Measurement and Application of Many-to-one Data Flows

HO, Po Yee

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Philosophy

in

Information Engineering

August 2007


香港中文大學
圖書館系統
0 2 SEP 2008
UNIVERSITY
LIBRARY SYSTEM
THE CHINESE UNIVERSITY OF HONG KONG

ACKNOWLEDGEMENTS

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. Wing C. Lau, who has contributed lots of valuable ideas and comments to my research.

I would also like to thank my dear colleagues, including Johnny, Gary, Brio and Rudolf. They provided me with earnest help and friendship during the past two years.

ABSTRACT

With the rapid deployment of multimedia contents in the Internet, the need to understand the characteristics of bandwidth availability in the Internet in general, and the capability to estimate and predict bandwidth availability in particular, is becoming increasingly important. Nevertheless, without any sort of resource reservation or explicit traffic regulation, the bandwidth availability of a one-to-one data flow across the Internet is still subject to a multitude of factors and thus is largely unpredictable. This study goes beyond one-to-one data flows to investigate the characteristics of many-to-one data flows, where multiple senders transmit data simultaneously to the same receiver, e.g., multi-source video streaming and peer-to-peer systems. In sharp contrast to one-to-one data flows, the measurement results reveal that having multiple senders not only achieves higher aggregate bandwidth, but the resultant aggregate data flows will also exhibit significantly more predictable properties even if the individual flows do not exhibit any consistent behavior. This newfound predictability thus could open up a new way to provide probabilistic quality-of-service guarantees for running bandwidth-sensitive applications over the best-effort Internet. This thesis presents the new findings obtained from extensive measurements conducted in the Internet and in PlanetLab, develops a mathematical framework to explain the observations from the measurement results, establishes the invariant properties of general many-to-one data flows, and proposes a novel predictive buffering algorithm to exploit the properties of many-to-one data flows in video streaming applications.

摘要

隨著多媒體內容在互聯網上迅速發展，了解頻寬可得性 (bandwidth availability) 的特性的需要，特別是其可估計和預測的能力，變得越來越重要。然而，在沒有任何資源保留或明確流量管制的情況下，互聯網上一對一數據流 (one-to-one data flow) 的頻寬可得性還是取決於很多因素，因而是難以預測的。本項研究超出一對一數據流的範圍，研究多對一數據流 (many-to-one data flow) 的特性，即多個發送者同時傳達數據給同一接收者，例如多源視訊串流和點對點系統。相對於一對一數據流，實驗測量結果顯示使用多個發送者不僅能取得更高的總頻寬 (aggregate bandwidth)，而且能令多對一數據流擁有更高的可預測性。這新發現的可預測性爲提供或然性質素保證給互聯網上頻寬敏感的應用開啓了一個全新的方向。本論文匯報從在互聯網及PlanetLab上大規模測量所獲得的新發現，建立一個數學框架來說明從測量結果得來的觀察，確立多對一數據流不變的特質並提出一個利用視訊串流中多對一數據流特質的預測緩衝演算法。

CONTENTS

Chapter 1

INTRODUCTION

With the rapid deployment of multimedia contents in the Internet, the need to understand the characteristics of bandwidth availability in the Internet in general, and the capability to estimate and predict bandwidth availability in particular, is becoming increasingly important. Previous studies which investigated the bandwidth availability of Internet flows are primarily focused on one-to-one data flows [1-7] and on aggregate flows passing through a common network link [8-12]. This one-to-one model, with a single source sending data to a single receiver, is a good representation of the client-server model used widely in many Internet applications.

Not surprisingly, without any sort of resource reservation or explicit traffic regulation, the bandwidth availability of a one-to-one data flow across the Internet is subject to a multitude of factors such as competing traffic, server load, and protocol dynamics, and thus is very difficult, if not impossible, to predict. This is a direct consequence of the Internet's best-effort nature and its lack of end-of-end quality-of-service control.

On the other hand, in recent years a new class of Internet applications is becoming extremely successful – distributed and peer-to-peer applications. These applications rely on replicating data across multiple hosts in the Internet, and then serve data to clients in a distributed, concurrent manner. A data transfer session in this class of

applications is inherently many-to-one, with multiple sources concurrently transmitting different parts of the requested data to a receiver. This emerging many-to-one data flow model presents many new challenges but at the same time also opens up a new opportunity to solving some of the long-standing challenges to deploying bandwidth-sensitive applications over the Internet.

This study reports results obtained from a large-scale measurement study of many-to-one data flows in the Internet, develops a mathematical framework to explain the observations and establishes the invariant properties of general many-to-one data flows. In sharp contrast to one-to-one data flows, the measurement results reveal that having multiple senders not only achieves higher aggregate bandwidth, but also results in many-to-one data flows with significantly more predictable properties even if the individual flows do not exhibit any consistent behavior. This newfound predictability thus could open up a new way to provide probabilistic quality-of-service guarantees for running bandwidth-sensitive applications over the best-effort Internet.

To exploit the properties of many-to-one data flows, this thesis presents a novel predictive buffering algorithm for streaming video from multiple sources across the Internet to a receiver. Unlike existing Internet video streaming systems where playback continuity is subject to the varying Internet bandwidth availability, the proposed predictive buffering algorithm can determine at runtime the buffering time required to ensure continuous playback for the entire duration of the video. Extensive trace-driven simulations showed that with sufficient number of senders the predictive buffering algorithm can achieve very high successful playback ratio while maintaining a buffering delay that is surprisingly close to the lower bound.

The rest of the thesis is organized as follows: Chapter 2 reviews some previous work in bandwidth measurements and multi-source video streaming; Chapter 3

summarizes the measurement methodology and the experimental setups employed in this study; Chapter 4 analyzes the measurement data; Chapter 5 develops a mathematical framework that explains some of the empirical observations as well as establishes some invariant properties of general many-to-one data flows; Chapter 6 presents the predictive buffering algorithm; Chapter 7 evaluates the performance of the predictive buffering algorithm using trace-driven simulations; Chapter 8 discusses some possible future works; Chapter 9 summarizes the study.

Chapter 2

BACKGROUND AND RELATED WORK

Many Internet bandwidth measurements have been reported in the literature, with different methodologies and software tools for measuring or estimating the bandwidth of network links and network paths. While all these previous works are related to bandwidth measurement, they are in many cases measuring different types of network bandwidth. We provide below precise definitions for three types of network bandwidth and review the relevant previous works.

2.1 Link/Path Capacity

The capacity of a link is the maximum data rate a flow can utilize when there is no other traffic flow sharing the same link. Note that this may or may not be the same as the link's physical bandwidth, depending on whether the network router implements any rate-limiting control over the data flows. Well-known tools for estimating the per-hop link capacity include pathchar [1], clink [2], and pchar [3].

If a data flow traverses N links from the sender to the receiver and C_i is the capacity of link i, then $C=\min\{C_1,C_2,\ldots C_N\}$ is the end-to-end path capacity of the data flow. Tools for estimating the path capacity include bprobe [4] and pathrate [5]. Knowledge of link and path capacities is useful to traffic engineering and network planning.

2.2 Unutilized Bandwidth

A second type of network bandwidth is the unutilized capacity of a link [6]. Let $u_i(t)$ be the utilization of link i, in normalized unit from 0 to 1, at time t. Then the average link utilization in the interval $[t, t + \tau)$, denoted by $u_i(t, t + \tau)$, can be computed from

$$u_i(t,t+\tau) = \frac{1}{\tau}\int_t^{t+\tau} u_i(t)dt \quad (1)$$

Given C_i as the capacity of link i, then the unutilized bandwidth of link i during the interval $[t, t + \tau)$, denoted by $U_i(t, t+\tau)$, can be computed from

$$U_i(t,t+\tau) = C_i[1-u_i(t,t+\tau)] \quad (2)$$

Similarly, the unutilized bandwidth of a network path is equal to the minimum unutilized bandwidth of the N links of the network path:

$$U(t,t+\tau) = \min_{i=1\ldots N}\{U_i(t,t+\tau)\} \quad (3)$$

Well-known tools for estimating the end-to-end unutilized bandwidth include pathload [6] and IGI [7]. Knowledge of unused bandwidth is also very useful to traffic engineering, e.g., guiding the routing or rerouting of traffic from congested network links to underutilized links.

2.3 Achievable Bandwidth

The third type of network bandwidth is the end-to-end throughput achievable by a transport flow in passing through a network path [13]. Of particular importance is the throughput achieves by the TCP transport protocol as it is the building blocks of most Internet applications. For this reason our measurement experiments are designed specifically to study the achievable bandwidth using the TCP transport protocol.

In contrast to unutilized bandwidth, a TCP flow will be able to obtain a fair share of bandwidth even if the network path is already fully utilized, assuming that the competing flows also share bandwidth fairly (e.g., TCP and TCP-friendly flows). Let $d(t, t+\tau)$ be the amount of data transferred in the interval $[t, t+\tau)$ by a TCP flow. Then the achievable bandwidth for the interval is given by

$$r(t,t+\tau) = \frac{1}{\tau} d(t,t+\tau) \tag{4}$$

In a many-to-one data flow there will be multiple TCP flows originating from multiple sources to the same receiver. The aggregate achievable bandwidth of this many-to-one data flow is equal to the sum of the achievable bandwidth of all the individual flows. Let there be N senders, with $d_i(t, t+\tau)$ denoting the amount of data received in the interval $[t, t+\tau)$ by the receiver from source i. Then the aggregate achievable bandwidth for the interval is given by

$$A(t,t+\tau) = \frac{1}{\tau} \sum_{i=1}^{N} d_i(t,t+\tau) \tag{5}$$

Note that in this definition we do not assume the individual flows to be independent. Some or even all of the flows could share a common network bottleneck and thus may exhibit correlated bandwidth variations. This inter-flow correlation issue will be studied in detail in Section 4.2.

On the other hand, the above definition also incorporates both network-limited and source-limited achievable bandwidth. The former represents the common notion of available bandwidth as constrained by the network's link capacities and utilizations while the latter represents the case where the source host is the bottleneck, i.e., the maximum data rate is limited by how fast the source host can send out data and the network path in this case does not limit the data throughput at all. This subtle distinction can be measured and is reported in Section 4.4.

Chapter 3

MEASUREMENT METHODOLOGY

Two measurement setups are employed in this study. The first one uses a custom-developed measurement system deployed in PlanetLab hosts around the world [14]. The second one makes use of another custom-developed measurement system to transfer large data files (Fedora Linux distribution images) from FTP mirror servers around the world. Both measurement setups are conducted continuously and automatically by a management software, which has begun operation since November 2005. The measurement datasets are collected and automatically posted to the measurement data archive [15].

Common to both measurement setups, a measurement session consists of up to 10 sources sending data simultaneously to the same receiver using TCP as the transport. Thus the measurements measure the achievable bandwidth (c.f. Section 2.3) in many-to-one data flows. Both the sender and the receiver software are implemented using the standard sockets [16] application programming interface (API), capturing the source, the time, and the amount of data received into a trace file. The use of the sockets API has two advantages. First, it enables the measurement software to run in the PlanetLab environment as applications are normally not allowed to access the network interface of PlanetLab host directly. Second, the throughput measured via the sockets API incorporates not only the effect of sending host and the network, but also

the buffering mechanism inside the operating system. This provides a more accurate representation of the achievable bandwidth as observed by real-world network applications.

In the following we describe detail operations of the two measurements setups and discuss their limitations.

3.1 PlanetLab Measurement

PlanetLab is a global research network with over seven hundred hosts located at over three hundred sites around the world, all connected through the Internet. PlanetLab hosts run a variant of the Linux operating system with virtual server capability. We installed our measurement system in 284 of the PlanetLab hosts. Each measurement node operates as either a sender or a receiver. As a sender it will send data to the receiver as fast as the transport allows; and as a receiver it will receive data from the transport as fast as it can (subject to data availability). A management server running in our local host continuously monitors and controls the operation of all the measurement nodes.

Each measurement run proceeds in four phases, namely setup phase, pre-measurement phase, measurement phase, and data collection phase. During the setup phase the management server first randomly selects a node from the pool of idle nodes, i.e., nodes which are not running measurements, and designates it to act as the receiver in the new measurement run. Next, the server sends the list of idle nodes to the designated receiver to begin the pre-measurement phase.

In this second phase the receiver randomly picks a node from the idle list to perform a pre-measurement test, which measures the average end-to-end achievable bandwidth from the selected sender to the receiver over a configurable test interval (currently 20

seconds). If the sender's average throughput is less than a configurable per-sender threshold (currently set at 1.6Mbps) then the sender will be included in the new measurement run. The process repeats until either (a) the desired number of senders is obtained; (b) the total achievable bandwidth of all senders exceeds a configurable per-receiver threshold (currently set at 8 Mbps), in which case the last sender added will be dropped; or (c) the idle sender list is exhausted.

The pre-measurement phase is introduced to overcome two problems. First, some PlanetLab hosts have very high-bandwidth network connections and thus could cause serious congestion at the receiver host, especially if more than one such high-bandwidth hosts send data to the receiver simultaneously. As each PlanetLab host is shared by many users using virtual servers, the induced congestion could cause significant performance degradation to other users sharing the same host. The per-receiver bandwidth threshold is designed to prevent this problem. Second, if a high-bandwidth sender is selected early in the process then only a few (or none at all) senders could be added to form the many-to-one data flow in order to comply with the per-receiver bandwidth threshold. Therefore the per-sender bandwidth threshold is introduced to filter out very high bandwidth senders to allow more senders to be included in the measurement run. In addition it also reduces the likelihood that the sender will hit the rate limit set by the local administrator of the PlanetLab host.

After the list of senders is determined the system then begins the measurement phase by triggering all senders to continuously send data to the receiver. Each measurement run lasts for 2 hours (constrained by the daily per-host data transfer limit in PlanetLab), after which the system enters the data collection phase where the trace data stored in the receiver are transferred back to the management server for processing and archival.

Conducting measurements using PlanetLab hosts has three distinctive advantages: scale – hundreds of hosts available, control – allows the use of user-written measurement applications, and reach – spans a wide geographical area covering over three hundred sites around the world (see Table 1). There are nevertheless some limitations as well. First, as most PlanetLab hosts are run by universities, research laboratories, and large corporations, their network connectivity characteristics naturally reflect this bias and thus may not be a good representation of residential user hosts where bandwidth is likely to be more limited, especially in the uplink connection (e.g., DSL [17] subscribers). Second, PlanetLab hosts are shared using virtual servers and thus experiments running in other virtual servers sharing the same host could interfere or even interact with the measurement runs and vice versa. The measured achievable bandwidth thus could be modulated by other experiments concurrently running at the sending hosts and the receiving host. However in many network applications one would also expect them to coexist with other applications competing for bandwidth and other host resources.

3.2 FTP Measurement

The second measurement setup makes use of 54 public FTP servers mirroring the Fedora Linux distribution CD image. This particular set of mirror servers is chosen for the large file available for download (FC4 Disc 1 of size 635 MB) and for their relatively wide availability and geographical distribution (see Table 1). Unlike PlanetLab hosts these FTP servers are production hosts serving real data to real users and thus provide another perspective to the bandwidth available in a real system setting.

Obviously in this case we could not implement our own sender software and have to rely on the FTP server software to act as the sender in the measurements. We developed a custom FTP client software running in a dedicated Linux hosts at our research laboratory to act as the receiver in the measurements. In each measurement run, it randomly draws N (1 to 10) FTP servers from the pool of 54 Fedora mirror sites and then initiate downloads from all N servers simultaneously. As FTP runs over TCP the measured throughput will reflect the achievable bandwidth between the FTP servers and the receiver. Again each measurement run lasts for 2 hours with N varies from 1 to 10 in subsequent runs. In addition to collecting bandwidth trace data, we also ran the tcpdump tool [18] at the receiver host to capture detailed packet traces for more in-depth analysis. Note that the system does not implement bandwidth filtering as in the PlanetLab setup because the receiver host is a dedicated machine with sufficient resources to handle the incoming traffics.

This FTP-based measurement setup differs quite substantially from the PlanetLab setup. First, the sender, i.e., FTP server, in this case is a concurrent server serving many users simultaneously with the same service as opposed to running completely different experiments in the same PlanetLab host. This environment thus more accurately reflects the characteristics of many-to-one data flows where the senders are Internet servers.

Second, while senders in the PlanetLab setup always send data as fast as the transport allows, our measurement results suggest that some FTP servers implement per-connection rate limiting. In case the available network bandwidth is higher than the server's rate limit the achievable bandwidth will then be limited by the source rather than by the network. We will return to this issue in Section 4.4.

Chapter 4

ANALYSIS OF MEASUREMENT DATA

This chapter analyzes the characteristics of data flows in the two measurement setups. The analysis is based on measurement data collected from December 2005 to December 2006, which consists of 879 and 628 measurement runs in the PlanetLab and FTP setups respectively. The total number of distinct hosts involved is 284 (PlanetLab) and 54 (FTP) respectively, with geographical distributions as listed in Table 1.

Table 1 The geographical distribution of sender nodes

Region	PlanetLab	FTP
Africa	0	6
America	152	19
Asia/Pacific	61	2
Europe	71	27

Table 2 Properties of per-flow achievable bandwidths

Bandwidth	PlanetLab	FTP
Min	2.65 Kbps	0.21 Kbps
Max	4,099.63 Kbps	12,576.38 Kbps
Median	671.75 Kbps	1,738.83 Kbps
Mean	570.66 Kbps	1,496.65 Kbps

4.1 Per-Flow Achievable Bandwidth

The following analysis is performed using one-second average throughput as the measurement sample. Specifically, let $X_i = \{x_{i,j} \mid j = 1, 2, \ldots, n\}$ be the measurement data sequence of sender i, where $x_{i,j}$ is the average achievable bandwidth between the $j-1^{th}$ and j^{th} second after the measurement started. Then the mean, μ_i, and the variance, σ_i^2, of the achievable bandwidth of sender i can be computed from

$$\mu_i = \frac{1}{n}\sum_{j=1}^{n} x_{i,j} \tag{6}$$

$$\sigma_i^2 = \frac{1}{n-1}\sum_{j=1}^{n}(x_{i,j} - \mu_i)^2 \tag{7}$$

Table 2 summarizes the minimum, maximum, median, and mean per-flow achievable bandwidth in the two measurement datasets. The achievable bandwidth in both cases vary across a very wide range, e.g., from a minimum of 0.21 Kbps up to a maximum of over 12 Mbps for the FTP dataset. The PlanetLab dataset has a lower maximum achievable bandwidth because high-bandwidth senders are excluded during the pre-measurement phase as described in Section 3.1. Note also that although the per-flow bandwidth threshold was set to 1.6 Mbps, some flows nonetheless end up with achievable bandwidth much higher than that (e.g., over 4 Mbps). This shows that some data flows exhibit substantial variations in their achievable bandwidth during the 2-hour measurement period.

Fig. 1 compares the histogram of per-flow achievable bandwidth for the two datasets. The distribution for the PlanetLab dataset has a peak around 200 Kbps, follows by a broad distribution, and finally drops rapidly beyond 1.6 Mbps (due to pre-measurement filtering). By contrast the FTP dataset exhibits a sharp peak at around 1.5Mbps which is significantly higher than the rest of the spectrum.

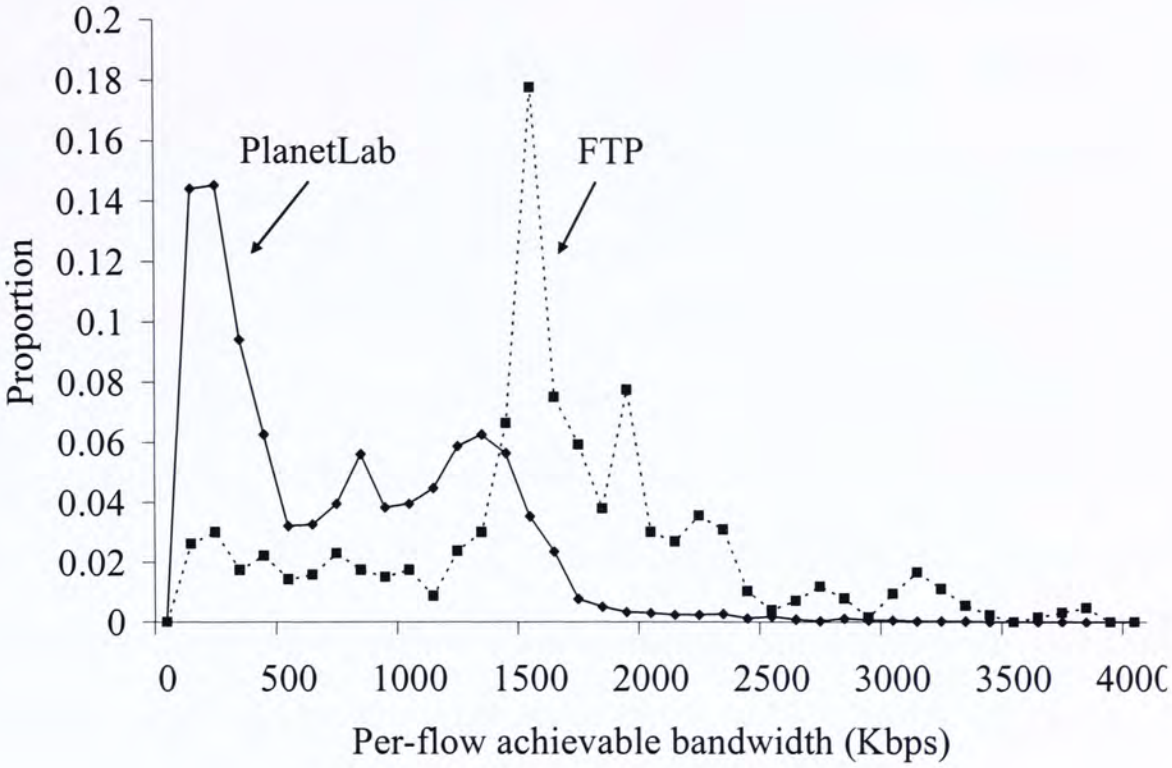

Fig. 1. Comparison of the histogram of per-flow achievable bandwidth of PlanetLab and FTP flows.

4.2 Inter-Flow Correlation

In a many-to-one data flow all the senders transmit data to the same receiver. Thus their network paths naturally converge to the same destination and hence could share some network links along the way. If two senders share the same network bottleneck then their flow-level properties will become correlated [19]. To investigate this phenomenon we compute the correlation coefficient, denoted by ρ, of different pairs of sender nodes in both datasets. Let X_i and X_j be the bandwidth sequences of two senders i and j in the same many-to-one data flow. The correlation coefficient of these two senders is given by

$$\rho(X_i, X_j) = \frac{\mathrm{cov}(X_i, X_j)}{\sigma_i \sigma_j} \tag{8}$$

Equation (8) measures the degree of correlation between the achievable bandwidth of two senders in the same many-to-one data flow. The correlation coefficient can range from -1 to 1, with values closer to either -1 or 1 representing stronger correlations which indicate that they may share a common network bottleneck.

Fig. 2 plots the distribution of the correlation coefficients for the two datasets. Using the threshold of 0.28 according to the study by Wang *et al.* [20] we observe that a large proportion of the sender pairs acquire a value below it, e.g., 85% and 90% for the PlanetLab and FTP datasets respectively. The rest of the sender pairs are likely to share a common network bottleneck, thus leading to correlated variations in their achievable bandwidths. We will return to this issue in Section 5.

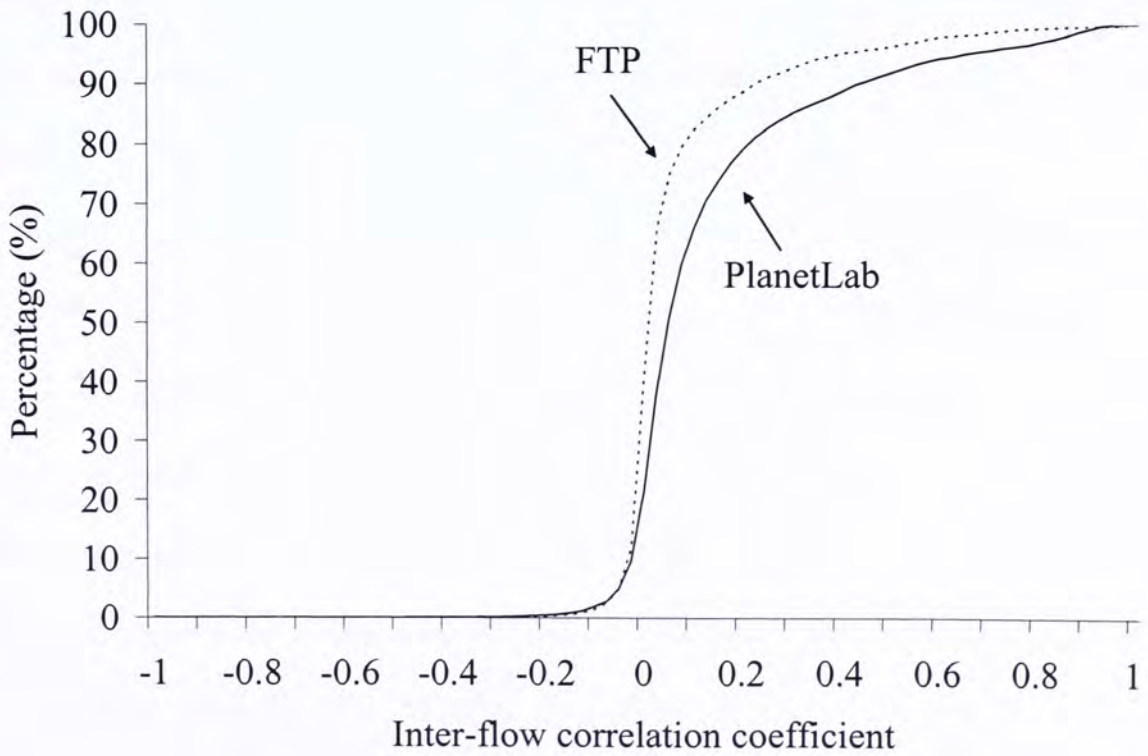

Fig. 2. The cumulative distribution of the inter-flow correlation coefficient of PlanetLab nodes and FTP servers.

4.3 Intra-Flow Temporal Correlation

To further study the correlation of achievable bandwidth across different times within the same data flow, we compute the normalized temporal correlation of a data flow i with a temporal distance of k measurement samples as follows:

$$R_i(k) = \left| \frac{\mathrm{cov}(X_i, X_i^k)}{\sigma_i^2} \right| \tag{9}$$

where cov(·) computes the covariance of two sequences, X_i is the sequence of measurement samples for achievable bandwidth, and X_i^k is the corresponding measurement sequence with a temporal distance of k samples, i.e., $X_i^k = \{x_{i,j} \mid i = 0+k, 1+k, \ldots, n+k\}$. The value can range from 0 to 1 where 0 implies the two sequences are uncorrelated and 1 implies that they are completely correlated. By varying the temporal distance k we can evaluate the temporal correlation over different time scales.

Fig. 3 plots the distribution of the normalized temporal correlation of PlanetLab data flows for k=1, 10, 50, and 100 respectively. Note that each measurement sample is a one-second average so the temporal distance is also equivalent to unit of seconds. The same distribution for FTP flows is very similar and is thus omitted. The key observation here is the rapid decrease in the temporal correlation for temporal distances larger than 1. For example, when k=1, i.e., the correlation of adjacent measurement samples, a substantial proportion of flows (over 84%) exhibits correlations above 0.2. By contrast, when we increase k to 10 the proportion drops to 42%, and it further reduces to 20% when k is further increased to 50. This shows that the achievable bandwidth is correlated only at very short time scales.

Next we consider the intra-flow temporal correlation of the aggregate achievable bandwidth in many-to-one data flows. Using a fixed temporal distance of k = 10 seconds and k = 100 seconds, we plot in Fig. 4 the average intra-flow temporal correlation for many-to-one data flows of 1 to 10 senders. The results show a gradual increase in temporal correlation along with more senders in the aggregate flow. However for larger temporal distance (e.g., k = 100 seconds) the temporal correlation remains insignificant.

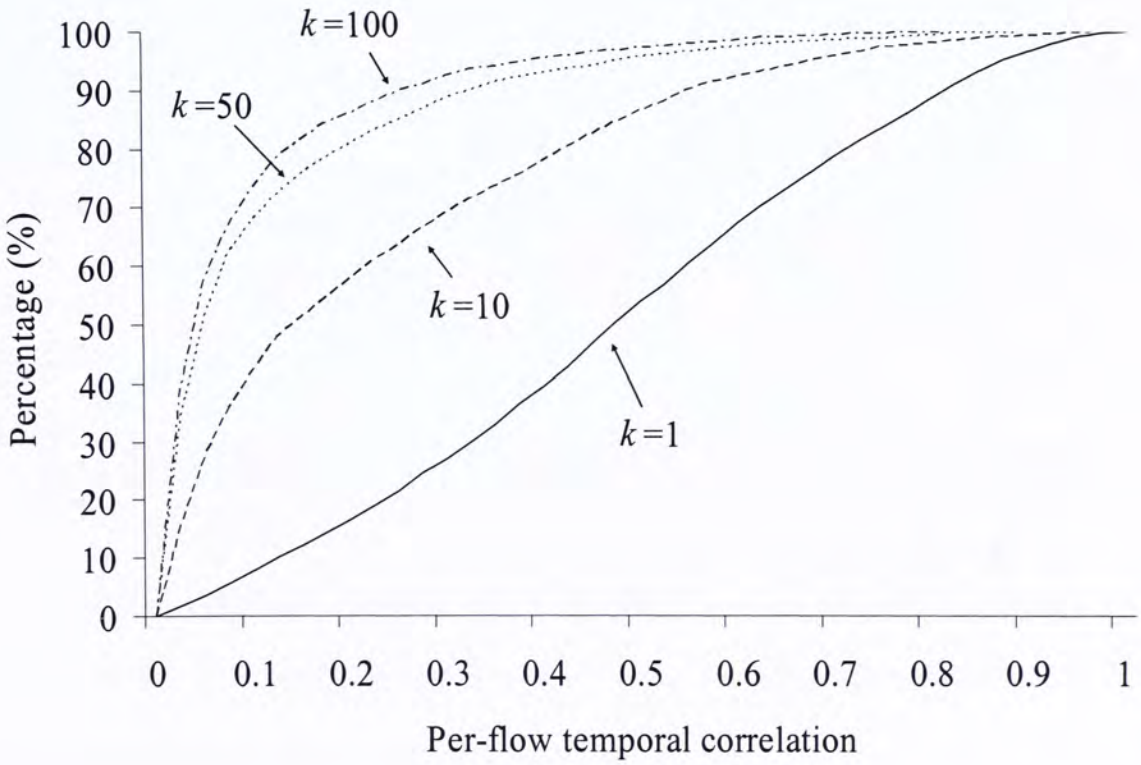

Fig. 3. The cumulative distribution of the per-flow temporal correlation for temporal distances of 1, 10, 50, and 100 seconds (PlanetLab dataset).

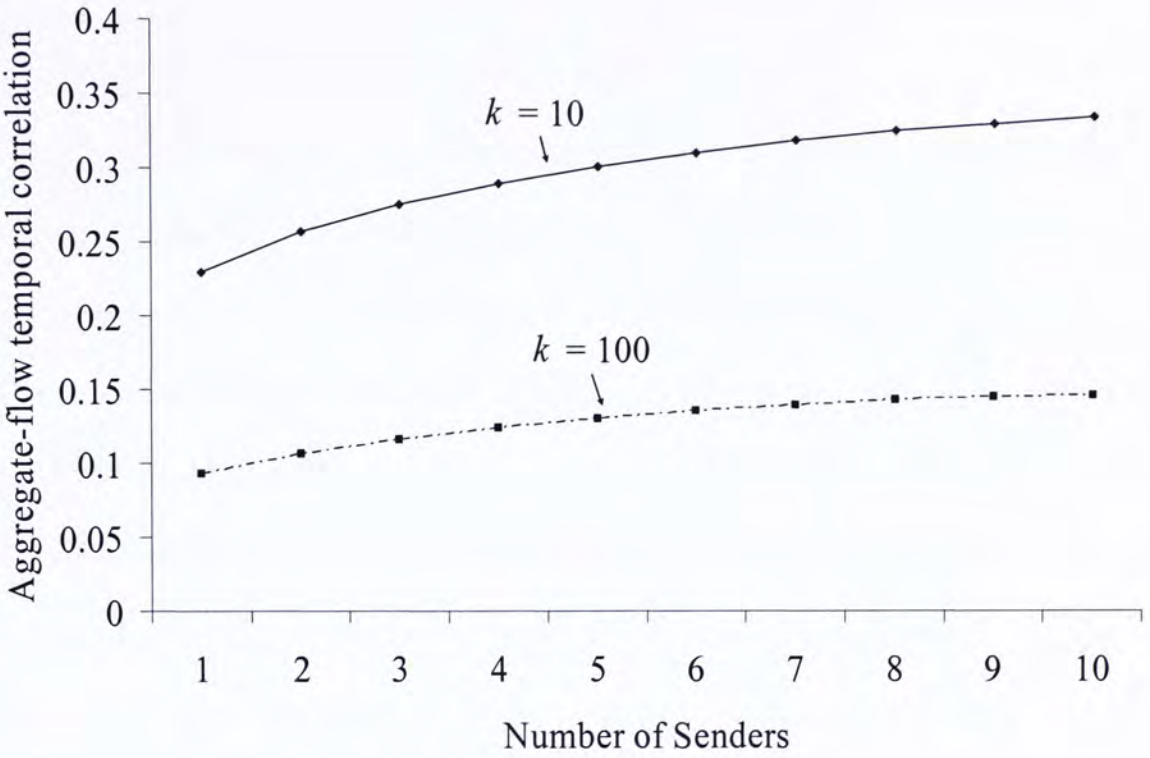

Fig. 4. Comparison of aggregate-flow temporal correlation for 1 to 10 senders (PlanetLab flows with temporal distances of 10 and 100 seconds).

4.4 Intra-Flow Bandwidth Variation

We further analyze in this section the bandwidth variations within the same data flow. This intra-flow variation can be quantified by the coefficient-of-variation (CoV) of a data flow's achievable bandwidth, defined as:

$$CoV_i = \frac{\sigma_i}{\mu_i} \tag{10}$$

for flow i, which is the standard deviation normalized by the mean. Since CoV is normalized we can compare the intra-flow variations of different data flows in Fig. 5, which plots the distribution of CoV for data flows in the two datasets. The results show that the FTP dataset, while spanning a wider range of achievable bandwidth across different flows than the PlanetLab dataset, had lower variations *within* the same data

flow than the PlanetLab dataset. For example, 70% of FTP flows have CoV less than 0.2 compare to only 30% in the PlanetLab dataset.

We conjecture that the FTP dataset's much lower intra-flow variation is due to rate-limiting imposed by some of the FTP servers. Specifically, many FTP servers have the option to set a limit on the maximum transmission rate for each connection as a mean to control resource allocation. If such a rate limit is substantially lower than the achievable bandwidth of the network path from the FTP server to the receiver, then the achievable bandwidth will be limited by the sender's transmission rate rather than the network bandwidth availability. In other words, variations in network bandwidth availability will have significantly less effect on the achievable bandwidth and thus resulting in lower intra-flow variation.

To further verify the conjecture we captured detailed packet traces at the receiver using the tcpdump tool [18]. If the achievable bandwidth of a data flow is rate-limited by the sender, then we would expect the TCP flow to experience little to no packet loss as the transmission rate is not sufficiently high to induce network congestion. By contrast, if the achievable bandwidth is congestion-limited then we would expect higher levels of packet loss due to frequent network congestions. From the packet traces we can extract and count the TCP retransmission events (by comparing sequence numbers in the TCP header) to deduce the packet loss rate for this purpose. Fig. 6 plots the distribution of packet loss rate for the FTP dataset. It is evident that the majority of the data flows has very low packet loss rate, e.g., approximately 60% of the data flows has packet loss rate lower than 0.05%. This strongly suggests that a sizable proportion of the FTP servers implemented rate-limiting which in turn affect properties of the end-to-end achievable bandwidth.

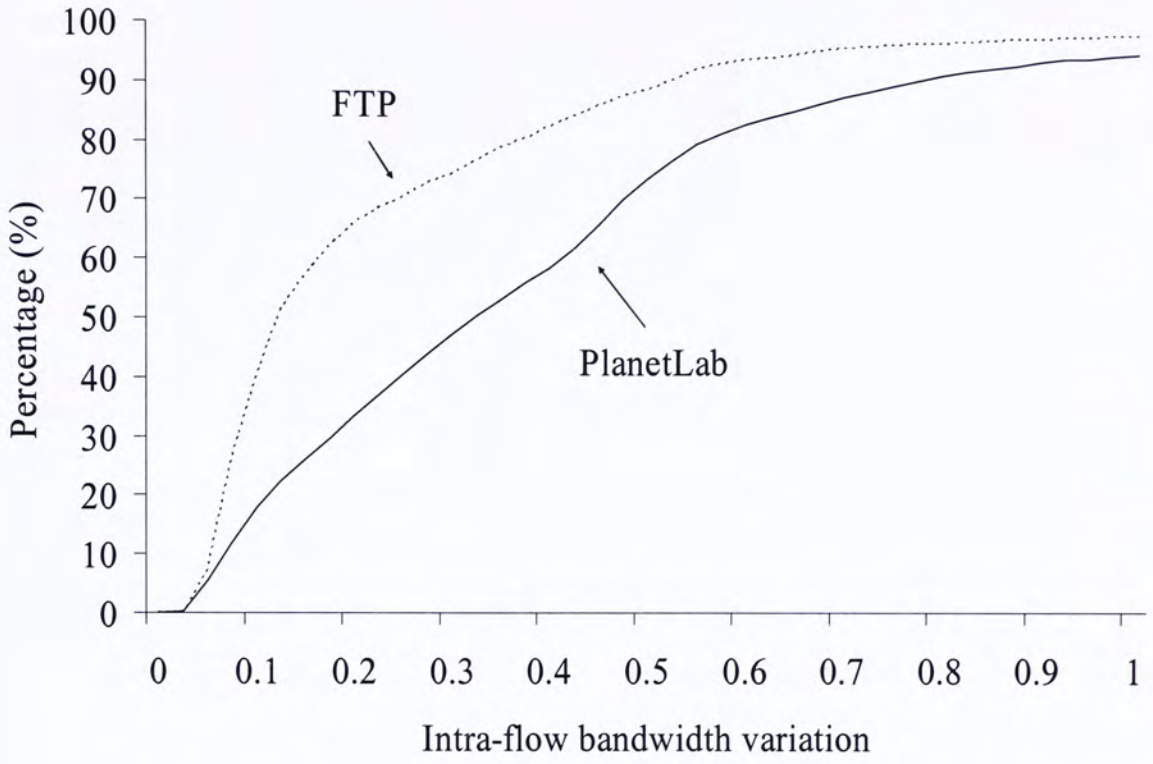

Fig. 5. The cumulative distribution of the intra-flow bandwidth variation in PlanetLab and FTP flows.

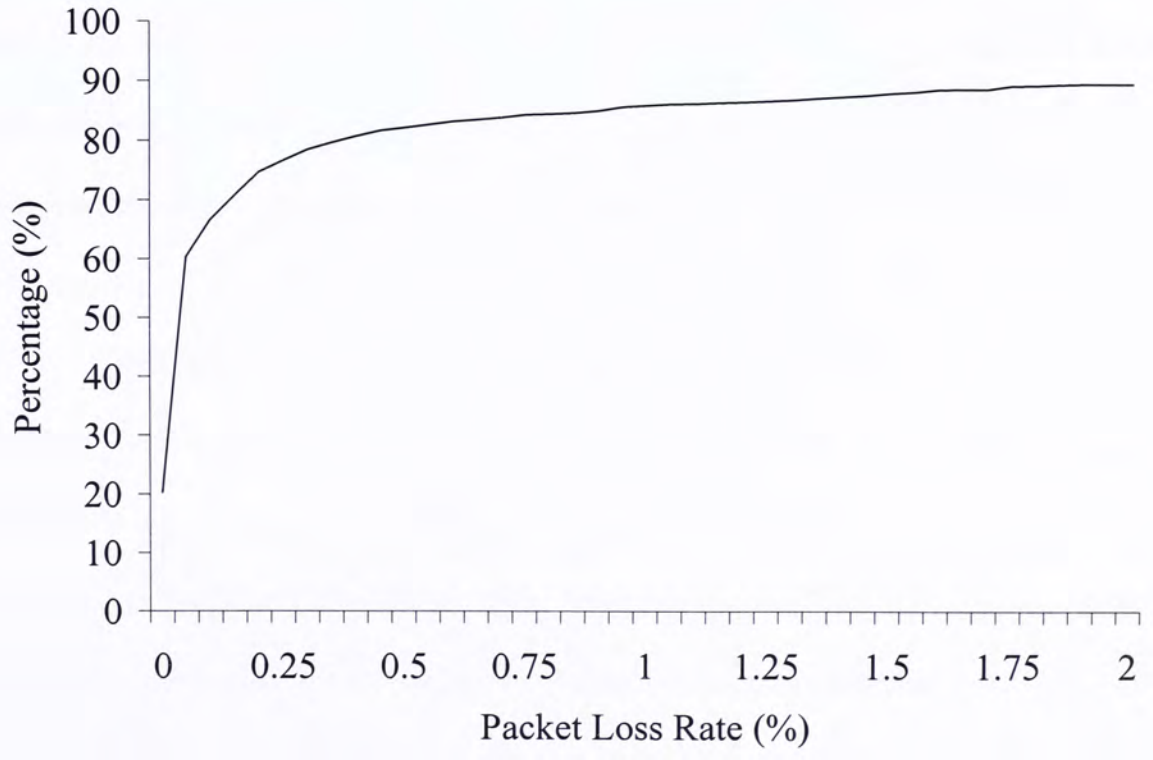

Fig. 6. The cumulative distribution of the packet loss rate of FTP flows.

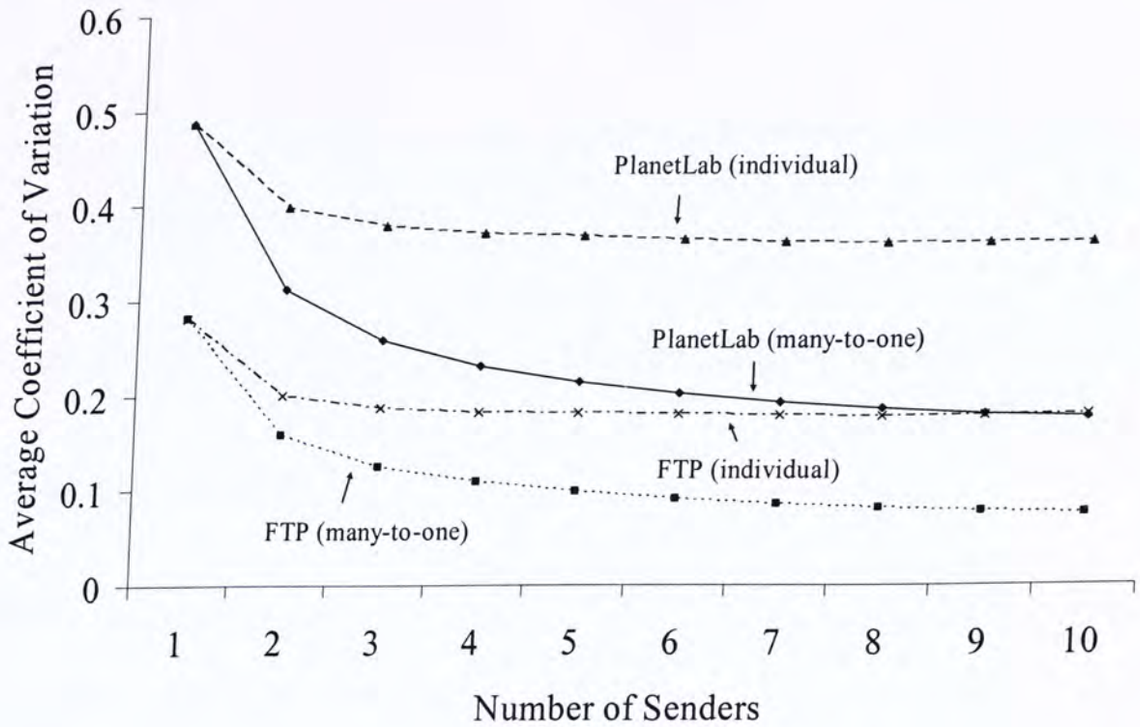

Fig. 7. Comparison of the average CoV for 1 to 10 senders.

Next we consider the CoV for the aggregate achievable bandwidth in many-to-one data flows. Fig. 7 compares the CoV of individual flows and many-to-one data flows with number of senders ranging from 1 to 10 for the two datasets. The former is computed from the weighted average of the per-flow CoV's of all individual flows, with the weight equal to the flow's mean achievable bandwidth. This corresponds to the case where the bandwidth resources of all senders are uniformly utilized by the receivers (see Section 5.1). The latter is computed from the CoV of the aggregate achievable bandwidth of all senders in the many-to-one data flow.

There are two distinctive observations from the results. First, the aggregate flows always have lower bandwidth variations when compared to the individual weighted average of the same number of sources (for more than one flow). Second, there is a clear trend of decreasing CoV for larger number of senders in many-to-one data flows. This implies that the aggregate achievable bandwidth generally exhibits less intra-flow

variations when the number of senders increases. By contrast the individual weighted average CoV levels off rapidly for two or more sources. In the next section we investigate whether these properties will result in increased predictability of aggregate data flows' bandwidth properties.

4.5 Predictability of Bandwidth Properties

One of the goals of bandwidth measurements is to enable the prediction of future bandwidth availability based on past measurement results. To investigate this issue we consider the simplest method to predict future achievable bandwidth – predicts the future achievable bandwidth to have the same mean as in the past. Specifically, the receiver measures the average bandwidth during the initial measurement period of T seconds:

$$\hat{\mu}_i = \frac{1}{T}\sum_{j=1}^{T} x_{i,j} \tag{11}$$

and then simply predicts the future achievable bandwidth to be the same.

Obviously the prediction will not be completely accurate. To quantify the prediction errors we compute the normalized deviation from

$$d_i = \frac{\sqrt{E[(X'_i - \hat{\mu}_i)^2]}}{\mu_i} \tag{12}$$

where $X'_i = \{x_{i,j} \mid j = T+1, T+2, \ldots, n\}$ is the actual achievable bandwidth of data flow i.

Similar to the well-known standard deviation in statistics, the normalized deviation d_i measures how far the future achievable bandwidth deviates from the one measured

during the initial measurement period. A smaller value implies less deviation, meaning that the future achievable bandwidth is more predictable.

We first consider the normalized prediction for individual data flows. Fig. 8 plots the distribution of the normalized deviation with $T = 500$ seconds for the two datasets. The results show that the mean achievable bandwidth of FTP flows is substantially more predictable than PlanetLab flows. This agrees with the observation in Section 4.3 (c.f. Fig. 5) where FTP flows exhibit less intra-flow bandwidth variations. In fact we found that the normalized deviation of a data flow is highly correlated with its intra-flow bandwidth variation (measured by CoV), with a correlation index of 0.429 and 0.617 for the PlanetLab and FTP flows respectively.

Next we consider the normalized deviation for the aggregate achievable bandwidth in many-to-one data flows. Fig. 9 plots the mean normalized deviation for individual flows and many-to-one data flows of 1 to 10 senders. The former is computed from the weighted average of individual flow's normalized deviation, with the weight equal to the individual flow's mean achievable bandwidth. The latter is computed from the aggregate achievable bandwidth of all individual flows in the many-to-one flow. Similar to Section 4.4, we observe that with the same number of sources, predicting the properties of many-to-one data flows are always more accurate than the case for individual flows. Moreover, the prediction error consistently decreases with more number of senders in the many-to-one data flow.

To illustrate the effect we compare in Fig. 10 the normalized mean achievable bandwidth of a 1-sender data flow and a 10-sender data flow over time for the two datasets. It is evident that the 10-sender data flow exhibit significantly less bandwidth fluctuations over time than the 1-sender data flow. This is a very useful property because if future bandwidth can be predicted with good accuracy then the performance

of bandwidth-sensitive applications could be significantly improved. We will explore this further in Section 4.6.

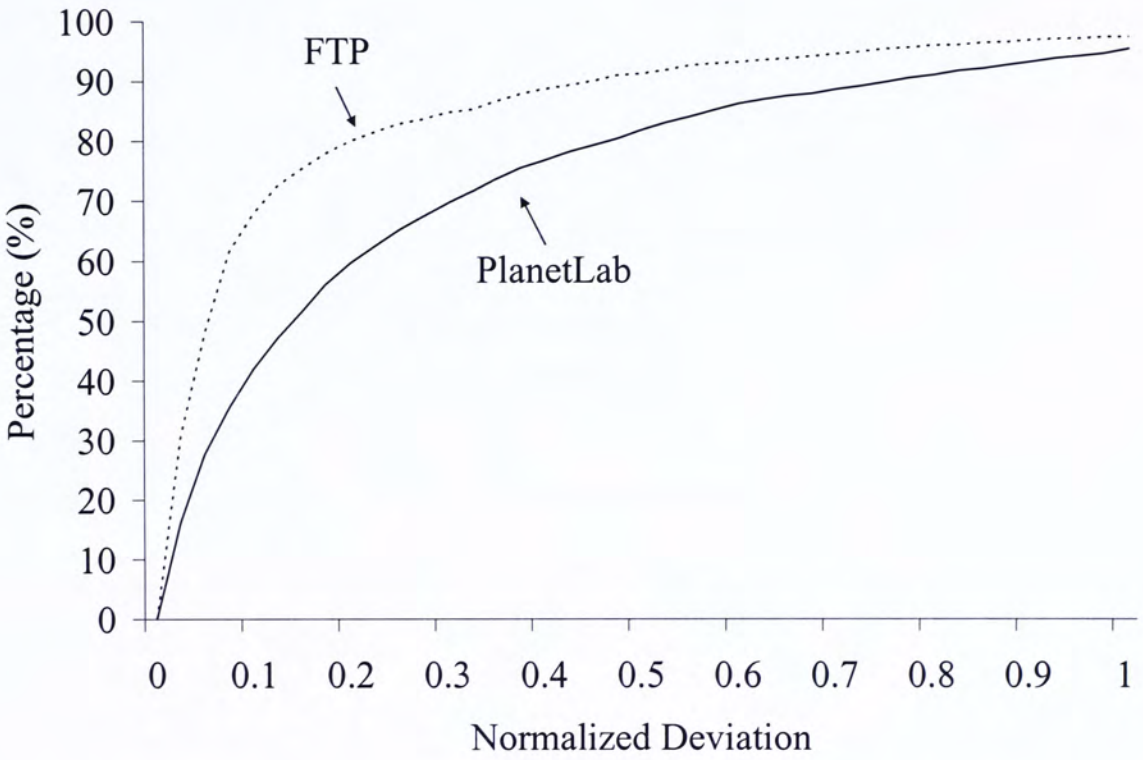

Fig. 8. The cumulative distribution of the normalized deviation of PlanetLab and FTP flows ($T = 500$).

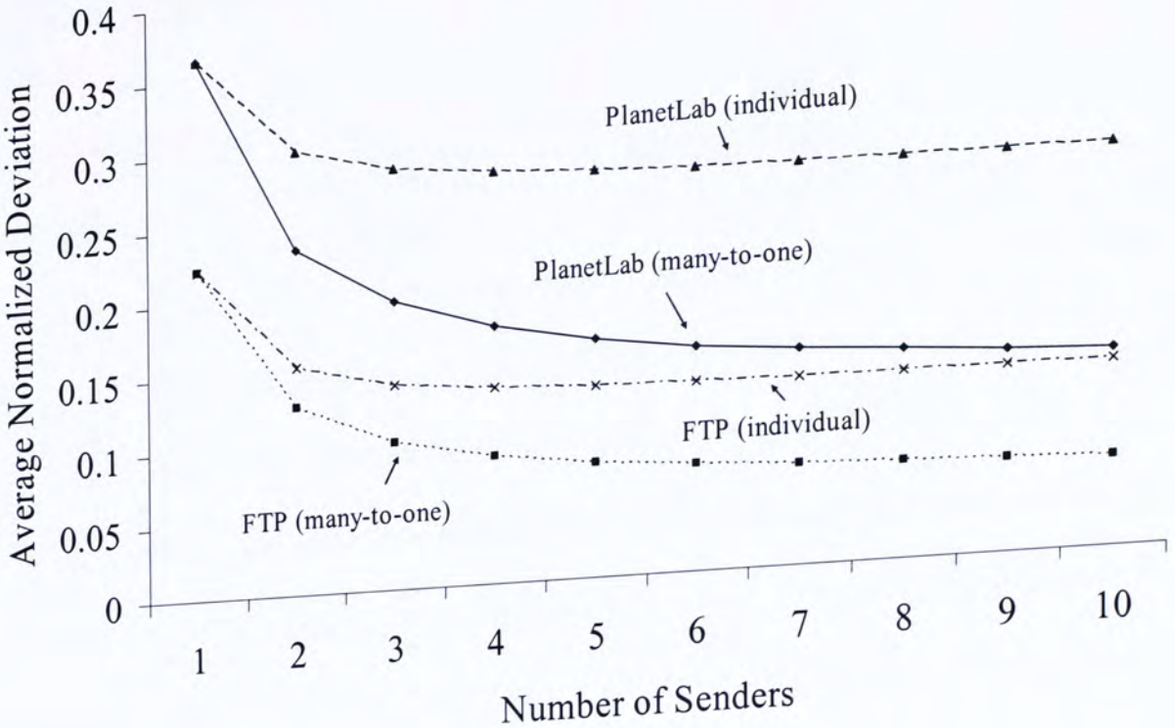

Fig. 9. Comparison of normalized deviation for 1 to 10 senders.

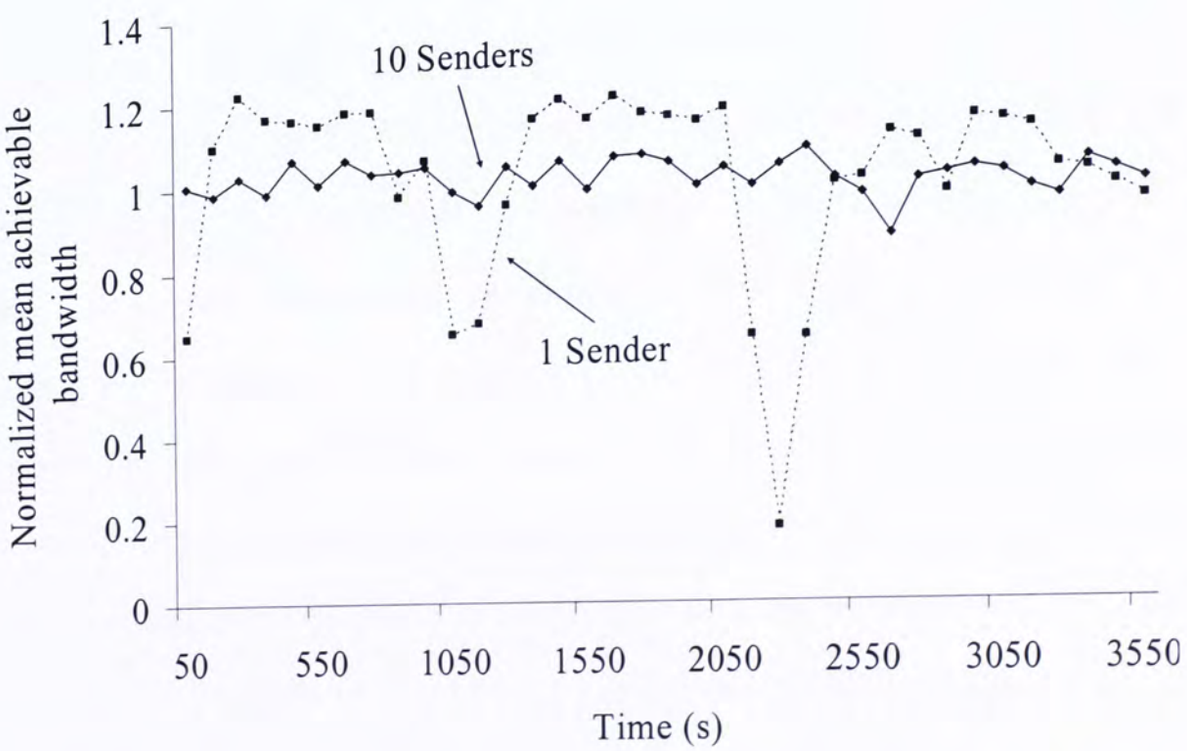

Fig. 10. Comparison of normalized mean achievable bandwidth for 1-sender and 10-sender data flows.

4.6 Long-term Flow Properties

The previous sections focus on the properties of a given many-to-one data flow, which last for 2 hours in the measurements. As we have accumulated measurement data for 12 months, we could also study the properties of the same sender over a longer time scale. This will shed light on whether long-term historical bandwidth data are useful in the estimation of future bandwidth availability.

To analyze the long-term flow properties we first grouped all the measurement data over the 12-month measurement period according to the identity of the sender. For each sender, say sender i, we compute the mean of the achievable bandwidth of measurement run j as $\mu_{i,j}$. Next we compute for each sender the CoV of all the per-run means $\{\mu_{i,j} \mid \forall j\}$, denoted by Ω_i. This per-sender CoV measures the variations of the mean achievable bandwidth of a particular sender over the 12-month period.

Fig. 11 plots the cumulative distribution of the per-sender CoV for the two datasets. We observe that FTP senders' properties are far more consistent over the long time scale. Two factors likely contribute to this result. First, as shown earlier in Section 4.4 some FTP servers implemented per-connection rate-limiting. As a result the achievable throughput for flows originating from these FTP servers will be more stationary. Second, in the PlanetLab measurements the receiver node is randomly selected from the pool of 284 hosts in each measurement run while the same receiver host is used throughout all the FTP measurements. As the network path will likely be different for different receivers even from the same sender, this will result in more variations in the sender's bandwidth properties in the PlanetLab dataset.

More importantly, we want to know whether historical information such as mean achievable bandwidth of a sender in the past could help in the selection of senders for

new data transfer sessions. To answer this question we can compare with the trivial alternative – assume no historical information is known and simply randomly pick a sender for each new data transfer session. Using this zero-knowledge approach the CoV of the mean achievable bandwidth are 0.82 and 0.96 for the PlanetLab and FTP datasets respectively. In comparison, if the historical CoV of all senders are known, then we can simply select the sender with the lowest CoV, and in both datasets around 80% of the senders have CoV values lower than the zero-knowledge case. This suggests that historical information of senders' bandwidth, even over a time scale of 12 months, could still be useful in estimating the future bandwidth availability.

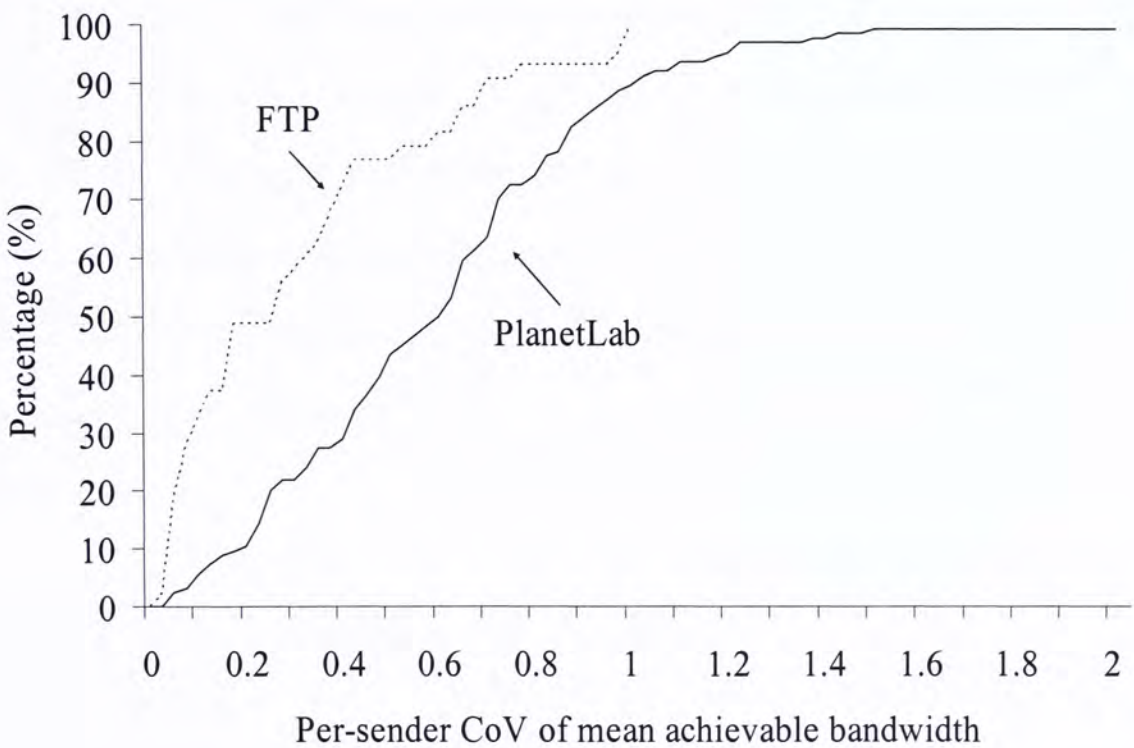

Fig. 11. The cumulative distribution of the per-sender CoV of mean achievable bandwidth over the 12-month period.

Chapter 5

A MATHEMATICAL FRAMEWORK

The measurement results reported in Section 4.4 and 4.5 strongly suggest that it is always better to transfer data using many-to-one data flows than using individual one-to-one data flows, as many-to-one data flows exhibit less intra-flow bandwidth variation (c.f. Fig. 7) and can be predicted more accurately (c.f. Fig. 9). Nevertheless empirical results are not proof per se and so in this section we develop a mathematical framework to formally establish these two invariant properties for general many-to-one data flows. More importantly, the framework provides deeper insights into the relations between properties of the individual flows and those of the aggregate flows which could be used to optimize the performance of many-to-one data flows.

5.1 Bandwidth Variations

Consider a pool of N senders, where $\mu_i > 0$ and $\sigma_i^2 > 0$ are the mean and variance of the achievable bandwidth of sender i. Assuming that the bandwidth resources of the pool of N senders are utilized in a uniform manner, then over the long run the bandwidth contribution of sender i, denoted by P_i, will be proportional to its mean achievable bandwidth:

$$P_i = \frac{\mu_i}{\sum_{j=1}^{N} \mu_j} \tag{13}$$

In other words, sender with more bandwidth will contribute proportionally more resources to serving the receivers and vice versa.

Now let Y be the aggregate achievable bandwidth of all N senders, i.e.,

$$\begin{aligned} Y &= \{y_j \mid j = 1,2,...,n\} \\ &= \{\sum_{i=1}^{N} x_{i,j} \mid j = 1,2,...n\} \end{aligned} \tag{14}$$

with the corresponding mean and the variance defined by

$$\mu_Y = \frac{1}{n}\sum_{j=1}^{n} y_j \tag{15}$$

and

$$\sigma_Y{}^2 = \frac{1}{n-1}\sum_{j=1}^{n} (y_j - \mu_Y)^2 \tag{16}$$

Then we can calculate the weighted average of the CoV of all N individual flows from

$$\begin{aligned} CoV^W &= \sum_{i=1}^{N} (CoV_i \cdot P_i) \\ &= \sum_{i=1}^{N} (\frac{\sigma_i}{\mu_i} \frac{\mu_i}{\sum_{j=1}^{N} \mu_j}) \\ &= \sum_{i=1}^{N} (\frac{\sigma_i}{\mu_i} \frac{\mu_i}{\mu_Y}) \\ &= \frac{1}{\mu_Y} \sum_{i=1}^{N} \sigma_i \end{aligned} \tag{17}$$

This represents the average performance for the case where receivers connect to just one sender at a time to transfer the requested data, i.e., using one-to-one data flow.

Similarly we can calculate the CoV of the aggregate flow for the case where the receivers always connect to all senders to transfer the requested data, i.e., using many-to-one data flow:

$$\begin{aligned} CoV^A &= \frac{\sigma_Y}{\mu_Y} \\ &= \frac{1}{\mu_Y}\sqrt{\sum_{p=1}^{N}\sum_{q=1}^{N}\sigma_p\sigma_q\rho(X_p,X_q)} \\ &= \frac{1}{\mu_Y}\sqrt{\sum_{i=1}^{N}\sigma_i^2+\sum_{p=1}^{N}\sum_{\substack{q=1\\p\neq q}}^{N}\sigma_p\sigma_q\rho(X_p,X_q)} \end{aligned} \tag{18}$$

With (17) and (18) we can then prove the first invariant property in Theorem 1 below.

Theorem 1 – *The weighted CoV of N individual data flows as in (17) is always larger than the CoV of the aggregate data flow formed from the same N data flows as in (18), assuming the individual flows are not perfectly and positively correlated.*

Proof:

Two flows p and q are perfectly and positively correlated if and only if their correlation coefficient $\rho(X_p, X_q) = 1$. Therefore we need to show that (18) ≤ (17) in general given that $\rho(X_p, X_q) < 1$ $\forall p,q$ where $p \neq q$.

From (18) we have

$$\begin{aligned} CoV^A &= \frac{1}{\mu_Y}\sqrt{\sum_{i=1}^{N}\sigma_i^2+\sum_{p=1}^{N}\sum_{\substack{q=1\\p\neq q}}^{N}\sigma_p\sigma_q\rho(X_p,X_q)} \\ &< \frac{1}{\mu_Y}\sqrt{\sum_{i=1}^{N}\sigma_i^2+\sum_{p=1}^{N}\sum_{\substack{q=1\\p\neq q}}^{N}\sigma_p\sigma_q} \qquad \because \rho(X_p,X_q)<1 \end{aligned} \tag{19}$$

Factoring the terms inside the square root we then obtain the desired result:

$$CoV^A < \frac{1}{\mu_Y}\sqrt{(\sum_{i=1}^{N}\sigma_i)^2} = \frac{1}{\mu_Y}\sum_{i=1}^{N}\sigma_i = CoV^W \quad (20)$$

■

Note that if two flows are perfectly and positively correlated it means that their achievable bandwidth varies in perfect synchrony. In practice this is clearly extremely improbable, if not impossible. Therefore Theorem 1 establishes that one can always reduce the intra-flow bandwidth variation by using many-to-one data flows instead of individual one-to-one data flows.

5.2 Bandwidth Predictability

The second invariant property is the bandwidth predictability reported in Section 4.5. First, consider the weighted average of the normalized deviation of all N individual flows:

$$ND^W = \sum_{i=1}^{N}(d_i \cdot P_i) = \frac{1}{\mu_Y}\sum_{i=1}^{N}\sqrt{E[(X'_i - \hat{\mu}_i)^2]} \quad (21)$$

Next we compute the corresponding normalized deviation for the aggregate flow from

$$ND^A = d_Y = \frac{1}{\mu_Y}\sqrt{\sum_{i=1}^{N}E[(X_i' - \hat{\mu}_i)^2] + \sum_{p=1}^{N}\sum_{\substack{q=1 \\ p\neq q}}^{N}E[(X_p' - \hat{\mu}_p)(X_q' - \hat{\mu}_q)]} \quad (22)$$

With (21) and (22) we can then prove the second invariant property in Theorem 2 below.

Theorem 2 – *The weighted normalized deviation of N individual data flows as in (21) is always larger than the normalized deviation of the aggregate data flow formed from the same N data flows as in (22), assuming the individual flows are not perfectly and positively correlated.*

Proof:

Recall that $\hat{\mu}_i = \frac{1}{T}\sum_{j=1}^{T} x_{i,j}$ and $X'_i = \{x_{i,j} \mid j = T+1, T+2, \ldots, n\}$.

Let e_i be the mean estimation error of sender i, where

$$e_i = \mu'_i - \hat{\mu}_i \tag{23}$$

and v_i^2 be the prediction variance of sender i, where

$$\begin{aligned} v_i^2 &= E[(X'_i - \hat{\mu}_i)^2] \\ &= \sigma'^2_i + e_i^2 \end{aligned} \tag{24}$$

and

$$\sigma'^2_i = \frac{1}{n-T-1}\sum_{j=T+1}^{n} (x_{i,j} - \mu'_i)^2 \tag{25}$$

Before proving Theorem 2, we first show that

$$E[(X'_i - \hat{\mu}_i)(X'_j - \hat{\mu}_j)] \le v_i v_j \tag{26}$$

and the equality holds only if X'_i and X'_j are perfectly and positively correlated, i.e., $\rho(X'_i, X'_j) = 1$.

As we know that

$$\begin{aligned} & E[((\frac{X'_i}{v_i} - \frac{X'_j}{v_j}) - (\frac{\hat{\mu}_i}{v_i} - \frac{\hat{\mu}_j}{v_j}))^2] \ge 0 \\ \Rightarrow\ & 2 - \frac{2}{v_i v_j} E[(X'_i - \hat{\mu}_i)(X'_j - \hat{\mu}_j)] \ge 0 \\ \Rightarrow\ & E[(X'_i - \hat{\mu}_i)(X'_j - \hat{\mu}_j)] \le v_i v_j \end{aligned}$$

When the equality holds,

$$
\begin{aligned}
&E[(X'_i-\hat{\mu}_i)(X'_j-\hat{\mu}_j)]=v_iv_j\\
&\Leftrightarrow E[(X'_i-\mu'_i+e_i)(X'_j-\mu'_j+e_j)]=v_iv_j\\
&\Leftrightarrow E[(X'_i-\mu'_i)(X'_j-\mu'_j)+(X'_i-\mu'_i)e_i+(X'_j-\mu'_j)e_j+e_ie_j]=v_iv_j\\
&\Leftrightarrow \sigma'_i\sigma'_j\rho(X'_i,X'_j)+e_ie_j=v_iv_j
\end{aligned}
\tag{27}
$$

Consider different values of $\rho(X'_i, X'_j)$ and e_ie_j,

If $\rho(X'_i, X'_j) < 0$, then

$$
\begin{aligned}
\sigma'_i\sigma'_j\rho(X'_i,X'_j)+e_ie_j &< e_ie_j\\
&< \sqrt{\sigma'^2_i\sigma'^2_j+e_i^2e_j^2+\sigma'^2_ie_j^2+\sigma'^2_ie_j^2}\\
&= v_iv_j
\end{aligned}
\tag{28}
$$

If $e_ie_j < 0$, then

$$
\begin{aligned}
\sigma'_i\sigma'_j\rho(X'_i,X'_j)+e_ie_j &< \sigma'_i\sigma'_j\rho(X'_i,X'_j)\\
&\le \sigma'_i\sigma'_j\\
&< \sqrt{\sigma'^2_i\sigma'^2_j+e_i^2e_j^2+\sigma'^2_ie_j^2+\sigma'^2_ie_j^2}\\
&= v_iv_j
\end{aligned}
\tag{29}
$$

If $\rho(X'_i, X'_j) \ge 0$ and $e_ie_j \ge 0$, then

$$
\begin{aligned}
&\sigma'_i\sigma'_j\rho(X'_i,X'_j)+e_ie_j=v_iv_j\\
&\Rightarrow \sigma'^2_i\sigma'^2_j\rho(X'_i,X'_j)^2+2\sigma'_i\sigma'_j\rho(X'_i,X'_j)e_ie_j=\sigma'^2_i\sigma'^2_j+\sigma'^2_ie_j^2+\sigma'^2_ie_j^2\\
&\Rightarrow \rho(X'_i,X'_j)=1 \quad \textit{and} \quad \sigma'_ie_j=\sigma'_je_i\\
&\Rightarrow \rho(X'_i,X'_j)=1 \quad \textit{and} \quad \frac{e_i}{\sigma'_i}=\frac{e_j}{\sigma'_j}
\end{aligned}
\tag{30}
$$

Combining (28) – (30), we obtain the desired result where $E[(X'_i-\hat{\mu}_i)(X'_j-\hat{\mu}_j)]=v_iv_j$ only if $\rho(X'_i, X'_j) = 1$.

Returning to the proof for Theorem 2, from (22) we have

$$
\begin{aligned}
ND^A &= \frac{1}{\mu_Y}\sqrt{\sum_{i=1}^{N} E[(X'_i-\hat{\mu}_i)^2]+\sum_{p=1}^{N}\sum_{\substack{q=1\\p\neq q}}^{N} E[(X'_p-\hat{\mu}_p)(X'_q-\hat{\mu}_q)]} \\
&\leq \frac{1}{\mu_Y}\sqrt{\sum_{i=1}^{N} v_i^2+\sum_{p=1}^{N}\sum_{\substack{q=1\\p\neq q}}^{N} v_p v_q} \\
&= \frac{1}{\mu_Y}\sum_{i=1}^{N} v_i \\
&= \frac{1}{\mu_Y}\sum_{i=1}^{N}\sqrt{E[(X'_i-\hat{\mu}_i)^2]} \\
&= ND^W
\end{aligned}
\tag{31}
$$

and the equality holds if and only if $E[(X'_p-\hat{\mu}_p)(X'_q-\hat{\mu}_q)]=v_p v_q \quad \forall p,q$ where $p \neq q$. From (26), we know that $ND^A = ND^W$ only if flows p and q are perfectly and positively correlated, i.e., $\rho(X'_p, X'_q) = 1) \ \forall p,q$ where $p \neq q$.

■

Again, perfectly and positively correlated flows are extremely improbable in practice and so in practice the achievable bandwidth of many-to-one data flows is more predictable than individual one-to-one data flows.

5.3 Sensitivity Analysis

The previous mathematical framework not only establishes two invariant properties of many-to-one data flows, but also provides insights into the relations between properties of individual flows and of the resultant aggregate flow. In this section we explore sensitivity of the aggregate flow's CoV to the individual flow's properties.

We consider a many-to-one flow with two senders. Using (17) and (18) we can compute the reduction in bandwidth variations when switching from using individual one-to-one flows to using a many-to-one flow as follows:

$$\frac{CoV^W - CoV^A}{CoV^W} = 1 - \frac{\sqrt{\sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2\rho}}{\sigma_1 + \sigma_2} \tag{32}$$

where σ_1^2 and σ_2^2, are the variances of bandwidth of flow 1 and flow 2 respectively, and ρ is the correlation coefficient of the two flows.

Define $r = \sigma_2 / \sigma_1$ as the ratio of the two flows' standard deviation. We can then rewrite (32) as

$$\begin{aligned} \frac{CoV^W - CoV^A}{CoV^W} &= 1 - \frac{\sqrt{\sigma_1^2 + (r\sigma_1)^2 + 2\sigma_1(r\sigma_1)\rho}}{\sigma_1 + (r\sigma_1)} \\ &= 1 - \frac{\sqrt{1 + r^2 + 2\rho r}}{1 + r} \\ &= f(r, \rho) \end{aligned} \tag{33}$$

which is a function of r and ρ.

Assume the correlation coefficient between the two senders, ρ, is fixed and $\rho \neq 1$ (i.e., not completely positively correlated), then we can determine the rate of change of $f(r, \rho)$ with respect to r from

$$\frac{\partial f(r,\rho)}{\partial r} = \frac{(1-r)(1-\rho)}{(1+r)^2(1+r^2+2\rho r)^{\frac{1}{2}}} \tag{34}$$

Solving for $\frac{\partial f(r,\rho)}{\partial r} = 0$ we obtain

$$\begin{aligned} &(1-r)(1-\rho) = 0 \\ &\Rightarrow r = 1 \quad \because \rho \neq 1 \end{aligned} \tag{35}$$

Since $f(r, \rho) > 0$ for all $r < 1$, and $f(r, \rho) < 0$ for all $x > 1$, $r = 1$ is the maximum point for $f(r, \rho)$. Thus, the maximum reduction in bandwidth variation is achieved when the standard deviations of the two senders' bandwidth are equal, in which case the reduction is given by

$$f(1,\rho) = \frac{\sigma_1 + \sigma_2 - \sqrt{\sigma_1^2 + \sigma_2^2 + 2\sigma_1\sigma_2\rho}}{\sigma_1 + \sigma_2}$$
$$= 1 - \frac{\sqrt{2\sigma_1^2 + 2\sigma_1^2\rho}}{2\sigma_1} \qquad (36)$$
$$= 1 - \sqrt{\frac{1+\rho}{2}}$$

Fig 12 plots the percentage of reduction against the ratio of the two senders' standard deviation r with different values of ρ. It further shows that the reduction increases when the correlation between senders decreases.

Fig. 13 is a scatter plot showing the actual percentage reductions obtained from different combinations of senders in the PlanetLab dataset. Results for the FTP dataset are similar and are thus omitted. The upper bound is computed from (36). The CoV reduction generally increases with decreases in the correlation coefficient. Note that negative correlation is desirable as the flows vary in opposite directions (i.e., one increase and the other decrease), thus canceling out some of the variations.

Still the function $f(r, \rho)$ depends on two factors: r and ρ. Thus the question is which one is more significant? To answer this question, we assume, without loss of generality, that $\sigma_2 \geq \sigma_1$, i.e., $r \geq 1$. Define S_r and S_ρ as the reduction sensitivity of r and ρ respectively:

$$S_r = \frac{\partial f}{\partial r} = \frac{(1-r)(1-\rho)}{(1+r)^2(1+r^2+2\rho r)^{\frac{1}{2}}} \leq 0 \qquad (37)$$

$$S_\rho = \frac{\partial f}{\partial \rho} = \frac{-2r}{(1+r)(1+r^2+2\rho r)^{\frac{1}{2}}} \leq 0 \qquad (38)$$

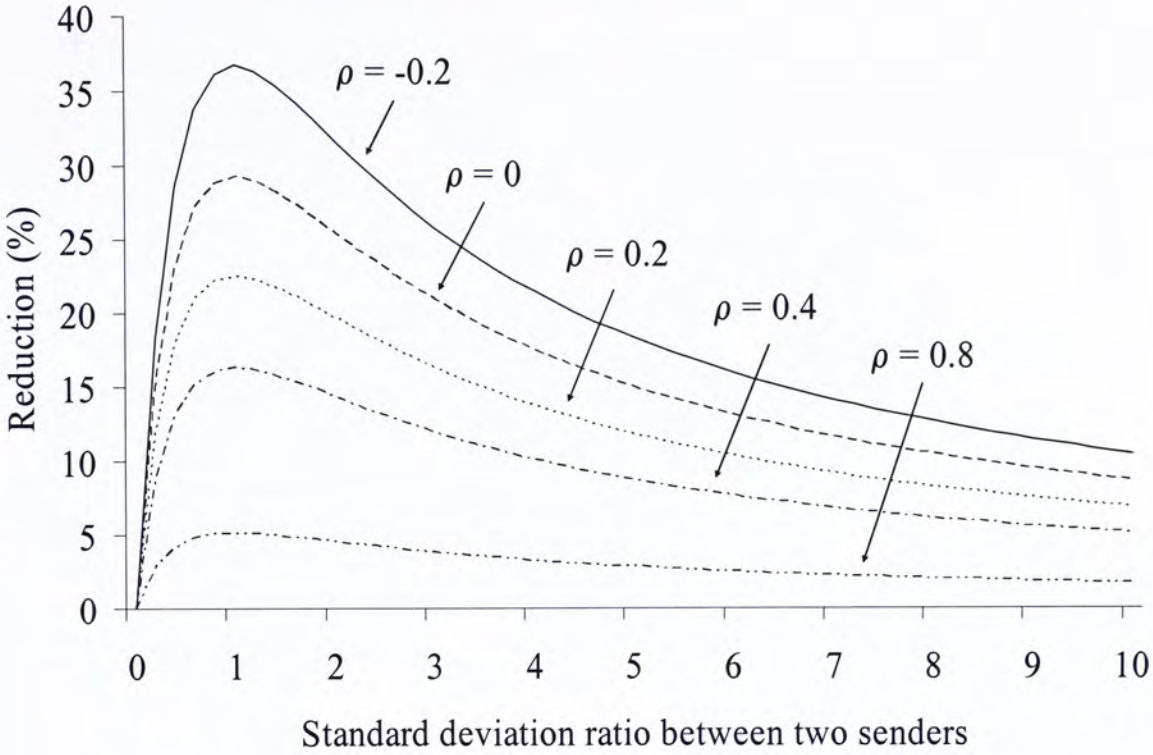

Fig. 12. Reduction in bandwidth variations versus the standard deviation ratio between two senders r.

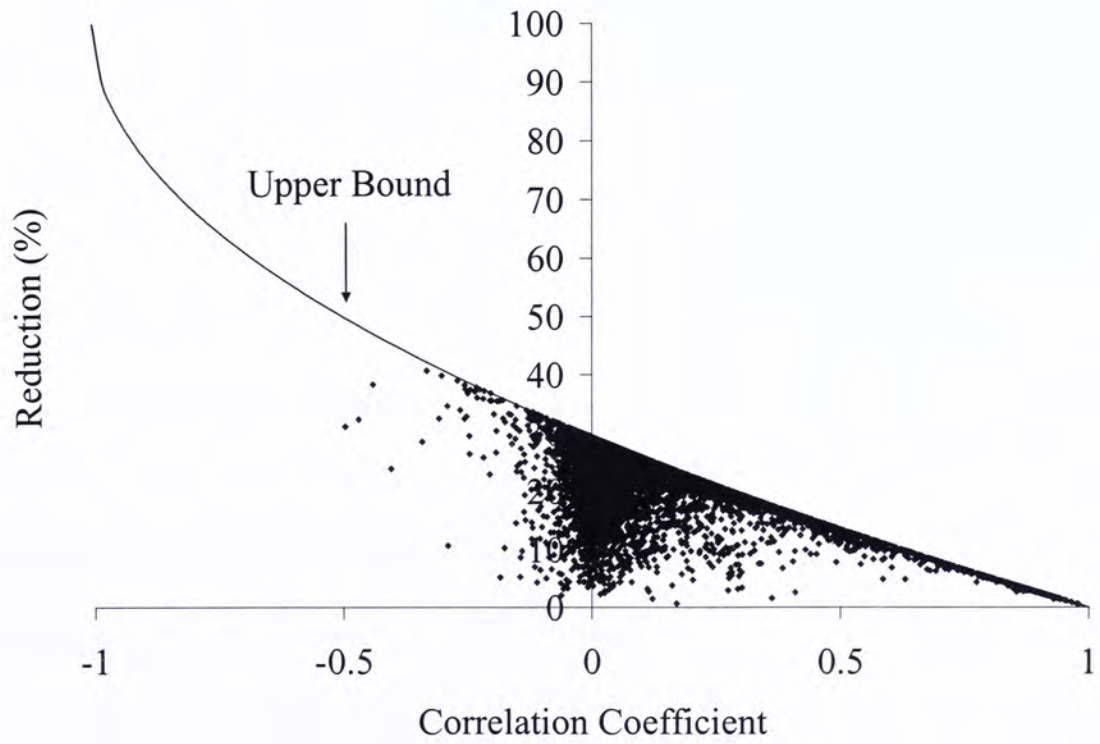

Fig. 13. Actual bandwidth variation reductions for different sender pairs in the PlanetLab dataset.

Now both S_r and S_ρ are negative meaning that the reduction increases when r or ρ decreases. To compare their relative significance we can compute the ratio of S_r and S_ρ from

$$\frac{S_r}{S_\rho} = \frac{(r-1)(1-\rho)}{2r(1+r)} \tag{39}$$

Define $g(r) = \frac{(r-1)}{r(1+r)}$, then

$$g'(r) = \frac{-r^2 + 2r + 1}{r^2(1+r)^2} \tag{40}$$

Solving for $g'(r) = 0$ we obtain

$$\begin{aligned} & r^2 - 2r - 1 = 0 \\ & \Rightarrow r = 1 \pm \sqrt{2} \end{aligned} \tag{41}$$

Now as $r \geq 1$, $r \neq 1-\sqrt{2}$ so $r = 1+\sqrt{2}$ is the maximum point of $g(r)$ since $g''(1+\sqrt{2}) < 0$. Thus,

$$\frac{S_r}{S_\rho} \leq \frac{\sqrt{2}(1-\rho)}{2(1+\sqrt{2})(2+\sqrt{2})} \approx 0.0858(1-\rho) \tag{42}$$

Since -1 $\leq \rho \leq$1, we have

$$0 \leq \frac{S_r}{S_\rho} \leq 0.172 \tag{43}$$

In other words the reduction sensitivity of ρ is at least 1/0.172 = 5.8 times that of r. If ρ = 0, i.e., the two flows are uncorrelated, then the sensitivity ratio is lower bounded by 11.655. Nevertheless this is still not the whole picture as the range of ρ is from -1 to 1, but the range of r is from 1 to ∞.

Thus to compare their effects we also need to consider the actual magnitudes of r. Fig. 14 and 15 plot the histogram of ρ and r respectively in the PlanetLab dataset. It is clear that the parameter r can indeed assume much larger values than ρ

(approximately by one order of magnitude). Similar results are also observed in the FTP dataset. Therefore we conclude that neither factor dominates the other one, and so one need to consider both factors to optimize the reduction in bandwidth variations.

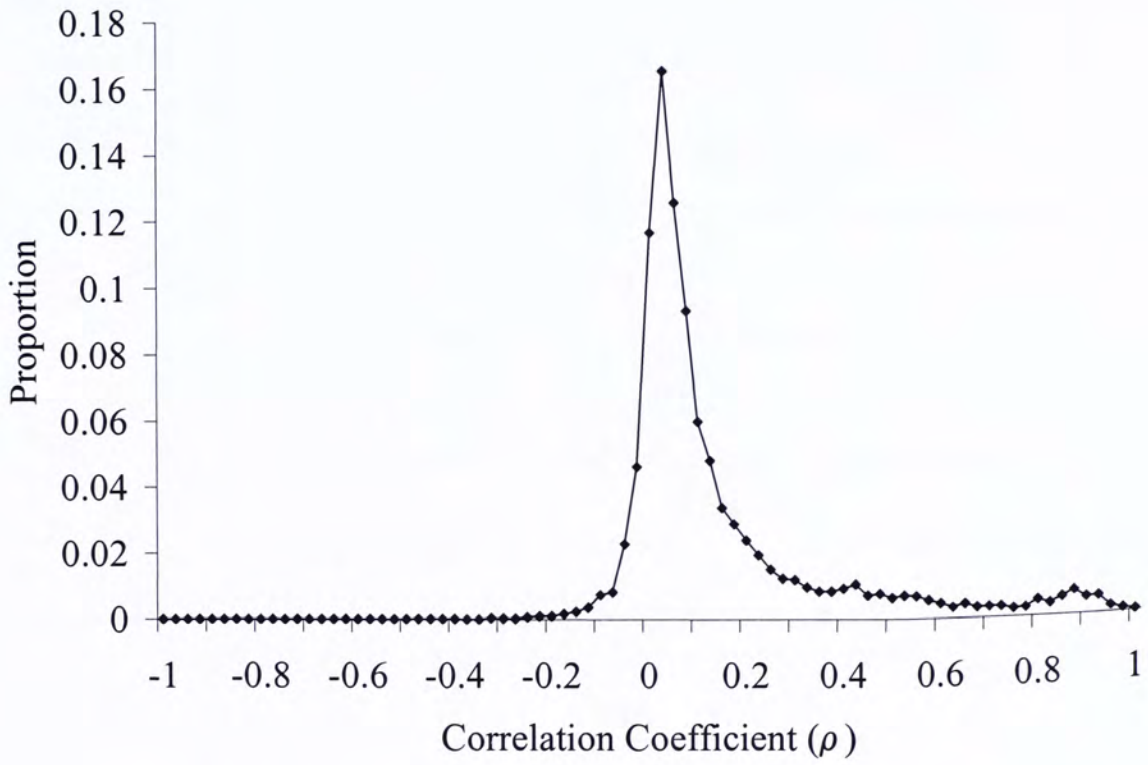

Fig. 14. The histogram of the correlation coefficient in the PlanetLab dataset.

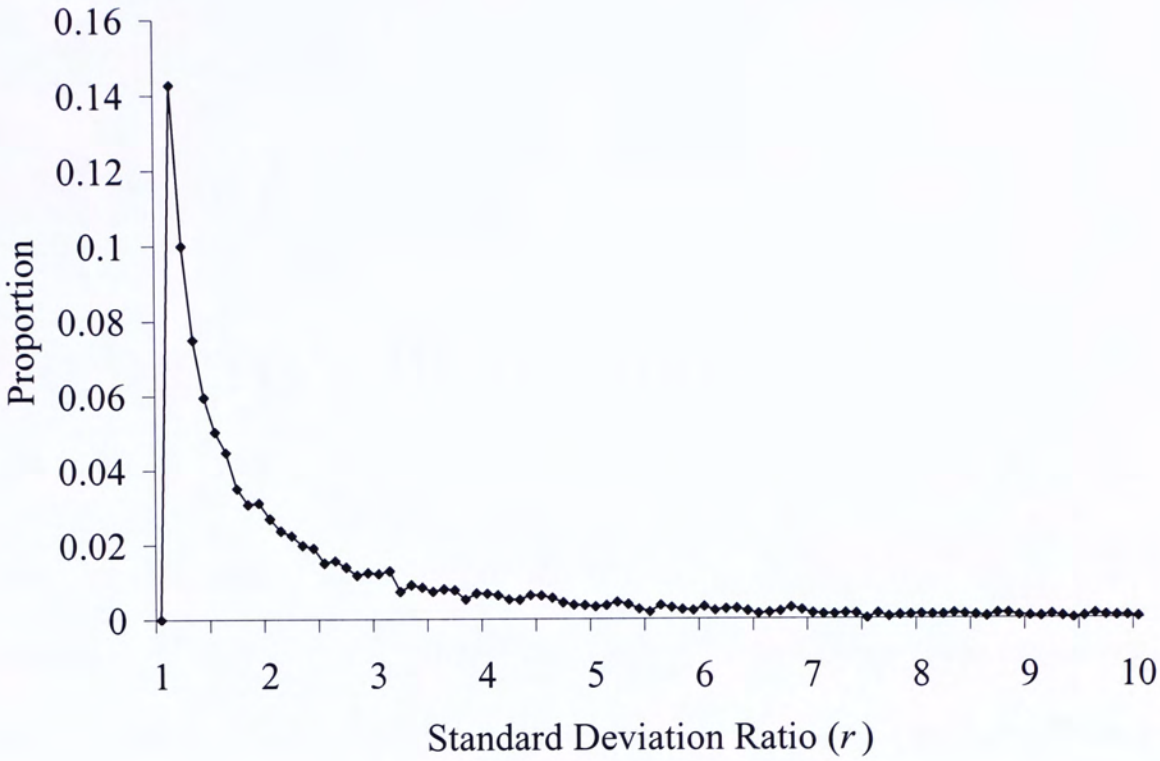

Fig. 15. The histogram of the standard deviation ratio in the PlanetLab dataset.

Chapter 6

PREDICTIVE BUFFERING ALGORITHM

The current Internet does not provide any end-to-end quality-of-service (QoS) control and thus presents a significant challenge to bandwidth-sensitive applications such as streaming video and TV contents over the Internet. The fluctuations in bandwidth availability can easily lead to frequent video playback interruptions that are extremely annoying to the end users.

To tackle this challenge researchers have developed novel adaptation mechanisms [21-25] to dynamically adjust the video bit-rate to match the varying bandwidth availability. However, this often requires the use of special compression algorithms (e.g., FGS [21-22]) or real-time media transcoders [24-25] that may not be feasible or available in some applications.

Without these advanced codecs or transcoders, today's content providers typically prepare a few versions of the same content in different bit-rates to cater for users of different connection bandwidth. Given the complexity and the time required to encode multiple versions of the same video, it is not surprising that there will only be a small number of versions of the same video content provided. Thus the selected video is often either of too low or too high a bit-rate for the client. The former case is trivial as streaming will likely be successful. The latter case will be far more complicated as the client now does not have sufficient bandwidth to streaming the video in the

conventional manner. Some existing video players will simply download the video and begins playback only after substantial portion of the video has been downloaded. However, due to the inherent variations in network bandwidth availability, even this conservative strategy may not be able to ensure continuous playback, especially for long video contents.

In this chapter we tackle this problem by developing a novel predictive buffering algorithm that can determine at runtime the buffering time required to ensure playback continuity, especially for longer videos (e.g., over a few minutes) and when the video bit-rate exceeds the available network bandwidth. The proposed predictive buffering algorithm is designed around two principles.

First, during the initial buffering period the client can measure the mean and variance of the available bandwidth over a given interval (e.g., 1 s). Assuming that the past and future available bandwidth is a stationary random process of unknown distribution, then the sum of future bandwidth availability over the next n intervals will approach normal as $n \to \infty$ due to the Central Limit Theorem. Thus knowing the distribution of the future bandwidth, the client can determine the minimum buffer time to ensure playback continuity.

Second, the success of the approach depends on the stationarity assumption of the future bandwidth availability, given that the available bandwidth from a sender to a receiver is often unpredictable. However, our measurements and analysis in previous chapters have established that if there are multiple senders transmitting data to the client simultaneously, then the aggregate available bandwidth will become far more stationary. Therefore, by employing sufficient number of senders, each transmitting a portion of the data, the stationarity assumption can then be satisfied and we can invoke the first principle to determine the minimum buffering time accordingly.

The proposed predictive buffering algorithm is evaluated using extensive trace-driven simulations, with two sets of traffic traces obtained from different networks and different time frames. The results confirm the relation between the aggregate bandwidth stationarity and the number of senders in the aggregate data flow, and also show that the predictive buffering algorithm can achieve buffer delays that are remarkably close to the optimal buffer time.

We present the predictive buffering algorithm in this chapter and then evaluate its performance using trace-driven simulations in Chapter 7.

6.1 Related Work

Streaming video from multiple sources to a receiver has previously been investigated by a number of researchers [26-30]. Compared to single-source streaming, multi-source streaming has several potential advantages, such as increasing the throughput by combining the bandwidth of multiple senders [26-28]; adapting to network bandwidth variations by shifting the workload among the multiple senders [29-30]; and reducing bursty packet loss by splitting the data transmission among the multiple senders [26-27].

For example, Nguyen and Zakhor [26-27] developed rate allocation and packet partition algorithms with Forward Error Correction (FEC) to minimize the packet loss rate and the probability of late packet arrivals. Xu et al. [28] proposed an algorithm for media data assignment to reduce buffering delay. Kwon and Yeom [29] proposed a dynamic rate allocation and packet partition scheme to adapt to the senders' varying throughput. Agarwal and Rejaie [30] proposed an adaptive layered streaming algorithm to compensate for variations in the measured available bandwidth from all congestion controlled senders.

The above studies exploited the availability of multiple sources and the diversity of multiple network paths to improve streaming performance. In another study, Reisslein and Ross proposed a novel call admission scheme [31] that can provide statistical QoS guarantee in streaming prerecorded variable-bit-rate (VBR) videos over ATM. In their study the network bandwidth is known but the video bit-rate can vary due to the VBR encoding and interactive playback controls. To guarantee QoS they proposed to multiplex multiple video streams over the network and then model the bit-rate of the multiplexed aggregate video flow as a stochastic process, and then apply the Central Limit Theorem and Large Deviation theory to obtain probabilistic bounds.

In comparison, the intra-flow bandwidth aggregation model developed in this chapter also appeals to the Central Limit Theorem (CLT) to obtain probabilistic bounds. However, there are two fundamental differences. First, Reisslein and Ross's work [31] solved the problem of varying video bit-rate but with constant network bandwidth, while our work solved the problem of constant video bit-rate but with varying network bandwidth. Second, the varying video bit-rate in Reisslein and Ross's work, although modeled as a random process, is known a priori as they are prerecorded. By contrast, our work does not assume *a priori* knowledge of the varying available bandwidth, and thus we need to develop an estimation algorithm to measure and estimate the parameters of the stochastic process.

6.2 System Model

To begin a new video session, a client will send requests to N senders to initiate data transfer. We assume that the video data are delivered from each sender to the receiver using a transport protocol with congestion control mechanisms such as TCP or

TCP-friendly streaming protocols (e.g., TFRC [32]) such that the bandwidth available to the video session will vary according to the instantaneous load of the network path.

The client upon receiving the initial video data will begin the buffering period, and then start playback once sufficient amount of video data are buffered. Specifically, let C_i be the total amount of data received from all N senders in time interval i after the buffering process begins; R be the video bit-rate and w be the time to start playback. To ensure continuous playback we must ensure that the amount of data received at any time must not be less than the amount of data consumed, i.e.,

$$\sum_{j=1}^{i} C_j \geq R(i-w), \forall i > w \tag{44}$$

or else buffer underflow will occur, causing playback interruptions. The challenge is to find, *at run time*, the smallest buffering period w that satisfies (44).

6.3 Prediction Algorithm for Constant Bit-Rate Videos

At each time interval, the client will check to see if sufficient data have been received to sustain continuous video playback for the rest of the video session. Let L be the total video length in number of time intervals and B_i be the amount of data received up to the time interval i. Then the client can guarantee continuous playback for the entire video session if the following constraint is satisfied:

$$B_i + \sum_{j=i+1}^{i+k} C_j \geq Rk, \forall k = 1, 2, \ldots, L \tag{45}$$

where the L.H.S. is the amount of data already buffered plus the amount of data to be received in the future k time intervals, and the R.H.S. is the amount of data to be consumed in the future k time intervals if playback is to begin from time interval i.

Otherwise the client will buffer for another time interval and then check (45) again, and repeat the process until (45) is satisfied. However the precise future bandwidth availabilities $\{C_j \mid j \geq i+1\}$ are obviously not known *a priori* and so we need to devise a way to estimate it.

From the results in Section 4.5 we know that many-to-one data flows exhibit more consistent properties over a long time scale. Thus we can assume that the future available bandwidth $\{C_j \mid j=i+1,i+2,\ldots\}$ will maintain the same mean and variance as the past available bandwidth up to the current time interval i: $\{C_j \mid j=1,2,\ldots,i\}$.

If we further assume that the $\{C_i$'s$\}$ are independent (c.f. Section 4.3), then the probability distribution of the summation term in the L.H.S. of (37) will be equal to the convolution of the probability distributions of the k aggregate bandwidths $\{C_j \mid j=i+1,i+2,\ldots,i+k\}$, denoted by $F_k(\cdot)$. Now as the $\{C_i$'s$\}$ are independent with the same mean μ and variance σ^2, the distribution $F_k(\cdot)$ will approach normal with mean $k\mu$ and variance $k\sigma^2$ as $k \to \infty$ according to the Central Limit Theorem.

Thus the minimum buffering time needed to guarantee playback continuity with a given probability of Δ can be computed from

$$w = \min_i \left\{ F_k(Rk - B_i) < (1-\Delta), \forall k = 1,2,\ldots,L \right\} \tag{46}$$

where the mean and variance of $F_k(\cdot)$ are estimated using the measured mean and variance of the aggregate bandwidth $\{C_i$'s$\}$ during the initial buffering period.

6.4 Prediction Algorithm for Variable Bit-Rate Videos

In the previous section we assumed that the video is encoded using constant-bit-rate (CBR) encoding in which their bit-rates are constant over the entire

duration of the video. However, the predictive buffering algorithm can be easily extended to support the streaming of variable-bit-rate (VBR) videos.

Assume we know in advance the VBR video's bit-rate profile $\{R_t \mid t = 1,2,\ldots,L\}$, where R_t is the short-term average video bit-rate in time interval t. To ensure continuous playback, the constraint (45) is replaced by

$$B_i + \sum_{j=i+1}^{i+k} C_j \geq \sum_{t=1}^{k} R_t, \forall k = 1,2,\ldots,L \tag{47}$$

And the minimum buffering time then becomes

$$w = \min_i \left\{ F_k(\sum_{t=1}^{k} R_t - B_i) < (1-\Delta), \forall k = 1,2,\ldots,L \right\} \tag{48}$$

The rest of the algorithm is the same as in the CBR case.

6.5 Parameter Estimation

During the initial buffering period the client measures the mean and variance of the aggregate available bandwidth. Being measurements of a stochastic process the measurement accuracy will depend on the number of samples used, i.e., the length of the measurement period. This latter point leads to another subtle issue as the length of the measurement period is simply equal to the buffering period, which can vary significantly depending on the ratio of the video bit-rate to the mean aggregate available bandwidth as well as the variances of the available bandwidth.

For example, if the available bandwidth is substantially lower than the video bit-rate then the buffering period will likely be longer, thus allowing more accurate measurement of the required parameters. On the other hand, if the available bandwidth is comparable to the video bit-rate then the buffering period as computed from (46) and (48) can be very short. In this case if the measured parameters are inaccurate then

the computation of (46) and (48) will become inaccurate as well, possibly resulting in playback interruptions.

To guard against this problem, we employ the method of confidence interval [33] in estimating the mean and variance during the buffering period. Specifically, when the sample size w is more than 30, we can assume that the sample mean distribution of μ is normally distributed. The $(1-\alpha)$ confidence interval of sample mean is given by

$$(\mu - z_{\alpha/2}\frac{\sigma}{\sqrt{w}}, \mu + z_{\alpha/2}\frac{\sigma}{\sqrt{w}}) \tag{49}$$

where σ is the samples' standard deviation and $z_{\alpha/2}$ is equal to 2.58 for $\alpha = 0.1$ (i.e., 99% confidence). Thus the client can use the lower limit of the confidence interval as the sample mean.

In extreme cases with sample size $w < 30$, the sample mean distribution is replaced by the Student's t-distribution with the corresponding $(1-\alpha)$ confidence interval given by

$$(\mu - t_{\alpha/2,w-1}\frac{\sigma}{\sqrt{w}}, \mu + t_{\alpha/2,w-1}\frac{\sigma}{\sqrt{w}}) \tag{50}$$

where the value of $t_{\alpha/2,w-1}$ is given in the t-table.

Our results show that using this confidence interval method can effectively prevent inaccurate parameter estimations without significantly increasing the estimated buffering time.

Chapter 7

PERFORMANCE EVALUATION

In this chapter we evaluate the performance of the predictive buffering algorithm using trace-driven simulations.

7.1 Trace-Driven Simulation Setup

There are two sets of trace data used in the simulations. The first set of trace data is obtained from our measurements conducted in the PlanetLab. The second set is generated using a well known network simulator – NS2 [34]. We first obtained traffic trace data from the NLANR PMA archive [35], which captured the packet-level trace data at an Internet gateway at Bell Labs in 2002. The one-day trace is then divided into separate one-hour sub-traces. The sub-traces are used as cross traffic in the simulation topology depicted in Fig. 16. There are up to N senders {S1, S2, …, SN} transmitting data simultaneously to the receiver R using TCP as the transport. The senders do not perform additional rate control and simply transmit data as fast as TCP allows. We choose TCP for its ability to automatically adapt to the network load (i.e., the cross traffic) to obtain a fair share of the available bandwidth for transporting video data. Other transport protocols such as TFRC can also be used as long as they have built-in congestion control algorithm. The predictive buffering algorithm operates independently from the actual transport protocol used.

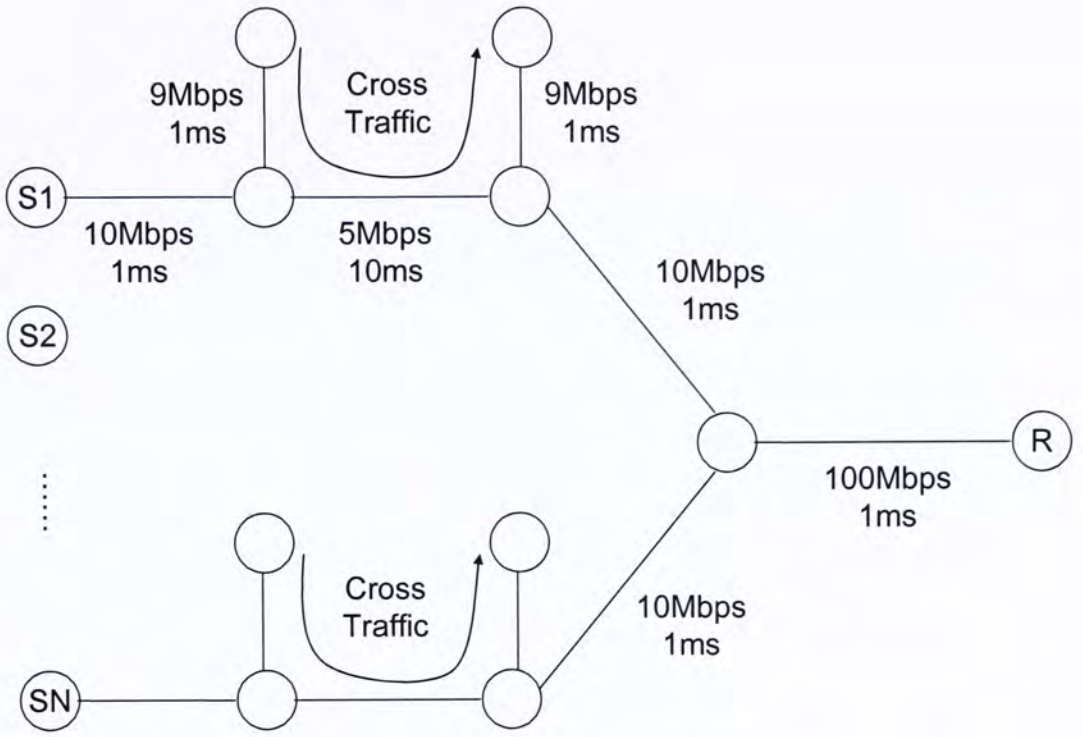

Fig. 16. Simulation topology.

7.2 Performance over CBR Videos

In this section we evaluate the performance of the proposed predictive buffering algorithm over CBR videos. The video length is set to 1800 seconds and the video bit-rate varies from 1 to 1.3 times the mean aggregate available bandwidth (i.e., R/μ = 1, 1.1, 1.2 and 1.3). Thus other the case of $R/\mu = 1$ all other cases suffer from insufficient bandwidth and so rely on the predictive buffering algorithm to determine the minimum buffer time needed to ensure continuous video playback. In case the client runs into buffer underflow due to data not arriving in time for playback, it will suspend playback and then rerun the predictive buffering algorithm to buffer sufficient video data before resuming playback. An alternative approach (not used in this study) would be to continue playback despite the missing data and then attempt to conceal the visual degradation through error concealment techniques. In this latter approach

playback performance will then be measured by the visual quality (e.g., PSNR) instead.

7.2.1 Video Playback Performance

Fig. 17 plots the successful playback ratio using the PlanetLab traces with video bit-rate ratios equal to 1.1 (i.e. $R/\mu = 1.1$). Successful playback ratio is the proportion of simulation runs with no playback interruption (i.e. buffer underflow) during the entire video playback session. To provide a finer scale for performance evaluation we also plot in Fig. 18 the average pause count – the average number of buffer-underflow-induced playback interruptions per streaming session, and in Fig. 19 the average underflow time – the average total duration of playback suspension per streaming session.

There are four curves in Fig. 17–19: the Sample Mean curve is plotted with the mean aggregate achievable bandwidth of the initial buffering period as input to compute the minimum buffering time using (46); the 90%, 95% and 99% CI Mean curves are plotted with the lower limit of the 90%, 95% and 99% confident interval (c.f. (49) and (50)) as input to (46).

The first observation is that the performance when using the sample mean is significantly worse than the case when the lower limit of confidence interval is used. This is because in this simulation the video bit-rate is only 1.1 times the mean available bandwidth and so the resultant buffering time is relatively short, thereby leading to inaccurate measurement of the bandwidth parameters. In our other simulations with higher video bit-rate ratios the difference will become substantially smaller as the buffering period lengthens.

Second, the performance of using 90%, 95% and 99% CI mean are similar while using 99% CI mean is always the best. But note that there is a tradeoff of buffering time. Using a higher percentage of the confidence interval, the buffering time will be longer. We will come back to the discussion of the average buffering time in Section 7.2.2.

Using the 99% CI mean we investigate further the performance of the predictive buffering algorithm at video bit-rate ratios ranging from 1 to 1.3 using the PlanetLab traces (Fig. 20–22) and the NLANR PMA traces (Fig. 23–25). The results show that increasing the number of senders generally results in better performance, i.e., higher successful playback ratios, fewer playback pauses, and shorter underflow time for all 4 cases of video bit-rate ratios in both traces. This is a direct result of the improved consistency of the aggregate available bandwidth properties when there are many senders. In fact the algorithm achieves a successful playback ratio higher than 90% when there are 6 or more senders.

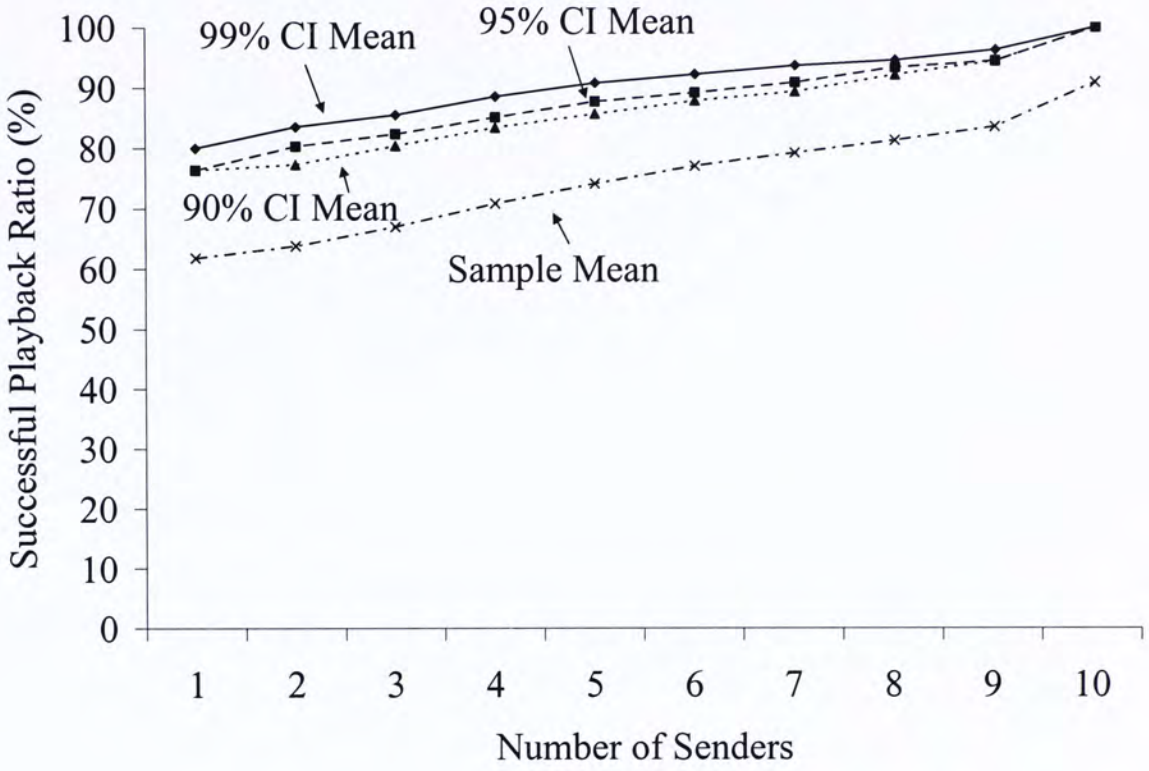

Fig. 17. Comparison of successful playback ratio when using Sample Mean, 90%, 95% and 99% CI Mean (PlanetLab traces, $R/\mu = 1.1$).

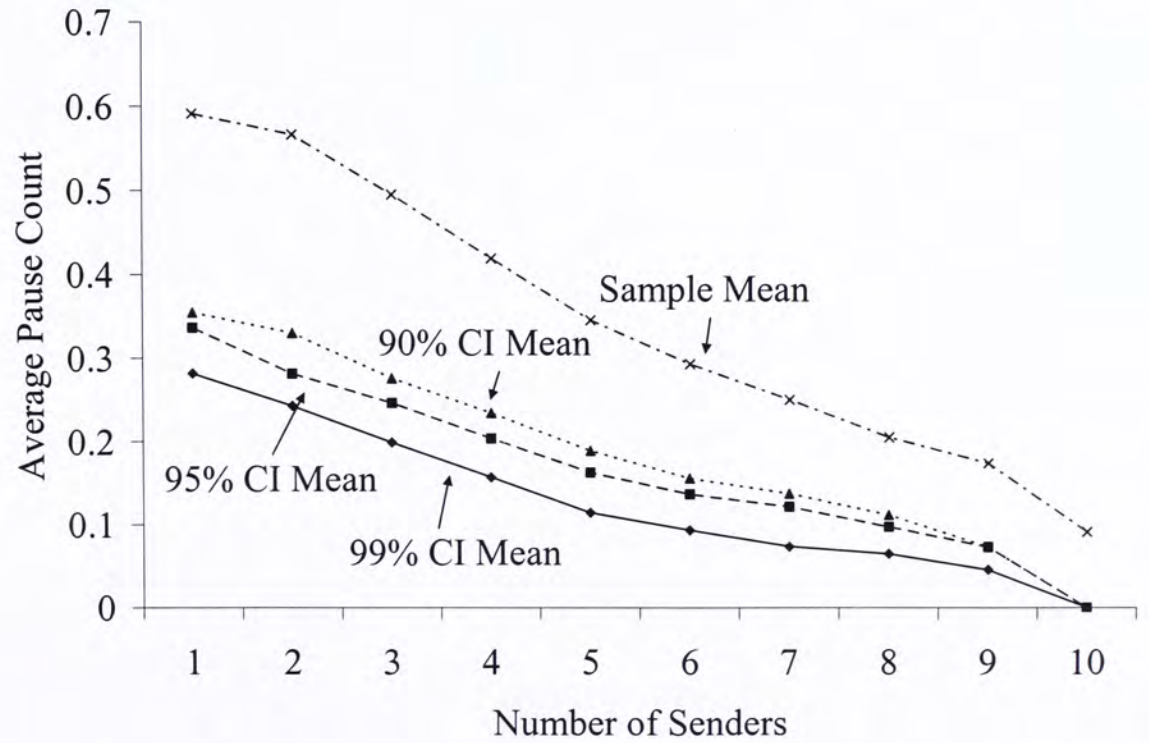

Fig. 18. Comparison of average pause count when using Sample Mean, 90%, 95% and 99% CI Mean (PlanetLab traces, $R/\mu = 1.1$).

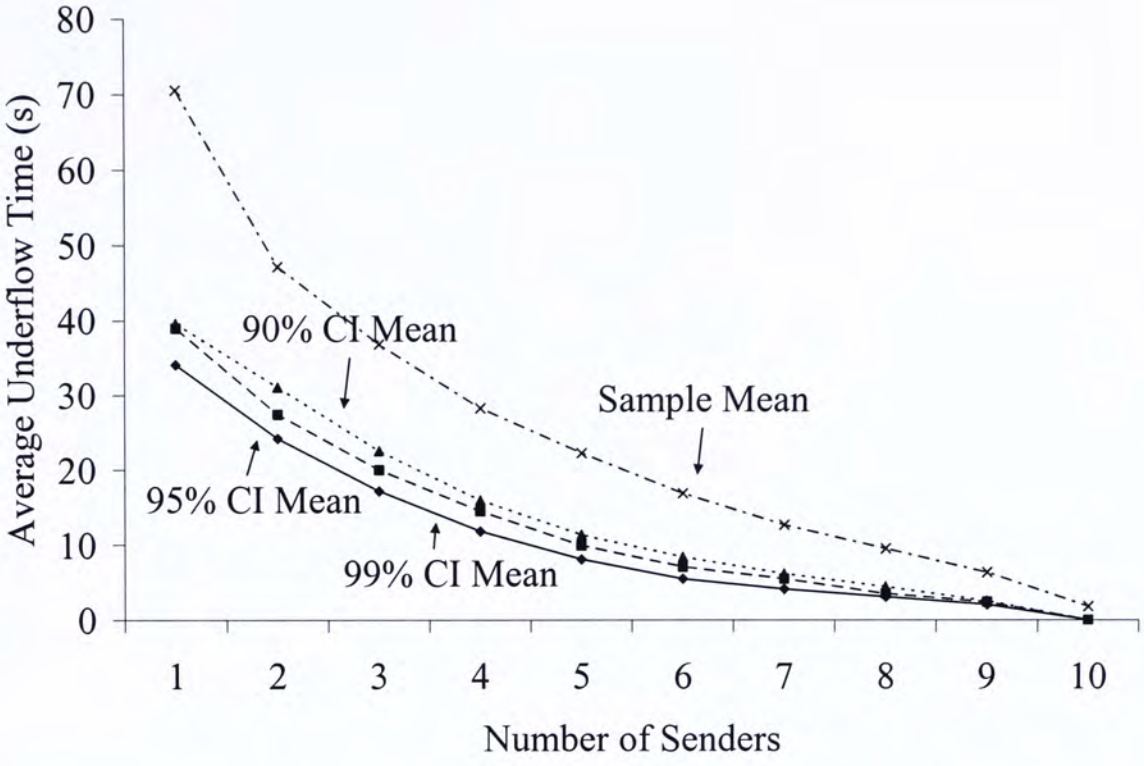

Fig. 19. Comparison of average underflow time when using Sample Mean, 90%, 95% and 99% CI Mean (PlanetLab traces, $R/\mu = 1.1$).

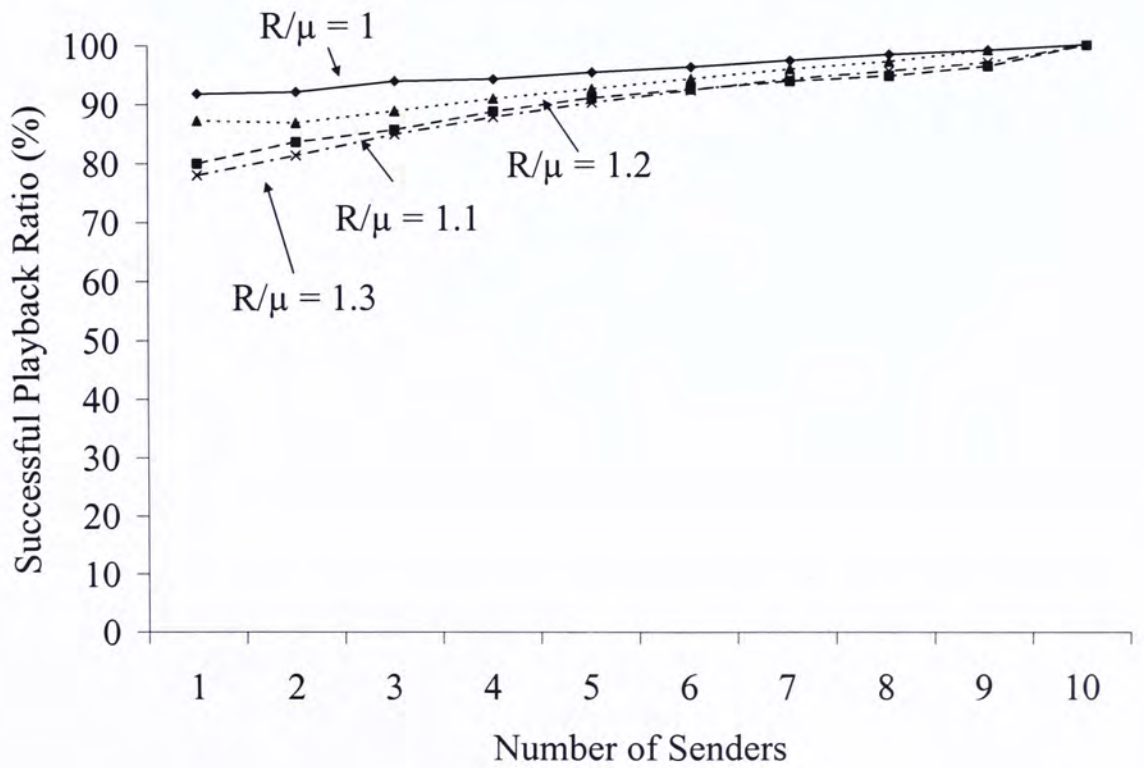

Fig. 20. Successful playback ratio for PlanetLab traces.

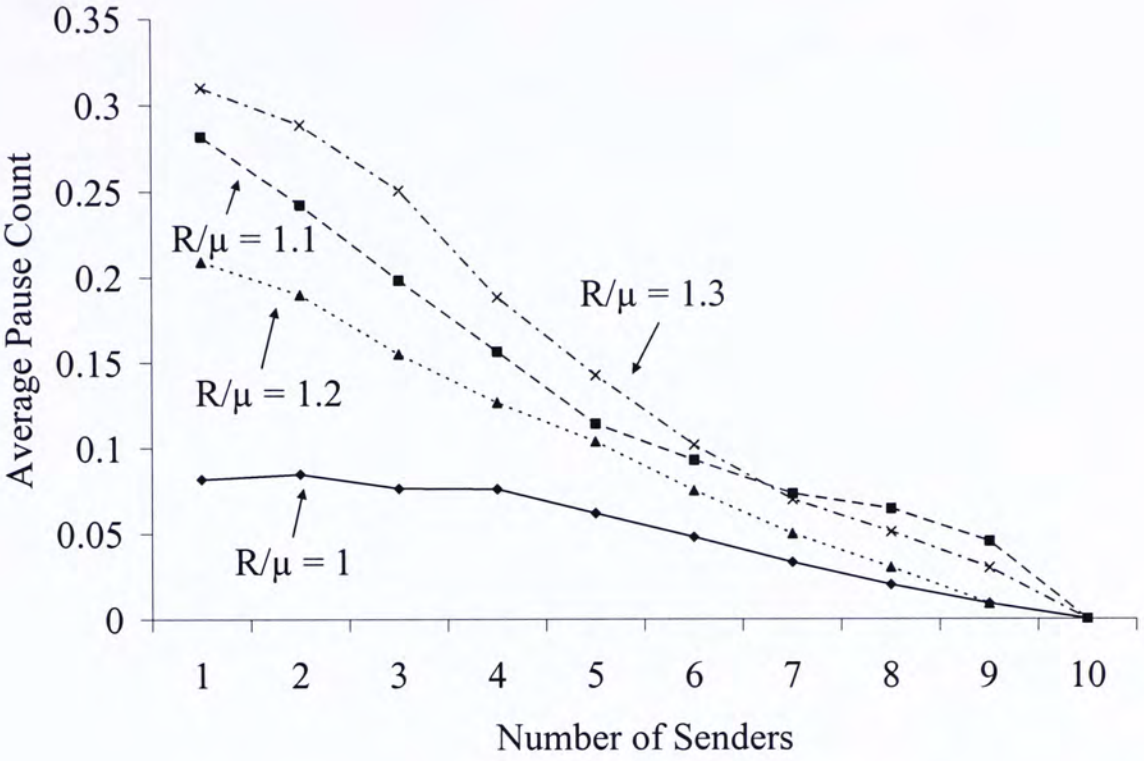

Fig. 21. Average pause count for PlanetLab traces.

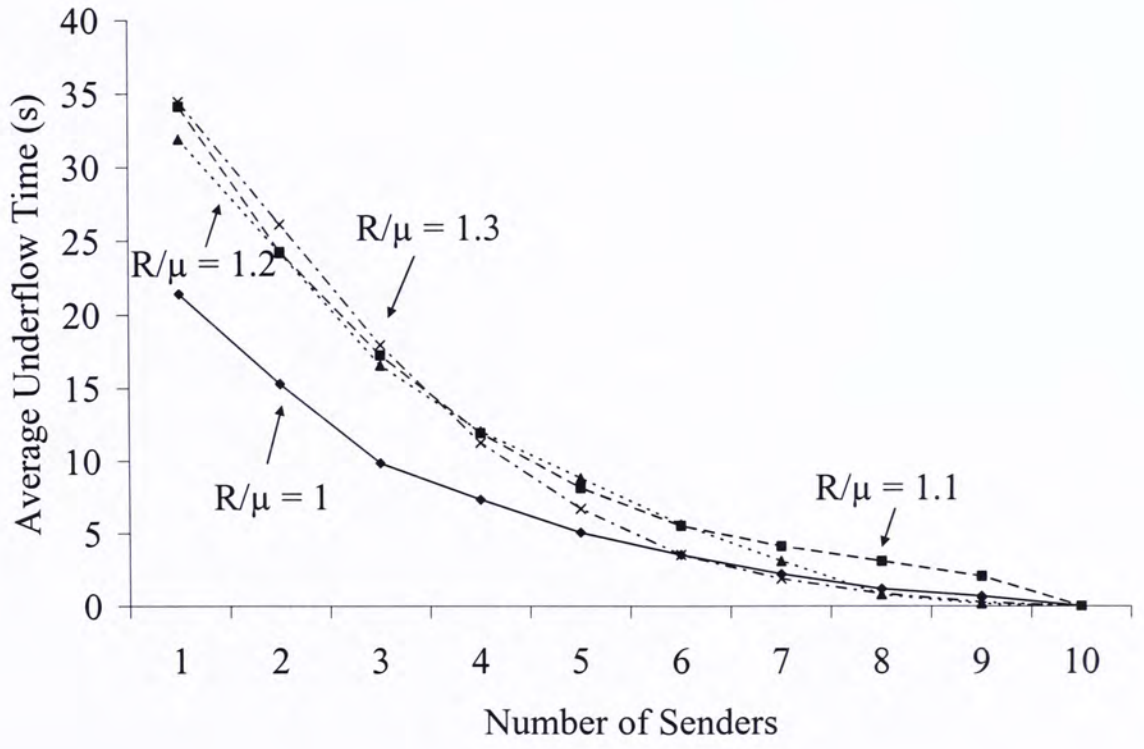

Fig. 22. Average underflow time for PlanetLab traces.

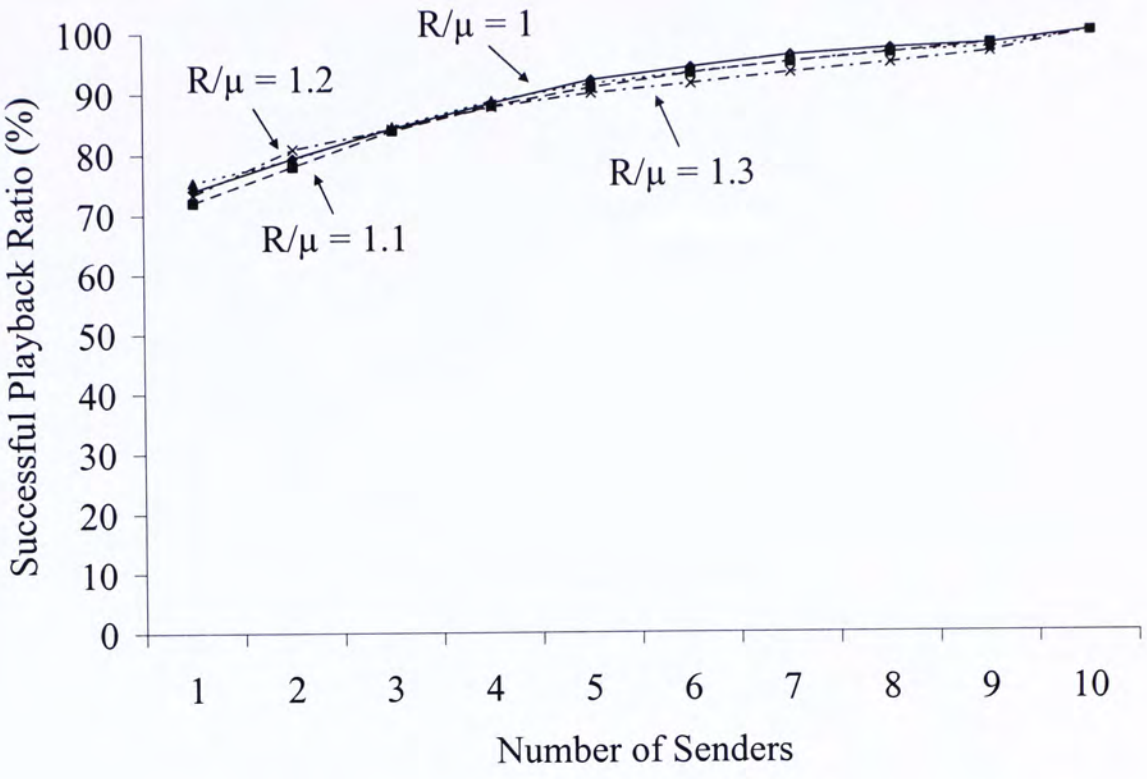

Fig. 23. Successful playback ratio for NLANR PMA traces.

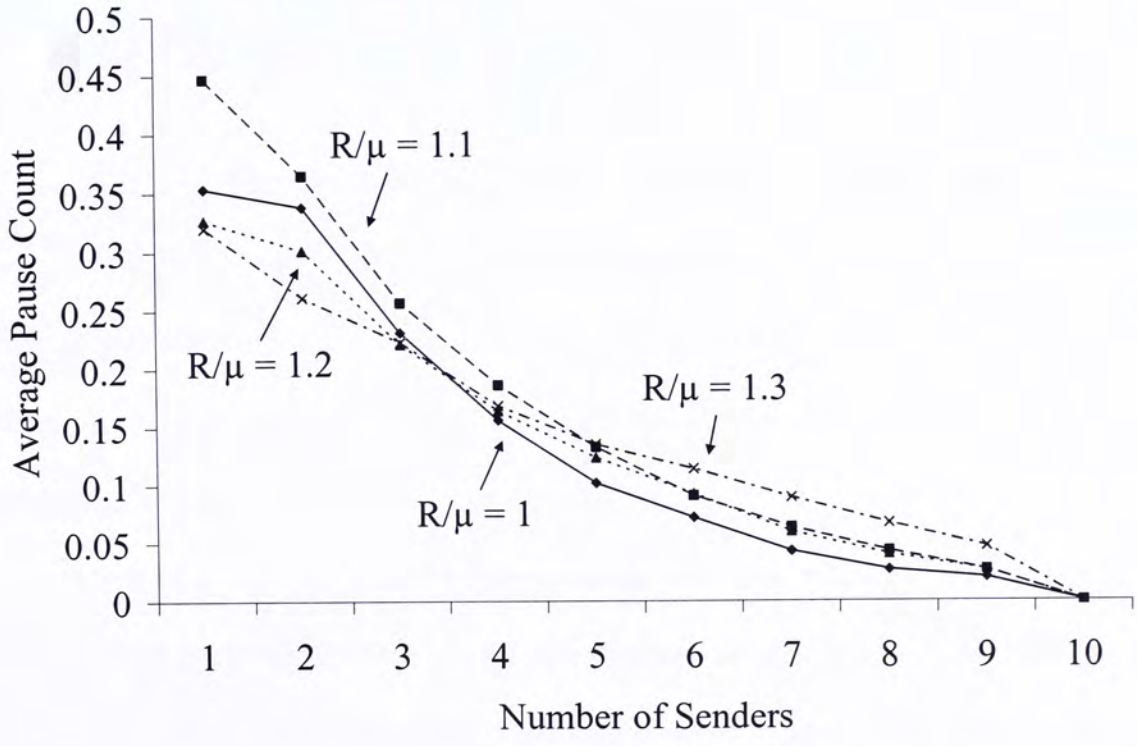

Fig. 24. Average pause count for NLANR PMA traces.

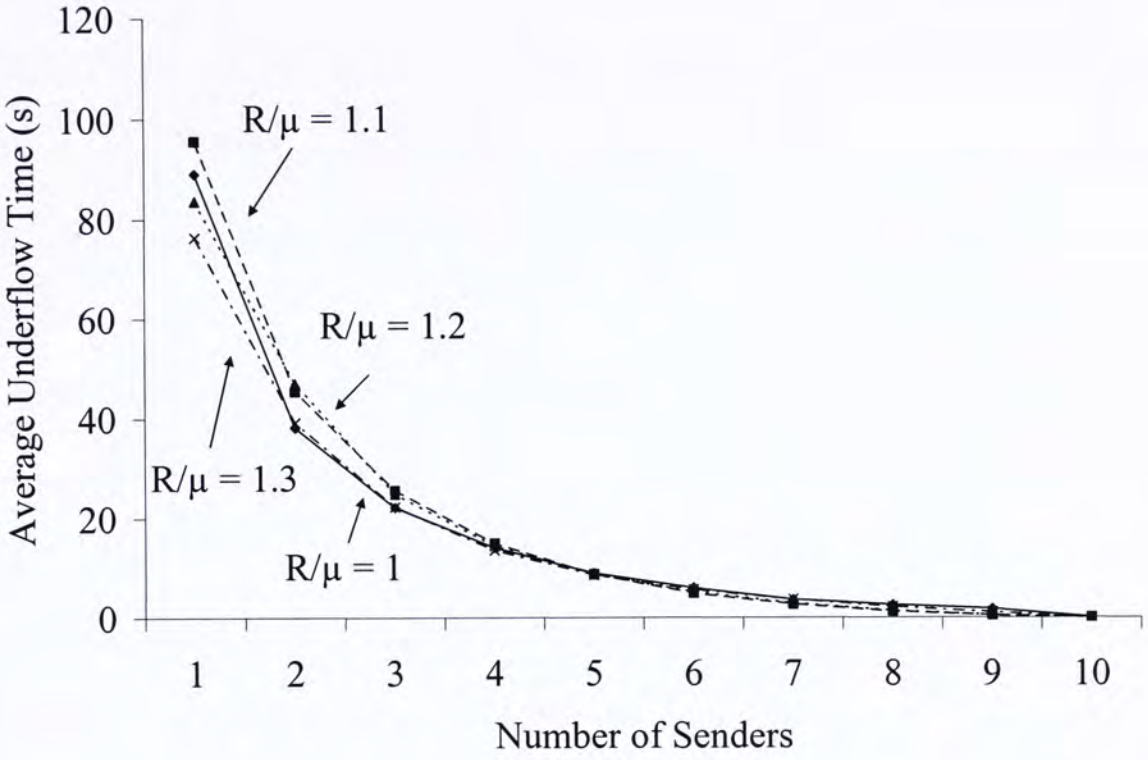

Fig. 25. Average underflow time for NLANR PMA traces.

7.2.2 Buffering Time

Next, we study how far the computed buffering time deviate from the lower bound, which is the minimum buffering time required for continuous video playback assuming all the future bandwidth availabilities are known a *priori*. This bound is not realizable in practice but provides a useful benchmark to evaluate the absolute performance of the predictive buffering algorithm.

First, we look into the effect on the average buffering time when using sample mean and the lower limit of the confidence interval. Fig. 26 plots the comparison of the ratio of the computed buffering time to the lower bound value when using the PlanetLab traces with $R/\mu = 1.1$. In the ideal case, the buffering time will be the same as the lower bound and the ratio will be equal to 1. If the ratio is larger than 1, then more than enough data will be buffered and video playback continuity is guaranteed.

On the other hand, if the ratio is less than 1, then the buffered data will not be sufficient to support continuous playback for the entire video. There are two sets of data in Fig. 26. One is calculated from all simulation runs with 7 or more senders. Another is obtained from only the successful simulation runs, again with 7 or more senders.

In the former set of results, the ratio when using sample mean is much smaller than those when using the lower limit of the confidence interval. It is expected since in the latter cases a smaller mean value is taken to compensate for inaccuracies in parameter estimation when computing the required buffering time.

However, the results from averaging the ratio of all runs do not reflect the whole picture. Fig. 27 plots the buffering time when using the sample mean and 99% CI mean together with the lower bound. It can be observed that the computed values when using the sample mean are smaller than the lower bound in some runs. Note that if the computed buffering time is less than the lower bound, then it is in fact undesirable as there will be playback interruptions.

Thus, to exclude these unsuccessful cases we plot the average ratio of successful runs in Fig. 26. The results show that when using the sample mean the ratio is still smaller than those using the lower limit of the confidence interval. But the differences are much smaller. All of the ratios are approximately equal to 1.8. This shows that by using the confidence interval mean we can achieve substantially better successful playback ratio (c.f. Fig. 17) and yet with insignificant increase in the buffering time.

Second, we study the relationship between the buffering time ratio and video bit-rate ratio. Fig. 28 plots the ratio of the buffering time to the lower bound ratio (successful runs only) with R/μ = 1, 1.1, 1.2 and 1.3 for the PlanetLab and the

NLANR PMA traces. There are two observations. First, the buffering time ratio drops when the video bit-rate ratio (R/μ) increases. It is because higher video bit-rate ratio requires longer buffering time to compensate for the insufficient bandwidth. This leads to a longer parameter estimation period and thus higher accuracy in parameter estimation. The computed buffering time is thus closer to the lower bound.

Second, the ratio for the NLANR PMA traces is higher than that for the PlanetLab traces except for $R/\mu = 1$. This is because the NLANR PMA traces generally exhibit significantly more and larger variations in the available bandwidth. Thus to compensate for the larger bandwidth variations the predictive buffering algorithm will extend the buffering time to ensure continuous playback. When $R/\mu = 1$, the ratio for the PlanetLab traces is higher because of the very small lower bound value (e.g., less than 5 seconds) in some runs. With the use of the lower limit of the confidence interval, the predictive buffering algorithm easily underestimate (or overestimate) the future bandwidth availability due to the short estimation period. The runs with underestimation lead to a very high ratio.

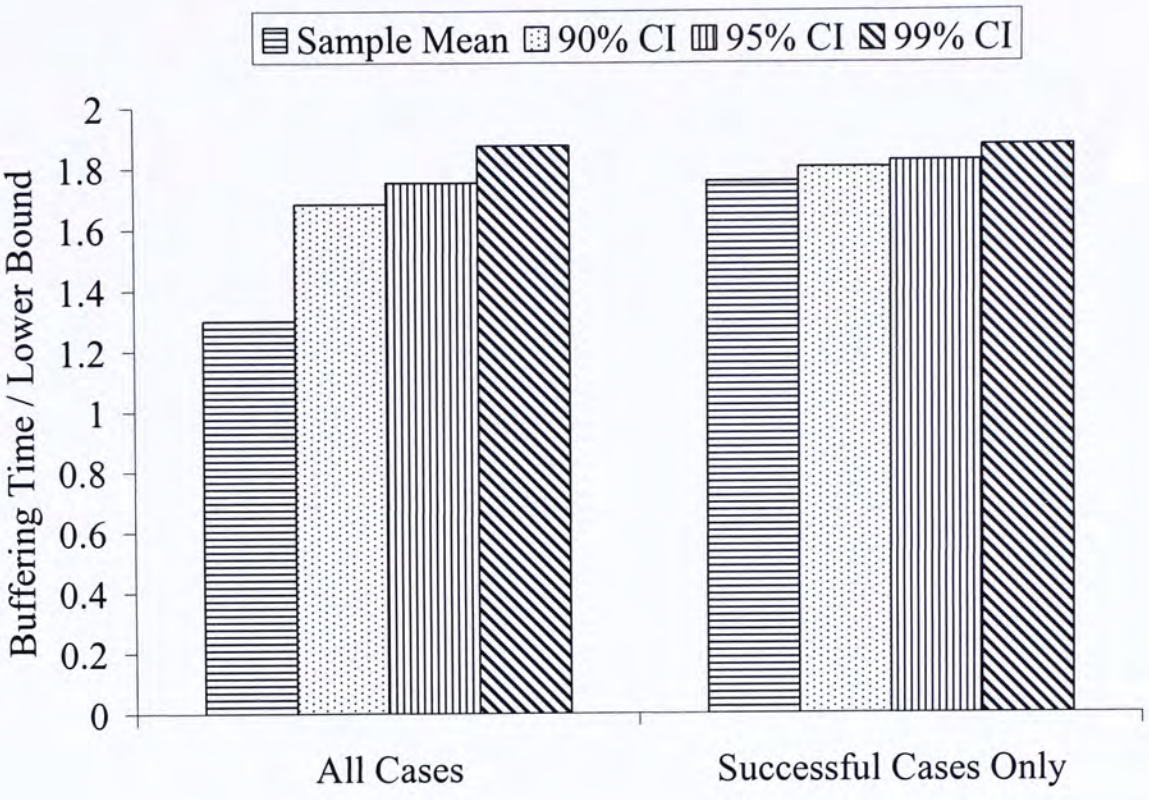

Fig. 26. Comparison of the ratio of the buffering time to the lower bound when using Sample Mean, 90%, 95% and 99% CI Mean (PlanetLab traces, $R/\mu = 1.1$).

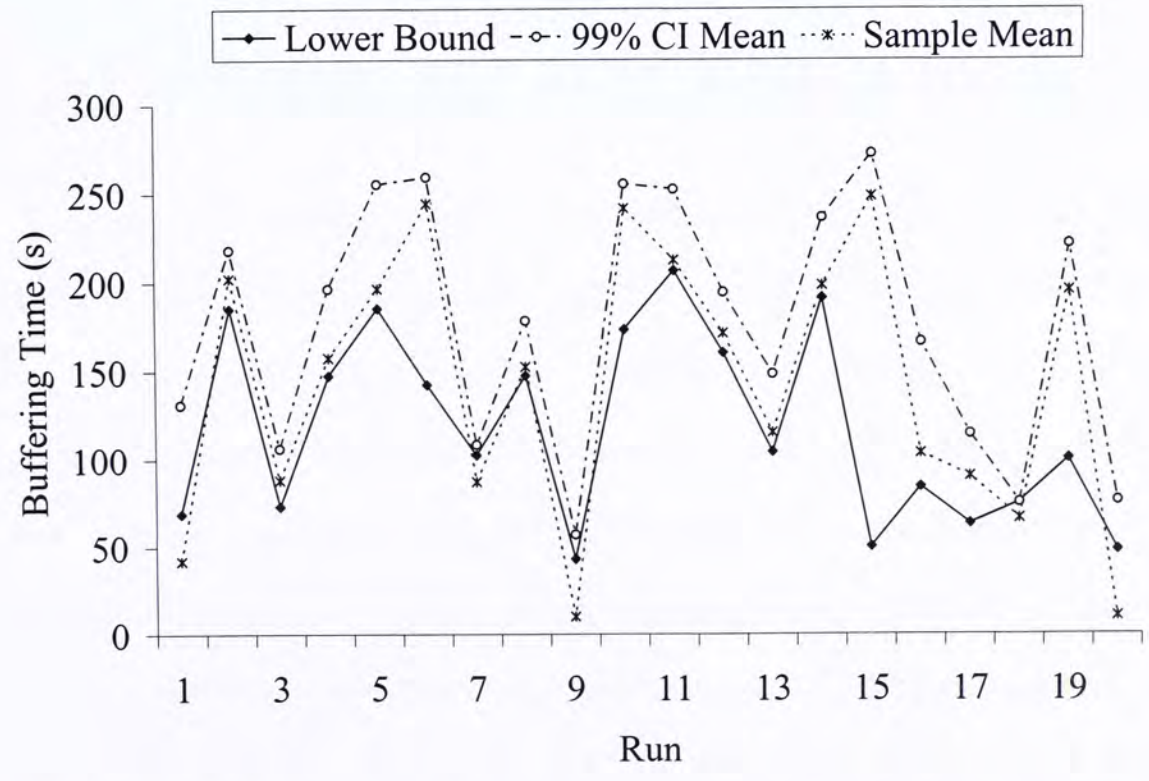

Fig. 27. Buffering time of 20 different simulation runs (PlanetLab traces, $R/\mu = 1.1$).

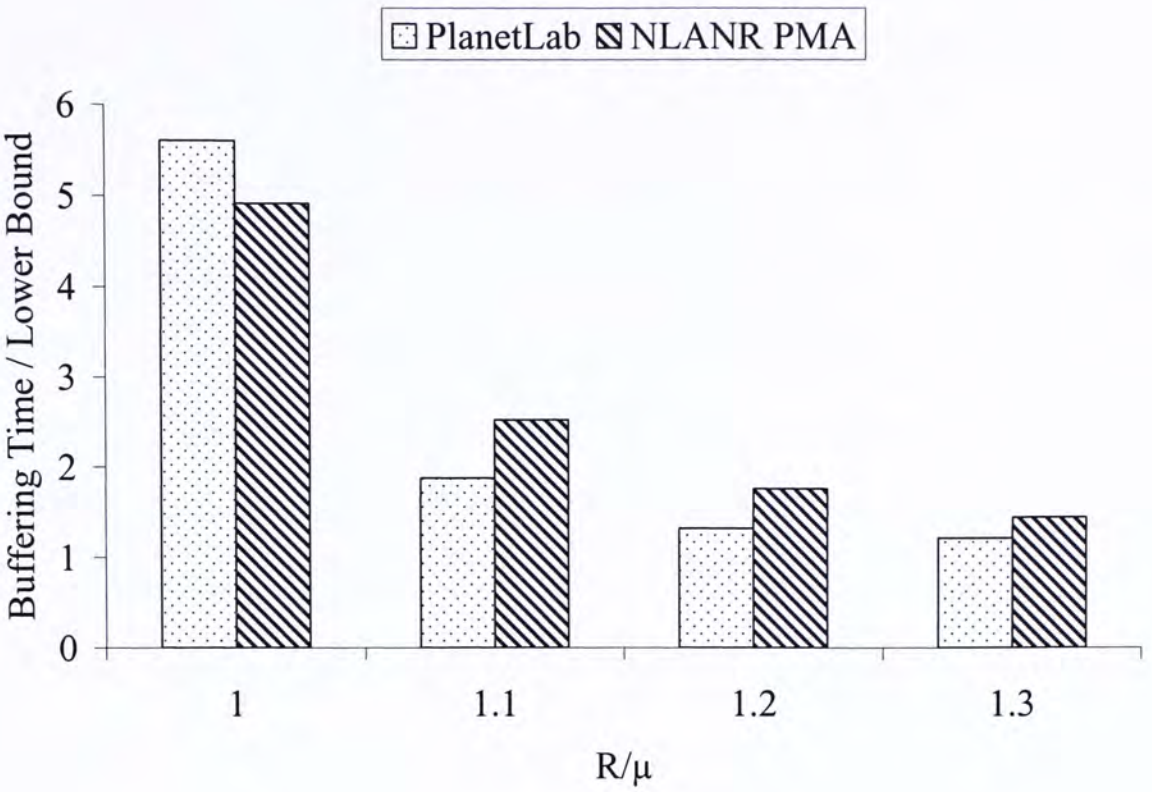

Fig. 28. The ratio of the buffering time to the lower bound for PlanetLab and NLANR PMA traces (successful runs only).

7.3 Performance over VBR Videos

To evaluate the performance of the predictive buffering algorithm streaming VBR videos, we perform trace-driven simulations using the PlanetLab traces using VBR video bit-rate profile from a movie DVD – Godzilla. As the total length of the movie is 8325 seconds which is longer than the PlanetLab traces (maximum of two hours), we split the entire movie into three parts, each lasting 2775 seconds to conduct the simulation. Moreover, to make the results comparable across different simulation runs, we scale the video bit-rate profile such that they all have the same average video bit-rate, which are set to 1.1 and 1.3 times the mean aggregate bandwidth available (i.e., $R/\mu = 1.1$ or 1.3, where $R = \mathrm{E}[R_t]$).

7.3.1 Video Playback Performance

Fig. 29 and 32 plot the successful playback ratio with R/μ = 1.1 and 1.3 respectively using the 99% CI mean. The average pause count and the average underflow time are shown in Fig. 30 and 31 with R/μ = 1.1 and Fig. 33 and 34 with R/μ = 1.3. Similar to the performance over CBR video, the successful playback ratio increases, average pause count reduces and average underflow time shortens when there are more senders.

When R/μ = 1.1 the performance of Part 1 and Part 2 is slightly better than that of CBR and Part 3, while the differences are negligible when R/μ = 1.3. This is due to the length of the buffering time (c.f. Fig. 37). The explanation of the differences in the buffering time will be discussed in Section 7.3.2. As the buffering time of Part 1 and Part 2 is longer, the accuracy in parameters estimation is higher and hence the performance is better. The effect is negligible when R/μ = 1.3 because the buffering time to compensate for the bandwidth insufficient is already long enough.

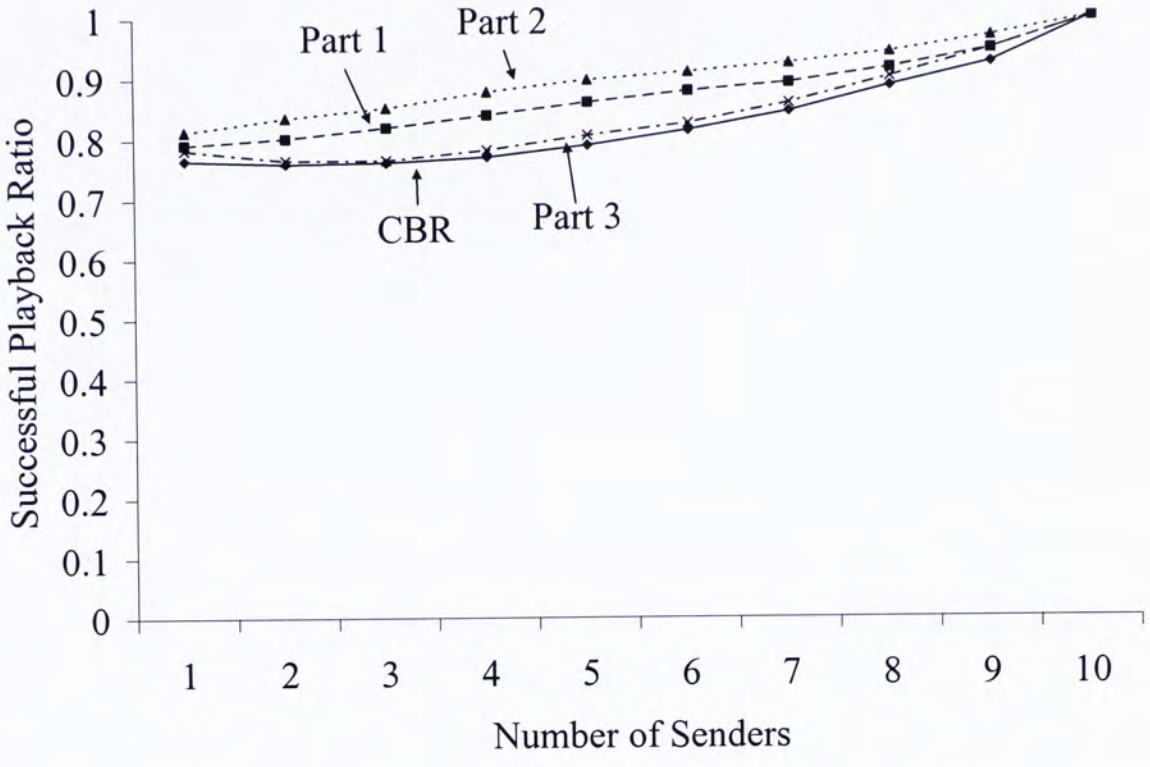

Fig. 29. Successful playback ratio ($R/\mu = 1.1$).

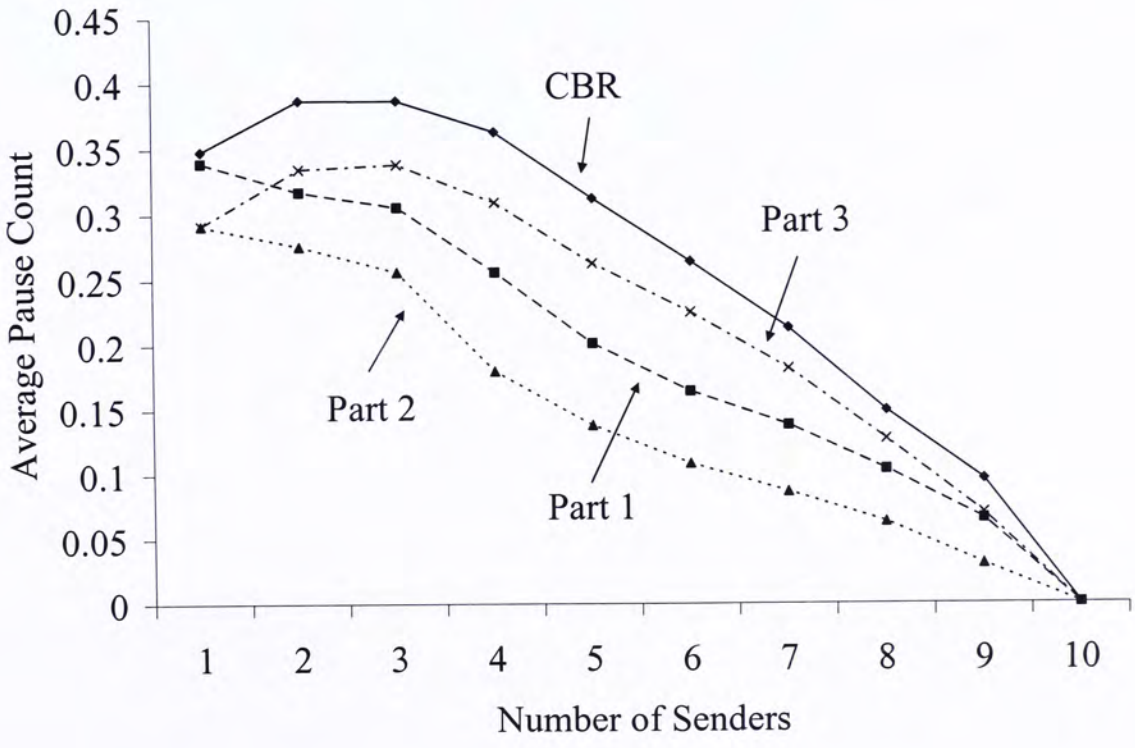

Fig. 30. Average pause count ($R/\mu = 1.1$).

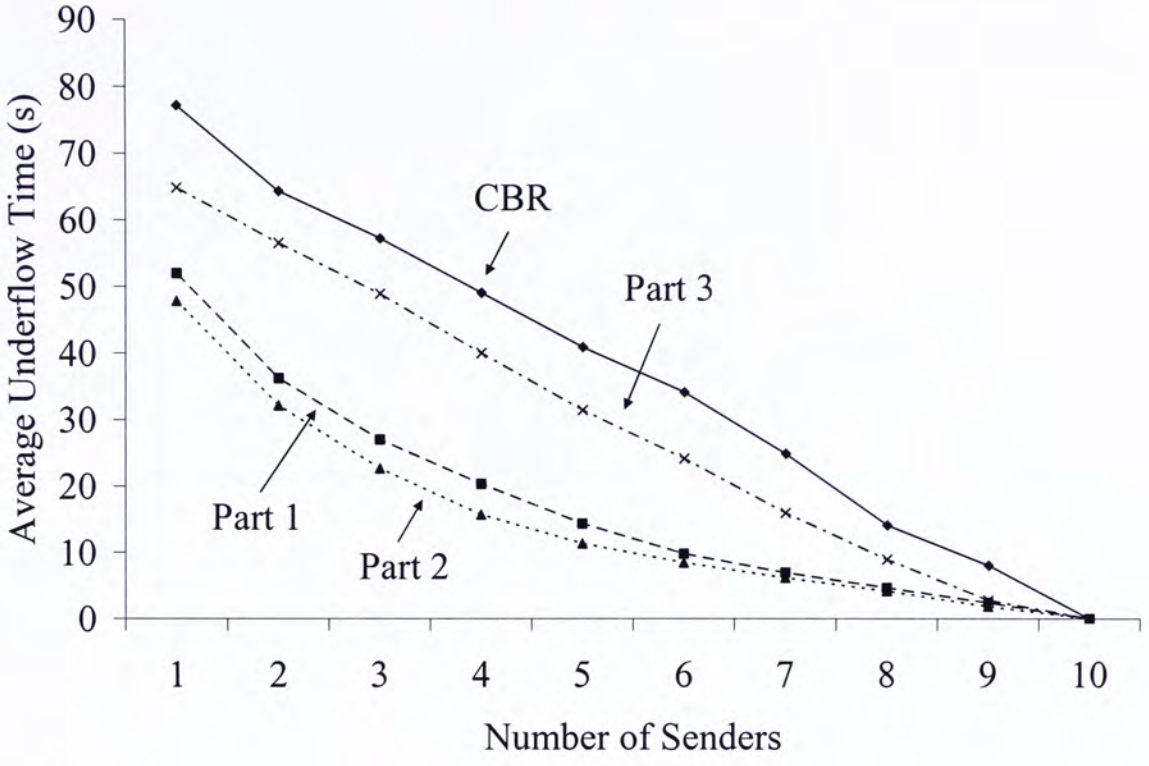

Fig. 31. Average underflow time ($R/\mu = 1.1$).

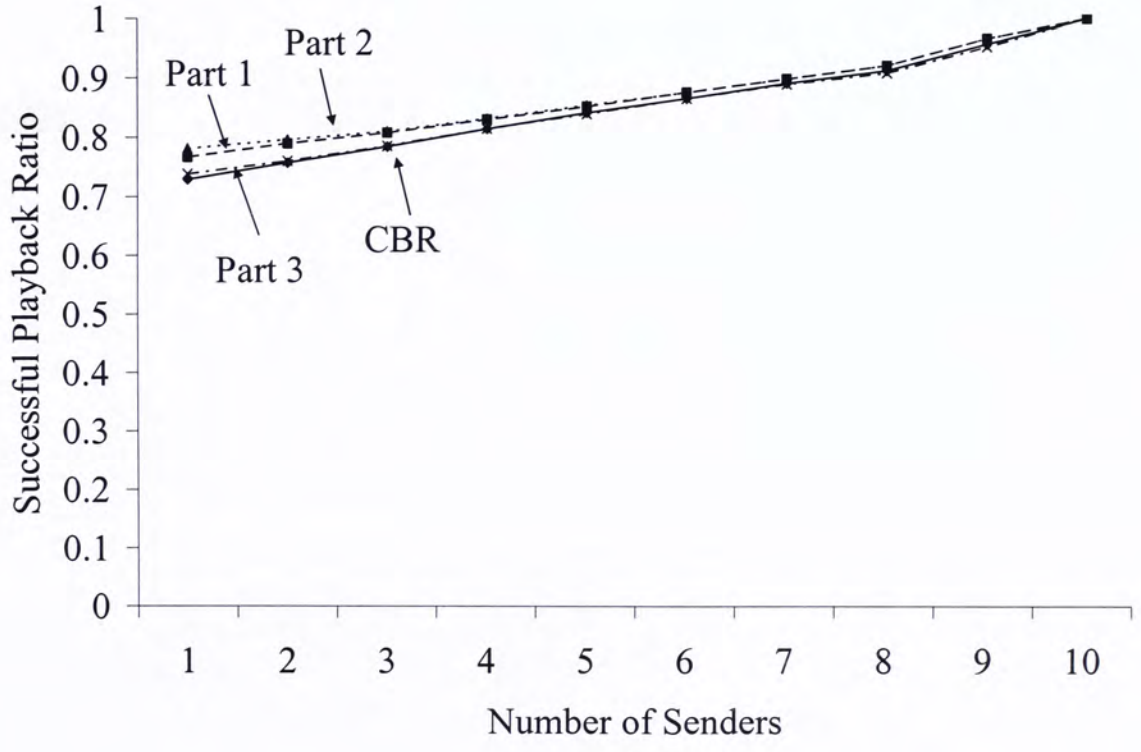

Fig. 32. Successful playback ratio ($R/\mu = 1.3$).

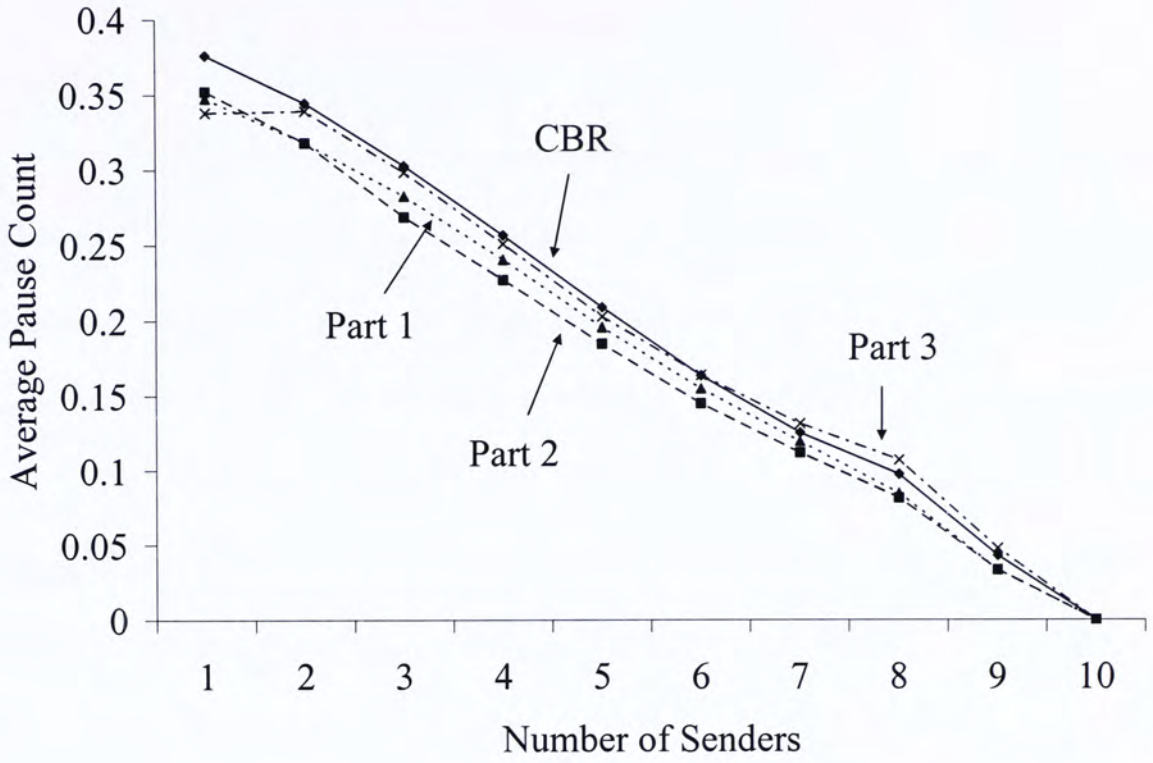

Fig. 33. Average pause count ($R/\mu = 1.3$).

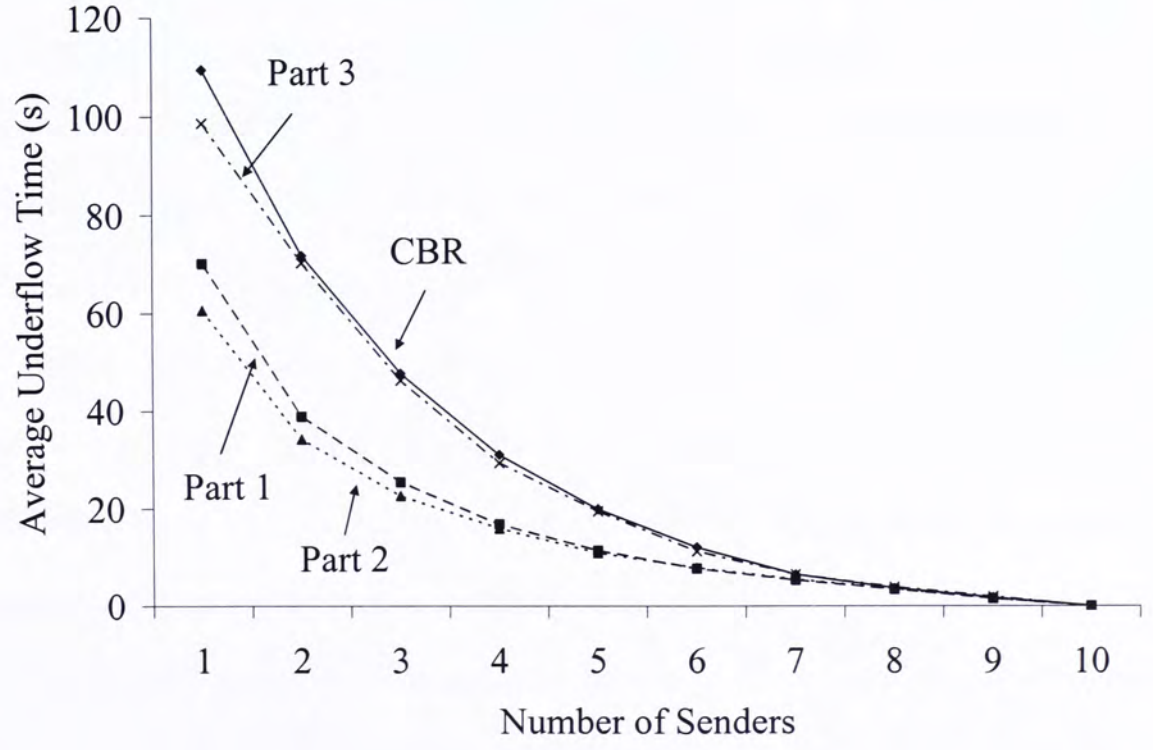

Fig. 34. Average underflow time ($R/\mu = 1.3$).

7.3.2 Buffering Time

We first evaluate the absolute average buffering time. For CBR videos, the buffering time is affected by the mean video bit-rate ratio and the bandwidth variations. For VBR videos, there is one more factor – the variation of video bit-rate during the entire video. For example, consider a VBR video stream where the first half of the video has twice the video bit-rate of the second half of the video. Compared to a CBR video of the same mean video bit-rate, it is easy to see that the amount of buffering required for the VBR video will be larger due to the higher bit-rate during the first half of the video.

Fig. 35 illustrates this observation by plotting the normalized video playback curves, i.e., the cumulative amount of video data consumed by the client, of three VBR video streams from the actual video bit-rate profile, together the CBR counterpart. We can see that Part 1 and Part 2 consume more while Part 3 consumes less video data at the beginning of the video compare to the CBR counterpart. In contrast, the scenario is reversed near the end of the playback as shown in Fig. 36, which plots the last 100 seconds of the playback curves in Fig. 35.

With no bit-rate variations at all we expect buffering time of the CBR video to be the shortest. Fig. 37 shows the average buffering time with R/μ = 1.1 and 1.3. We observe that the average buffering time of the CBR video is indeed the shortest, while Part 2 requires the longest buffering time.

Fig. 38 plots the ratio of the buffering time to the lower bound (successful runs only) with R/μ = 1.1 and 1.3. The ratios are around 1.5 (R/μ = 1.1) and 1.2 (R/μ = 1.3) for all the four videos. There is no significant difference between CBR and VBR videos.

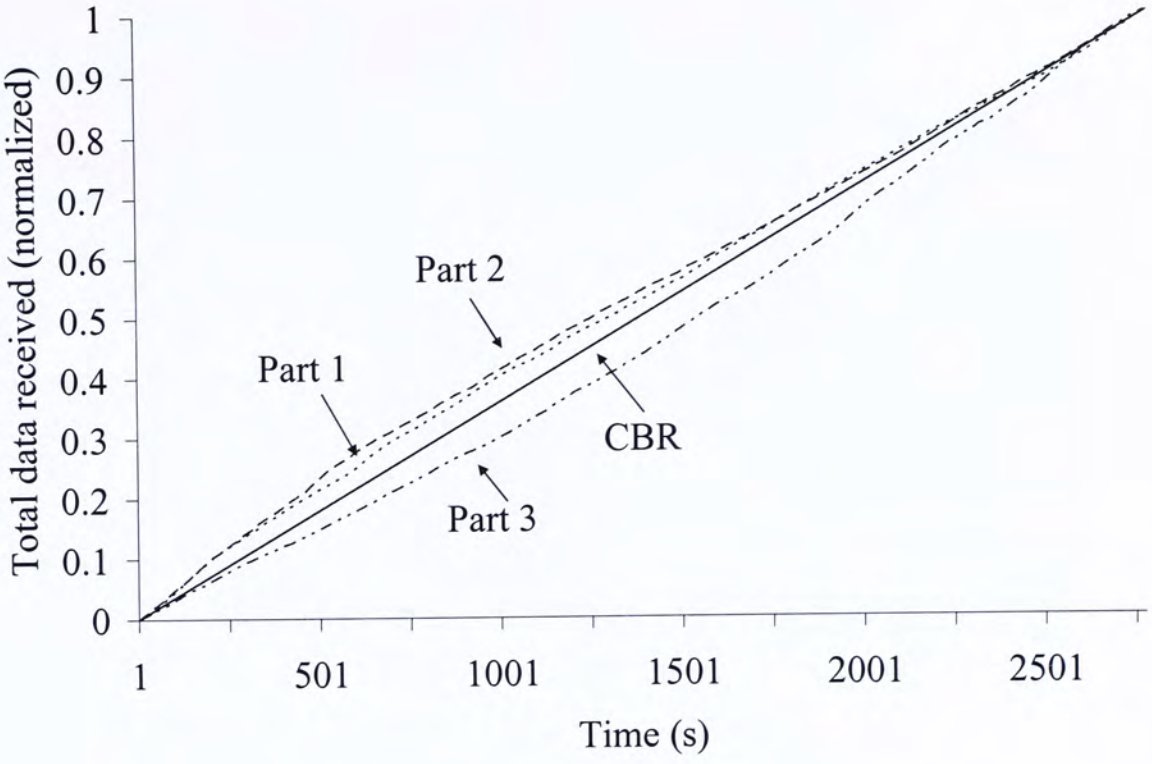

Fig. 35. Normalized video playback curve (the entire video).

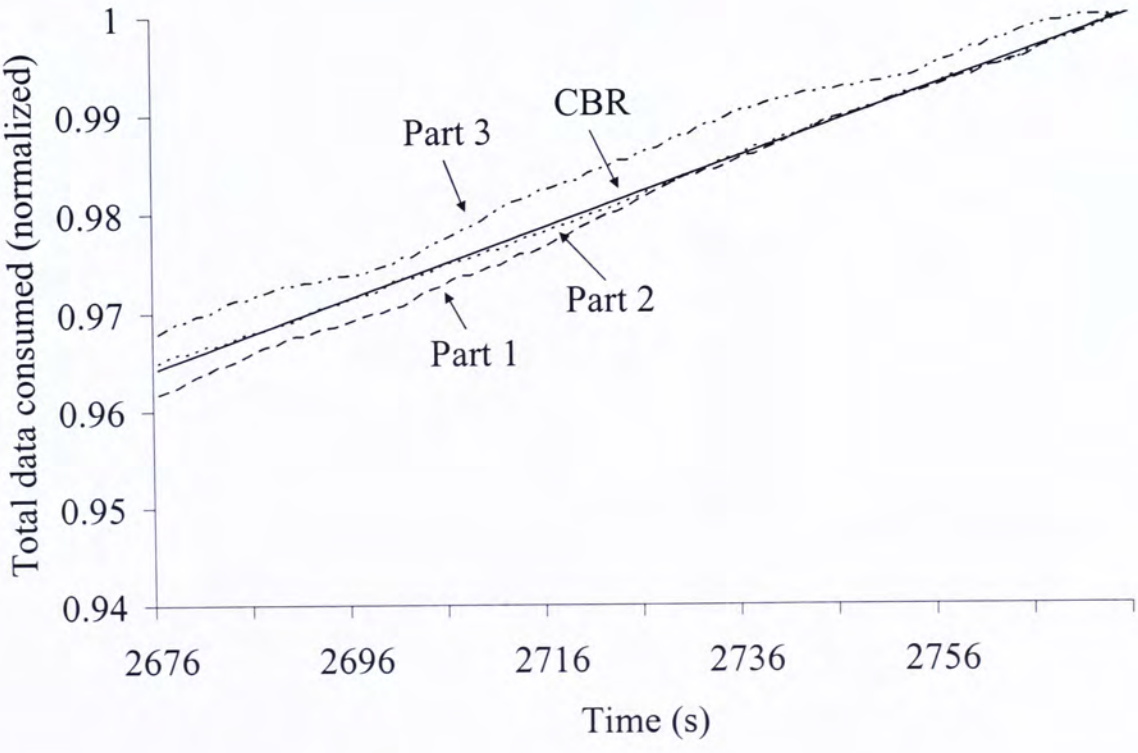

Fig. 36. Normalized video playback curve (last 100s).

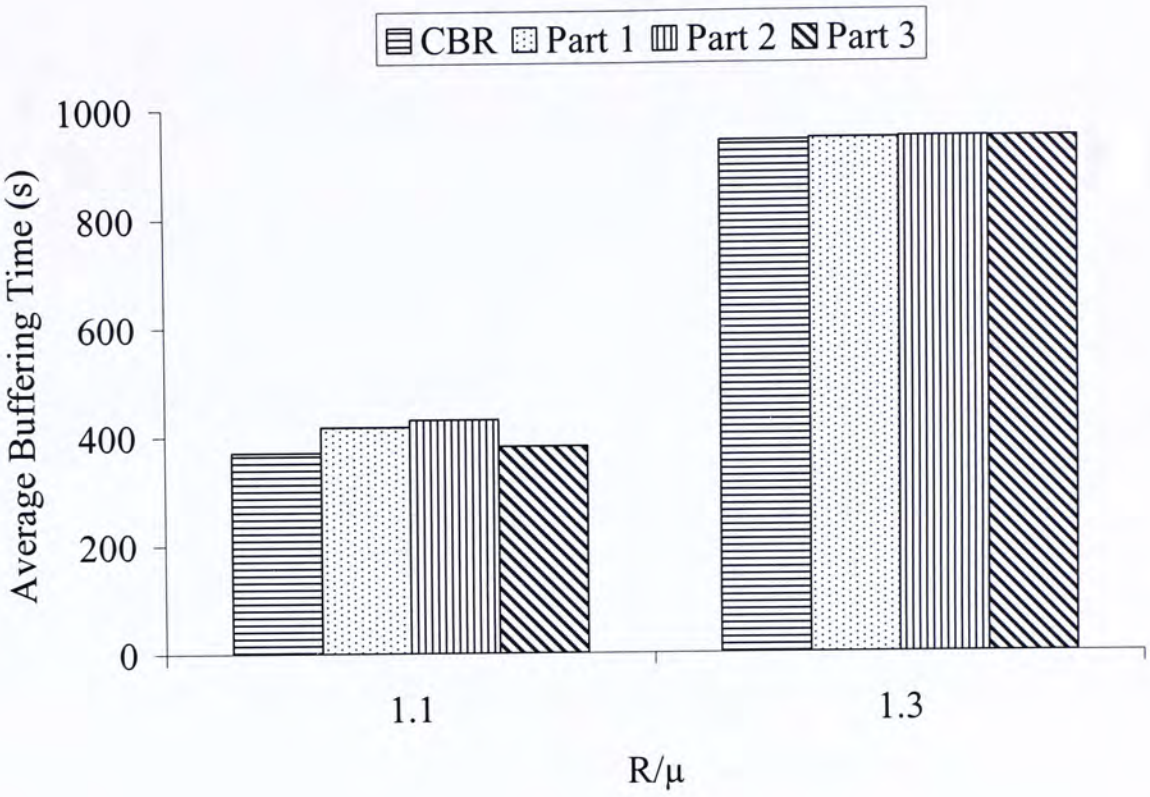

Fig. 37. Average buffering time.

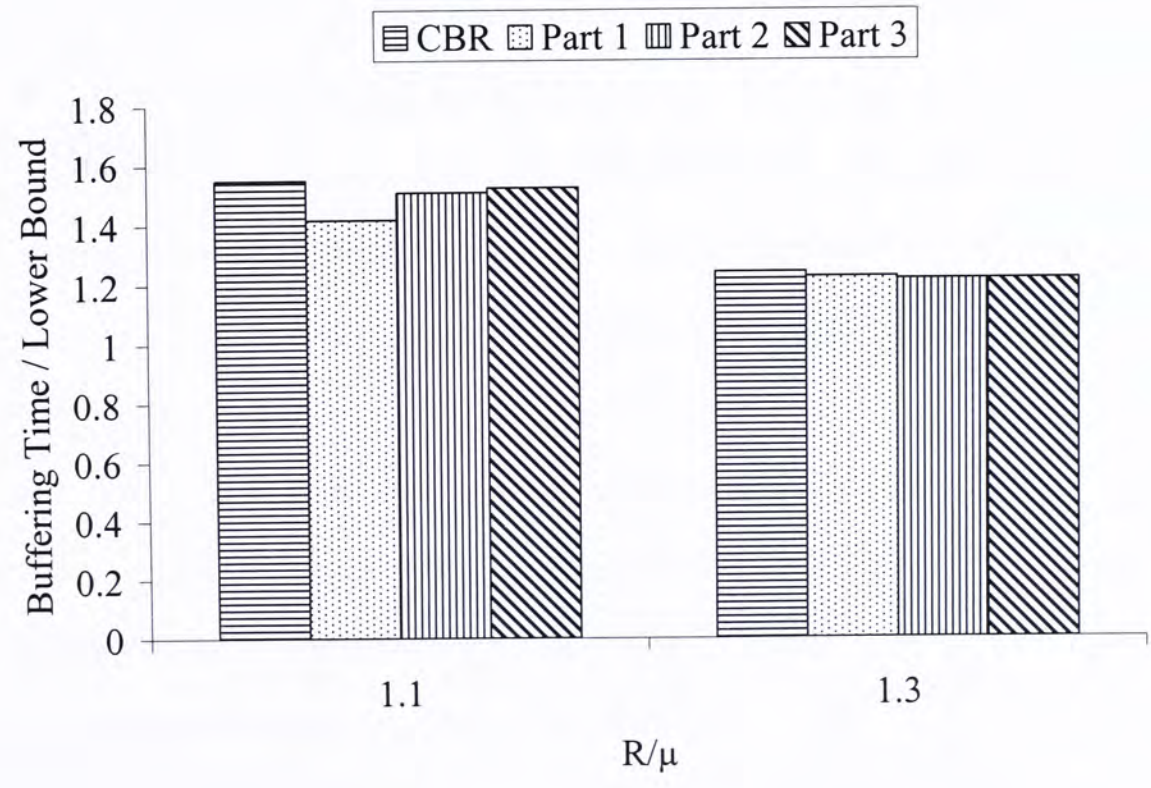

Fig. 38. The ratio of the buffering time to the lower bound (successful runs only).

Chapter 8

FUTURE WORK

The predictive buffering algorithm presented in Chapter 6 and 7 shows how we can apply the many-to-one data transfer model to provide probabilistic performance guarantees. It is merely one of the simplest applications. There are still a number of works can be done to extend the usage and improve the performance of the proposed algorithm.

First, the predictive buffering algorithm assumes the video length and video bit-rate profile (for VBR video) is known in advance. Without this information it will be impossible to estimate the future bandwidth assumption. This could be case if the streaming content is encoded live in real-time. On the other hand, for very long video such as video conference proceedings, the bandwidth estimated at the beginning of the video session may eventually deviate after a long period of time (e.g., tens of hours) and thus may eventually cause playback interruptions. To tackle these problems it is necessary to adopt post-playback adaptation in addition to pre-playback prediction. In Section 8.1 we discuss one possible approach through the use of playback rate adaptation and content adaptation.

Second, so far we only employed a simple random sender selection algorithm in Chapter 7. When there are more than enough senders to choose from, we can expect different selections of the set of senders will likely affect the final streaming

performance. For example, if the resultant sender set exhibits less bandwidth variation then the performance should be better (c.f. Section 8.2). Furthermore, if multiple users receive data through a proxy server then the proxy server can also monitor the flow properties continuously and when needed, further optimize performance by reallocating the senders dynamically (c.f. Section 8.3 – 8.4).

Finally, we discuss in Section 8.5 the challenges to applying our results in peer-to-peer (P2P) applications.

8.1 Playback Rate Adaptation

In the predictive buffering algorithm, once playback starts the algorithm completes and so any future bandwidth availability information will not be used unless there are playback interruptions. In order to support real-time encoded live video feeds that have unpredictable bit-rate variations and to adapt to very long time scale bandwidth fluctuations, playback rate adaptation [36-37] and content adaptation algorithms [21-22] can be integrated with the predictive buffering algorithm.

Specifically, after playback has begun, the client will continue to measure the statistical properties of the aggregate data flow. At periodic time intervals the client will re-predict the future bandwidth availability and check if the constraints in (45) or (47) are still satisfied. If not then it will slow down the video playback rate, e.g., by decreasing the video frame rate and using time-scale modification [38] to elongate the audio, to compensate for the bandwidth reductions. The playback rate will be returned to normal once the playback constraints are satisfied again.

On the other hand, if the video is encoded using scalable codec such as multiple description coding (MDC) [39-40] or if online transcoding [25] is available, the

client can feedback the bandwidth information to the streaming servers which can then adaptively fine-tune the video bit-rate to transmit to the client.

8.2 Sender Selection Algorithm

In our measurements and simulations, the senders are picked up randomly from the pool of available senders. In practice the system/client could keep historical information of senders' bandwidth properties and exploit that in selecting senders for a new video session. As reported in Section 4.6, properties of a substantial portion of the senders exhibit high-degree of correlations over very long time scales (e.g., months). Together with the findings in Section 5.3, which showed that we can minimize bandwidth variation of the resultant many-to-one data flow by carefully matching the correlations between two senders (ρ) and the ratio of two senders' standard deviations (r), it is possible to design an optimization algorithm to select the subset of senders that will result in minimal or at least lower bandwidth variations.

One approach is to focus on one of the two factors in the sender-selection process. Specifically, a sender is first randomly selected from a pool of senders. Next we find another sender whose value of ρ (correlation between itself and the selected sender) is smallest or r (the ratio to the selected sender) is closest to 1, and then combine the two senders as one, i.e., treat their combined aggregate flow as a single flow. We repeat this process by adding more and more senders to the many-to-one flow until sufficient number of senders is selected. Thus this is a simple greedy algorithm that optimizes one of the two sender selection criteria (i.e., either ρ or r). To further refine this algorithm we can modify it to select M candidate senders according to one criterion (say ρ), and then from these M candidates select one to be combined

according to the other criterion (say r). Similarly this process is to be repeated until sufficient number of senders is selected.

In addition to the greedy algorithm approach it is also possible to approach the problem as a general optimization problem, where a subset of senders is to be selected to optimize given performance objectives (e.g., buffering time) subject to system constraints. Clearly this is not a trivial optimization problem due to the non-linearity in the formulations and thus warrants further research to study the existence of, and the algorithm to obtain optimal or near-optimal solutions.

8.3 Dynamic Flow Allocation

So far we have assumed that the client interacts with the senders directly (Fig. 39a). In some network environments such as a corporation, an ISP, or a university campus, it may not be the complete picture. In particular, to reduce Internet bandwidth usage, such network environments often employ one or more proxy servers to intercept and forward client requests to the servers, and then caching and forwarding the data received from the servers to the clients (Fig. 39b).

Now as all traffic passes through the proxy, the proxy server can measure the statistical bandwidth properties of all the servers across all the many-to-one flows. This equipped the proxy server with far more extensive information of the servers' bandwidth properties, both current and historical. This information can then be used to guide the selection of senders as described in Section 8.2, or take one step further – to select and even dynamically reallocate the right combination of senders on behalf of the clients.

For example, consider two clients where each of them is served by two streaming servers, say Server 1 and 2 for Client 1, and Server 3 and 4 for Client 2. Assume the

Client 2. Obviously this requires that the contents are replicated across t servers – not uncommon in large content providers using content distrib networks, and that the servers support selective delivery of data – feasible for m types of servers, including HTTP servers and some media servers. Neverthele further investigations are needed to study the gains and tradeoffs of this approach.

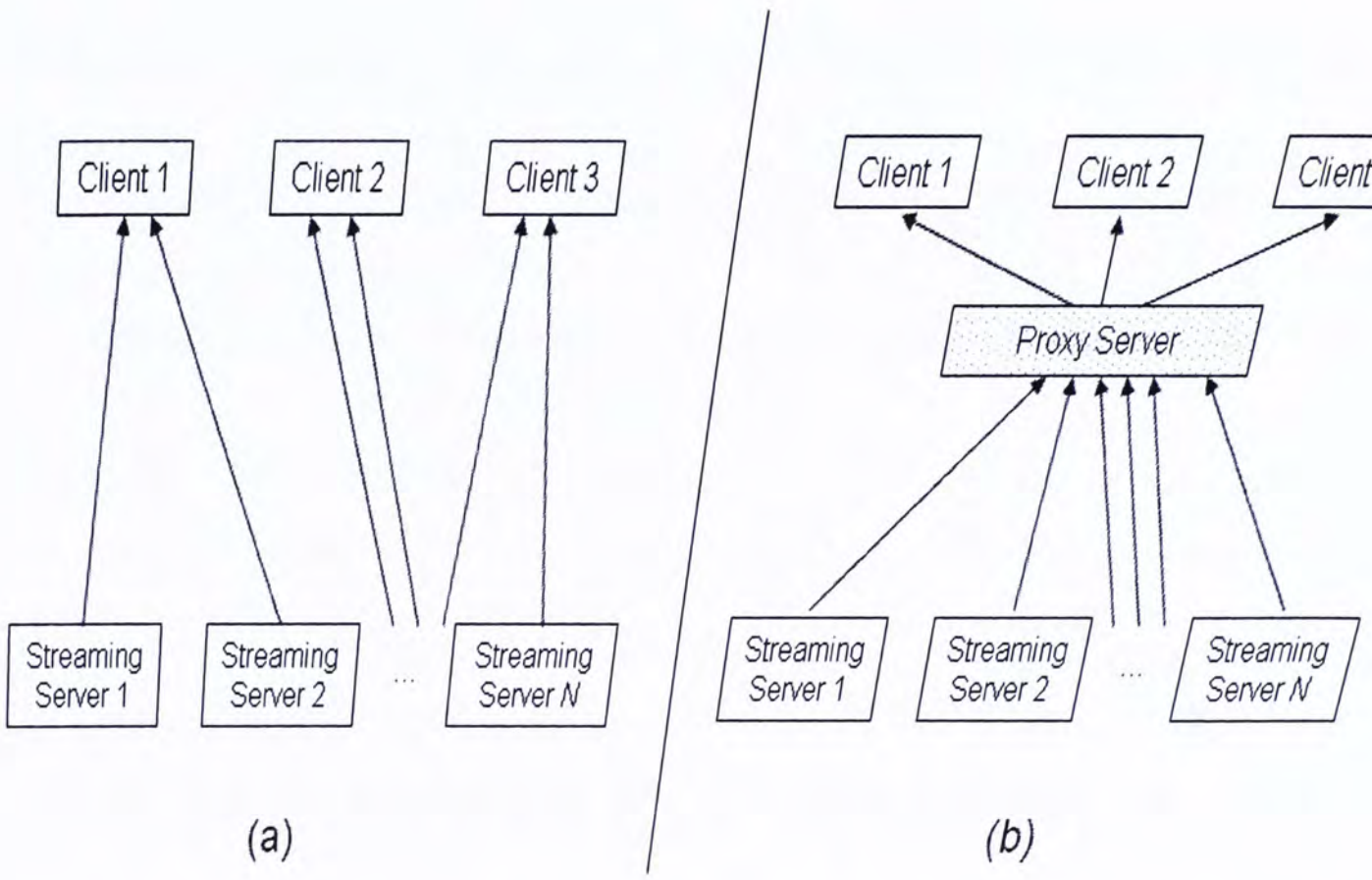

Fig. 39. Media data delivery from the streaming servers to the clients (a) without proxy and (b) with proxy.

8.4 Predictive *Flow Allocation*

Taking the idea in Section 8.3 one step further the proxy server can implement even more intelligent algorithms to improve system performance. Specifically, the proxy server could periodically review the state of a many-to-one session to identify

sessions which turn out to have received more than sufficient bandwidth, e.g., large amount of data have been accumulated at the receiver. When admitting new streaming session the proxy server can then take-away some of the senders from these existing *wealthy* sessions subject to the constraint that the target quality of service of these wealthy sessions are not compromised, and reallocate them to the new streaming session to reduce the buffering time, or reallocate them to other existing *poor* sessions which are suffering or predicted to suffer from insufficient bandwidth.

This predictive flow allocation approach is particularly suitable for VBR video streaming as some VBR videos have higher than average bit-rate at the beginning, which requires more senders to achieve an acceptable startup delay. Once the peak bandwidth demands are over, these VBR video streaming sessions will have more than sufficient senders, which could then be reallocated to other streaming sessions to improve overall system performance.

8.5 Challenge in P2P Applications

Our work studies the characteristics of aggregate data flow from multiple senders. Consider a P2P application, the peers exchange data with each others and most of the time peers receive data from multiple peers simultaneously. Thus, our results seem to be naturally applicable to this type of applications. However, there are still many challenges when applying the many-to-one data flow model in P2P applications.

Specifically, in a P2P system peers are highly dynamic and unlike dedicated server, peers may join and leave the system at any time. Thus the data flow model will not only need to handle bandwidth fluctuations, but also changes of the sources

themselves. On the other hand, P2P systems tend to have a large number of peers available and thus a many-to-one data flow can be constructed from a larger number of sources to reduce the impact of peer departure. Together with a dynamic source discovery scheme it is thus possible to replace departed peers with other available peers to maintain the data flow's performance. More research is warranted to investigate these challenges further and to develop practical solutions for the future P2P systems.

Chapter 9

CONCLUSION

In this work, we studied the characteristics of achievable bandwidth in many-to-one data flows through extensive measurements conducted in the PlanetLab and in the Internet. Our results confirmed that the achievable bandwidth of individual one-to-one data flows can vary over a very wide range and is very difficult to predict accurately based on past measurements. By contrast, the aggregate achievable bandwidth of a many-to-one data flow exhibit significantly more consistent properties over a time scale of hours. Moreover, increasing the number of senders in a many-to-one data flow guarantees that the bandwidth variations will reduce and the future bandwidth will be more predictable. These two findings are of particular importance to bandwidth-sensitive applications.

By applying the many-to-one data transfer model, we developed a novel predictive buffering algorithm for multi-source video streaming. The proposed algorithm incorporates the impact of variations in the network available bandwidth and uses that knowledge to inform the buffering operation. The trace-driven simulation results show that the predictive buffering algorithm can achieve very high successful playback ratio while keeping the buffering time short.

BIBLIOGRAPHY

[1] V. Jocobson. "Pathchar: A Tool to Infer Characteristics of Internet Paths," ftp://ftp.ee.lbl.gov/pathchar/, Apr. 1997.

[2] A. Downey, "Using Pathchar to Estimate Internet Link Characteristics," in *Proc. of ACM SIGCOMM*, Sep, 1999.

[3] B. A. Mah, "pchar: a Tool for Measuring Internet Path Characteristics," http://www.kitchenlab.org/www/bmah/Software/pchar/, Feb. 2005.

[4] R. L. Carter and M. E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," *Performance Evaluation*, vol.27-28, Oct 1996, pp.297-318.

[5] C. Dovrolis, P. Ramanathan, and D. Moore, "What do Packet Dispersion Techniques Measure?" *Proc. of IEEE INFOCOM*, Apr. 2001, pp.905-914.

[6] M. Jain and C. Dovrolis, "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput," *IEEE/ACM Trans. on Networking*, vol.11, no. 4, Aug. 2003, pp.537-549.

[7] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE Journal on Selected Areas in Communications*, vol.21, no.6, Aug. 2003, pp.879-894.

[8] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson,"On the Self-similar Nature of Ethernet Traffic," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, February 1994, pp.1–15.

[9] V. Paxson and S. Floyd, "Wide-area Traffic: The Failure of Poisson Modeling," *Proc. ACM SIGCOMM'94*, London, England, UK, 1994, pp.257–268.

[10] R. Morris and D. Lin, "Variance of Aggregated Web Traffic," *Proc. IEEE INFOCOM'2000*, Tel Aviv, Israel, 2000, pp.360–366.

[11] J. Kilpi and I. Norros, "Testing the Gaussian Approximation of Aggregate Traffic," *Proc. of Internet Measurement Workshop (IMW)*, Nov. 2002.

[12] M. F. T. Karagiannis, M. Molle and A. Broido, "A Nonstationary Poisson View of Internet Traffic," *Proc. of IEEE INFOCOM*, Mar. 2004, vol.3, pp.1558-1569.

[13] S. C. Hui and J. Y. B. Lee, "Modeling of Aggregate Available Bandwidth in Many-to-One Data Transfer," *Proc. of the Fourth International Conference on Intelligent Multimedia Computing and Networking*, July 21-26, 2005, Utah, USA.

[14] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *Comp Comm Review*, vol. 33(3), pp. 3-12, July 2003.

[15] Many-to-One Measurement Data Archive available at http://many2one.mclab.info.

[16] *IEEE Std. 1003.1-2001, Standard for Information Technology -- Portable Operating System Interface (POSIX). Open Group Technical Standard: Base Specifications, Issue 6*, December 2001. ISO/IEC 9945:2002.

[17] J. Bourne, D. Burstein, *DSL: A Wiley Tech Brief*, John Wiley and Sons, 2002.

[18] tcpdump Homepage : http://www.tcpdump.org/.

[19] D. Arifler, G. de Veciana, and B. L. Evans, "Inferring path sharing based on flow level TCP measurements," *Proc. IEEE Int. Conf. on Communications*, Paris, France, June 2004.

[20] L. Wang, J. N. Griffioen, K. L. Calvert and S. Shi, "Passive inference of path correlation," *Proc. of the 14th international workshop on Network and operating systems support for digital audio and video (2004)*, pp. 36–41.

[21] P. de Cuetos and K.W. Ross, "Adaptive Rate Control for Streaming Stored Fine-Grained Scalable Video," Proc. NOSSDAV, May 2002, pp.3-12.

[22] P. de Cuetos, P. Guillotel, K.W. Ross and D. Thoreau, "Implementation of Adaptive Streaming Of Stored MPEG-4 FGS Video Over TCP," Proc. ICME 2002, pp.405-408.

[23] S. Jacobs and A. Eleftheriadis, "Streaming Video using Dynamic Rate Shaping and TCP Congestion Control," Journal of Visual Comm and Image Representation, Vol. 9, No. 3, 1998, pp.211-222.

[24] L. S. Lam, Jack Y. B. Lee, S. C. Liew, and W. Wang, "A Transparent Rate Adaptation Algorithm for Streaming Video over the Internet," Proc. 18th International Conference on Advanced Information Networking and Applications, Fukuoka, Japan, March 29-31, 2004.

[25] A. Vetro, C. Christopoulos, H. F. Sun, "Video Transcoding Architectures and Techniques: An Overview," IEEE Signal Process Magazine, vol.20(2), March 2003, pp.18–29.

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

[27] Nguyen and A. Zakhor, "Distributed Video Streaming with forward error correction," Packet Video Workshop, PA, USA, April 2002

[28] D.Y. Xu, M. Hefeeda, S. Hambrusch and B. Bhargava, "On Peer-to-Peer Media Streaming," Proc. Int'l Conf on Distributed Computing Systems 2002, Vienna,

Austria, pp.363-371, July 2002.

[29] J. B. Kwon and H. Y. Yeom, "Distributed Multimedia Streaming over Peer-to-Peer Network," Proc. 9th Int'l Conf on Parallel and Distributed Computing, Klagenfurt, Austria, August 2003.

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

[31] M. Reisslein and K.W. Ross, "Call Admission for Prerecorded Sources with Packet Loss," IEEE Journal Selected Areas in Communications, vol. 15, pp.1167-1180, August 1997.

[32] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," RFC 3448, January 2003.

[33] L. Lapin, Modern Engineering Statistics, Duxbury Press, 1997.

[34] NS2 official homepage at http://www.isi.edu/nsnam/ns/.

[35] NLANR PMA data set: http://pma.nlanr.net/Traces/long/bell1.html.

[36] S. C. Hui and Jack Y. B. Lee, "Playback-Adaptive Multi-Source Video Streaming," Proc. of the 4th Int'l Conf on Intelligent Multimedia Computing and Networking, July 21-26, 2005, Utah, USA.

[37] M. Kalman, E. Steinbach, and B. Girod, "Adaptive Media Playout for Low-Delay Video Streaming Over Error-Prone Channels," IEEE Trans on Circuits and Systems for Video Technology, vol.14(6), June 2004, pp.841-851.

[38] 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.

[39] A. R. Reibman, H. Jafarkhani, Y. Wang, M. T. Orchard, and R. Puri R, "Multiple Description Coding for Video Using Motion Compensated Prediction," Proc. International Conference on Image Processing, Oct 1999, Kobe, Japan, vol.3, pp.837–41.

[40] E. Setton, Y. Liang, B. Girod, "Adaptive Multiple Description Video Streaming over Multiple Channels with Active Probing," Proc. International Conference on Multimedia and Expo, Baltimore, Maryland, July 2003, vol.1, pp.509-12.