Playback-Adaptive Multi-Source Video Streaming

S. C. Hui and Jack Y. B. Lee
{schui3, yblee}@ie.cuhk.edu.hk
Department of Information Engineering
The Chinese University of Hong Kong
Hong Kong

***Abstract* - The best-effort nature of the Internet poses significant challenges to guaranteeing performance of video streaming applications. In particular, the available bandwidth between a sender and a receiver is often unpredictable and can vary substantially from time to time, causing buffer underflows and consequently playback starvations. This work tackles this problem by proposing a novel video playback adaptation and rebuffering algorithm for multi-source streaming. Unlike single-source streaming, the aggregate bandwidth of multiple senders closely follows the normal distribution when the number of senders exceeds four. Based on this discovery this paper presents a video playback rate adaptation and rebuffering algorithm that can significantly reduce playback starvations. This adaptive multi-source streaming scheme is simple to implement, does not require complex media adaptation processing at the servers, and can be applied to the vast amount of video content already available in the Internet.**

***Keywords* - multi-source streaming, Internet measurement, aggregate bandwidth estimation, video adaptation, rebuffering control**

I. INTRODUCTION

Today's Internet only provides best-effort data delivery and so does not guarantee bandwidth availability. While the best-effort model works well for data applications such as the WWW and email, it presents significant challenges to bandwidth-sensitive applications such as video streaming.

Specifically, to successfully stream a video over the Internet we need to ensure that the video bit-rate does not exceed the network bandwidth available, or else congestion will occur, leading to dropped packets and playback hiccups. Unfortunately the available network bandwidth between a sender and a receiver is not known a priori and worst, often varies from time to time.

One approach to tackle this problem is to adapt the video bit-rate according to the network bandwidth available. As it is not possible to know a priori the future bandwidth, one needs to apply prediction techniques [1-3] to estimate the short-term future bandwidth availability based on past bandwidth measurements. Armed with knowledge of the bandwidth availability, the sender can then adapt the video content to the desired bit-rate using techniques such as scalable video codec [4,5] or real-time transcoders [6,7].

In this work we investigate a new adaptive multi-source video streaming (AMSS) scheme that does not require support from the streaming server at all. AMSS is developed based on two principles. First, unlike streaming video from a single sender, we discovered that the aggregate data rate from multiple senders to a receiver is in fact normally-distributed, even for as few as 4 senders [8]. Therefore by measuring the mean and variance of the aggregate streaming bandwidth, we can obtain a probabilistic estimate of the streaming bandwidth available. Second, instead of using scalable video codec, which is not widely supported in current streaming applications, we propose the use of playback rate adaptation to cope with the random fluctuations in the streaming bandwidth available.

Combining these two principles, we develop an adaptation algorithm to dynamically vary the playback rate locally at the client according to the estimated bandwidth availability and the client buffer occupancy. This adaptation algorithm is run locally at the client and thus eliminates the added processing complexity at the server. To our knowledge this is the first study to combine playback rate adaptation with multi-source streaming. Our trace-driven simulations show that good streaming performance can be achieved with as few as 4 senders and a playback rate variation of only 5%, despite the fact that TCP is used as the underlying transport running over the best-effort Internet.

The rest of the paper is organized as follows: Section II discuss the related work. Section III the multi-source video adaptation algorithm; Section IV evaluates the performance of these schemes using trace-driven simulations; Section V summarizes the paper.

II. BACKGROUND AND RELATED WORK

Streaming from multiple senders has been investigated by a number of researchers. For example, Xu *et al.* [9] proposed a data assignment algorithm to allocate and schedule the senders' data transmissions to the receiver to reduce the receiver's buffering delay. *Jin et al.* [10] further generalized the data assignment algorithm for arbitrary bandwidth ratios among the senders. Both studies, however, employed static data assignment, i.e., the data each sender has to transmit are fixed. In another study [1], Nguyen and Zakhor proposed a dynamic rate allocation and packet partition scheme for multi-sender streaming. The video data each sender will transmit is not fixed, but adapted according to the senders' actual throughput. On the other hand, if there are many senders the receiver may need to select a subset of them rather than streaming from all senders in order to reduce complexity.

This research is funded in part by a Direct Grant, an Earmarked Grant (CUHK4211/03E) from the HKSAR Research Grant Council, and the UGC Area of Excellence in Information Technology Scheme (AoE/E-01/99).

The above studies are focused on the selection of sources, assignment of data, and the scheduling of transmissions of the multiple senders in streaming. They did not employ any video adaptation algorithm to shape the video bit-rate to fit the network bandwidth available, and thus is limited in their capability to compensate for network bandwidth variations.

Current work on video adaptation primarily focused on one of three approaches: (a) encode multiple versions of the same video in different bit-rates and send the version that best match the network bandwidth available (e.g., SureStream in RealVideo [11]); (b) encode the video using scalable techniques such as multiple description codec (e.g., [4]), layered video codec (e.g., FGS [5]), and then adjust the video bit-rate by adding or dropping video layers; and (c) use a real-time video transcoder (e.g., [6,7]) to dynamically shape the video bit-rate.

All these approaches however, are primarily designed for single-source streaming as the adaptation processing has to be done at the server. Conceivably it is possible to extend these approaches to multi-source streaming but then the servers will need to coordinate their adaptation processing which is far from trivial. Therefore further investigation is needed to uncover the potential problems and devise the solutions.

In comparison, the AMSS scheme presented in this work does not employ server-side video adaptation. Instead, AMSS employs client-side playback rate adaptation to vary the video data consumption rate.

For video stream this can be achieved simply by changing the display rate (i.e., inter-frame interval) of video frames. Changing the playback rate of audio is more challenging as increasing/decreasing the playback sampling rate will also change the pitch of the audio, which is audible. To address this problem, we can apply a technique called Time Scale Modification (TSM) [12] that can shorten or elongate the audio stream while preserving the pitch. These techniques are well known and have been applied successfully in many applications, including voice over IP, adaptive piggybacking [13], etc.

Clearly, there is still a limit on which we can change the display rate without causing noticeable degradation. However, our experiments show that even with a very small playback rate change of 5%, which is not noticeable [13], we can already achieve significant performance improvement in terms of frequency and length of playback interruptions.

III. Client-Side Video Adaptation

Armed with the discovery that the aggregate bandwidth of multiple senders is approximately normally distributed [8], we develop in this section a video playback rate adaptation scheme to compensate for the bandwidth estimation inaccuracies so that smooth, continuous video playback can be maintained. The adaptation algorithm is composed of two parts, namely the playback rate adjustment algorithm and the rebuffering algorithm.

A. Playback Rate Adjustment Algorithm

Assume there are N senders transmitting a video encoded in a constant bit-rate R. Let T be the averaging time window for computing the average bandwidth availability, i.e., the bandwidth availability is taken at intervals of T seconds. Furthermore, let $a_{i,j}$ be the amount of data received from sender i at time interval j. Then, the total amount of data received from all N senders at time interval j, denoted by A_j, is then given by

$$A_j = \sum_{i=0}^{N-1} a_{i,j} \tag{1}$$

Let C_j be the amount of data consumed at interval j. With a playback rate of R, we can compute C_j from

$$C_j = TR \tag{2}$$

The client buffer occupancy at interval j, denoted by B_j, can be calculated from the difference between the amount of data received and consumed, i.e.,

$$B_j = \max(0, (A_j - C_j)) \tag{3}$$

According to the argument in the previous section, the playback rate can be adjusted within a small range, say α, without noticeable by the user. Thus a video segment (say segment j) of original playback duration T seconds can in fact be played back in a range of durations:

$$T(1-\alpha) \le T_j \le T(1+\alpha) \tag{4}$$

Intuitively, the receiver should increase the playback rate (i.e., shorten the playback duration) when the buffer is about to overflow, and decrease the playback rate (i.e., extend the playback duration) when the buffer is about to underflow. In practice, the buffer constraint is typically far less of a problem than bandwidth constraint and so for simplicity we ignore buffer overflow and the constraint in (4) is simplified to

$$T_j \le T(1+\alpha) \tag{5}$$

Now as T_j is no longer a constant we will need to modify (1) and (2) to

$$r_j = \sum_{i=0}^{N-1} \frac{a_{i,j}}{T_j} \tag{6}$$

and

$$m_j = \frac{TR}{T_j} \tag{7}$$

where r_j and m_j represent respectively the data reception rate and data consumption rate at interval j.

Using this model the playback rate adjustment problem is then equivalent to determining T_j given the current estimated aggregate bandwidth availability as well as the client buffer occupancy.

Specifically, let B_j be the *actual* buffer occupancy at interval j. Then the *estimated* buffer occupancy at the next interval j+1, denoted by B'_{j+1} will be equal to

$$B'_{j+1} = r_j T_j - RT + B_j \quad (8)$$

where the first term is the amount of data received, and the second term is the amount of data consumed at interval j. The goal is to maintain the buffer occupancy to a level, say X, larger than zero, i.e.,

$$B'_{j+1} \geq X > 0 \quad (9)$$

Revisiting (8) we already know the exact values for R, T, and B_j. The aggregate bandwidth r_j is normally distributed and the receiver has been measuring its mean and variance since the beginning of the streaming session. Thus the only unknown is the playback duration T_j, which we can adjust in order to satisfy the constraint in (9).

Assume that the client can tolerate a probability of Δ of failing the constraint in (9). Then we can rewrite the constraint in (9) as

$$\Pr\left\{B'_{j+1} < X\right\} \leq \Delta \quad (10)$$

Substituting (8) into (10) we have

$$\Pr\left\{r_j T_j - RT + B_j < X\right\} \leq \Delta \quad (11)$$

Rearranging gives

$$\Pr\left\{r_j < \frac{X - B_j + RT}{T_j}\right\} \leq \Delta \quad (12)$$

which the L.H.S. probability is given by the normal distribution and hence we can compute T_j accordingly.

In practice, most streaming video player software performs prefetch buffering before beginning playback to absorb network delay variations. Assuming the amount of prefetch video data is equal to B_{pre}, then we can simply set X=B_{pre} to maintain the client buffer occupancy at the prefetch level.

B. Rebuffering Algorithm

Despite the use of multiple senders and the playback rate adaptation algorithm described in the previous section, the client may still occasionally experience buffer underflow. When underflow occurs it is necessary to temporary pause the video playback until some amount of video data are accumulated.

The simplest rebuffering algorithm is to rebuffer up to the prefetch buffer level, i.e., B_{pre}. However, this method may not be optimal. On one hand, the prefetch buffer level could be unnecessary large. While a longer delay is acceptable at startup, it is far less tolerable when the video is suddenly suspended due to buffer underflow. On the other hand, if bandwidth availability is low, it would be better to prefetch more video data to reduce the occurrences of buffer underflows.

Instead of using a fixed rebuffer size, we can compute the amount of video data to rebuffer using methods similar to (8) and (9). Specifically, when buffer underflow occurs at say time interval j, then B_j=0. Let P be the rebuffer size. Then we can calculate P from

$$P = \min\left\{p \mid \Pr\left\{r_j < \frac{X - p + RT}{T_j}\right\} \leq \Delta\right\} \quad (13)$$

The playback will resume after the client buffer occupancy reaches P. In this adaptive rebuffering algorithm the variability of the available bandwidth is then incorporated into the calculation of the rebuffer size P.

IV. Performance Evaluation

In this section we use trace-driven simulations to evaluate the AMSS scheme and compare the performance of different design choices. The performance metric used is the total underflow time – defined as the total time at which playback is suspended due to buffer underflow, and the number of playback pauses (i.e., number of buffer underflow occurrences) during the streaming session. We set N = 5, T = 1s, B_{pre} = 5s, Δ = 0.15%, and α is in the range from 0.005 to 0.05. The video bit rate, R is set to equal to the average aggregate bandwidth of traffic traces obtained from PlanetLab [14]. The simulation result is obtained from the average of five simulation runs.

A. Comparison of Different Algorithms

Three different algorithms are compared in the following results: (a) "No Scheme" – no playback rate adaptation nor rebuffering; (b) "Adaptation Only" – using playback rate adaptation but not rebuffering; and (c) "Adaptation and Rebuffering" – using both playback rate adaptation and rebuffering.

Fig. 1 and 2 shows the average total underflow time and pause count with respect to the playback rate adjustment limit α. First, without playback rate adaptation and rebuffering (i.e., "No Scheme" in the figures) the system performed poorly with long underflow time (over 14 seconds) and large number of playback pauses (over 70 occurrences). Second, by introducing playback rate adaptation the performance is improved significantly.

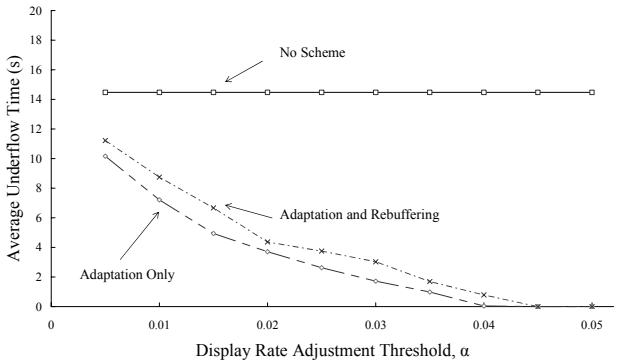

Fig. 1. Average underflow time versus playback rate adjustment limit.

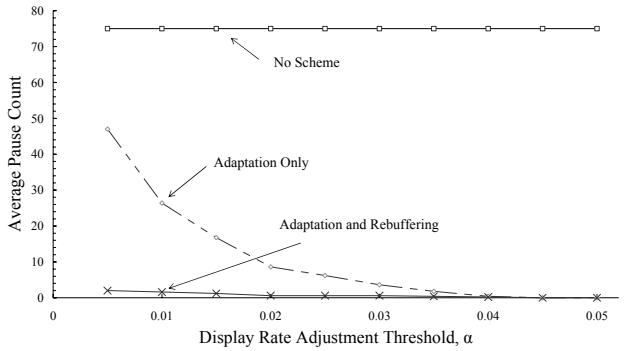

Fig. 2. Average pause count versus playback rate adjustment limit.

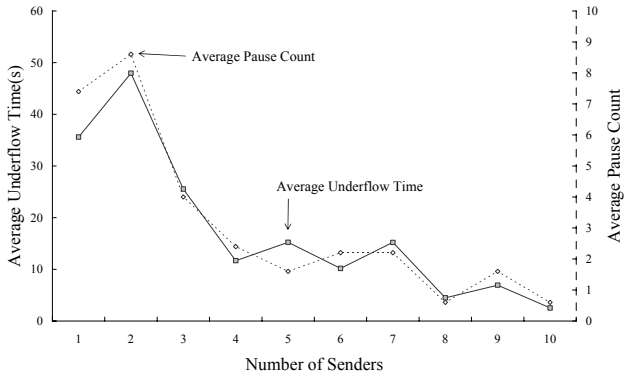

Fig. 3. Average underflow time and pause count versus number of senders.

Further improvement is obtained when rebuffering is also employed. For rate adjustment threshold larger than 0.02 (i.e., up to 2% playback rate variation) both the underflow time and playback pause count decrease to negligible levels.

B. Effect of Number of Senders

In this experiment we investigate the effect of number of senders on the system performance. Fig. 3 plots the average underflow time and pause counts for number of senders ranging from 1 to 10 where $\alpha = 0.05$.

There are two observations in these results. First, the system performs poorly when there are fewer than 4 senders. This result matches our measurements in [8] as the aggregate bandwidth does not conform to a normal distribution when the number of senders is fewer than 4. This leads to estimation errors in the adaptation algorithms and thus degrades the system performance substantially. Second, we observe that the system performance continue to improve for more senders. This suggests that the proposed AMSS scheme is particularly suitable for applications with many sources (e.g., in peer-to-peer applications).

V. SUMMARY

This work investigates the integration of playback-rate adaptation with multi-source streaming to achieve high streaming performance in the best-effort Internet. These two techniques are complementary as either one alone is not sufficient to compensate for the inherent bandwidth variations in the Internet. Only when used together the client can effectively adapt to the changing bandwidth availability, and without the need to implement complex media adaptation algorithms (e.g., scalable video coding, transcoding, etc.) at the servers. Therefore the proposed AMSS algorithm can be applied to the vast amount of existing video contents already in the internet, which are often encoded in non-scalable formats.

REFERENCES

[1] T. Nguyen and A. Zakhor, "Distributed Video Streaming over the Internet" *SPIE Conference on Multimedia Computing and Networking*, San Jose, California, January 2002.

[2] S. Vazhkudai, J. M. Schopf, and I. Foster, "Predicting the Performance of Wide Area Data Transfers," *16th International Parallel and Distributed Processing Symposium*, Fort Lauderdale, FL, April 2002.

[3] R. Wolski, "Dynamically Forecasting Network Performance Using the Network Weather Service," *Journal of Cluster Computing*, vol. 1, pp.119-132, January 1998.

[4] Reibman, H. Jafarkhani, Y. Wang, M. Orchard, and R. Puri, "Multiple Description Coding for Video Using Motion Compensated Prediction," *International Conference on Image Processing*, Kobe, Japan, vol. 3, pp.837-41, October 1999.

[5] Y. Q. Liang and Y. P. Tan, "Methods and Needs for Transcoding MPEG-4 Fine Granularity Scalability Video," *IEEE International Symposium on Circuits and Systems 2002*, Scottsdale, Arizona, vol.4, pp.719-722, May 2002.

[6] A. Vetro, C. Christopoulos and Huifang Sun, "Video Transcoding Architectures and Techniques: an Overview," *IEEE Signal Processing Magazine*, vol. 20, Issue 2, pp.18 – 29, March 2003.

[7] L. S. Lam, Jack Y. B. Lee, S. C. Liew, and W. Wang, "A Transparent Rate Adaptation Algorithm for Streaming Video over the Internet," *18th International Conference on Advanced Information Networking and Applications*, Fukuoka, Japan, March 2004.

[8] S. C. Hui, Jack Y. B. Lee, "Modeling of Aggregate Available Bandwidth in Many-to-One Data Transfer" *Fourth International Conference on Intelligent Multimedia Computing and Networking*, Salt Lake City, Utah, USA, July 2005.

[9] D.Y. Xu, M. Hefeeda, S. Hambrusch and B. Bhargava, "On Peer-to-Peer Media Streaming," *International Conference on Distributed Computing Systems 2002,* Vienna, Austria, pp.363-371, July 2002.

[10] Jin B. Kwon and Heon Y. Yeom, "Distributed Multimedia Streaming over Peer-to-Peer Network" *Euro-Par 2003, 9th International Conference on Parallel and Distributed Computing*, Klagenfurt, Austria, August 2003.

[11] RealVideo 10 Home Page: http://www.realnetworks.com/products/codecs/realvideo.html

[12] Y. J. Liang, N. Farber and B. Girod, "Adaptive Playout Scheduling Using Time-Scale Modification in Packet Voice Communications," *IEEE International Conference on Acoustics, Speech, and Signal Processing 2001*, Salt Lake City, Utah, vol. 3, pp.1445-1448, May 2001.

[13] L. Golubchik, John C. S. Lui and R. R. Muntz, "Reducing I/O demands in Video-on-Demand Storage Servers," *ACM SIGMETRICS and PERFORMANCE'95, International Conference on Measurement and Modeling of Computer Systems,* Ottawa, Canada, May, 1995.

[14] PlantLab Home Page: http://www.planet-lab.org/