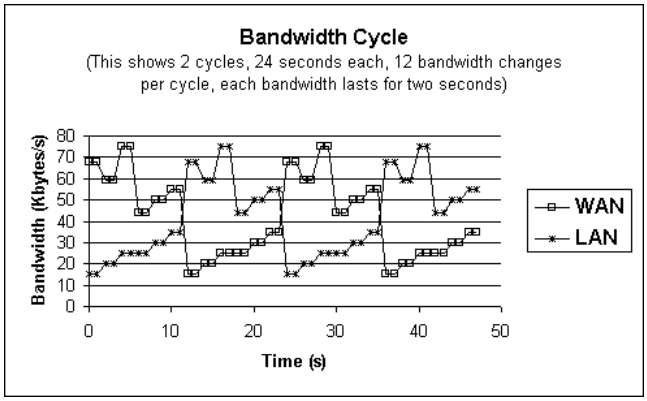

Figure 4. Bandwidth for smoothing experiment.

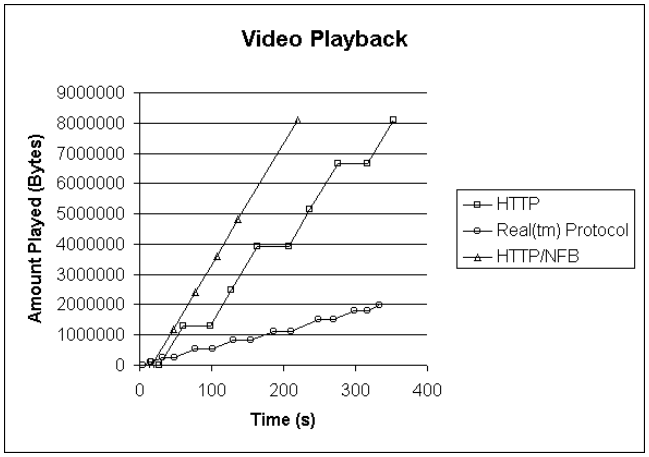

Figure 5. Playback in each of the three scenarios.

In Fig. 6, we show the amount of buffering at the player for direct HTTP, Real streaming, and NFB-enhanced HTTP (in the latter case, the combined amount buffered at both the player and intermediary for NFB is presented).

The large spikes in buffering for direct HTTP result from the player trying to manage its buffer when the bandwidth is bursty. The maximum buffer size for the player in this case is 1.1 MB (1103346 bytes). The buffering when using Real streaming is also bursty. However, the drastic reduction in amount of video data played, at the cost of playback quality, results in a maximum client buffer size of 258 KB (257805 bytes). Finally, the smoothing effect of NFB-enhanced HTTP can be seen by the significantly reduced burstiness of its (combined) buffering.

The combined buffering when NFB-enhanced HTTP is being used is further broken down into its two components, the client buffering and the intermediary (for NFB) buffering, in Fig. 7. When the intermediary buffer is peaking because the WAN bandwidth is higher than that of

the LAN, the client buffer is draining because the LAN bandwidth is lower than the video rate. When the LAN bandwidth increases, the client buffer will start to increase, but the intermediary buffer begins to drain since the WAN is now the bottleneck. The maximum *combined* buffering (client + intermediary) peaks at 785 KB (785057 bytes), less than the maximum buffer requirement at the client when direct HTTP is used. The maximum buffering *at the client* when the NFB is used is 660 KB (660514 bytes). This is due to the fact that without NFB, the average effective bandwidth over the course of the playback is lower, so more buffering is required to compensate.

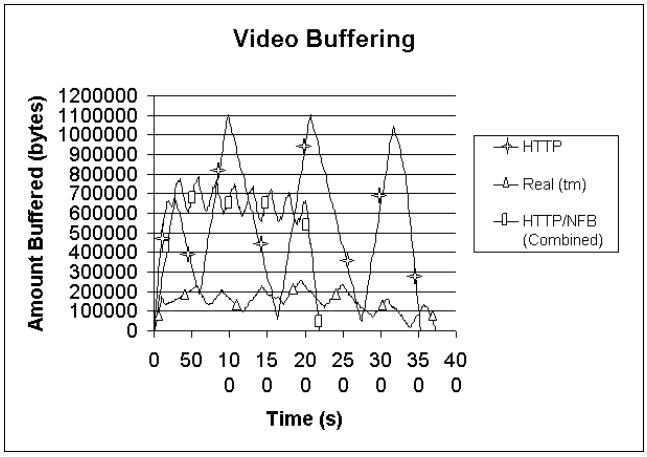

Figure 6. Buffering during the smoothing experiment.

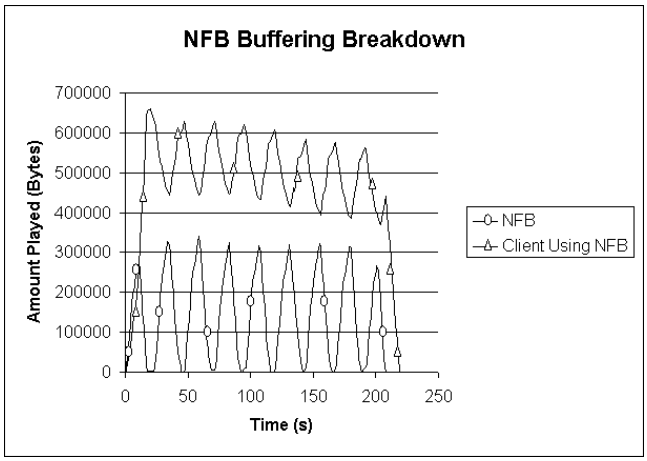

Figure 7. Breakdown of buffering at the client and at the network flow buffer.

3.3 Packet Loss

In this experiment, we show how NFB reduces retransmission delay under packet loss for HTTP streaming, which can result in video performance comparable to that of Real streaming. Here, we configured DummyNet to randomly drop packets in the LAN portion, to compare how direct HTTP, Real streaming, and NFB-enhanced HTTP tolerate packet loss.

The random loss of packets is expected to reduce the effective bandwidth. However, under TCP, the limited latency between client and NFB-intermediary allows for a higher tolerance of packet loss than under the end-to-end latency of the entire network, since it will take less time to retransmit lost packets. (In our experiments, the LAN delay was 10ms, WAN delay was 50ms, for a total end-to-end delay of 60ms.) When Real streaming is used, the full latency is still a factor, but the protocol may selectively choose to allow packets to drop rather than retransmitting. Thus, Real streaming, if it adapts well, is expected to avoid video interruptions at the cost of reduced playback quality.

We ran three trials, each with DummyNet configured with a different packet loss rate that remained constant throughout the trial. We chose loss rates of 2%, 4.5%, and 5% because each is a threshold where we observed that one of the scenarios typically starts to encounter interruptions to playback. The 2% rate was chosen because it was the threshold where direct HTTP started to degrade in performance and was no longer competitive with the other approaches (interruptions in playback occur at this loss rate). The 4.5% loss rate is the corresponding threshold for Real player; interruptions in playback start occurring at this point. Finally, at 5% loss rate we see interruptions in playback even with the NFB approach.

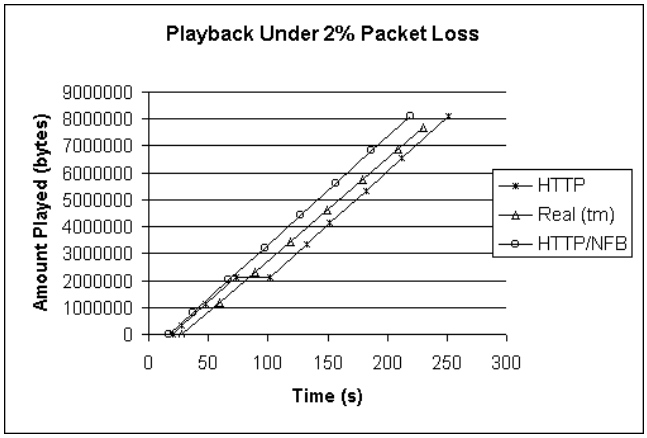

Figure 8. Playback under 2% packet loss when using Real streaming vs. network flow buffering.

We expect direct HTTP to be least tolerant of packet loss. Fig. 8 shows playback under 2% packet loss for each scenario. With direct HTTP, there is a 20 second interval of buffering before playback starts, and there is an additional 29 second interruption in playback. Neither of the other two scenarios incurs an interruption during playback. With NFB-enhanced HTTP, there is an initial 14-second buffering period before playback begins, after which the video plays at full quality without interruption. When Real streaming is used, the video takes the longest to start, as there is an initial 28-second buffering period. The video then plays at nearly full quality without interruption, with about 95% of the video data being played. As the packet loss rate increases and the effective bandwidth over the network (with a 60ms delay) decreases, we expect direct HTTP playback to continue to degrade in the form of increased frequency and duration of interruptions of video playback for re-buffering.

We now compare how NFB-enhanced HTTP and Real streaming perform at higher packet loss rates. Fig. 9 shows the playback with a 4.5% packet loss rate. With Real streaming, there is an initial buffering period of 30 seconds before playback begins, and an additional interruption of 12 seconds to buffer more data, that occurs about 14 seconds into the playback. Up until this point, the playback is very choppy, with only between 1 and 4 frames displayed per second and many artifacts on the screen. However, after this interruption, the playback smoothens out, although the frame rate does occasionally drop to about 2/3 its normal rate. With NFB-enhanced HTTP, there is an initial buffering period of 21 seconds before full quality, uninterrupted, playback begins. Both protocols allow smooth playback when the packet loss rate is more than double what direct HTTP can tolerate. The amount of time that playback is delayed or suspended for buffering using NFB-enhanced HTTP is about half that of when Real streaming is used.

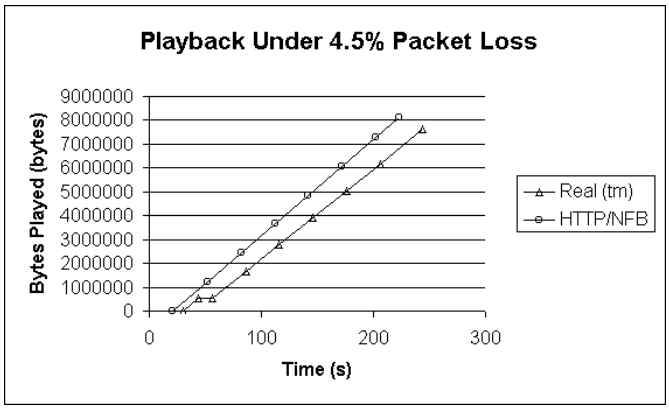

Figure 9. Playback under 4.5% packet loss when Real streaming vs. NFB-enhanced HTTP.

Finally, we examine what happens with a packet loss rate of 5%, where we begin to see interruptions during playback when NFB-enhanced HTTP is used, and erratic behavior from Real streaming. This case is shown in Fig. 10. When the packet loss rate is 5%, Real streaming exhibited mixed behavior. About half of the trials, it would adapt well to the packet loss, resulting in smooth playback at slightly reduced quality. But the other half of the trials, it would not adapt well, resulting in frequent interruptions in playback, where each interruption might last for 1 to 2 minutes! The poor performance example of Real streaming, which is cut off to fit into the chart, continues the same trend of frequent, long interruptions for the entire video, and playback can last 20 minutes or more. When Real streaming adapts well, it buffers for about 12 seconds before playback begins, and has one buffering interruption of 18 seconds about 6 seconds into the playback.

When NFB-enhanced HTTP is used, there is an initial buffering period of 26 seconds before playback begins. About 57 seconds into the playback, there is an interruption that lasts for about 23 seconds. Thus, the general behavior of NFB-enhanced HTTP is similar to Real streaming when the latter is operating well. As the packet loss rate rises above 5%, we expect an increase in the frequency and duration of interruptions to playback, and that Real streaming will be less able to adapt and will more frequently exhibit the behavior of the poor playback case.

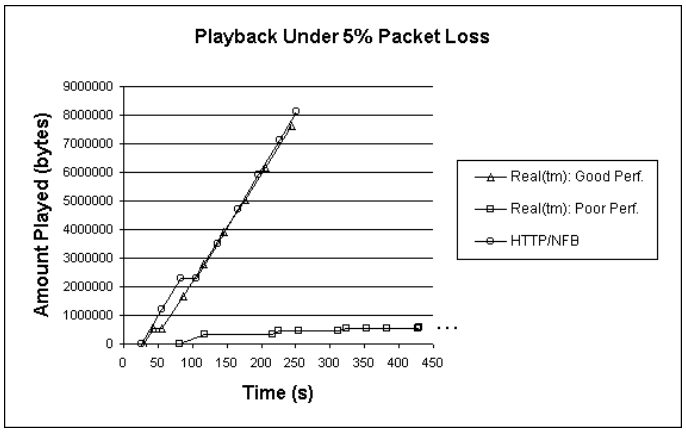

Figure 10. Playback under 5% packet loss when using Real streaming vs. NFB-enhanced HTTP.

4. Related Work

There is a large body of research on smoothing compressed video streams by sending frames to the client buffer ahead of their playback times to deal with bandwidth burstiness. In this approach, a video transmission schedule is calculated at the server based on the bit rate over the course of the video, the server bandwidth, and the amount client memory available for buffering. Two notable contributions [6, 12] take into account video, network, and host characteristics, which are known *a priori*, and a bandwidth schedule is calculated. In [12], the smoothing takes into account the possibility of having multiple intermediary buffers (such as proxies) along the path between server and client, and considers bandwidth limitations of network hops among the intermediary buffers, the client, and the server. In [6], the constraint of minimizing peak server bandwidth is relaxed to keep the client and server in closer synchronization, which better supports interactive functionality such as fast-forward and rewind. Optimal smoothing can be approximated for live video by calculating the transmission schedule over a window of time based on the currently available video data in conjunction with predictive techniques [14]. We are solving the different problem of dynamic changes in network performance and availability as a result of such factors as wireless network variability and congestion.

Another area of relevant video research involves using intermediaries, either fixed or dynamically deployable, to improve video streaming using techniques such as transcoding and protocol adaptation for unicast or multicast [2, 3, 8, 9, 10, 11]. These approaches differ from ours in that NFB does not do any transcoding of the video data. This type of functionality is more CPU intensive than buffering. CPU intensive approaches such as transcoding can certainly be used in conjunction with NFB.

Another approach to improving video performance is to use proxy caching, where proxies cache all or part of commonly accessed videos to improve performance for a group of clients. A number of approaches perform partial caching of video streams to improve performance, while reducing resource consumption at the proxy [13, 20, 22]. Other approaches use cooperative caches to improve hit rates as well as reduce server load and WAN bandwidth utilization [1, 19]. There are a number of important differences between NFB and proxy caching. In proxy caching, videos are cached at a shared server accessed by many clients, so resources must be shared and the proxy can become a bottleneck. NFB is a client-specific approach, so there is no competition for resources among multiple users (unless they just happen to use the same intermediary for NFB). Furthermore, the location of a shared proxy cache may not be ideal for a particular user. NFB can operate under dynamic deployment, and the location can be chosen to best serve one particular client. Finally, NFB does not require that any video data be stored beyond the duration of the streaming session. Since a single user will typically only be streaming one video at a time, the resource requirements for NFB are small enough that it does not place undue burden on its host (e.g., a user's PC).

5. Conclusions

Network flow buffering makes HTTP-based video competitive with specialized streaming protocols, under the conditions of high LAN packet loss rates or in networks with highly variable relative LAN/WAN bandwidths. We showed that the smoothing benefits of NFB-enhanced HTTP support full-quality playback, while requiring less client buffering than direct HTTP, in cases where Real streaming will reduce buffering requirements at the cost of drastically reduced playback quality.

The ability to dynamically deploy a simple flow-regulating buffer has the added benefit of reducing retransmission delay in order to increase playback performance under high LAN packet loss rates. NFB-enhanced HTTP allows the video player to tolerate more than double the packet loss rate that direct HTTP can tolerate. The lower the delay is between the NFB-intermediary and the client, the more tolerant the streaming will be to packet loss. Furthermore, NFB-enhanced HTTP outperforms Real streaming at medium packet loss rates, and is competitive with Real streaming at higher loss rates where Real streaming begins to exhibit erratic behavior.

6. References

[1] S. Acharya and B. Smith, "MiddleMan: A Video Caching Proxy Server," *Proc. 10th Intl.. Workshop on Network and Op. Sys. Support for Digital Audio and Video (NOSSDAV)*, June 2000.

[2] E. Amir, S. McCanne, and R. Katz, "An Active Service framework and its application to real-time multimedia transcoding," *Proc. ACM SIGCOMM*, pp. 178-189, Aug 1998.

[3] F. Baschieri, P. Bellavista, and A. Corradi, "Mobile Agents for QoS Tailoring, Control and Adaptation over the Internet: the ubiQoS Video on Demand Service," *Proc. Symp. Applications and Internet (SAINT)*, Jan 2002.

[4] S. Chandra, "Wireless Network Interface Energy Consumption Implications of Popular Streaming Formats," *Proc. Multimedia Computing and Networking (MMCN02),* Jan 2002.

[5] M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and Analysis of a Streaming-Media Workload," *Proc. 3rd USENIX Symp. Internet Tech. Sys. (USITS)*, Mar. 2001.

[6] W. Feng, "Rate-Constrained Bandwidth Smoothing for the Delivery of Stored Video," *Proc. IS&T/SPIE Multi-media Networking and Computing*, pp. 316-327, Feb 1997.

[7] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," *Proc. 1st ACM SIGCOMM Workshop on Internet Meas.*, pp. 169-182, Nov 2001.

[8] Z.Lei and N.D.Georganas, "Rate Adaptation Transcoding For Video Streaming Over Wireless Channels"," *Proc. IEEE ICME*, June 2003.

[9] J. Meggers, T. Strang, and A.S. Park, "A Video Gateway to Support Video Streaming to Mobile Clients," *Proc. ACTS Mobile Communication Summit*, Oct 1997.

[10] R. Mohan, J. Smith, and C.-S. Li, " Adapting Multimedia Internet Content For Universal Access," *IEEE Trans. Multimedia*, pp. 104-114, March 1999.

[11] W. T. Ooi, and R. van Renesse, "Distributing Media Transformation Over Multiple Media Gateways," *Proc. ACM Multimedia*, Oct. 2001.

[12] J. Rexford, and D. Towsley, "Smoothing variable-bit-rate video in an internetwork," *IEEE/ACM Trans. Networking*, pp. 202-215, April 1999.

[13] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," *Proc. IEEE INFOCOM*, March 1999.

[14] S. Sen, J. Rexford, J. Dey, J. Kurose, and D. Towsley, "Online smoothing of variable-bit-rate streaming video," *IEEE Trans. Multimedia*, pp. 37-48, March 2000.

[15] J. Steinberg and J. Pasquale, "A Web Middleware Architecture for Dynamic Customization of Content for Wireless Clients," *Proc 11th Intl. WWW Conf.*. May 2002.

[16] J. Steinberg, "The Web Stream Customizer Architecture: Improving Performance, Reliability and Security for Wireless Web Access," *Ph.D. Dissertation*, *UC San Diego, Dept. Computer Science and Engineering*, 2004.

[17] J. Steinberg and J. Pasquale, "Using Network Flow Buffering to Improve the Performance of Video over HTTP," UCSD Technical Report CS2004-0776, January 14, 2004.

[18] M. Stemm, Gauthier, D. Harada, R.H Katz, "Reducing power consumption of network interfaces in hand-held devices," *Proc. 3rd Workshop on Mobile Multimedia Comm. (MoMuC-3),* 1996.

[19] D. A. Tran, K. A. Hua, and S. Sheu, "A New Caching Architecture for Efficient Video Services on the Internet," *Proc. IEEE Symp. Applications and the Internet (SAINT)*, Jan 2003.

[20] Y. Wang, Z. Zhang, D. H.C. Du, and D. Su, "A Network Conscious Approach to End-to-End Video Delivery over Wide Area Networks Using Proxy Servers," *Proc. IEEE INFOCOM*, April 1998.

[21] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy., "Organization-Based Analysis of Web-Object Sharing and Caching," *Proc. 2nd USENIX Conf. on Internet Tech. and Systems* (USITS), Oct. 1999.

[22] K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," *Proc. 10th Intl. WWW Conference*, May 2001.