On Aggregate Available Bandwidth in Many-to-One Data Transfer

HUI SHUI CHEUNG

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

In

Information Engineering

©The Chinese University of Hong Kong

August 2005

ACKNOWLEDGEMENT

First, I would like to express my deepest gratitude to my supervisor Prof. Jack Y. B. Lee, who whole-heartedly provided guidance and invaluable advice throughout the whole duration of my research.

Next, I would like to thank Prof. S. C. Liew, Prof. O. C. Yue, Prof. Gary Chan and Prof. Angela Zhang who have contributed lots of valuable ideas and comments to my research.

I would also like to thank my dear colleagues, including Dr Poon, Dr Wong, Ki, Tony, Johnny, Alex, Felix, Kin and Rudolf. They provided me with earnest help and friendship during the past two years.

Finally, I am grateful to my family and Mavis for their dedicated love and support.

ABSTRACT

The best effort nature of the current Internet has caused significant difficulties in 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 thesis tackles this challenge by developing a new model for use in multi-sender network applications. We investigate the modeling of aggregate available bandwidth through extensive experiments conducted in the Internet. The result shows that the available bandwidth of multiple senders when combined together does exhibit consistent properties and thus can be modeled and estimated. This discovery opens a new way to providing probabilistic performance guarantees in bandwidth-sensitive applications such as video streaming even in the current best-effort Internet. This thesis also explores the model's applications by developing a new hybrid download-streaming algorithm and a new playback-adaptive streaming algorithm for video delivery under different bandwidth availability scenarios with probabilistic performance guarantee.

摘要

在具盡量傳輸(Best Effort)的性質下，現今的互聯網很難確保視訊串流系統的影像質素。尤其在單一供應者的情況，接收者得到的頻寬會隨時間改變而難以預計，從以出現緩衝區下溢(Buffer Undeflows)及播放斷續的情形。本論文提出一個新方案應用於多個供應者的網絡服務。我們提出一個在互聯網上透過反覆的實驗總結而成的總可得頻寬(Aggregate Available Bandwidth)模型。實驗結果顯示由多個供應者得到的總可得頻寬具有一個相同的性質，可藉此建立模型及根據模型作出估計。這個方法是以機會率來估計所得影像的質素，它的發現爲頻寬敏感的(Bandwidth-sensitive)應用(例如影像串流)開啓了一個全新的方向。我們亦透過這個模型而發展出整合下載串流(Hybrid Download-Streaming)的演算法和播放調節串流(Playback-adaptive Streaming)的演算法，並運用於不同可得頻寬下的影像串流上。

CONTENTS

Chapter 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.

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 Chapter 3, modeling the bandwidth availability for a single sender is very difficult, if not impossible.

Given the limitation of this single-sender approach, researchers have begun to investigate approaches employing multiple senders [1-3] 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 numbers 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 streaming experiments in the global PlanetLab testbed [4]. 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 [5].

This work has three contributions. First, to the best of our knowledge, this is the study investigating the 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, this discovery opens a new way to providing probabilistic performance guarantees in bandwidth-sensitive applications such as video streaming even in the current best-effort Internet.

The rest of the thesis is organized as follows. In Chapter 2, we discuss some of the related works on bandwidth modeling. In Chapter 3, we present the single-source bandwidth availability. In Chapter 4, we present the multi-source bandwidth availability for comparison. In Chapter 5, details and tools for the measurement are given. Chapter 6 and 7 present two algorithms to provide probabilistic performance guarantees in streaming video over the current Internet.

Chapter 2

RELATED WORK

Modeling of network traffic has been studied extensively in the literature. It is generally accepted that the Internet traffic cannot be adequately modeled by simple models such as a Poisson process [6]. A number of studies showed that network traffic is in fact self-similar [7-10], 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 streaming media data 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 itself (e.g., due to other concurrently running applications), as well as dynamics of the transport protocol (e.g., TCP).

Not surprisingly, with so many system factors in the equation the temporal bandwidth availability between a sender and a receiver can vary significantly across different senders and as a result does not conform to any conventional models. On the other hand, if we combine the available bandwidth of multiple senders, then the aggregate

Chapter 2

RELATED WORK

Modeling of network traffic has been studied extensively in the literature. It is generally accepted that the Internet traffic cannot be adequately modeled by simple models such as a Poisson process [6]. A number of studies showed that network traffic is in fact self-similar [7-10], 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 streaming media data 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 itself (e.g., due to other concurrently running applications), as well as dynamics of the transport protocol (e.g., TCP).

Not surprisingly, with so many system 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 models. On the other hand, 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 Chapter 4 by computing the correlation coefficient [11] of different senders' bandwidths.

Chapter 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 [4] 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 [12] 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 connect to the sender to initiate data transmission. We measure 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. The receiver captures the average throughput for each source once every 10 seconds (the default setting in Iperf). The measurement lasts for 3 hours, which generated 1,080 measurement samples.

Although Iperf also supports the use of UDP in measurements, 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 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 [13], etc.).

3.2 Measurement Results

We first examine the throughput of individual senders. Fig. 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. 2 plots the bandwidth distribution of 4 out of the 47 senders, over the measurement period of 3 hours. 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.

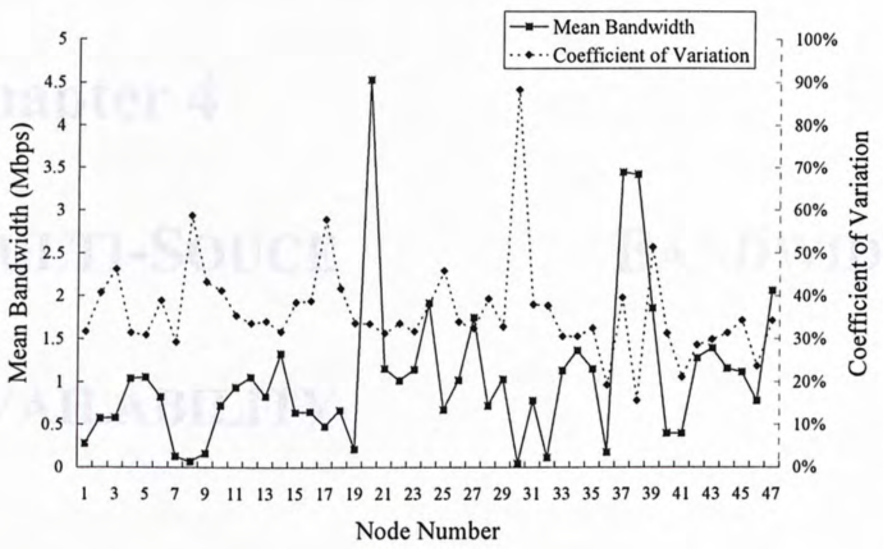

Fig. 1. Average bandwidth and Coefficient-of-variations of the 47 senders.

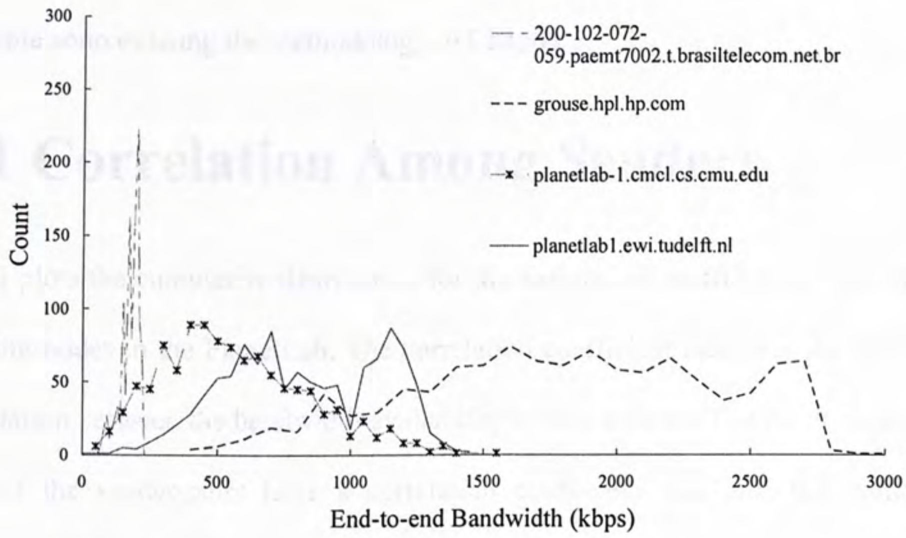

Fig. 2. End-to-end bandwidth distribution, sample from (a) planetlab-1.cmcl.cs.cmu.edu (b) planetlab1.ewi.tudelft.nl (c) grouse.hpl.hp.com (d) 200-102-072-059.paemt7002.t.brasiltel ecom.net.br.

Chapter 4

MULTI-SOUCE BANDWIDTH AVAILABILITY

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

4.1 Correlation Among Senders

Fig. 3 plots the cumulative distribution for the correlation coefficient [11] of the 47 sending nodes in the PlanetLab. The correlation coefficient measures the degree of 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.

4.2 Aggregate Bandwidth

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 [14] 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 further 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 [14], the results show that the measured distribution becomes normally distributed even when there are only 4 senders. This suggests that even with only a few senders, we can still approximate the aggregate available bandwidth using the normal distribution.

4.3 Sensitivity Analysis

The previous results are primarily averaged over a large number of randomly-generated sender combinations. To investigate the sensitivity of the model to different combinations of senders, we randomly form different combinations of senders and then compute the p-value of their aggregate available bandwidth. The distribution of the p-values is plotted in Fig. 6 for 4, 6, 8, and 10 senders respectively. We observe that the p-value does vary across different combinations of senders but in most cases (over 85%) the resultant aggregate available bandwidth remains conformance to normal (i.e., with p-value not less than 0.05).

Another key parameter in our measurement is the time interval in computing the available network bandwidth. To investigate the model's sensitivity to this parameter we plot the p-value in Fig. 7 versus different measurement time intervals ranging from 1 to 10 seconds. The results are averaged over 10 sets of trace data from PlanetLab. We observe that although there are variations in the p-value, which all exceeds the threshold of 0.05, there is no consistent trend with respect to the length of the measurement time interval. This suggests that the length of the measurement time interval is not critical to the validity of the model.

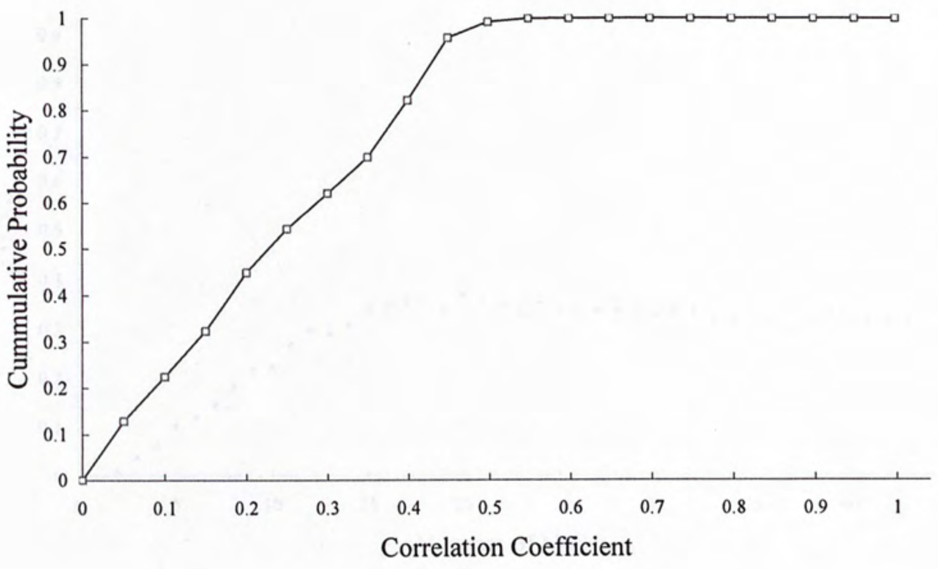

Fig. 3. Correlation of senders' bandwidth availability.

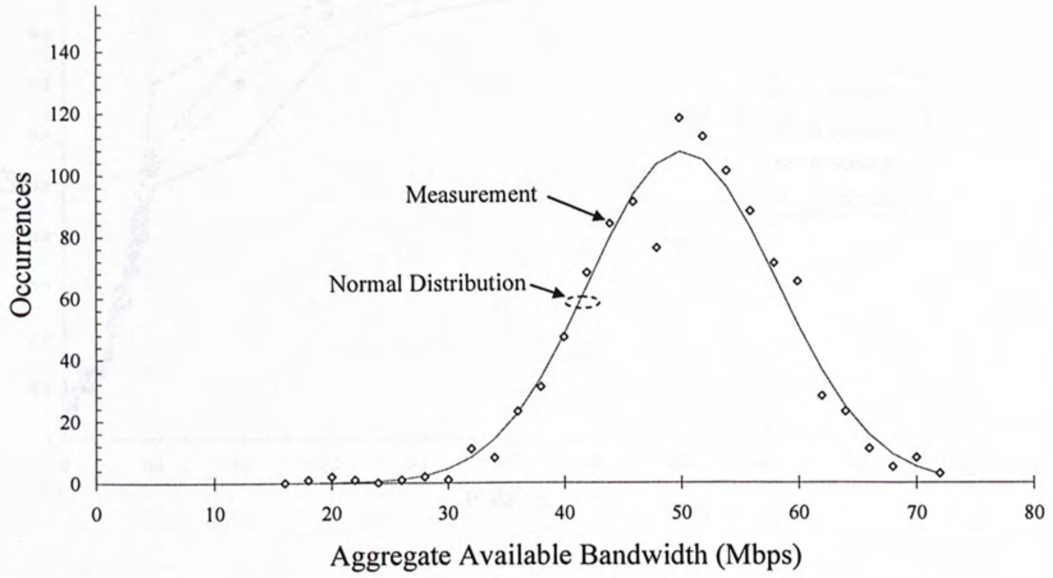

Fig. 4. Distribution of measured aggregate available bandwidth of 47 senders.

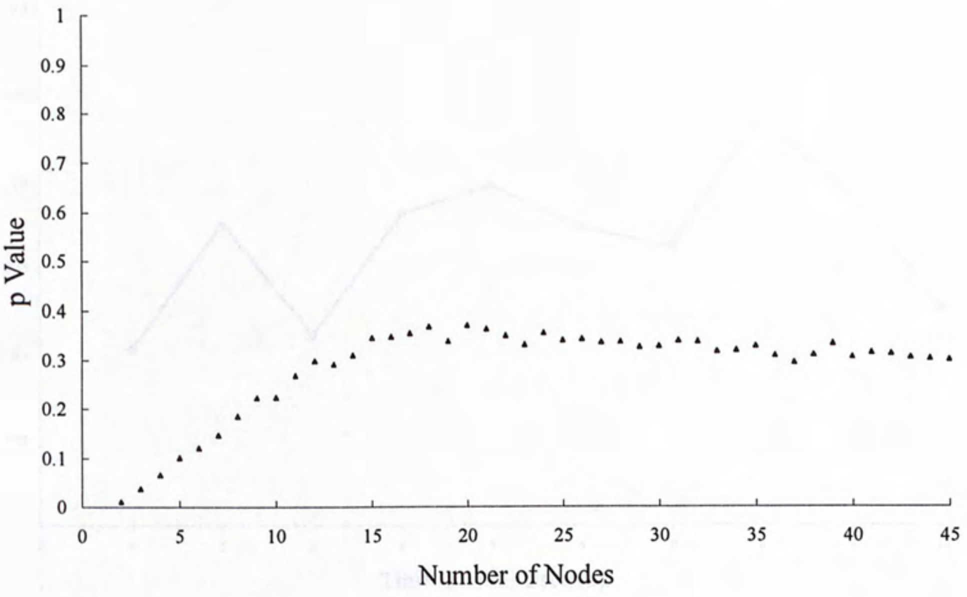

Fig. 5. The effect of the number of senders on normal conformance.

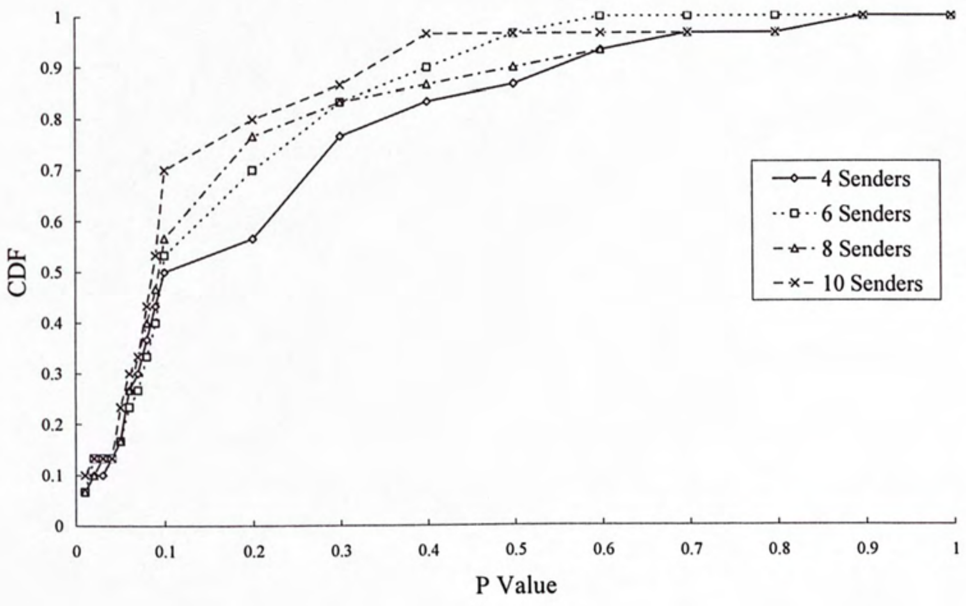

Fig. 6. Distribution of p-value for different sender-combinations.

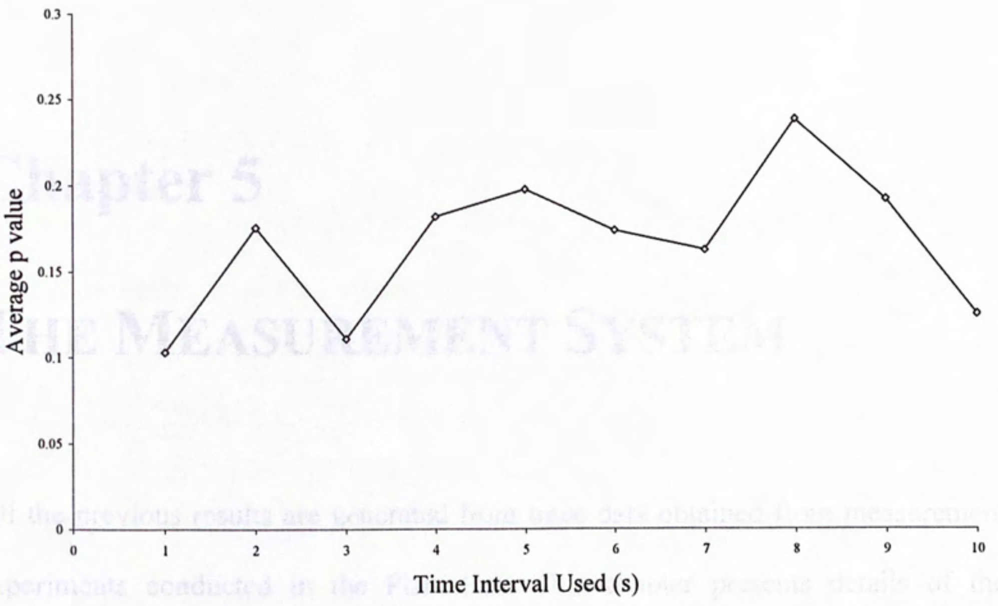

Fig. 7 Average p value for different time intervals

Chapter 5

THE MEASUREMENT SYSTEM

All the previous results are generated from trace data obtained from measurement experiments conducted in the PlanetLab. This chapter presents details of the measurement system we developed and the measurement algorithms employed.

5.1 Overview of PlanetLab

The PlanetLab Consortium consists of academic, industrial, and government institutions. Planetlab is an overlay in the Internet, which aims at providing researchers an open platform to perform network tests or simulations.

Currently, PlanetLab is running the Fedora Core 2 platform in all its nodes. Experimental programs can be developed in C, C++, or java (provided that JVM is installed) and deployed to run in any of the PlanetLab nodes (there are currently over 500 nodes in PlanetLab and the number is growing).

Resource management in PlanetLab nodes are managed by the use of Slices. A Slice is a collection of PlanetLab resources assigned to a set of users. Each PlanetLab node is shared by creating virtual servers, with each virtual server allocated to different Slices. A Slice can include virtual servers from multiple PlanetLab nodes. As these virtual servers ultimately share the computing and network resources of a node, activities of different Slices at the same node are not physically isolated, e.g., they compete for and share the available network bandwidth and CPU processing cycles. This lack of isolation however, is in fact more suitable for our experiment as it renders the measurement data more realistic.

5.2 Measurement Tool

In our project, we employed a modified version of Iperf (based on Version 1.70) for bandwidth measurements. Iperf [12] is a software tool for measuring the available TCP bandwidth through sending bulk data from a sender to a receiver. It allows the tuning of various TCP parameters and UDP characteristics (e.g., binding port, window size, MSS, etc.) and captures numerous statistics such as bandwidth, delay jitter, and loss. To increase the flexibility in using the trace data, we modified the Iperf's to capture not only the average TCP throughput, but also the individual time at which the receiver receives data from TCP (e.g., via the socket API's recv() function).

The modified Iperf tool generates three trace files: IPfile.txt and result_X.bin, where X represents the trace number.

IPfile.txt is a text-based file capturing the IP addresses of the sending nodes and the receiver node. The format is shown in Fig. 8.

result_X.bin is a binary file capturing the raw trace data (Fig. 9). The receiver allocates a 4096-byte buffer for receiving the incoming data through the sockets connected to the senders. The first two fields (Fig. 9) record the time when the data is received (via socket API's recv() function). The last field shows the amount of data received (as reported by the recv() function). Note that the binary format in Fig. 9 consumes 10 bytes for each trace datum. Given a Slice in a PlanetLab node has a maximum storage of 5GB the system can store only limited trace data, e.g., up to 6 hours at 6Mbps.

This more detailed trace data allows one to generate average throughput data using arbitrary measurement time intervals without the need to rerun the same experiments (which is impossible in practice due to the constantly changing network conditions).

Receiver IP	<Space>	No. of sender

Figure 8 IPfile.txt data format

Time <in seconds>	Time <micro seconds>	Size
4 bytes	4 bytes	2 bytes

Figure 9 result_X.bin data format

5.3 Process Control

As there are often tens of PlanetLab nodes participating in a measurement experiment, manual control of individual measurement processes quickly become impractical. Therefore we developed an automatic system based on automated remote login and scripting to automatically identify PlanetLab nodes that are available for measurement experiment; to setup the measurement environment; to execute and initiate the sender and receiver measurement processes; and finally to collect the captured trace data for further processing.

```
Function Auto_Measurement_Program()
do
{
        1. Generate a healthy node list in PlanetLab
        2. Discard those nodes in the same subnet
        3. Run the experiment
        4. Download the trace data
        5. Validation
        6. Parse the result
        7. Output
}
```

Figure 10 pseudo code for the measurement program

Fig. 10 uses pseudo code to outline the execution flow of the measurement system. Nodes in Planetlab have very different capabilities (e.g., CPU processing power, memory, and network bandwidth) as well as availability. Our experiments revealed that many nodes did randomly shut down from time to time, and the total number of concurrently working nodes amounts to only one-fourth of the total number.

Therefore the first problem in the measurement experiments is to find out the current healthy nodes before activating the bandwidth measurement tools (Step 1 in Fig. 10). To address this problem, we developed a controller software to connect and upload the updated measurement tools to all nodes and then test run the tools (e.g., Iperf) after a waiting time of 60 seconds. A responding node will be added to the healthy list while a non-responding node will be excluded from the measurement experiments. In our current setup the system will proceed with the experiment once there are at least 65 healthy nodes available.

Next, the system will eliminate nodes in the same subnet (Step 2 in Fig. 10) as these nodes are likely to be network-wise close neighbors which has two undesirable consequences. First, these nodes will likely to connect to the larger Internet through the same network link and so the available bandwidth of them will be highly correlated. Second, if two nodes from the same subnet are chosen as sender and receiver, the throughput will be limited only by their local connections, which is likely to be high-speed LAN connections. This will usually result in the receiver becoming the bottleneck as sending data consumes less processing time than receiving data. Given that out experiments are really to capture the available network bandwidth, having the receiver becoming the bottleneck will obviously distort the trace data.

For these two reasons the measurement system will eliminate all but one node in the same subnet from the healthy nodes list. The filtering tool is implemented in a software module called "filter_samesubnet".

In Step 3, the system will randomly divide the healthy nodes into groups of senders and receivers and then start the measurements using the modified Iperf. After the measurements are completed, the system will download the trace data from the receiver nodes using PSCP [15] (Step 4).

Given the dynamic nature of PlanetLab nodes, even nodes in the healthy list may later go offline during the measurement experiment. In such case the trace data will not be consistent as the number of senders will change (i.e., drop) in the middle of the measurement. Therefore the system will conduct post-measurement processing to identify and discard those inconsistent trace data (Step 5).

Finally, the raw trace data are in compressed format to reduce storage but are not readily useable for computations or simulations. Therefore we developed a set of post-processing parsers to generate data sets for graphing, computations, and simulations (e.g., for use in the NS2 simulator). For example, the results in Fig. 7 in Chapter 4 are computed from data sets of different measurement time intervals generated by these post-processing tools.

Chapter 6

Hybrid-Download Streaming

6.1 Introduction

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 chapter, we propose a hybrid-streaming algorithm to tackle the second scenario and in Chapter 7 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.

6.2 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 this thesis. 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 10 seconds 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.,

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

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

Substituting the definition of A_i and B_i into (1) and rearranging we can obtain

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

Now as C_j's are normally-distributed random variables, the sum of n C_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

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

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

$$w = \min\left\{ v \mid F^{(n)}(\mathrm{R}(n-v)) \le \Delta, \forall n \ge v \right\} \tag{4}$$

6.3 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. Fig. 11 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 Ci's are assumed to be known a priori. Obviously this algorithm is not realizable in practice and is thus included for comparison only.

We run 29 different experiments each with a different traffic trace collected from PlanetLab. The video length and its bit rate range from 500 seconds to 1,800 seconds and 200 kbps to 1 Mbps respectively. In each experiment, there are 5 to 10 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 as discuss earlier in this thesis.

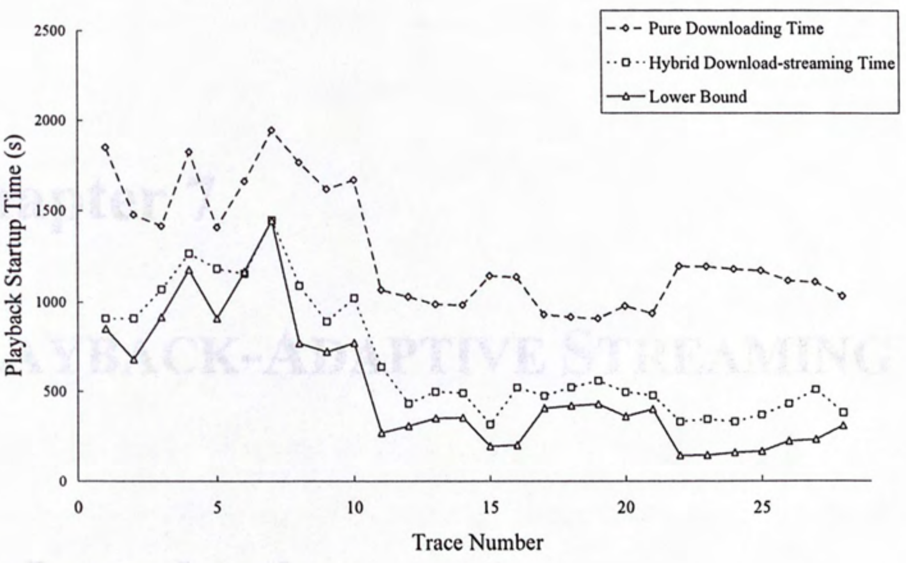

Fig. 11 Comparison of startup time.

Chapter 7

Playback-Adaptive Streaming

7.1 Introduction

In this chapter 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 in chapter 4 of the thesis 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) [16] 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 [17], 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 [17], 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 [18-19] and transcoding [13,20], 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 3.2 and the adaptive rebuffering algorithm in Section 3.3. We then evaluate their performance using trace-driven simulations in Section 3.4.

7.2 Streaming 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.,

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

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

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

Substituting (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.

7.3 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 methods similar to (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

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

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

7.4 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 = 1s, Bpre = 5s, 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 [4]. 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 seconds; (b) "Adaptation Only" – using playback rate adaptation with a constant rebuffering duration of 5 seconds; and (c) "Adaptation and Rebuffering" – using both playback rate adaptation and adaptive rebuffering.

Fig. 12 and 13 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 (about 60 seconds) and large number of playback pauses (nearly 12 occurrences). Second, by introducing playback rate adaptation the performance is improved significantly and the improvement increases with larger display rate adjustment limit.

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. 12). 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. 13 when compared to using a fixed rebuffering time of 5 seconds.

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.

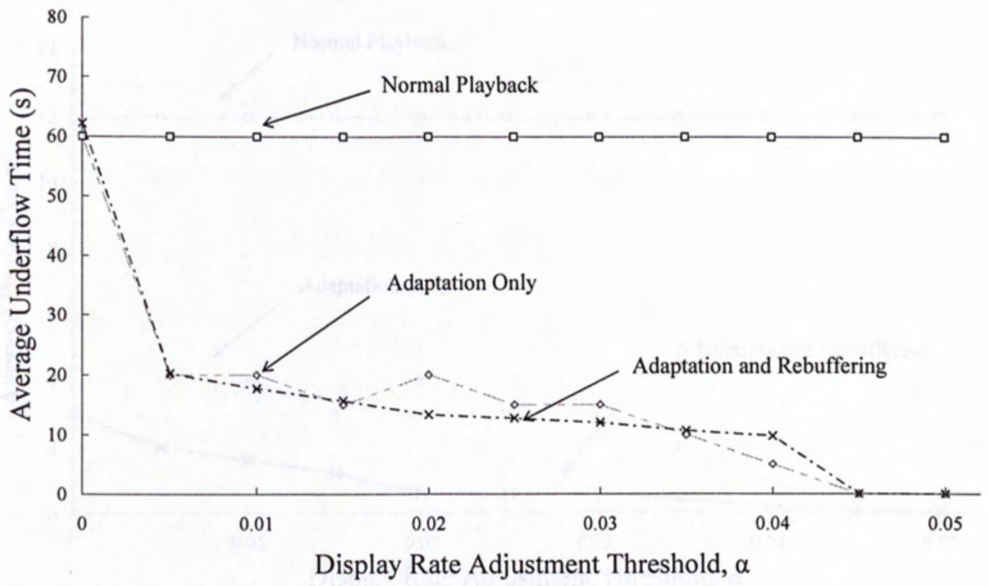

Fig. 12. Average underflow time versus playback rate adjustment limit.

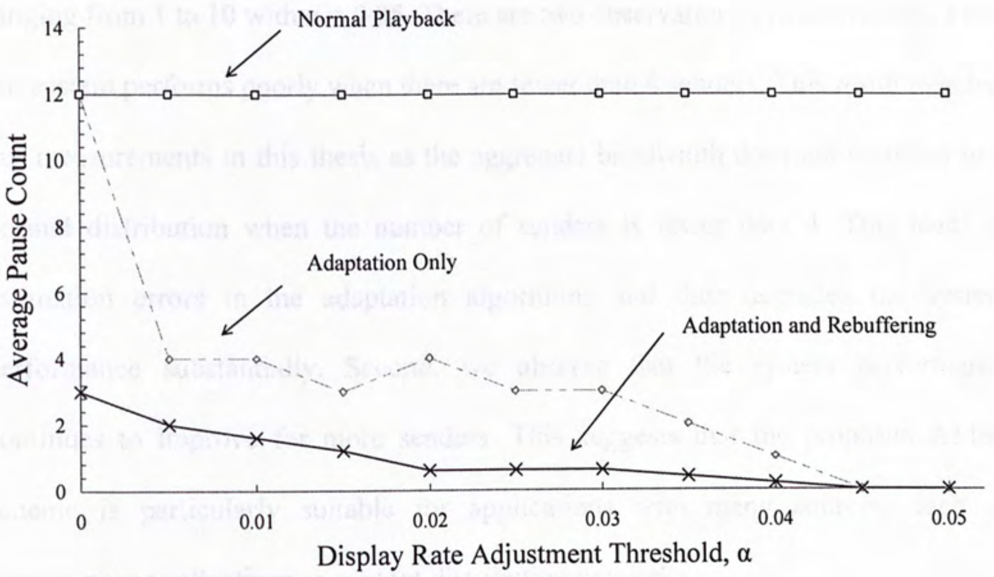

Fig. 13. Average pause count versus playback rate adjustment limit

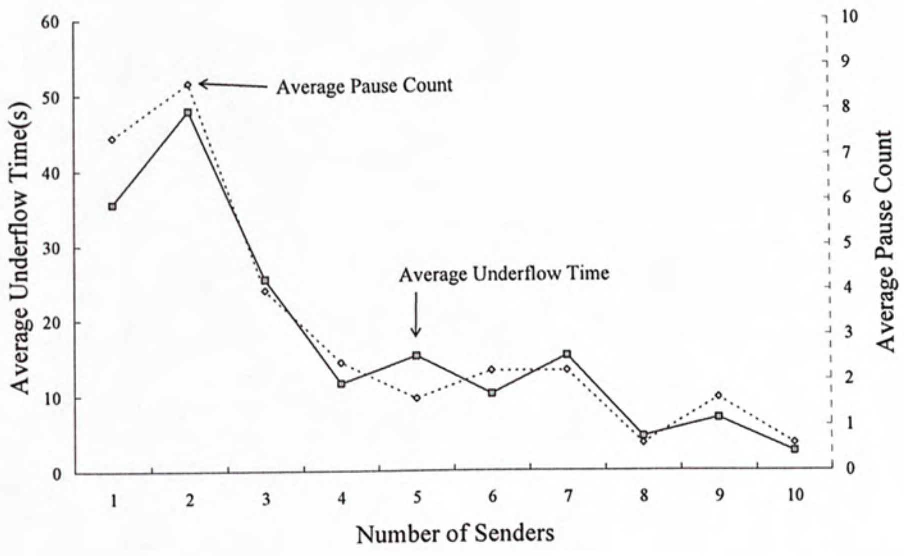

Fig. 14. Average underflow time and pause count versus number of senders.

Fig. 14 plots the average underflow time and pause counts for number of senders ranging from 1 to 10 with $\alpha = 0.05$. 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 this thesis 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.

Chapter 8

Conclusions and Future Work

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 PlanetLab 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.

The next step is to expand the measurement studies beyond the PlanetLab to investigate further the validity of the model in the wider Internet and under different network scenarios. On the other hand, in addition to approaching the problem from an experimental angle, it is also important to develop the theoretical foundations for the bandwidth to obtain deeper insights and intuitions.

BIBLIOGRAPHY

[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] V. Agarwal and R. Rejaie, "Adaptive Multi-source Streaming in Heterogeneous Peer-to-Peer Networks" SPIE Conference on Multimedia Computing and Networking, San jose, California, January 2005.

[3] E. Setton, Yi Liang, B. Girod, "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.

[4] Planetlab Homepage : http://www.planet-lab.org/

[5] S.C. Hui, Jack Y.B. Lee, "Playback-Adaptive Multi-Source Video Streaming" Fourth International Conference on Intelligent Multimedia Computing and Networking, Salt Lake City, Utah, USA, July 2005.

[6] V. Paxson, "Fast approximation of self-similar network traffic", Tech. Rep., Lawrence Berkeley Laboratory and EECS Division, University of California, Berkeley. April 1995.

[7] K. Park, G. Kim, and M. Crovella, "On the Relation Between File Sizes, Transport Protocols, and Self-Similar Network Traffic," International Conference on Network Protocols, Columbus, Ohio, USA, p 171-180, Oct. 1996.

[8] K. Park, G. Kim, and M. Crovella, "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,.

[9] T. Tuan and K. Park, "Congestion Control for Self-Similar Network Traffic," Department of Computer Science., Purdue University, CSD-TR 98-014, May 1998,

[10] P. R. Morin, "The Impact of Self-Similarity on Network Performance

Analysis," Ph.D. dissertation, Carleton University, Dec. 1995.

[11] R. Morris and Dong Lin, "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.

[12] Iperf Homepage : http://dast.nlanr.net/Projects/Iperf/

[13] 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.

[14] Hahn and Shapiro, Statistical Models in Engineering, Wiley, 1994

[15] PSSH, PSCP Homepage : http://www.theether.org/pssh/

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.