Multimed Tools Appl (2007) 34:139–154
DOI 10.1007/s11042-006-0088-1

On aggregate available bandwidth in many-to-one data transfer—modeling and applications

S. C. Hui · Jack Y. B. Lee

Published online: 7 February 2007

Abstract This work investigates the modeling of aggregate available bandwidth in multi-sender network applications. Unlike the well-established client–server model, where there is only one server sending the requested data, the available bandwidth of multiple senders when combined together does exhibit consistent properties and thus can be modeled and estimated. Through extensive experiments conducted in the Internet this work proposed to model the aggregate available bandwidth using a normal distribution and then illustrates its application through a hybrid download-streaming algorithm and a playback-adaptive streaming algorithm for video delivery under different bandwidth availability scenarios. This new multi-source bandwidth model opens a new way to provide probabilistic performance guarantee in best-effort networks such as the Internet, and is particularly suitable for the emerging peer-to-peer applications, where having multiple sources is the norm rather than the exception.

Keywords Multi-sender transmission · Bandwidth modeling · Internet measurement · Multi-source streaming

1 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.

An early version of this work was presented in the *Fourth International Conference on Intelligent Multimedia Computing and Networking*, Salt Lake City, Utah, USA, July 2005. This work was 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).

S. C. Hui · J. Y. B. Lee (✉)
Department of Information Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong
e-mail: jacklee@computer.org

Specifically, to successfully stream a video we need to ensure that the video bit-rate does not exceed the network bandwidth available, or else the client will run into buffer underflow, leading to 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.

Ideally, if the bandwidth availability can be accurately modeled by a random process, then the sender can simply select a video bit-rate such that performance can be guaranteed probabilistically. However, as we will show in Section 3, modeling the bandwidth availability for a single sender is very difficult, if not impossible, in the current Internet.

Given the limitation of this single-sender approach, researchers have begun to investigate approaches employing multiple senders [1, 11, 17] to exploit three potential benefits: (a) increasing the throughput by combining the bandwidth of multiple senders; (b) adapting to network bandwidth variations by shifting the workload among the multiple senders; and (c) reducing bursty packet loss by splitting the data transmission among the multiple senders.

In this work we go one step further to argue that if there are sufficient number of independent senders, we not only can achieve higher throughput, but also be able to model the aggregate available bandwidth as a normal distribution according to the Central Limit Theorem. We verify this conjecture experimentally by conducting experiments in the global PlanetLab testbed. Our experimental results strongly suggest that this model is applicable in the current Internet and thus, can be used for designing multi-sender streaming protocols that supports probabilistic performance guarantees.

This work has three contributions. First, to the best of our knowledge, this is the first study to investigate modeling of *aggregate* available bandwidth of multiple senders. Second, this is the first study to report experimental results to show that the aggregate available bandwidth is normally distributed, and under what conditions. Finally, we illustrate applications of this new discovery by developing two algorithms to provide probabilistic performance guarantees in streaming video over the current best-effort Internet.

The rest of the paper is organized as follows: Section 2 discusses the background and related work. Section 3 and Section 4 investigate single-source and multi-source bandwidth availability, respectively. Sections 5 and 6 present two algorithms to provide probabilistic performance guarantees in streaming video over the current Internet. Section 7 discusses extension to support the streaming of variable-bit-rate videos. Section 8 concludes the paper.

2 Background and related works

Modeling of network traffic has been studied extensively in the literature. It is generally accepted that Internet traffic cannot be adequately modeled by simple models such as a Poisson process [14]. A number of studies showed that network traffic is in fact self-similar [9, 12, 13, 18], exhibiting long-range dependency with heavy-tailed distribution. There are many other traffic models proposed in the last decade but due to space limitation they will be not reviewed here.

It is worth noting that the abovementioned studies primarily focused on modeling properties of the network traffic itself. By contrast, our work focuses on the modeling of the bandwidth available for data transfer in an end-to-end manner. In particular, our measurements include the effects of network link capacity, competing traffics, limits and variations of the sender (e.g., due to other concurrently running applications), as well as dynamics of the transport protocol (e.g., TCP).

Not surprisingly, with so many factors in the equation the resultant bandwidth availability between a sender and a receiver can vary significantly across different senders and as a result does not conform to any consistent model. However, if we combine the available bandwidth of *multiple* senders, then the *aggregate* available bandwidth will become far more consistent.

Specifically, let X_i $\{i \in 1..N\}$ denotes a set of N independent random variables representing the available bandwidth from sender i to the receiver. Assume each X_i to have an arbitrary probability distribution with finite mean μ_i and finite variance σ_i^2. Then according to the Central Limit Theorem (CLT), the combined available bandwidth has a limiting cumulative function which approaches a normal distribution. Note that for the CLT to be applicable, we need to ascertain that the X_i's are independent, i.e., the senders' available bandwidths are not correlated. We investigate this issue in Section 4 by computing the correlation coefficient [10] of different senders' bandwidths. In the following section we first investigate the properties (or lack thereof) of bandwidth availability between a single sender and a receiver.

3 Single-source bandwidth availability

3.1 Measurement methodology

To obtain realistic results it is necessary to conduct experiments in the Internet rather than in a simulator or a closed test-bed. Therefore we conducted all experiments in the PlanetLab [15] global test-bed which has hundreds of hosts residing in many different countries around the world connected through the Internet. For the actual bandwidth measurement we used the Iperf [5] tool, which can measure the network throughput averaged over a given period of time. A total of 47 different hosts in PlanetLab are employed in the experiments. We manually removed hosts local to the receiver host to prevent overflowing the receiver and skewing the results. We also tested the receiver's throughput to ensure that the local network and the receiver will not become the bottleneck in the measurements.

We installed the Iperf server in the 47 sender hosts, and let the receiver connects to the sender to initiate data transmission. We measured the bandwidth availability by sending data using TCP from the senders to the receiver. The senders all send data as fast as TCP will allow, subject to TCP's flow and congestion control. The receiver captures the average throughput for each source once every 10 s (the default setting in Iperf). The measurement lasted for 3 h, which generated 1,080 measurement samples.

Although Iperf also supports the use of UDP in measurement, we choose TCP for two reasons. First, sending UDP datagrams at very high data-rate will likely cause serious network congestion and affect other users. Second, even for video streaming it is desirable to keep the video data traffic TCP-friendly to minimize adverse impact to other traffic flows. Thus we employed TCP instead of UDP in the bandwidth measurements and the results should also be applicable to other TCP-friendly protocols (e.g., TFRC [4], etc.).

3.2 Measurement results

We first examine the throughput of individual senders. Figure 1 plots the mean throughput and the coefficient-of-variation (CoV) of the 47 senders. We can observe that the bandwidth availability of the 47 senders varies substantially from a minimum of 0.04 Mbps to a maximum of 4.53 Mbps. Moreover, the senders' temporal bandwidth variations, represented by their CoV, also vary substantially across different senders, ranging from 0.16 to 0.88.

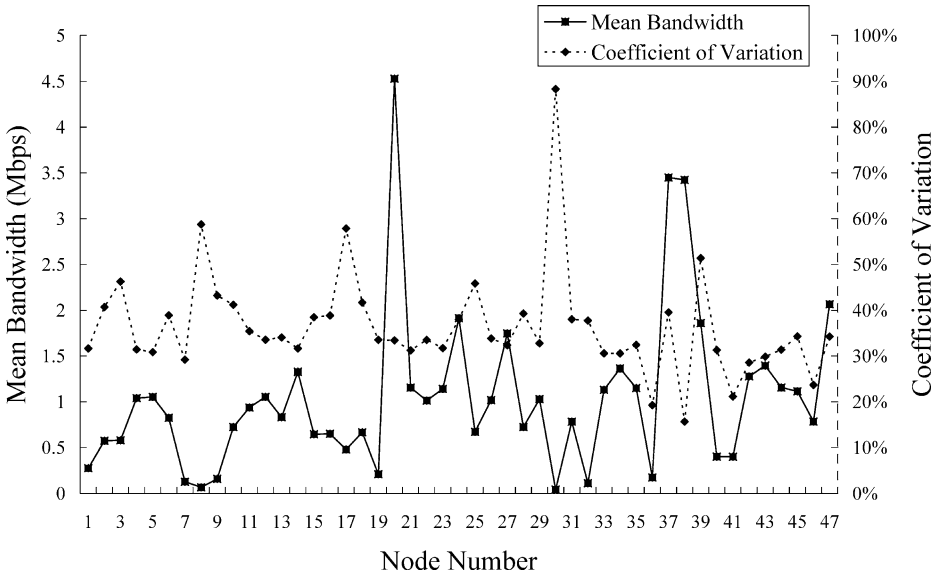

Fig. 1 Average bandwidth and coefficient-of-variations of the 47 senders

Figure 2 plots the bandwidth distribution of 4 out of the 47 senders, over the measurement period of 3 h. We observe that their distributions also vary substantially from one sender to another and do not conform consistently to any known distributions. These results clearly illustrate the difficulty in estimating and modeling the bandwidth availability of individual senders.

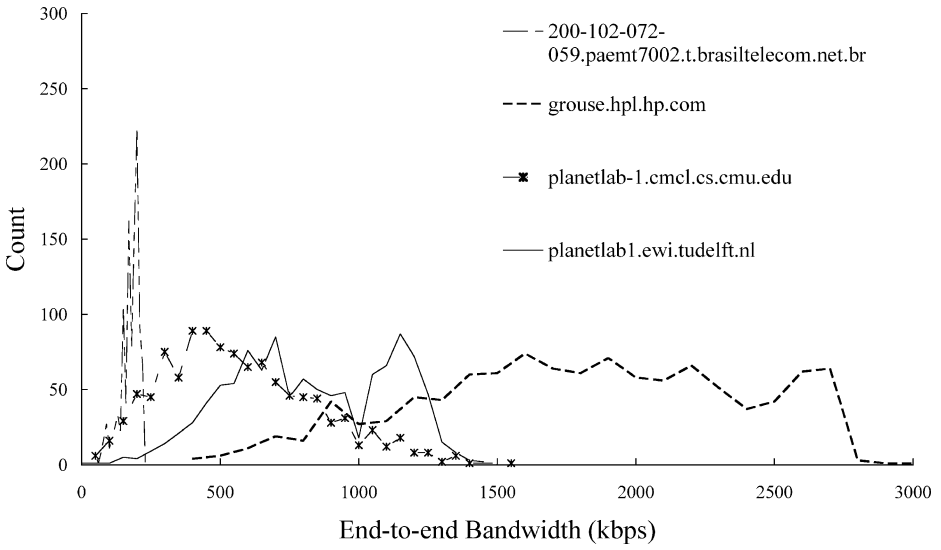

Fig. 2 End-to-end bandwidth distribution, sample from **a** http://www.planetlab-1.cmcl.cs.cmu.edu **b** http://www.planetlab1.ewi.tudelft.nl **c** http://www.grouse.hpl.hp.com **d** http://www.200-102-072-059.paemt7002.t.brasiltel ecom.net.br

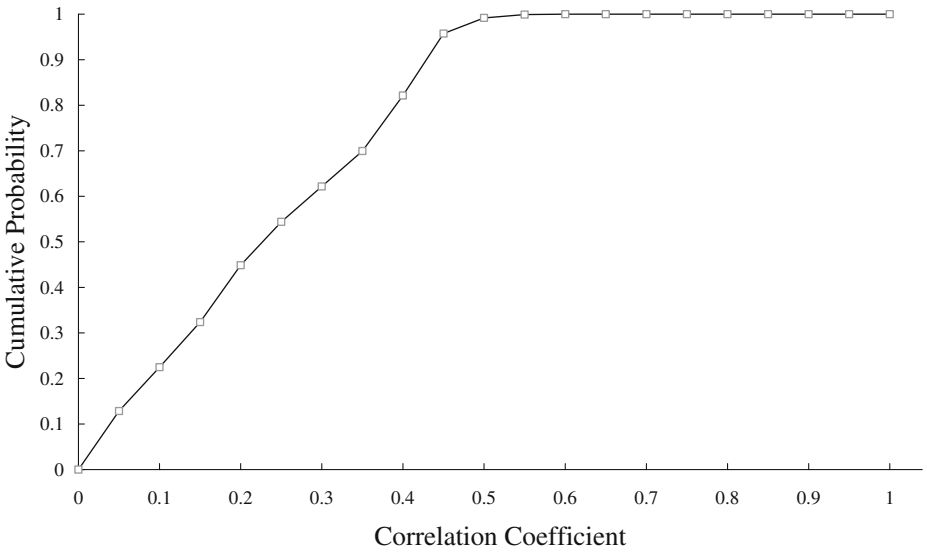

Fig. 3 Correlation of senders' bandwidth availability

4 Multi-source bandwidth availability

While the properties of individual senders are difficult to model and predict, the properties of aggregate bandwidth of multiple senders are far more consistent. We examine in this section the characteristics of aggregate available bandwidth from multiple sources using the methodology in Section 3.

We first investigate the correlations between different senders in the experiment. Figure 3 plots the cumulative distribution for the correlation coefficient [10] of the 47 sending nodes in the PlanetLab. The correlation coefficient measures the degree of

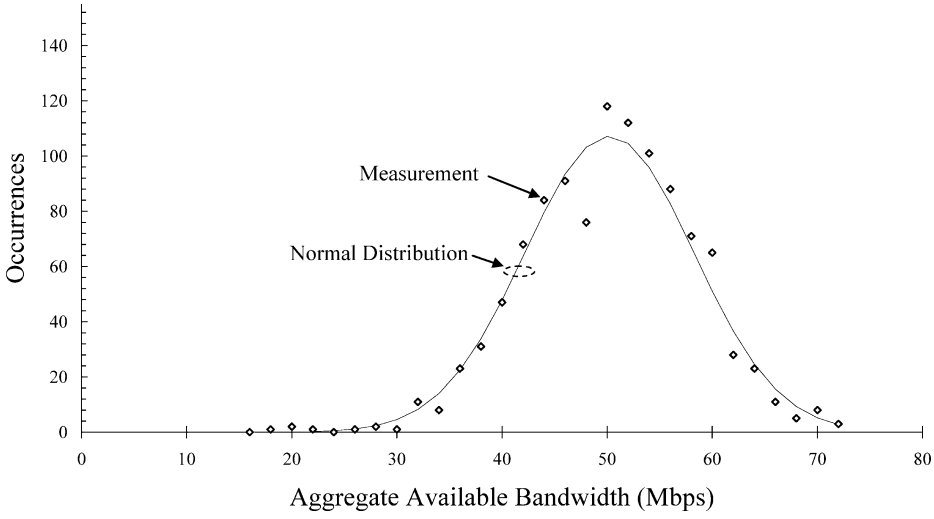

Fig. 4 Distribution of measured aggregate available bandwidth of 47 senders

correlation between the bandwidth availability of two senders. The result shows that half of the sender-pairs have a correlation coefficient less than 0.2 while the correlation coefficients of all sender-pairs are less than 0.6. This suggests that the bandwidth availability of most nodes is relatively uncorrelated. Therefore if we treat the bandwidth availability of each sender as a random variable, we will expect the sum of these random variables and hence the aggregate available bandwidth to approach the normal distribution.

This is confirmed in Fig. 4 which plots the distribution of the aggregate bandwidth of all 47 senders as well as the normal distribution with the same mean and variance as the measurement samples. By inspection we can see that the empirical distribution closely follows the normal distribution.

To further quantify the similarity, we apply the Shapiro–Wilk test [3] which computes from the measurements a p value to quantify the measurements' conformity to the normal distribution. The range of the p value is from 0 to 1, with larger values representing better conformity to the normal distribution. For example, a p value of 0.05 represents a 95% confidence level and is generally considered to be conformance to normal.

To investigate the effect of the number of senders on normal-conformance, we vary the number of senders from 2 to 45 and plot the resultant p value in Fig. 5. Note that each data point is computed from the average of up to 1,000 different combinations of senders.

There are three observations. First, as expected the p value increases with the number of senders (up to 18 senders). Second, we also note that beyond 18 senders the p value actually decreases slightly. This is an artifact of the Shapiro–Wilk test as the test is more sensitive to non-conformity when there are more samples. Third, using the p value threshold of 0.05 as a threshold for normal-conformity [3], the results show that the measured distribution becomes normally distributed even when there are only four senders. This suggests that even with only a few senders, we can still approximate the aggregate available bandwidth using the normal distribution.

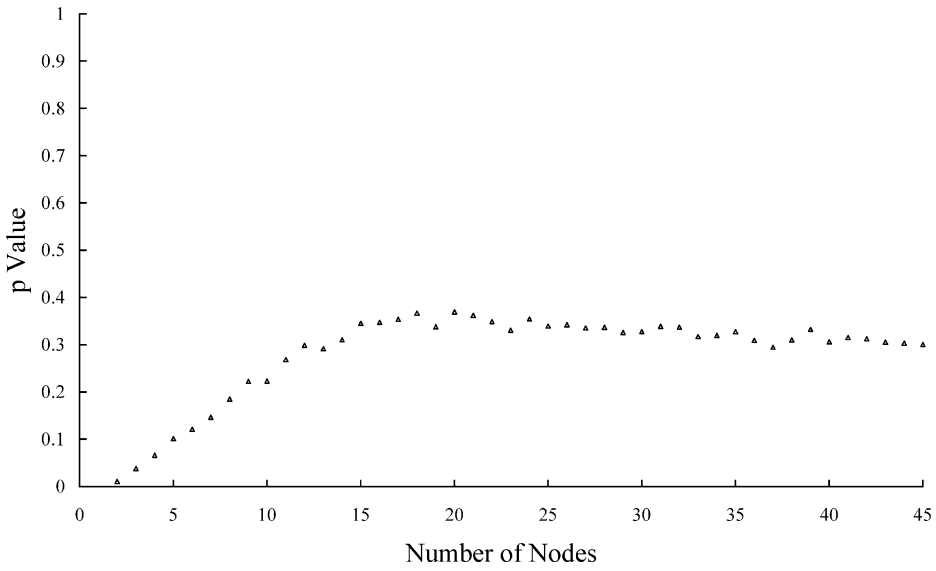

Fig. 5 The effect of the number of senders on normal conformance

5 Hybrid-download streaming

The significance of multi-sender data transfer is that the aggregate available bandwidth can be described by the normal distribution despite the fact that the underlying Internet is a best-effort network. This discovery enables one to build content delivery systems with probabilistic performance guarantees to improve the quality of service to end users.

We consider the streaming of constant-bit-rate (CBR) encoded video from multiple senders to a receiver over the best-effort Internet. Depending on the bandwidth available at the time of streaming, we can classify it into three scenarios. First, if the available bandwidth is substantially higher than the video bit-rate, then conventional streaming algorithm will work just fine without further complications. Second, if the available bandwidth is substantially lower than the video bit-rate, then streaming is simply not possible. Third, for the case where the available bandwidth is comparable to the video bit-rate, then streaming may be possible provided that additional steps are taken to compensate for fluctuations in the available bandwidth.

In this section we propose a hybrid-streaming algorithm to tackle the second scenario and in Section 6 we develop a playback-adaptive streaming algorithm to tackle the third scenario. These two streaming algorithms exploit knowledge of the aggregate available bandwidth through modeling and measurement to guarantee video playback continuity probabilistically.

5.1 Streaming algorithm

We first consider the second scenario where the available bandwidth is expected to be substantially lower than the video bit-rate, e.g., downloading of media content (e.g., MPEG video) from a web server for local playback at the client. In this case conventional streaming is obviously not possible and currently the only option is to completely download the media file before playback to ensure that playback will be continuous, or else risks frequent playback interruptions which can be very annoying.

However, if the media file is to be downloaded from multiple web servers (with the data properly divided across the servers), then the aggregate data transfer rate will exhibit the normal distribution as demonstrated earlier in Section 4. This enables us to estimate the data transfer time and thus start the playback process earlier *before* the download is completed, and still be able to guarantee (probabilistically) continuous playback.

Specifically, let C_i be the aggregate data transfer rate at time interval i after the beginning of the download process and assume playback begins w intervals after download begins. For simplicity, the length of a time interval is chosen to be the same as the measurement interval as explained in Section 3 and the video is encoded in a constant bit rate of R.

To ensure that playback is continuous, we then need to ensure that the total amount of media data received at any time interval i, denoted by A_i, must not be lower than the total amount consumed by playback, denoted by B_i, i.e.,

$$A_i \geq B_i, \forall i \geq 0 \tag{1}$$

where $A_i = \sum_{j=1}^{i} C_j$ and $B_i = \begin{cases} R(i-w), & \text{if } i > w \\ 0, \text{otherwise} \end{cases}$.

Substituting the definition of A_i and B_i into Eq. 1 and rearranging we can obtain

$$\sum_{j=1}^{i} C_j \geq R(i-w), \forall i > w \tag{2}$$

Now as C_j's are normally-distributed random variables, the sum of nC_j's will also be normally distributed, and is described by the n-times autoconvolution of C_j's CDF $F(x)$, denoted by $F^{(n)}(x)$.

Assuming the user can tolerate a probability of Δ of playback interruption, we need to ensure that

$$F^{(n)}(R(n-w)) \leq \Delta \tag{3}$$

Thus the earliest time for playback to begin can be obtained from

$$w = \min\left\{v \middle| F^{(n)}(R(n-v)) \leq \Delta, \forall n \geq v\right\} \tag{4}$$

5.2 Performance evaluation

To evaluate the latency reduction achievable by this hybrid download-streaming algorithm, we conducted trace-driven simulations using available bandwidth traces collected from PlanetLab. Figure 6 compares the playback latencies for three algorithms: (a) *pure download*—begin playback only after video file is completely downloaded; (b) *hybrid download-streaming*; and (c) *lower bound* of the download time.

The lower bound is obtained from a priori knowledge of the traffic traces, i.e., all the C_i's are assumed to be known a priori. Obviously this algorithm is not realizable in practice and is thus included for comparison only.

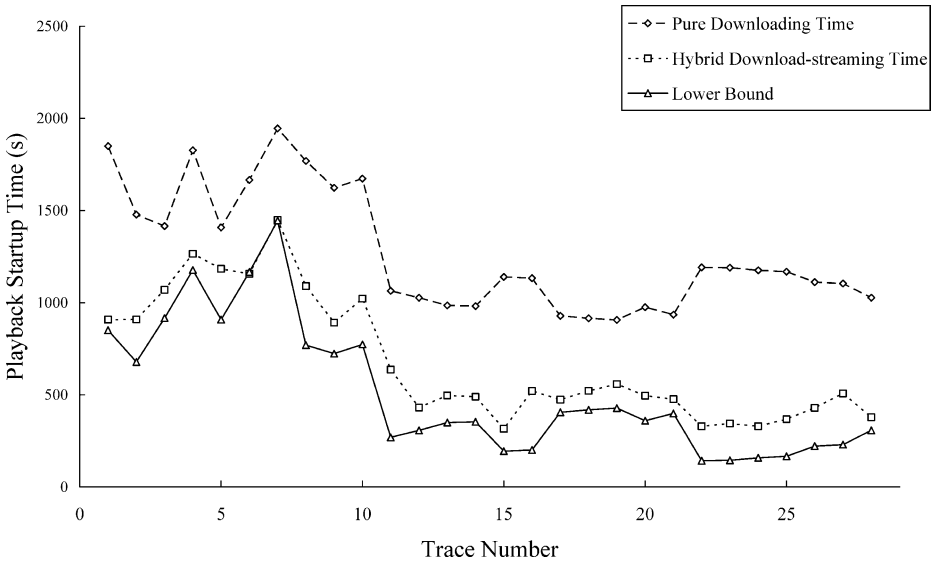

Fig. 6 Comparison of startup time

We run 29 different experiments each with a different traffic trace collected from PlanetLab. The video length and its bit rate range from 500 to 1,800 s and 200 kbps to 1 Mbps, respectively. In each experiment, there are five to ten servers sending disjoint subsets of the video file and the mean aggregate bandwidth available is lower than the video bit-rate (i.e., conventional streaming is not feasible).

As expected, the playback latency for pure download is very long—longer than the video duration, due to the limited bandwidth available. This represents the upper bound for the startup latency. By contrast, the hybrid download-streaming algorithm performs very well, closely tracking the lower bound. Considering the fact that the lower bound algorithm requires a prior knowledge of the available bandwidth and thus is not realizable, the hybrid download-streaming algorithm achieves near-optimal performance through the use of multiple senders together with the bandwidth model in Section 4.

6 Playback-adaptive streaming

In this section we consider the scenario where the available network bandwidth is comparable to the video bit-rate. Now if the available network bandwidth does not vary, then streaming will succeed as long as the available bandwidth is not lower than the video bit-rate. In practice, however, even though the average available bandwidth may be equal to or higher than the video bit-rate, the short term bandwidth fluctuations can still cause frequent playback hiccups.

To tackle this problem, we develop an Adaptive Multi-Source Streaming (AMSS) algorithm that combines the use of aggregate bandwidth model of Section 4 with playback rate adaptation to compensate for fluctuations in the available network bandwidth.

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) [8] 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 [2], etc.

Clearly, there is still a limit on how far we can change the display rate without causing noticeable degradation. Surprisingly, our experiments show that even with a very small playback rate change of 5%, which is not noticeable [2], we can already achieve significant performance improvement in terms of number and duration of playback interruptions.

In comparison to alternative video adaptation techniques such as layered video coding [7, 16] and transcoding [6, 19], the AMSS algorithm does *not* require any support from the server-side to implement video adaptation and so can be readily implemented without revamping the existing media content or media servers.

In the following sections we present the two key components of the adaptation algorithm, namely the playback rate adjustment algorithm in Section 6.1 and the adaptive rebuffering algorithm in Section 6.2. We then evaluate their performance using trace-driven simulations in Section 6.3.

6.1 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 given by

$$A_j = \sum_{i=0}^{N-1} a_{i,j} \tag{5}$$

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

$$C_j = TR \tag{6}$$

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.,

$$B_j = \max\left(0, \left(A_j - C_j\right)\right) \tag{7}$$

With the use of adaptation, 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 now be played back in a range of durations:

$$T(1-\alpha) \le T_j \le T(1+\alpha) \tag{8}$$

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 (i.e., assuming the receiver can buffer the whole video) and the constraint in Eq. 8 is simplified to

$$T_j \le T(1+\alpha) \tag{9}$$

Now as T_j is no longer a constant we will need to modify Eqs. 5 and 6 to

$$r_j = \sum_{i=0}^{N-1} \frac{a_{i,j}}{T_j} \tag{10}$$

and

$$m_j = \frac{TR}{T_j} \tag{11}$$

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

$$B'_{j+1} = r_j T_j - RT + B_j \tag{12}$$

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.,

$$B'_{j+1} \geq X > 0 \tag{13}$$

Revisiting Eq. 12 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 Eq. 13.

Assume that the client can tolerate a probability of Δ of failing the constraint in Eq. 13. Then we can rewrite the constraint in Eq. 13 as

$$\Pr\left\{B'_{j+1} < X\right\} \leq \Delta \tag{14}$$

Substituting Eqs. 12 into 14 we have

$$\Pr\left\{r_j T_j - RT + B_j < X\right\} \leq \Delta \tag{15}$$

Rearranging gives

$$\Pr\left\{r_j < \frac{X - B_j + RT}{T_j}\right\} \leq \Delta \tag{16}$$

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.

6.2 Adaptive 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 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, thus leading to long delay. 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. A single longer playback suspension is far more tolerable than numerous playback suspensions, even if the individual suspensions are shorter.

Therefore, instead of using a fixed rebuffer size, we can again exploit knowledge of the aggregate network bandwidth to compute the amount of video data to rebuffer, using

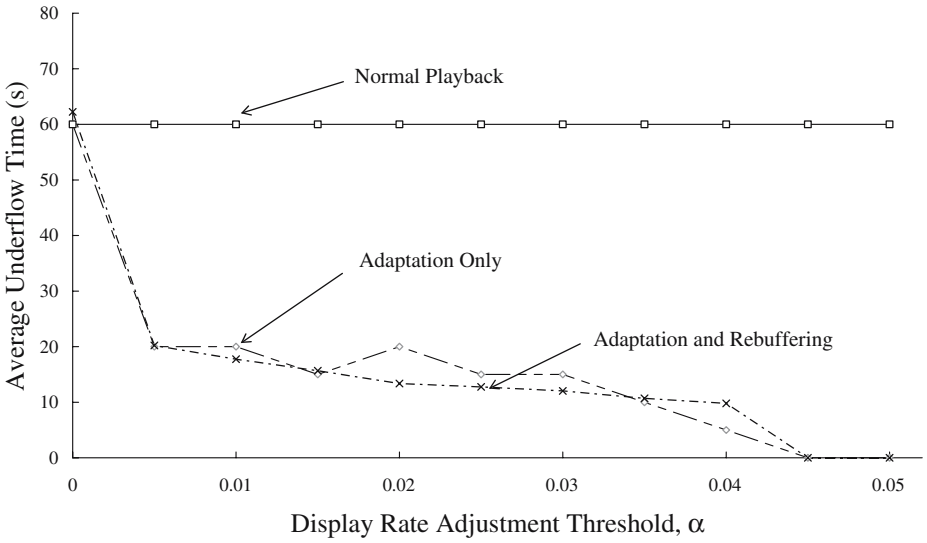

Fig. 7 Average underflow time versus playback rate adjustment limit

methods similar to Eqs. 12 and 13. 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

$$P = \min\left\{p \middle| \Pr\left\{r_j < \frac{X - p + RT}{T_j}\right\} \leq \Delta\right\} \tag{17}$$

Video 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 so as to shorten the rebuffering delay when the bandwidth variation is small, or to reduce the occurrences of rebuffering when the bandwidth variation is large.

6.3 Performance evaluation

In this section we use trace-driven simulations to evaluate the AMSS scheme and analyze the performance impact of playback rate adaptation and adaptive rebuffering. We use 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 as the performance metric. We set N=5, T=1 s, B_{pre}=5 s, Δ=0.15%, and α is in the range from 0.01 to 0.05. The video bit rate, R is set to the average aggregate available bandwidth from traffic traces obtained from PlanetLab [15]. The simulation result is obtained from the average of 35 simulation runs.

Three different algorithms are compared in the following results: (a) "Normal playback"—simple playback without using any adaptation, with a constant rebuffering duration of 5 s; (b) "Adaptation Only"—using playback rate adaptation with a constant rebuffering duration of 5 s; and (c) "Adaptation and Rebuffering"—using both playback rate adaptation and adaptive rebuffering.

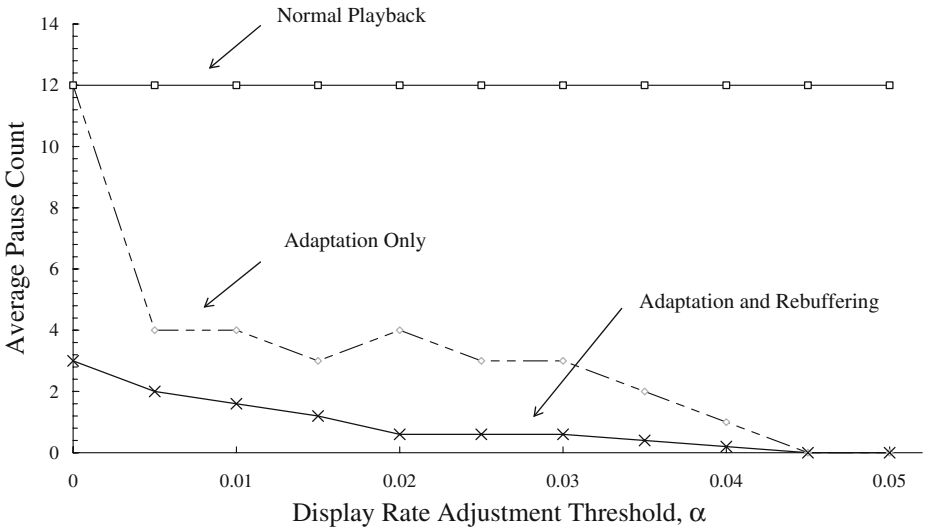

Fig. 8 Average pause count versus playback rate adjustment limit

Figures 7 and 8 show the average total underflow time and pause counts with respect to the playback rate adjustment limit α. First, without playback rate adaptation and the proposed rebuffering scheme (i.e., "Normal Playback" in the figures) the system performed poorly with long underflow time (60 s) and large number of playback pauses (12 occurrences). Second, by introducing playback rate adaptation the performance is improved significantly and the improvement increases with larger display rate adjustment limit.

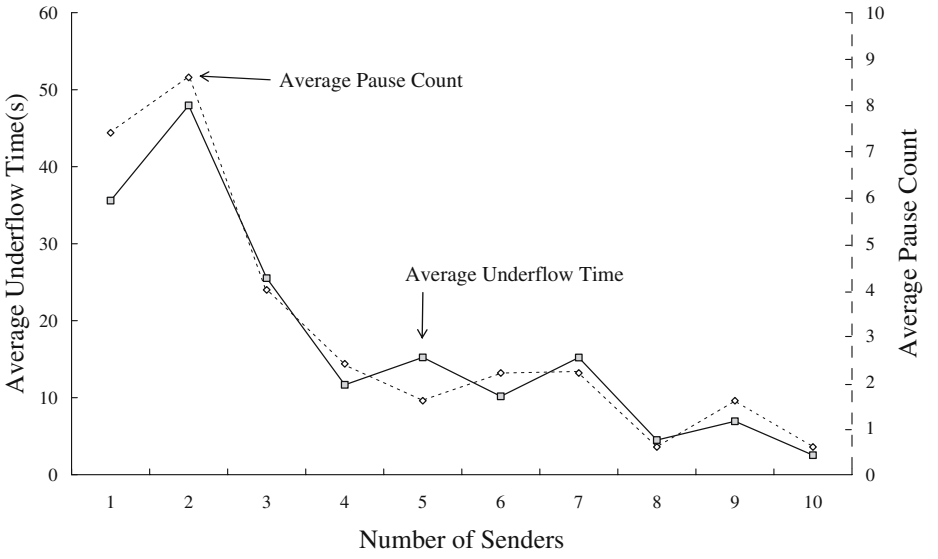

Fig. 9 Average underflow time and pause count versus number of senders

When combined with adaptive rebuffering, the average underflow time increases slightly in some cases (e.g., when α=0.015 and 0.04 in Fig. 7). This is because the adaptive rebuffering algorithm computed longer rebuffering time to compensate for bandwidth variations. This resulted in significantly lower number of average pause count as shown in Fig. 8 when compared to using a fixed rebuffering time of 5 s.

In the next experiment we investigate the effect of different number of senders on the system performance. A separate set of traffic traces was collected from PlanetLab using the same methodology but with number of senders varying from 1 to 10.

Figure 9 plots the average underflow time and pause counts for number of senders ranging from 1 to 10 with α=0.05. There are two observations in these results. First, the system performs poorly when there are fewer than four senders. This result matches our measurements in Section 4 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 continues to improve for more senders. This suggests that the proposed AMSS scheme is particularly suitable for applications with many sources, such as peer-to-peer applications or content distribution networks.

7 Streaming variable-bit-rate videos

While the algorithms in Section 5 and Section 6 are designed around CBR videos with constant playback data-rate, the same principle can be easily extended for streaming variable-bit-rate (VBR) videos. We illustrate this in the following by extending the hybrid-download streaming algorithm in Section 5 to support VBR video streaming.

We assume that the video playback bit-rate is known and is defined by the series $\{R_i | i = 1, 1, ..., L\}$, where R_i is the amount of video data consumed in time interval i, and L is the length of the video in number of intervals. With reference to Eq. 1, we can express the cumulative data consumption variable B_i in terms of R_i as follows:

$$B_i = \begin{cases} \sum_{j=1}^{i-w} R_j, & \text{if } i > w \\ 0, & \text{otherwise} \end{cases}. \tag{18}$$

Substituting this new definition of B_i into Eq. 2 we have the new constraint for continuous playback:

$$\sum_{j=1}^{i} C_j \geq \sum_{j=1}^{i-w} R_j, \quad \forall i > w \tag{19}$$

and the earliest time for playback to begin can be obtained from

$$w = \min \left\{ v \left| F^{(n)} \left(\sum_{j=1}^{n-v} \mathrm{R}_j \right) \geq \Delta, \forall n \geq v \right. \right\} \tag{20}$$

The playback-adaptive streaming algorithm in Section 6 can be extended in a similar way to support streaming of VBR videos.

8 Conclusion

This work is a first step in exploring the feasibility and performance gain achievable through the modeling of aggregate available bandwidth from multiple senders. The experiments conducted in the Internet strongly support the bandwidth model and the applications to hybrid download-streaming and playback-adaptive streaming produced very promising results. In addition to these applications, the proposed multi-source bandwidth model can also be applied to other applications such as file transfer or synchronized multimedia presentations, and to other system architectures such as peer-to-peer applications, where having multiple sources is the norm rather than the exception.

References

1. Agarwal V, Rejaie R (2005) Adaptive multi-source streaming in heterogeneous peer-to-peer networks. SPIE conference on multimedia computing and networking, San Jose, California, January 2005
2. Golubchik L, Lui JCS, Muntz RR (1995) 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
3. Hahn GJ, Shapiro SS (1994) Statistical models in engineering. Wiley Classics Library Edition, John Wiley & Sons, Inc, USA
4. Handley M, Floyd S, Padhye J, Widmer J (2003) TCP Friendly Protocol Specification (TFRC): protocol specification. RFC 3448, January 2003
5. Iperf Homepage: http://www.dast.nlanr.net/Projects/Iperf/
6. Lam LS, Lee Jack YB, Liew SC, Wang W (2004) A transparent rate adaptation algorithm for streaming video over the internet. 18th International conference on advanced information networking and applications, Fukuoka, Japan, March 2004
7. Liang YQ, Tan YP (2002) 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
8. Liang YJ, Farber N, Girod B (2001) 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
9. Morin PR (1995) The impact of self-similarity on network performance analysis. Ph.D. dissertation, Carleton University, Dec. 1995
10. Morris R, Dong Lin (2000) Variance of aggregated web traffic. INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Tel-Aviv, Israel, vol 1, pp 360–366, March 2000
11. Nguyen T, Zakhor A (2002) Distributed video streaming over the internet. SPIE Conference on Multimedia Computing and Networking, San Jose, California, January 2002
12. Park K, Kim G, Crovella M (1996) On the relation between file sizes, transport protocols, and self-similar network traffic. International conference on network protocols, Columbus, Ohio, USA, pp 171–180, Oct. 1996
13. Park K, Kim G, Crovella M (1997) On the effect of traffic self-similarity on network performance. SPIE international conference on performance and control of network system, pp 296–310, November 1997
14. Paxson V (1995) Fast approximation of self-similar network traffic. Tech. Rep., Lawrence Berkeley Laboratory and EECS Division, University of California, Berkeley, April 1995
15. Planetlab Homepage: http://www.planet-lab.org/
16. Reibman AR, Jafarkhani H, Wang Y, Orchard MT, Puri R (1999) Multiple description coding for video using motion compensated prediction. International Conference on Image Processing, Kobe, Japan, vol 3, pp 837–41, October 1999
17. Setton E, Liang Yi, Girod B (2003) Adaptive multiple description video streaming over multiple channels with active probing. International conference on multimedia and expo, Baltimore, Maryland, vol 1, pp I-509-12, July 2003
18. Tuan T, Park K (1998) Congestion control for self-similar network traffic. Department of Computer Science, Purdue University, CSD-TR 98-014, May 1998
19. Vetro A, Christopoulos C, Sun Huifang (2003) Video transcoding architectures and techniques: an overview. IEEE Signal Process Mag 20(2):18–29, March

Shui-cheung Hui received his B.Eng. degree in Electronic and Communication Engineering from the University of Hong Kong in 2003, and M.Phil degree in Information Engineering from the Chinese University of Hong Kong in 2005. He was a member of the Multimedia Communications Laboratory, participated in the research of algorithms and models for multi-source peer-to-peer streaming.

Jack Y. B. Lee received his B.Eng and Ph.D degree in Information Engineering from the Chinese University of Hong Kong in 1993 and 1997, respectively. He participated in the research and development of video streaming systems from 1997 to 1998 where he and his team developed a novel parallel video server architectures for building cost-effective, scalable and fault-tolerant video-on-demand systems. This work had resulted in numerous publications, two US Patents, and the technologies are subsequently transferred to a spin-off technology company for commercialization. From 1998 to 1999, he was a faculty member at the Department of Computer Science, Hong Kong University of Science and Technology. In 1999, he joined the Department of Information Engineering at the Chinese University of Hong Kong to establish the Multimedia Communications Laboratory (http://www.mcl.ie.cuhk.edu.hk) to spearhead research in distributed multimedia systems, peer-to-peer networks, multicast communications, Internet protocols and applications.