CHARACTERIZING END-TO-END INTERNET PERFORMANCE

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yin Zhang

August 2001

CHARACTERIZING END-TO-END INTERNET PERFORMANCE

Yin Zhang, Ph.D.

Cornell University 2001

The Internet has undergone dramatic growth in connectivity, use, and quality of service over the past several years. As this growth continues and the Internet is used for increasingly diverse and demanding purposes, it is vital to understand how the network and applications behave across a wide range of scenarios.

In this thesis, we have attempted to characterize end-to-end Internet performance at both the network layer and the application layer. At the network layer, we concentrate on the fundamental question of the constancy of Internet path properties. Today's networking practices often assume various forms of constancy of the underlying network properties. In order to provide general guidance about when and over what time scales these assumptions are valid, we have gathered a large measurement dataset using the NIMI infrastructure and then examined various notions of constancy in the context of four core Internet path properties: routes, loss, delay, and throughput.

At the application layer, we provide a detailed discussion and analysis of a large-scale Internet application, content distribution networks (CDNs). CDNs have recently emerged as a popular mechanism for delivering content on behalf of the origin Web sites. To test whether CDNs improve Web performance in reality, we have conducted the first large-scale measurement based study on CDNs, focusing on how they are used on the Web and how well they perform in terms of client

download time. Our study not only is timely, but also provides a baseline on an effective way to evaluate large-scale Internet systems and applications.

Throughout our study, we have paid considerable effort to deal with various measurement and analysis difficulties introduced by the tremendous diversity of the Internet. We hope that such effort has allowed us to draw a realistic picture of today's end-to-end Internet performance and to provide useful insights on how to effectively design and provision the future network. Meanwhile, we have attempted to gather the right set of concepts and tools that can be applied to understand various aspects of end-to-end Internet performance even when the network traffic condition changes. We hope these fundamental concepts and tools might have long-lived value.

Biographical Sketch

Yin Zhang was born in 1974, the Year of Tiger according to the Chinese Zodiac—in Chinese, *Yin* means the *Tiger*. Partly influenced by his parents, who are both electrical engineers, he has been very interested in mathematics ever since he learned how to count. Solving interesting mathematics problems had been one of his favorite pastimes throughout his childhood. During his school years, he won numerous awards in many mathematics competitions. In 1991, he became the national champion in the Chinese National Mathematics Competition. In 1992, he was selected as one of the six representatives of China to participate in the 33rd International Mathematics Olympiad (IMO) held in Moscow. He won a Gold Medal at the IMO and his score (41 out of 42) ranked fifth among over three hundred contestants from all over the world. In recognition of his excellent performance at the IMO, he was admitted by Peking University (PKU) without entrance examinations. Fascinated by the explosion of information technology, he decided to become a student in the Department of Computer Science and Technology. In 1997 he finished his B.S. and went to Cornell University to pursue his Ph.D. degree in Computer Science. At Cornell University, he worked in the field of computer networking under the advice of Dr. Srinivasan Keshav and Dr. Robbert van Renesse. Between June 1999 and February 2001, he worked as a research intern at the AT&T

Center for Internet Research at ICSI (ACIRI), collaborating with Dr. Vern Paxson and several other networking researchers. His research at ACIRI spanned several areas in computer networking—network measurement, content distribution, and Internet security—and forms the basis of his PhD thesis.

To Mum, Dad, Sister, and Lili.

Acknowledgments

First and foremost, my deep appreciation goes to my advisors, Dr. Robbert van Renesse and Dr. Vern Paxson. Robbert patiently read and marked up my thesis drafts, and gave me invaluable advice and suggestions that have significantly improved the quality of this thesis. Vern is a remarkable advisor in many aspects. He interacts closely with the students, but meanwhile grants them a lot of freedom to explore new ideas. From Vern, I have learned how excellent networking research is done — rigorous, energetic, open to new ideas, and dedicated to research. I greatly appreciate and enjoy the collaboration with Vern.

I would also like to thank Professor Zygmunt Haas and Professor Sam Toueg for serving on my special committee. Their excellent instruction and technical advice have significantly broadened my view in wireless networking and distributed systems. I would also like to thank Professor Kenneth P. Birman for serving as proxy for Professor Sam Toueg in my defense.

I am very grateful to my former advisor Dr. Srinivasan Keshav, who introduced me to the wonderful world of computer networking and initiated my passion in doing research in this area. He also kindly arranged internships for me at Bell Labs and the AT&T Center for Internet Research at ICSI (ACIRI).

I greatly benefited from the fruitful collaboration with Dr. Nick Duffield, Dr.

Balachander Krishnamurthy, Dr. Scott Shenker, and Dr. Craig Wills, which has contributed significantly to this thesis. I also got valuable comments and suggestions on various aspects of this thesis from Dr. Lee Breslau, Dr. Mark Handley, Dr. Lili Qiu, and anonymous reviewers. I would like to thank them all.

I feel very fortunate to have great colleagues and friends during my stay at Cornell. In particular, I would like to thank Zhiyuan Chen, Li Li, Wei-Tsang Ooi, Lili Qiu, Jia Wang, Zhen Xiao, Yu Zhang, and Lidong Zhou.

I spent one and a half years visiting the AT&T Center for Internet Research at ICSI (ACIRI). I had a wonderful time with the stellar team of networking researchers at ACIRI, including Scott Shenker, Sally Floyd, Mark Handley, Jitendra Padhye, Sylvia Ratnasamy, and Vern Paxson. I also enjoyed the friendship with the other interns at ACIRI, including Deepak Bansal, Neal Cardwell, Ramakrishna Gummadi, Orion Hodson, Brad Karp, Dina Katabi, Li Li, Ratul Mahajan, and Hongsuda Tangamunrunkit.

I owe a special gratitude to my family and my girlfriend Lili Qiu. My parents and sister have always given me selfless care, support, and love. Their passion for career and dedication to work have also been role models for me, and always inspired me to excel. Lili is both my collaborator and my girlfriend. Over the past three years, we have shared a lot in our lives — passion, love, happiness, worries, and even pain. It is her constant love, support, and inspiration that makes my dissertation quest such an enjoyable and memorable experience in my life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the past several years, we have witnessed a dramatic growth of the Internet in terms of connectivity, use, and the development of novel network services. As this growth continues and the Internet is used for increasingly diverse and demanding purposes, it is crucial to collect and analyze data about a range of network functions, from low-level Internet path properties, to the performance of large-scale Internet systems and applications. Such data analysis is critical to understanding how the network and applications behave, and how to effectively design and provision the future network.

1.1 The growth of the Internet

The Internet, as we know it today, is an outgrowth of ARPANET, a research network sponsored by the U.S. Department of Defense. The original configuration in December 1969 consisted of four nodes (hosts), all located in the U.S. The ARPANET traffic was estimated to be a few thousand bytes per month at its inception. Since then traffic has grown exponentially at a rate of approximately

100% per year, and shows no sign of stopping. The only exception occurred during 1995 and 1996, when Internet traffic exhibited spectacular growth—doubling traffic every three or four months, which corresponds to an annual growth of around 1,000%. By the end of 1999, the Internet traffic on U.S. long distance networks had grown to 10,000 - 16,000 Terabytes per month [14]. By August 2001, the number of hosts on the Internet had grown beyond 117 million [69]!

The growth of the Internet has facilitated the explosive growth of Internet applications, which in turn further accelerates the growth of the Internet. Today, the dominant Internet application is the World Wide Web (WWW). Recent measurements show that Web traffic constitutes over 70% of the total on wide-area backbone links in the Internet [70]. The ease of use of major Web browsers Netscape, and Internet Explorer has often been cited as an important factor in the continued growth and popularity of the Web and the Internet. The graphical user interface (GUI) of web browers is inviting, and the point-and-click ease of hypertext links makes accessing information easy enough for most young children to explore with confidence.

The content delivered by the Web is quickly evolving. In the early days of the Web, content was mostly text, which changed in a few years to be mostly images [23]. Recently, the content mix has changed to include streaming media. Our study [35] and another study [10] show that although streaming media is a small fraction of the number of resources, it can contribute to a significant fraction of the bytes. The recent dramatic increase in interest in peer-to-peer networks (e.g. Napster [46], Gnutella [22]) has led to significant increase in streaming content. Delivering large streaming media resources raises new performance challenges and can potentially alter Internet traffic dynamics.

Meanwhile, new Internet systems and applications are emerging. One example is the content distribution networks (CDNs), which have also received a lot of attention recently. A content distribution network is a mechanism for delivering content to end users on behalf of origin Web servers. It offloads work from origin servers and moves content closer to end users by serving some or all of the contents of Web pages using a number of techniques (which will be described in Section 3.1) Today an appreciable amount of Web content is being served in this fashion.

1.2 Challenges

The continuing growth of the Internet has made it increasingly challenging to measure and characterize Internet performance. Below we describe three major difficulties.

1. Part of the difficulty in characterizing Internet performance is due to how quickly the network changes. Depending on the metric of interest, the network grows by 80% to 100% each year, and has sustained this growth for well over a decade. Furthermore, the dominant protocols and their patterns of use can change radically over just a few years, or even a few months [52, 11]. Consequently, the value of detailed results from measurements often tend to be short-lived.

2. Another difficulty is the network's substantial—and increasing—heterogeneity. As of this writing, the Internet has over 110 million hosts interconnected through a wide variety of link technologies [69]. Therefore, it is difficult to measure a plausible representative cross-section of Internet behavior.

3. The third difficulty comes from the different layers involved in network protocols and applications. Protocol layering is critical for building, operating, and maintaining networks. However, the presence of multiple layers also increases the analysis complexity. Different protocol layers may have different effects on the network performance. Sometimes, the subtle interaction across certain layers (e.g. TCP and HTTP) can also have significant performance impact. This means, in order to thoroughly understand the behavior of the network and applications, we need to open up the "box" and take into account all the relevant protocol layers.

1.3 Thesis overview

In this thesis, we have attempted to measure and characterize end-to-end Internet performance at both the network layer and the application layer. At the network layer, we concentrate on the fundamental question of the constancy of Internet path properties. At the application layer, we provide a timely discussion and analysis of a particularly large-scale Internet application—content distribution networks (CDNs), focusing on how CDNs are used on the Web and how they perform. Below we give an overview of each part and then describe the common methodology that unites them.

1.3.1 The constancy of Internet path properties

Many Internet protocols and operational procedures use measurements to guide future actions. This is an effective strategy if the quantities being measured exhibit a degree of *constancy*: that is, in some fundamental sense, they are not changing.

Using a large measurement data set gathered from the NIMI infrastructure [57], we have explored various notions of constancy in the context of four key Internet path properties: routes, loss, delay, and throughput [73, 74]. Our aim is to provide guidance as to when assumptions of various forms of constancy are sound, versus when they might prove misleading.

Among our results in [73, 74], we have discussed how different notions of constancy sometimes overlap, and sometimes differ substantially. That they can differ substantially highlights how it remains essential to be clear which notion of constancy is relevant to the task at hand.

One surprise in our findings is that many of the processes are well-modeled as *independent and identically distributed* (IID), once we identify change-points in the process's median and aggregate fine-grained phenomena into episodes. IID models are a mixed blessing; they are very tractable, but IID processes are very hard to predict.

Another general finding is that almost all of the different classes of predictors frequently used in networking (moving average, EWMA, S-shaped moving average) produced very similar error levels. Sometimes the predictor performed well, such as when predicting RTTs, and sometimes poorly, because of the IID nature of the data (loss, throughput).

Finally, the answer to the question "how steady is the Internet?" depends greatly on the aspect of constancy and the dataset under consideration. However, it appears that for all three aspects of constancy we considered, and all four quantities we investigated, one can generally count on constancy on at least the time scale of minutes.

1.3.2 The use and performance of content distribution networks

Over the last few years, content distribution networks (CDNs) have emerged as a mechanism to deliver content to end users on behalf of origin Web sites. Yet, there has been little work to soundly evaluate the role of CDNs.

In our work [35], we examine the performance of CDNs from various perspectives. Our study uses multiple data streams with both static and streaming content: active measurements obtained via repeated crawls over a period of time, and passive measurements representing a large number of users from different organizations. As part of understanding performance, the study takes into account all the relevant protocols used by CDNs (DNS, TCP, HTTP) along with the protocol flavors and compliance exhibited by CDNs.

The study, conducted over a period of months on the most prominent set of CDN companies, shows that some of the widespread practices do not necessarily improve performance, and interactions across certain protocol layers merit closer attention. Among our results, we find a significant increase in CDN-served content over the past year. Meanwhile, there is considerable variation in the performance of different CDNs studied. A common technique, that of DNS-based redirection which requires frequent DNS lookups, generally cannot be justified in terms of improved client download times.

1.3.3 Themes and contributions

The two parts in our work are united by the common methodology we use to address the challenges identified in Section 1.2.

First of all, to cope with the great speed at which the network and applications evolve, we have attempted to conduct both studies over a relatively long period of time. More specifically, for the constancy study, our two main datasets were collected one year apart. We have also compared our routing stability results with results obtained from a similar study conducted six years ago [53]. For the study of CDNs, the primary experiments on CDN performance have been repeated multiple times over a period of several months. While our study is still relatively short, we feel it constitutes a step in the right direction.

Meanwhile, to tackle the substantial network heterogeneity, we conducted most of our measurements using the NIMI measurement infrastructure [57], in which a number of measurement platforms are deployed across the Internet and used to perform end-to-end measurements. For the constancy study, we coordinate measurements between pairs of NIMI sites, which allows us to measure end-to-end performance along $O(N^2)$ paths for a framework consisting of N sites. For the CDN study, we measure performance between all NIMI sites and all CDN and origin Web sites, covering a total of $O(N \times (M + C))$ paths, given N NIMI sites, M origin sites, and C CDN companies. The scaling property of our measurement infrastructure serves to measure a sufficiently diverse set of Internet paths, so that we might plausibly interpret the resulting analysis as accurately reflecting general Internet behavior.

Finally, to accommodate the different layers involved in network protocols and applications, we have used a multi-layer methodology throughout our work. First of all, the two distinct parts are complementary to each other in that they try to characterize Internet performance at two different layers—the network layer and the application layer. In addition, within each part, we have considered different levels of Internet performance. The four quantities being measured in the constancy study—routes, loss, delay, and throughput—represent three levels of Internet behavior. The routing data represents the stability—or constancy—of a very basic infrastructure. It is typically invisible to higher layers. In contrast, packet loss and delay are end-to-end path properties quite visible to the transport layer, but typically hidden from applications. Finally, throughput is precisely what many applications care about most. It can be thought of as the application-relevant manifestation of the underlying loss and delay behavior on a path. In the CDN study, the primary performance analysis focuses on the client download time, which is an application-layer performance metric. However, as part of understanding performance, we have also taken into account various protocols at different layers (TCP, DNS, HTTP). We hope that our holistic approach can provide a more complete picture of Internet behavior.

The contributions of this work can be viewed on two levels. On one level, by collecting and analyzing data about a wide range of functions, from low-level Internet path properties, to large-scale Internet systems, our study sheds light on today's Internet performance and can provide insights on how to design and provision the future network.

On another level, we have attempted to gather the right set of concepts and tools needed to understand different aspects of Internet performance. While the detailed results from our measurements may soon prove ephemeral (due to changing traffic conditions), or rendered obsolete (by subsequent and better measurement efforts), we hope that the fundamental concepts and tools developed here might prove longer-lived.

1.4 Thesis outline

The remainder of this thesis is organized as follows. In Chapter 2, we explore various notions of constancy in the context of four key Internet path properties: routing, loss, delay, and throughput. In Chapter 3, we give a thorough evaluation of the role of content distribution networks, focusing on how they are used on the Web and how they perform. We conclude with a summary of our major contributions in Chapter 4. Here we also point out avenues for future research. In Appendix A, we discuss various statistical tests involved in our study.

Chapter 2

The Constancy of Internet Path Properties

There has been a recent surge of interest in network measurements. These measurements have deepened our understanding of network behavior and led to more accurate and qualitatively different mathematical models of network traffic. Network measurements are also used in an operational sense by various protocols to monitor their current level of performance and take action when major changes are detected. For instance, RLM [40] monitors the packet loss rate and, if it crosses some threshold, decreases its transmission rate. In addition, several network protocols and algorithms use network measurements to predict future behavior. For example, TCP uses delay measurements to estimate the timeout value for detecting packet loss, and measurement-based admission control algorithms use measurements of past load to predict future loads.

Measurements are inherently bound to the present—they can merely report the state of the network at the time of the measurement. However, measurements are most valuable when they are a useful guide to the future; this occurs when the relevant network properties exhibit what we will term *constancy*. We use a new term for this notion, rather than an existing term like "stationarity", in an attempt to convey our goal of examining a broad, general view of the property "holds steady and does not change," rather than a specific mathematical or modeling view. We will also use the term *steady* for the same notion, when use of "constancy" would prove grammatically awkward.

In this section, we introduce three different notions of constancy: mathematical, operational, and predictive.

We say that a dataset of network measurements is *mathematically steady* if it can be described with a single time-invariant mathematical model. The simplest such example is describing the dataset using a single independent and identically distributed (IID) random variable. More complicated forms of constancy would involve correlations between the data points. More generally, if one posits that the dataset is well-described by some model with a certain set of parameters, then mathematical constancy is the statement that the dataset is consistent with that set of parameters throughout the dataset.

One example of mathematical constancy is the finding by Floyd and Paxson [56] that session arrivals are well described by a fixed-rate Poisson process over time scales of tens of minutes to an hour. However, they also found that session arrivals on longer time scales can only be well-modeled using Poisson processes if the rate parameter is adjusted to reflect diurnal load patterns, an example of mathematical *non-constancy*.

When analyzing mathematical constancy, the key is to find the appropriate model. Inappropriate models can lead to misleading claims of non-constancy because the model doesn't truly capture the process at hand. For instance, if one tried to fit a highly correlated but stationary arrival process to a Poisson model, it would appear that the Poisson arrival rate varied over time.

Testing for constancy of the underlying mathematical model is relevant for modeling purposes, but is often too severe a test for operational purposes because many non-constancies are completely irrelevant to protocols. For instance, if the loss rate on a path was completely constant at 10% for thirty minutes, but then changed abruptly to 10.1% for the next thirty minutes, one would have to conclude that the loss dataset was not mathematically steady, yet one would be hard-pressed to find an application that would care about such a change. Thus, one must adopt a different notion of constancy when addressing operational issues. The key criterion in operational, rather than mathematical, constancy is whether an application (or other operational entity) would care about the changes in the dataset. We will call a dataset *operationally steady* if the quantities of interest remain within bounds considered operationally equivalent. Note that while it is obvious that operational constancy does not imply mathematical constancy, it is also true that mathematical constancy does not imply operational constancy. For instance, if the loss process is a highly bimodal process with a high degree of correlation, but the loss rate in each mode does not change, nor does the transition probability from one mode to the other, then the process would be mathematically steady; but an application will see sharp transitions from low-loss to high-loss regimes and back which, from the application's perspective, is highly non-steady behavior.

Operational constancy involves changes (or the lack thereof) in perceived application performance. However, protocols and other network algorithms often make use of measurements on a finer level of granularity to predict future behavior. We will call a dataset *predictively steady* if past measurements allow one to reasonably predict future characteristics. As mentioned above, one can consider TCP's time-out calculation as using past delays to predict future delays, and measurement-based admission control algorithms do the same with loss and utilization. So unlike operational constancy, which concerns the degree to which the network remains in a particular operating regime, predictive constancy reflects the degree to which *changes* in path properties can be tracked.

Just as we can have operational constancy but not mathematical, or vice versa, we also can have predictive constancy and none or only one of the others, and vice versa. Indeed, as we will illustrate, processes exhibiting the simplest form of mathematical constancy, namely IID processes, are generally impossible to predict well, since there are no correlations in the process to leverage.

Another important point to consider is that for network behavior, we anticipate that constancy is a more useful concept for coarser time scales than for fine time scales. This is because the effects of numerous deterministic network mechanisms (media access, FIFO buffer drops, timer granularities, propagation delays) manifest themselves on fine time scales, often leading to abrupt shifts in behavior, rather than stochastic variations.

An important issue to then consider concerns different ways of how to look at our fine-grained measurements on scales more coarse than individual packets. One approach is to aggregate individual measurements into larger quantities, such as packets lost per second. This approach is quite useful, and we use it repeatedly in our study, but it is not ideal, since by aggregating we can lose insight into the underlying phenomena. An alternative approach is to attempt to *model* the fine-grained processes using a model that provides a form of aggregation. With this approach, as long as the model is sound, we can preserve the insight into the underlying phenomena because it has already been captured by the model.

For example, instead of analyzing packet loss per second, we show that individual loss events come in *episodes* of back-to-back losses (Section 2.3.4). We can then separately analyze the characteristics of individual loss episodes versus the constancy of the process of loss episode arrivals, retaining the insight that loss events often come back-to-back, which would be diminished or lost if we instead went directly to analyzing packets lost per second.

Another important distinction is between the concepts of *persistence* and *prevalence* [53]. Persistence reflects how long one set of characteristics will remain unchanged if observed continuously. Prevalence, in contrast, quantifies the percentage of time the system will exhibit a particular set of characteristics if observed sporadically. The context usually determines which aspect of constancy is more relevant. Also note that the two notions are orthogonal: you can have one and not the other, or both, or neither, depending on whether the property is short-lived or long-lived, and whether it tends to primarily manifest itself in many different ways or in just a few ways.

For the routing data we present, we focus mainly on prevalence. The motivating example is protocols which cache path information for the next time they access the same address; routing prevalence is a measure of how often an access to an address will travel the same route (and thus the cached information would be of use).

For the loss, delay, and throughput data, we concentrate on persistence. Our basic model for various time series is of piecewise steady regions delineated by *change-points*. With a parameterized family of models (e.g. Poisson processes with some rate), the time series in each change-free region (CFR) is modeled through a particular value of the parameter (e.g. the Poisson arrival rate). In fitting the time series to this model, we first identify the change-points. Within each CFR we determine whether the process can be modeled by IID processes. When occurring, independence can be viewed as a vindication of the approach to refocus to coarser time scales, showing the simplicity in modeling that can be achieved after removing small time scale correlations. Furthermore, we can test conformance of inter-event times with a Poisson model within each CFR. Given independence, this entails testing whether inter-event times follow an exponential distribution.

To focus on the network issues, we defer discussion of the statistical methodology for these tests—the presence of change-points, IID processes, and exponential inter-event times—to Appendix A. However, one important point to note is that the two tests we found in the literature for detecting change-points are not perfect. The first test—$\mathcal{CP}/RankOrder$—is *biased* towards sometimes finding extraneous change-points. The effect of the bias is to underestimate the duration of steady regions in our datasets. The second test—$\mathcal{CP}/Bootstrap$—does not have the bias. However, it is *less sensitive* and therefore misses actual change-points more often. The effect of the insensitivity is to overestimate the duration of steady regions and to underestimate the number of CFRs within which the underlying process can be modeled by IID processes. (See Appendix A.1 for a detailed Monte-Carlo based study on the performance of both tests.) To accommodate the imperfection, we apply both tests whenever appropriate and then compare the results. Our hope is to give some bound on the duration of steady regions.

The remainder of the chapter is organized as follows. We first describe the sources of data in Section 2.1. We discuss the routing data and its constancy analysis in Section 2.2. We then examine loss, delay, and throughput data in Section 2.3, 2.4, and 2.5. Of these three sections, the first one is much more detailed, as we develop a number of our analysis and presentation techniques therein. We then conclude in Section 2.6 with a brief summary of our results.

2.1 Measurement methodology

We gathered three basic types of measurements: routes, using the `traceroute` utility ([27]; see [67] for detailed discussion); Poisson packet streams (for assessment of loss and delay characteristics), using the `zing` utility that comes with the NIMI infrastructure (see below); and bulk throughput, using 1 MB TCP transfers.

Most of our measurements were made using the NIMI measurement infrastructure [57]. NIMI is a follow-on to Paxson's NPD measurement framework [53], in which a number of measurement platforms are deployed across the Internet and used to perform end-to-end measurements. NIMI attempts to address the limitations and resulting measurement biases present in NPD [54]. All the NIMI data analyzed in this chapter were captured either during Winter 1999–2000, or during Winter 2000–2001. For the first period, the infrastructure consisted of 31 hosts, 80% of which were located in the United States, and for the second, 49 hosts, 73% in the USA. About half are university sites, and most of the remainder research institutes of different kinds. Thus, the connectivity between the sites is strongly biased towards conditions in the USA, and is likely not representative of the commercial Internet in the large. That said, the paths between the sites do traverse the commercial Internet fairly often, and we might plausibly argue that our observations could apply fairly well to the better connected commercial Internet of the not-too-distant future, if not today.

In addition, to gain a broader view of Internet routing behavior, we made use of a pool of 189 public traceroute servers, a third located within the United States, and the other two-thirds spread across 31 different countries. Data from these sources is not as clean as that from the NIMI infrastructure, because the collection process suffers from some of the same biases as the NPD framework (failure to connect to the measurement server may preclude an opportunity to measure a network problem). The data is nevertheless valuable due to its rich diversity.

We gathered two sets of routing measurements during Winter 1999–2000, one from NIMI and one from the public traceroute servers mentioned above. For NIMI, we measured 36,724 routes, which included 707 of the 930 possible host pairs. Measurements were made at Poisson intervals with a mean of 10 minutes between measurements initiated by the same host. By using Poisson intervals, time averages computed using the measurements are unbiased [71].

12,655 of the measurements were made by pairing the source host with a random destination host in the mesh each time a new measurement was made; these measurements assured broad coverage of the mesh. The remaining 24,069 paired a single source with the same destination over the course of a day. These measurements were made as part of the `zing` packet data discussed in Section 2.3 below. Thus, this dataset gives us a fairly detailed look at a smaller number of Internet paths.

Using the public servers, we made 287,206 route measurements (so for both datasets we have an average of over 1,000 measurements from each host). Due to the size of the mesh, it was impractical to fully measure it in depth, so we split our measurements into one group scattered across the mesh, comprising 220,551 of the measurements, in an attempt to capture the breadth of routing anomalies, and another of 66,655 in-depth measurements of pairs, for assessing routing prevalence and persistence, similar to our NIMI measurements. The former set of measurements covered 97% of the mesh, with a median of 5 traceroutes per pair of hosts.

For Poisson packet streams and bulk throughput, we collected two main sets of data, one during Winter 1999–2000 ($\mathcal{W}_1$), and one during Winter 2000–2001 ($\mathcal{W}_2$), both using the NIMI infrastructure. For Poisson packet streams we used the "`zing`" utility, provided with the NIMI infrastructure, to source UDP packets at a mean rate of 10 Hz ($\mathcal{W}_1$) or 20 Hz ($\mathcal{W}_2$). For the first of these, we used 256 byte payloads, and for the second, 64 byte payloads. `zing` sends packets in selectable patterns (payload size, number of packets in back-to-back "flights," distribution of flight interarrivals), recording time of transmission and reception. While `zing` is capable of using a packet filter to gather kernel-level timestamps, for a variety of logistical problems this option does not work well on the current NIMI infrastructure, so we used user-level timestamps.

Again we used Poisson intervals for sending the packets, which makes time averages computed from the measurements unbiased [71]. Packets were sent for an hour between random pairs of NIMI hosts, and were recorded at both sender and receiver, with some streams being unidirectional and some bidirectional. We used the former to assess patterns of one-way packet loss based on the unique sequence number present in each `zing` packet, and the latter to assess both one-way loss and round-trip delay. We did not undertake any one-way delay analysis since the NIMI infrastructure does not provide synchronized clocks.

For throughput measurements we used TCP transfers between random pairs of NIMI hosts, making a 1 MB transfer between the same pair of hosts every minute

Table 2.1: Summary of datasets used in the study.

Dataset	# pkt traces	# pairs	# pkts	# thruput	# xfers
$\mathcal{W}_1$	2,375	244	160M	58	16,900
$\mathcal{W}_2$	1,602	670	113M	111	31,700

for a 5-hour period. We took as the total elapsed time of the transfer the interval observed at the receiver between accepting the TCP connection and completing the close of the connection. Transfers were made specifying 200 KB TCP windows, though some of the systems clamped the buffers at 64 KB because the systems were configured to not activate the TCP window scaling option [28]. The NIMI hosts all ran versions of either FreeBSD or NetBSD.

Table 2.1 summarizes the datasets $\mathcal{W}_1$ and $\mathcal{W}_2$. The second column gives the number of hour-long `zing` packet traces, the third the number of distinct pairs of NIMI hosts we measured (lower in $\mathcal{W}_1$ because we paired some of the hosts in $\mathcal{W}_1$ for an entire day, while all of the $\mathcal{W}_2$ measurements were made between hosts paired for one hour), and the total number of measured packets. The fifth column gives the number of throughput pairs we measured, each for 5 hours, and the corresponding number of 1 MB transfers we recorded.

In our preliminary analysis of $\mathcal{W}_1$, we discovered a deficiency of `zing` that biases our results somewhat: if the `zing` utility received a "No route to host" error condition, then it terminated. This means that if there was a significant connectivity outage that resulted in the `zing` host receiving an ICMP unreachable message, then `zing` stopped running at that point, and we missed a chance to further measure the problematic conditions. 47 of the $\mathcal{W}_1$ measurement hours (4%) suffered from this problem. We were able to salvage 6 as containing enough data to still warrant analysis; the others we rejected, though some would have been rejected anyway due to NIMI coordination problems. This omission means that the $\mathcal{W}_1$ data is, regrettably, biased towards underestimating significant network problems, and how they correlate with non-constancies. This problem was fixed prior to the $\mathcal{W}_2$ data collection.

One other anomaly in the measurements is that in $\mathcal{W}_2$ some of the senders and receivers were missynchronized, such that they were not running together for the entire hour. This mismatch could lead to series of packets at the beginning or ending of traces being reported as lost when in fact the problem was that the receiver was not running. We removed the anomaly by trimming the traces to begin with the first successfully received packet and end with the last such. This trimming potentially could bias our data towards underestimating loss outages; however, inspection of the traces and the loss statistics with and without the trimming convinced us that the bias is quite minor.

Finally, our focus in this paper is on constancy, but to soundly assess constancy first requires substantial work to detect pathologies and modal behavior in the data and, depending on their impact, factor these out. We then can identify quantities that are most appropriate to test for constancy. Thus, while our goal is lofty—understanding constancy—we necessarily devote considerable attention in our discussion to more mundane methodological issues.

2.2 Routing constancy

We begin our analysis with a look at the routing data. The routing data represents the stability—or constancy—of a very basic infrastructure. Compared to the other path properties (loss, delay, and throughput), routes are more lower-level—they are typically invisible to higher layers. The dataset we collected, and our analysis of it, is quite similar to that by Paxson in [53]. Whenever appropriate, we present detailed comparisons with the results of the previous work to illuminate long-term trends in route stability.

2.2.1 Routing pathologies

Following the approach used in Paxson's earlier Internet routing study [53], we begin our routing analysis with characterizations of unusual or non-functioning routing behavior, i.e., "pathologies." We do so with three goals: first, as a sanity check on the data, to ensure it is not plagued with problems; second, so we can distinguish between ordinary routing fluctuations and apparent fluctuations that in fact are instead pathologies; and third, to form an impression on whether the quality of Internet routing has changed since Paxson's study, which was based on data 4–5 years older than ours.

To do so, we first categorize three different types of problems that we can associate with a traceroute measurement: measurement failures (the tool did not run or failed to produce useful output); connectivity problems (an end user would notice that there was some sort of problem); and eccentricities (unusual behavior, but not likely to affect end-to-end performance), which we do not further analyze due to their limited impact on end-to-end performance. These last two categories are somewhat blurred in Paxson's analysis, but his pathologies were dominated by outages (30 seconds or more of no connectivity), which we categorize as a connectivity problem.

For the NIMI data, we restrict our pathology analysis to the 12,655 traceroute measurements of the random mesh, as these reflect broad, even coverage of the different routes, rather than restricted, detailed coverage of a small subset of the routes. Of these, about 10% were marred by measurement errors occurring on the NIMI host themselves, so missing this data is unlikely to bias our samples. Of the remainder, 6% exhibit connectivity problems, with nearly all of these being connectivity outages. This figure is *double* that of Paxson's, even though the NIMI sites should enjoy better connectivity than the NPD sites due to the higher prevalence of university and research labs with high-quality Internet connections.

However, the NIMI pathologies are heavily skewed by two sites. If we remove these as outliers, then the total pathology rate falls to 3.2%, still dominated by outages, with the next most common pathology being unresolved routing loops, but these being 20 times as rare. The 3.2% figure is virtually unchanged from Paxson's 1995 data, which had a 3.3% pathology rate. Some pathologies are much more rare (persistent loops), others somewhat more common (30+ sec. outages). All in all, we would conclude that routing has not gotten significantly worse, but neither has it improved; furthermore, we had to discard a pair of our sites to get there, while Paxson did not need to resort to removing pathology outliers.

To attempt to assess the quality of broader Internet routing, we analyzed the public traceroute server data, as follows. First, we again restricted our analysis to the measurements made with random pairing (220,551 total). For those, we found that 11% of measurements completely failed, and another 4% were incomplete due to the connection to the server failing before it delivered all of its data. Of the remainder, we found that 4.3% suffered from a connectivity problem, the most common of which were outages (2.0%) and rapid route changes (1.4%).

This indicates that for the general Internet, routing is degraded compared to that measured in 1995. But this may not be a fair comparison, since the dataset is only one-third USA sites, while Paxson's data was about two-thirds USA sites. To assess the effects of this discrepancy, we repeated the analysis but limiting it to the 33,018 measurements made between two USA sites. Of these, 12.5% failed or were incomplete, and of the remainder, 2% exhibited a connectivity problem, with almost all (1.6%) of these being outages. From this we conclude that the evidence is solid that routing has neither improved nor degraded significantly since 1995, in terms of routing problems.

2.2.2 Routing prevalence

As noted above, in general we can think about two types of consistency for a network path property, its prevalence and its persistence. In this section, we characterize the prevalence of Internet routes as manifest in our datasets: that is, how often the most commonly occurring ("dominant") route is observed. The finding in [53] concerning routing prevalence was that in general Internet paths were strongly dominated by a single route, though there was significant site-to-site variation.

To assess prevalence, we use the second type of measurements discussed above, namely repeated measurements between particular pairs of hosts. For the NIMI data, we had 50 or more successful, non-pathological measurements of 94 distinct paths (source/destination pairs), comprising a total of 17,627 measurements. For the public traceroute servers, we had 50 or more such measurements of 367 distinct paths, for a total of 52,872 measurements.

An important consideration when assessing routing stability (both prevalence and persistence) is how exactly to determine whether two routes are the same. The problem arises in two ways. First, traceroute measurements report IP addresses, and some routers do not always return the same address. For example, we have traceroutes in our datasets that differ only in one hop sometimes being reported as address 205.171.18.114 and other times as 205.171.5.129. However, both of these addresses have DNS entries as *sjo-core-01.inet.qwest.net*, and these do in fact refer to the same router. In addition, we sometimes have a similar situation in which the IP addresses resolve to *different*, but very similar, hostnames, such as *s8-0-0-14.nyc-bb5.cerf.net* and *s0-0-0-25.nyc-bb5.cerf.net*. These may be the same router, or they may in fact be different routers but ones which are co-located or at least functionally very similar. Accordingly, we might well argue that two routes that differ only by one of these two cases ought to be considered the same route, since either they are in fact the same route, or they at least should in many ways share the same properties. Consequently, we would like to merge the two addresses into the equivalent of a single router prior to performing our analysis.

We identified addresses A and B as a pair to merge if they occurred in the same positions in adjacent traceroutes, and their hostnames were either identical or agreed both in domain and in whatever geographic clues were present in the hostname (e.g., *nyc*). For borderline cases we allowed as additional evidence of equivalence the fact that the next hop in both traceroutes was identical. Our approach is similar to but less strict than the one used by Paxson [53], which required A and B to be the only different hop in adjacent traceroutes. For the NIMI data, we identified 64 equivalent addresses (out of 1,602 total), and for the public server data, 220 (out of 12,663 total)—enough in both cases to seriously skew our analysis were they not merged, since a number were frequently observed routers.

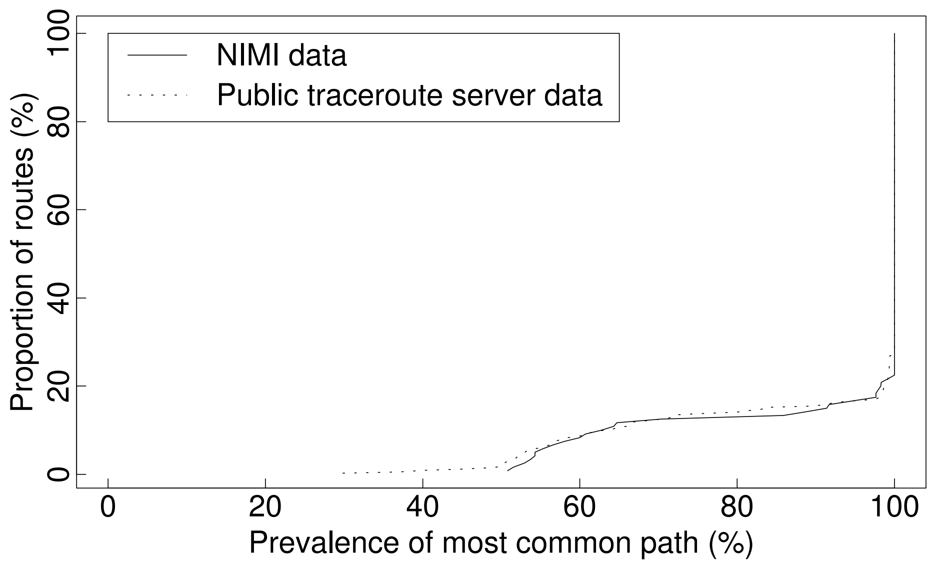


Figure 2.1: Routing prevalence in NIMI and public traceroute server datasets.

Figure 2.1 shows CDF's of the prevalence of the dominant route for the NIMI and public traceroute server datasets. For the NIMI routes, 78% always exhibited the same path, and 86% of the routes had a prevalence of 90% or higher. For the public servers, the corresponding figures are 73% and 85%, respectively.

These figures are considerably higher than those given by Paxson in [53]. The difference may reflect that routing has changed such that today the dominance effect is even stronger than in 1995, or it may reflect differing measurement methodologies; in particular, Paxson's data was spread over more days than ours. However, for the most obvious way to exploit routing prevalence—caching path properties for future use—it is plausible that the primary concern is the validity of routing prevalence over time scales of minutes to perhaps hours, a regime well covered by our data. In addition, the striking agreement between the two distributions taken from very different datasets suggests that the finding is well-grounded and quite plausibly general. Finally, we observe that even for the 15% of routes for which the dominant path does not completely dominate, it still is almost always observed the majority of the time, so it remains useful to cache information about its properties.

2.2.3 Routing persistence

Our basic approach for assessing routing persistence is to look at how many consecutive traceroute measurements each observed the same route. Because our measurements of the same route are made on average 10 min. apart, this approach is sound except when the route is rapidly changing, in which case we may miss a change to another, short-lived route that then changed back, all between our two measurements. [53] faced this problem too, and addressed it by first identifying paths with any evidence of rapidly changing routes and characterizing these separately. We follow the same approach.

For the NIMI data, there were 85 routes for which we had a successful series of day-long measurements. Of these, 8 exhibited rapid changes at some point, while for the public traceroute servers, 85 routes out of 383 did so.

Figure 2.2 gives the distribution of the route duration for the remaining routes. We see that very often routes persist for at least a day, the upper limit of what we can observe from our data. (The steps in the NIMI data reflects that some datasets were 22 or 23 hours long rather than a full 24 hours.) The long lower tail agrees with the finding for routing persistence in [53], namely that most paths are persistent over time scales of many hours to days, but a fair number of paths are persistent only for quite shorter time scales.

From the figure, we see that about 10% of the commercial Internet routes have lifetimes of a few hours or less, and about 5% of the NIMI routes (highlighting that the routing between the NIMI infrastructure is considerably more stable than

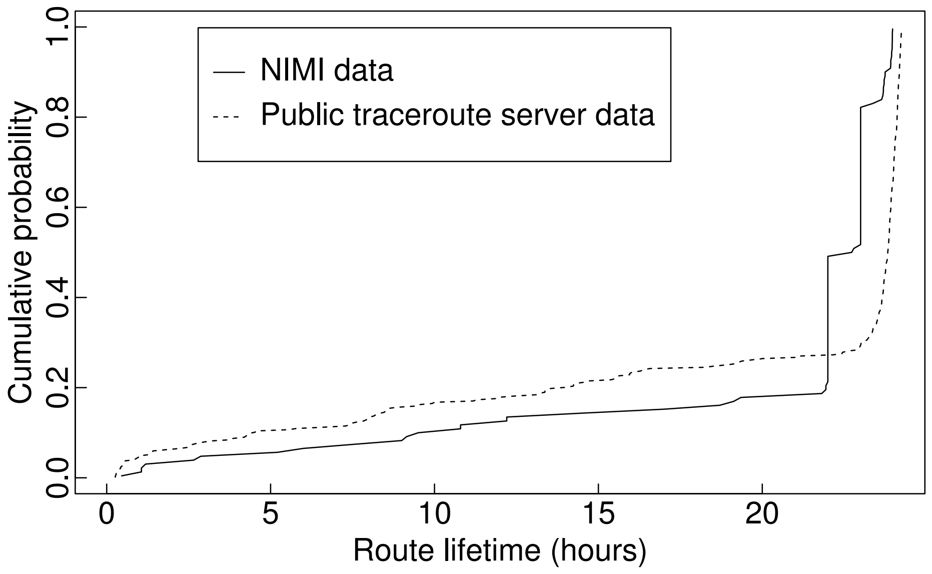


Figure 2.2: Routing persistence in NIMI and public traceroute server datasets, for routes not identified as exhibiting rapid changes.

that of the Internet in the large). When we include the routes we had to factor out of our persistence assessment because they exhibited rapid changes at some point, we find good evidence that a total of about 1/3 of Internet routes in general, and 1/6 of the NIMI routes, are short-lived.

2.3 Loss constancy

Now we turn to examine packet loss. We devote significantly more discussion to this section than to the subsequent sections analyzing delay and throughput because herein we develop a number of our analysis and presentation techniques.

Correlation in packet loss was previously studied in [9, 54, 72]. The first two of these focus on conditional loss probabilities of UDP packets and TCP data/ACK packets. [9] found that for packets sent with a spacing of $\leq$ 200ms, a packet was much more likely to be lost if the previous packet was lost, too. [54] found that for consecutive TCP packets, the second packet was likewise much more likely to be lost if the first one was. The studies did not investigate correlations on larger time scales than consecutive packets, however. [72] looked at the autocorrelation of a binary time series representation of the loss process observed in 128 hours of unicast and multicast packet traces. They found correlation time scales of 1000 ms or less. However, they also note that their approach tends to underestimate the correlation time scale.

While the focus of these studies was different from ours—in particular, [72] explicitly discarded non-steady samples—some of our results bear directly upon this previous work. In particular, in this section we verify the finding of correlations in the loss process, but also find that much of the correlation comes only from back-to-back loss episodes, and not from "nearby" losses. This in turn suggests that congestion epochs (times when router buffers are running nearly completely full) are quite short-lived, at least for paths that are not heavily congested.

As discussed in the previous section, we measured a large volume (270M) of Poisson packets sent between several hundred pairs of NIMI hosts, yielding binary-valued time series indexed by sending time and indicating whether each packet arrived at the receiver or failed to do so. For this analysis, we considered packets that arrived but with bad checksums as lost.

Packet loss in the datasets was in general low. Over all of $\mathcal{W}_1$, 0.87% of the packets were lost, and for $\mathcal{W}_2$, 0.60%. However, as is common with Internet behavior, we find a wide range: 11–15% of the traces experienced no loss; 47–52% had some loss, but at a rate of 0.1% or less; 21–24% had loss rates of 0.1–1.0%; 12–15% had loss rates of 1.0–10%; and 0.5–1% had loss rates exceeding 10%.

Because we sourced traffic in both directions during our measurement runs, the data affords us with an opportunity to assess symmetries in loss rates. We find for $\mathcal{W}_1$ that, similar to as reported in [54], loss rates in a path's two directions are only weakly correlated, with a coefficient of correlation of 0.10 for the 70% of traces that suffered some loss in both directions. However, the logarithms of the loss rates are strongly correlated (0.53), indicating that the order of magnitude of the loss rate is indeed fairly symmetric. While time-of-day and geographic (trans-continental versus intra-USA) effects contribute to the correlation, it remains present to a degree even with those effects removed. For $\mathcal{W}_2$, the effect is weaker: the coefficient of correlation is -0.01, and for the logarithm of the loss rate, 0.23.

2.3.1 Pathologies: reordering and replication

As with routing, before analyzing stability patterns in packet loss, we first assess the presence of unusual packet behavior. We again do this both as a sanity check on the data, and to compare with [54] to see if we can discern significant changes since 1995.

Three types of pathologies are characterized in [54]: out-of-order delivery, replication (the delivery of multiple copies of a single packet), and corruption. As our measurements were made at user-level, and hence only recorded packet arrivals with good UDP checksums, we cannot accurately assess corruption. (`zing` packets include an MD5 checksum, which never failed for our data.)

We first needed to remove 354 traces from our analysis because clock adjustments present in the trace rendered facets of reordering ambiguous. On the remainder we then used the same definition of reordering as in [54]—that is, packets arriving with a sending sequence number lower than a packet that arrived previously are counted as "late"and hence an instance of reordering. We find that about 0.3% of the 136 million packets arrived out of order, but that only 7% of our measured hours had no reordering at all. The highest reordering rate we observed sustained for one hour was 8.9%, and 25 datasets had rates exceeding 5% (three different sites dominated these). The largest reordering gap spanned 664 msec.

The 0.3% reordering rate is equal to the 1995 figure given in [54] of 0.3% of all data packets arriving out of order (and 0.1% for ACKs). However, we must be careful equating the agreement with a lack of change in reordering rates, because the data packets analyzed in [54] were often sent two back-to-back, due to TCP slow start and delayed acknowledgments acking every second packet, while our `zing` data was sent with an average of 50 msec. between packets (a mean sending rate of 10/sec. plus a mean reply rate to incoming `zing` packets of 10/sec), so the transit time difference to reorder our `zing` packets is quite high.

In agreement with [54], we find that reordering is dominated by just a few sites (the top three having seven times the median reordering rate), so another possible explanation of the relative increase in reordering we see is that it is simply due to chance in the selection of NIMI sites.

Also in agreement with [54], we find replication rare, with a total of 27 packets of the 160 million we studied arriving at the receiver more than once. This ratio is very low (significantly lower than in [54]), and accordingly does not merit further characterization.

2.3.2 Periodicities

Because of the frequent use of timers in network protocols, and because such timers can sometimes synchronize in surprising ways [20], it behooves us to analyze our loss data for periodicities, which form an important class of non-constancies. Based on our analysis in [74], we found persistent periodic loss behavior for packets sent to a particular NIMI site, "`nasa`". Consequently, we removed `nasa` from our subsequent loss-constancy analysis. We also identified several other periodicities in our data, with cycle times of 60, 90, and 300 sec, and one set of traces with at least two periodic loss processes active at the same time. However, none of these periodicities was as strong or as pervasive as that for `nasa`, and we judged the traces could be kept for our constancy analysis.

2.3.3 Individual loss vs. loss episodes

Previously we discussed how an investigation of mathematical constancy should incorporate looking for a good model. In this section, we apply this principle to understanding the constancy of packet loss processes.

The traditional approach for studying packet loss is to examine the behavior of individual losses [9, 45, 54, 72]. These studies found correlation at time scales below 200–1000 ms, and left open the question of independence at larger time scales. We introduce a simple refinement to such characterizations that allows us to identify these correlations as due to back-to-back loss rather than "nearby" loss, and we relate the result to the extended Gilbert loss model family [21, 60, 29]. We do so by considering not the loss process itself, but the loss *episode* process, i.e., the time series indicating when a series of consecutive packets (possibly only of length one) were lost.

For loss processes, we expect congestion-induced events to be clustered in time, so to assess independence among events, we use the autocorrelation-based Box-Ljung test developed in Appendix A.2, as it is sensitive to near-term correlations.

We chose the maximum lag k to be 10, sufficient for us to study the correlation at fine time scales. Moreover, to simplify the analysis, we use lag in packets instead of time when computing autocorrelations.

We first revisit the question of loss correlations as already addressed in the literature. In $\mathcal{W}_1$, for example, we examined a total of 2,168 traces, 265 of which has no loss at all. In the remaining 1,903 traces, only 27% are considered IID at 5% significance using the Box-Ljung Q statistic. The remaining traces show significant correlations at lags under 10, corresponding to time scales of 500–1000 ms, consistent with the findings in the literature.

These correlations imply that the loss process is not IID. We now consider an alternative possibility, that the loss *episode* process is IID, and, furthermore, is well modeled as a Poisson process. We again use Box-Ljung to test the hypothesis. Among the 1,903 traces with at least one loss episode, 64% are considered IID, significantly larger than the 27% for the loss process. Moreover, of the 1,380 traces classified as non-IID for the loss process, half have IID loss episode processes. In contrast, only 1% of the traces classified as IID for the loss process are classified as non-IID for the loss episode process.

Figure 2.3 illustrates the Poisson nature of the loss episode process for eight different datasets measured for the same host pair. The X-axis gives the length of the loss-free periods in each trace, which is essentially the loss episode interarrival time, since nearly all loss episodes consist of only one lost packet. The Y-axis gives the probability of observing a loss-free period of a given length or more, i.e., the complementary distribution function. Since the Y-axis is log-scaled, a straight line on this plot corresponds to an exponential distribution. Clearly, the loss episode interarrivals for each trace are consistent with exponential distributions,

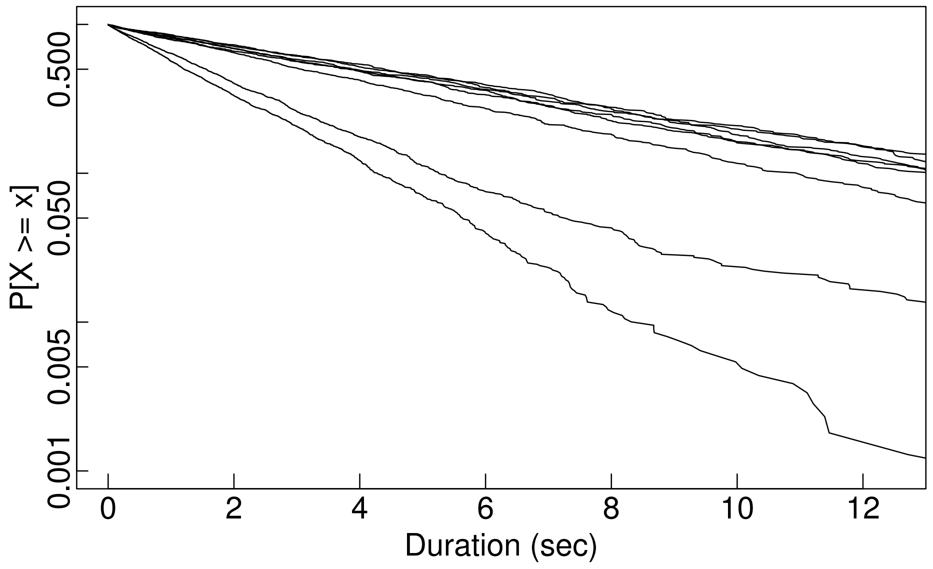


Figure 2.3: Example log-complementary distribution function plot of duration of loss-free runs.

even though the mean loss episode rate in the traces varies from 0.8%–2.7%, and this in turn argues strongly for Poisson loss episode arrivals.

If we increase the maximum lag in the Box-Ljung test to 100, the proportion of traces with IID loss processes drops slightly to 25%, while those with IID loss episodes falls to 55%. The decline illustrates that there is some non-negligible correlation over times scales of a few seconds, but even in its presence, the data becomes significantly better modeled as independent if we consider loss episodes rather than losses themselves.

If we continue out to still larger time scales, above roughly 10 sec, then we find exponential distributions become a considerably poorer fit for loss episode interarrivals; this effect is widespread across the traces. It does not, however, indicate correlations on time scales of 10's of seconds (which in fact we generally find are absent), but rather mixtures of exponentials arising from differing loss rates present at different parts of a trace, as discussed below. Note that, were we not open to considering a loss of constancy on these time scales, we might instead wind up misattributing the failure to fit to an exponential distribution as evidence of the need for a more complex, but steady, process.

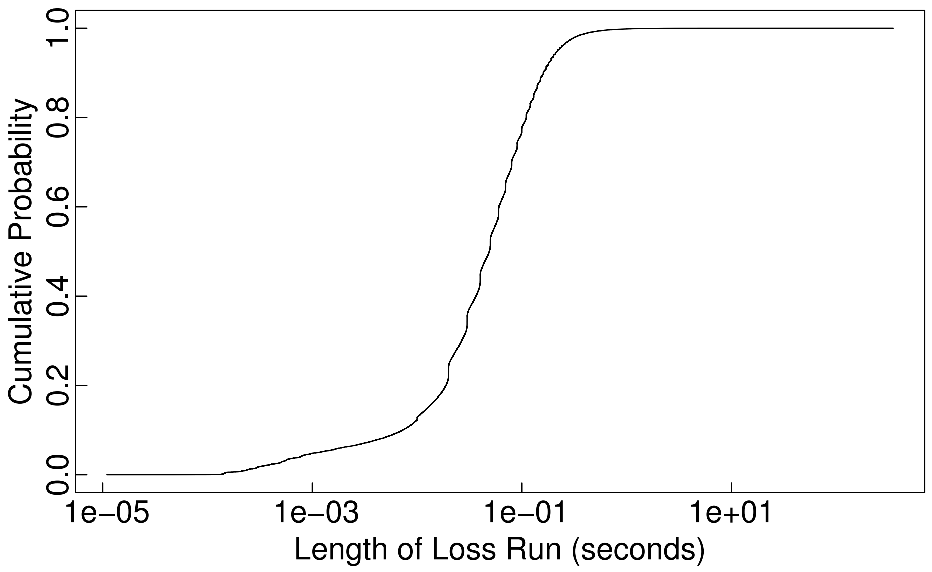


Figure 2.4: Distribution of loss run durations.

All in all, these findings argue that in many cases the fine time scale correlation reported in the previous studies is caused by trains of consecutive losses, rather than intervals over which loss rates become elevated and "nearby" but not consecutive packets are lost. Therefore, loss processes are better thought of as spikes during which there's a short-term outage, rather than epochs over which a congested router's buffer remains perilously full. A spike-like loss process accords with the Gilbert model [21], which postulates that loss occurs according to a two-state process, where the states are either "packets not lost" or "packets lost," though see below for necessary refinements to this model.

A related finding concerns the size of loss runs. Figure 2.4 shows the distribution of the duration of loss runs as measured in seconds. We see that virtually all of the runs are very short-lived (95% are 220 ms or shorter), and in fact near the limit of what our 20 Hz measurements can resolve. Similarly, we find that loss run sizes are uncorrelated according to Box-Ljung. We also confirm the finding in [72] that loss run lengths in packets often are well approximated by geometric distributions, in accordance with the Gilbert model, though the larger loss runs do not fit this description, nor do traces with higher loss rates ($> 1\%$); see below.

2.3.4 Mathematical constancy of the loss episode process

While in the previous section we homed in on understanding loss from the perspective of looking at loss episodes rather than individual loss, we also had the finding that on longer time scales, the loss episode rates appear to changing, i.e., *non-constancy.*

To assess the constancy of the loss episode process, we apply change-point analysis to the binary time series $\langle T_i, E_i \rangle$, where T_i is the time of the ith observation and E_i is an indicator variable taking the value 1 if a loss episode began at that time, 0 otherwise. In constructing this time series, note that we collapse loss episodes *and* the non-lost packet that follows them into a single point in the time series. For example, if the original binary loss series is: $0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,$ then the corresponding loss episode series is: $0, 0, 1, 1, 0, 1, 0, 0.$

$$\underbrace{0}_{0}\quad \underbrace{0}_{0}\quad \underbrace{1\,0}_{1}\quad \underbrace{1\,1\,1\,0}_{1}\quad \underbrace{0}_{0}\quad \underbrace{1\,0}_{1}\quad \underbrace{0}_{0}\quad \underbrace{0}_{0}$$

I.e., $\langle T_{i+1}, E_{i+1} \rangle$ reflects the observation of the second packet after the ith loss episode ended. We do this collapsing because if the series included the observation of the *first* packet after the loss episode, then E_{i+1} would always be 0, since episodes

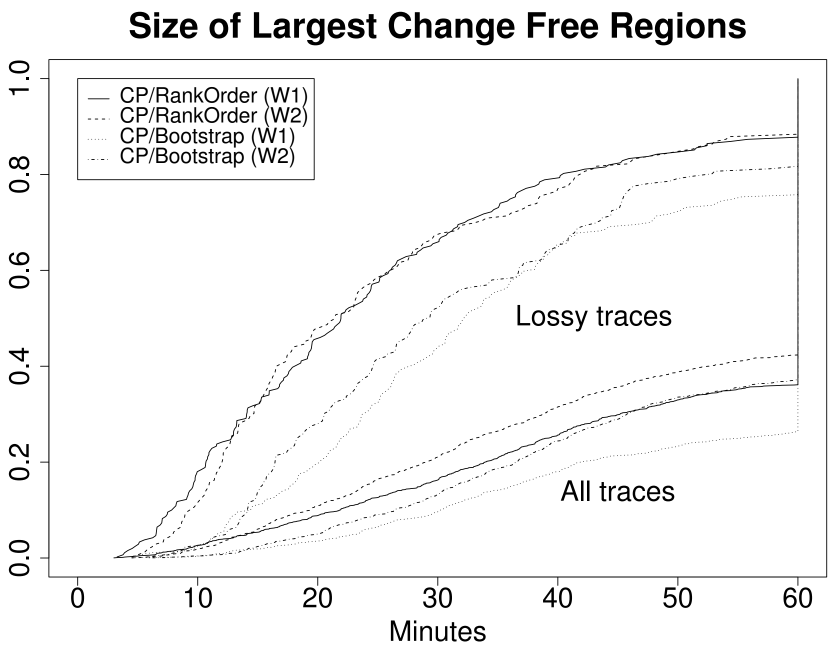


Figure 2.5: CDF of largest change-free region (CFR) for loss episodes in $\mathcal{W}_1$ and $\mathcal{W}_2$ datasets. "Lossy traces" is the same analysis restricted to traces for which the overall loss rate exceeded 1%.

are always ended by a non-lost packet, and we would thus introduce a negative correlational bias into the time series.

Using the methodology developed in Appendix A.1, we then divide each trace up into 1 or more change-free regions (CFRs), during which the loss episode rate appears well-modeled as steady. Figure 2.5 shows the cumulative distribution function (CDF) for the size of the largest CFR found for each trace in $\mathcal{W}_1$ (solid) and $\mathcal{W}_2$ (dashed). We also plot CDFs restricted to just those traces for which the overall loss rate exceeded 1% ("Lossy traces"). We see that more than half the traces are steady over the full hour. Of the remainder, the largest period of constancy runs the whole gamut from just a few minutes long to nearly the full hour. However, the situation changes significantly for lossy traces, with half of the traces having no CFR longer than 20 minutes for $\mathcal{CP}/RankOrder$ (or 30 minutes for $\mathcal{CP}/Bootstrap$). The behavior is clearly the same for both datasets. Meanwhile, the difference between the results for $\mathcal{CP}/RankOrder$ and those for $\mathcal{CP}/Bootstrap$ is also relatively small—about 10-20% more traces are change-free over the entire hour with $\mathcal{CP}/Bootstrap$ than with $\mathcal{CP}/RankOrder$. This suggests the effect of the bias/insensitivity is not major.

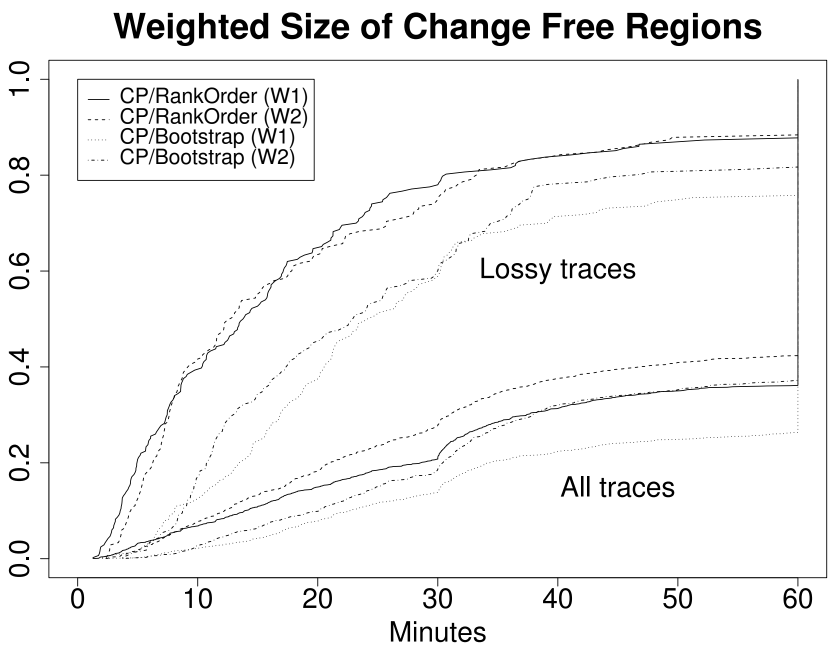


Figure 2.6: Weighted CDF of the size of change-free regions (CFR) for loss episodes in $\mathcal{W}_1$ and $\mathcal{W}_2$ datasets. "Lossy traces" is the same analysis restricted to traces for which the overall loss rate exceeded 1%.

We also analyzed the CDFs of the CFR sizes weighted to reflect the proportion of the trace they occupied. For example, a trace with one 10-minute CFR and one 50-minute CFR would be weighted as $\frac{1}{6}10 + \frac{5}{6}50 = 43.3$ minutes, meaning that if we pick a random point in a trace, we will on average land in a CFR of 43.3 minutes total duration. As shown in Figure 2.6, the CDFs for the weighted CFRs have shapes quite similar to those shown above, but shifted to the left about 7 minutes, except for the 60-minute spike on the righthand side, which of course

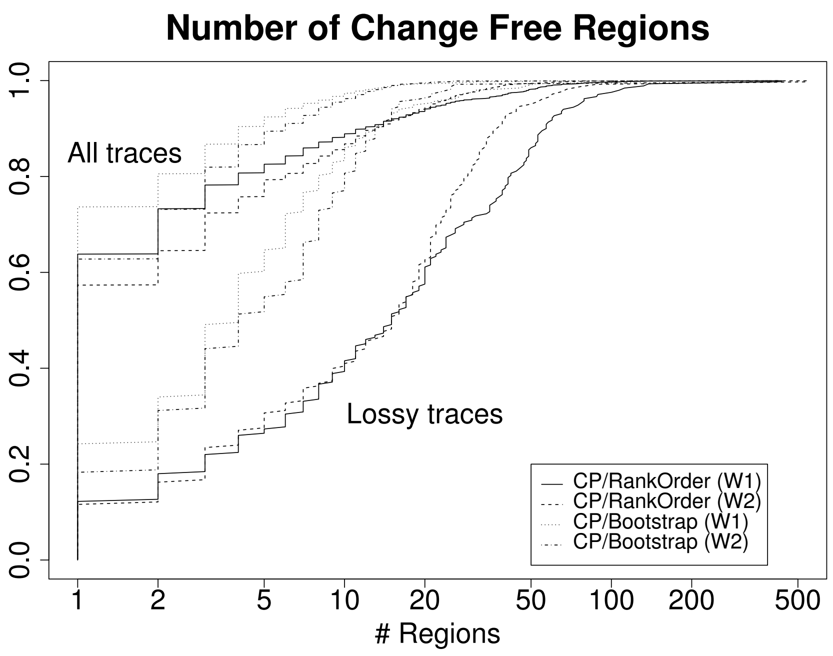


Figure 2.7: CDF of number of CFRs for loss episodes in $\mathcal{W}_1$ and $\mathcal{W}_2$ datasets. "Lossy traces" is the same analysis restricted to traces for which the overall loss rate exceeded 1%.

does not change because its weight is 1.

Figure 2.7 shows the distribution of the number of CFRs per trace. Again, the two datasets agree closely. Over all the traces there are usually just a handful of CFRs, but for lossy traces the figure is much larger, with the average rising from around 5 over all traces to around 20 over the lossy traces. Clearly, once we are in a high-loss regime, we also are in a regime of changing conditions. In addition, sometimes we observe a huge number of CFRs. Figure 2.8 shows an example of the latter, a trace whose loss episode process divides into hundreds of CFRs.

Once we have divided traces into one or more CFRs, we can then analyze each region separately from the others, having confidence that within the region the overall loss episode rate does not change. Table 2.2 summarizes our major results. Upon applying the Box-Ljung test, we find that 88-92% of the regions are consistent

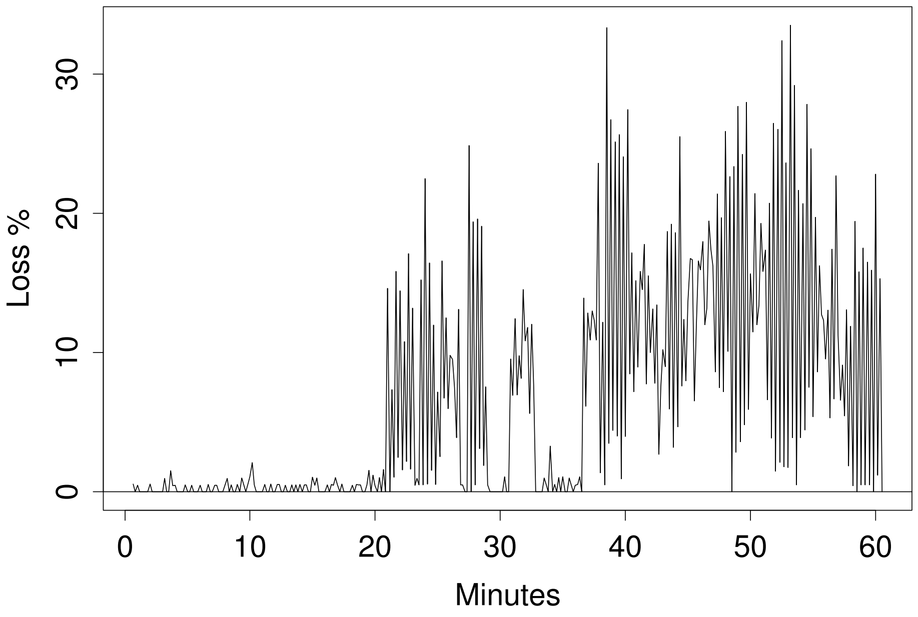


Figure 2.8: Example of a trace with hundreds of change-free regions. Loss rate is computed over 10-second intervals.

with an absence of lag 1 correlation, and 77-86% are consistent with no correlation up to lag 100. Clearly, within a CFR the loss episode process is well modeled as IID better than over the entire trace (previous section). In addition, applying the Anderson-Darling test (Section A.3) to the interarrivals between loss episodes in a region, we find that 77-85% of the regions are consistent with exponential interarrivals.

Table 2.2: Poisson nature of the loss episode process within change-free regions.

Test	CFRs w/o correlation		CFRs with exponential interarrivals
	lag=1	lag up to 100	
$\mathcal{CP}/RankOrder$	92%	86%	85%
$\mathcal{CP}/Bootstrap$	88%	77%	77%

Together, these findings solidly support modeling loss episodes as homogeneous Poisson processes within change-free regions. In particular, correlations in loss processes are due to the presence of consecutive losses, rather than nearby losses.

It remains to describe the structure of loss episodes. We do so in the context of the aforementioned Gilbert and extended Gilbert models. For the two-state Gilbert model to hold, we should find that within a loss episode the probability of observing each additional loss remains the same. In particular, the probability that we observe a 2nd loss in an episode, given that we have seen the initial loss of an episode, should be the same as the probability of observing a 3rd loss given that we have seen the 2nd loss. Similarly, the extended Gilbert model allows for k different loss rates for the first k losses after the initial loss, each corresponding to a different state in the model.

Accordingly, we can assess whether k states suffice to describe a given loss process by seeing whether the $k+1$ loss after the initial loss occurs (conditioned on the kth loss) with the same probability as the kth loss does (conditioned on the $k-1$ loss). We made these tests using Fisher's Exact Test [59], and found that, for both $\mathcal{W}_1$ and $\mathcal{W}_2$, 40% of the traces are consistent with Bernoulli loss; 89% with the Gilbert two-state model; 98% with 3 states (extended Gilbert); and 99% with 4 states. However, the models work less well for lossy traces: only 6% are well-modeled as Bernoulli, 68% with 2 states, 85% with 3 states, and 96% with 4 states.

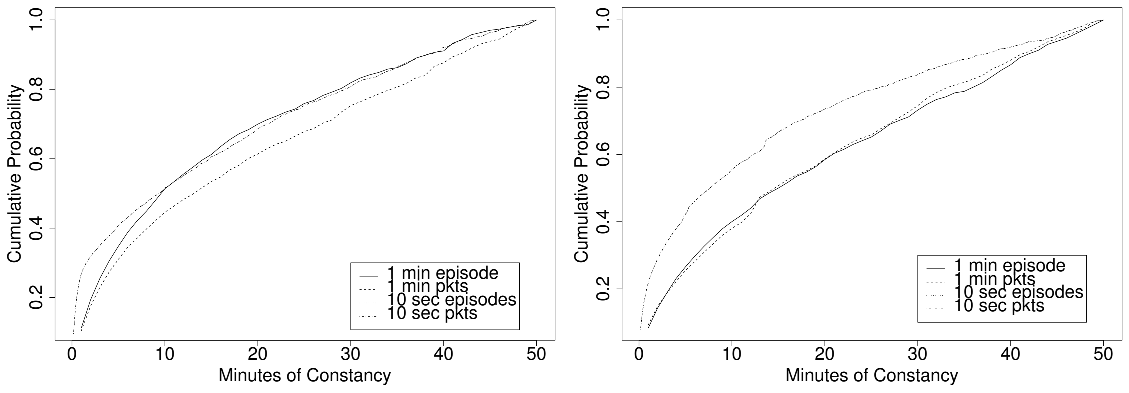
2.3.5 Operational constancy of loss rate

We now turn to analyzing a different notion of loss rate constancy, namely from an *operational* viewpoint. To do so, we partition loss rates into six different categories with ranges and operational meanings summarized in Table 2.3.

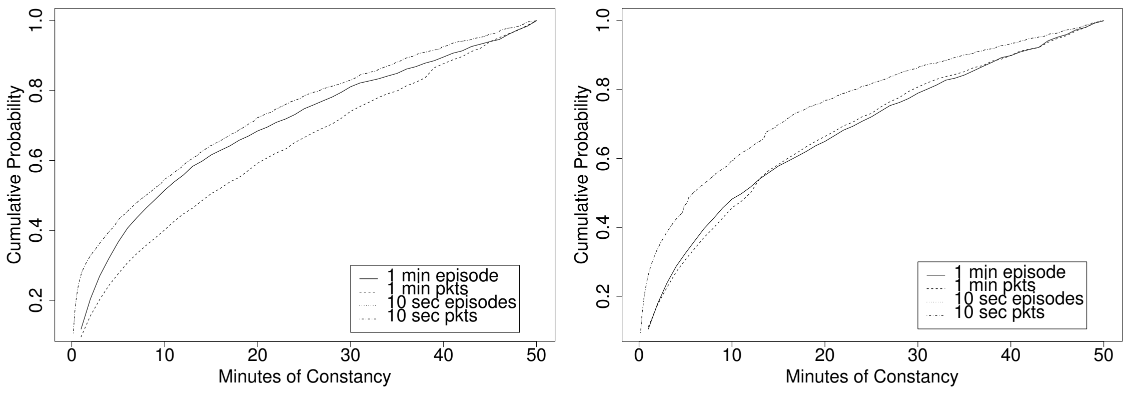
Table 2.3: Ranges and operational roles of different loss rate categories.

Range	Operational Role
0–0.5%	"no loss"
0.5–2%	"minor loss"
2–5%	"tolerable loss"
5–10%	"serious loss"
10–20%	"very serious loss"
20+%	"unacceptable loss"

For each trace we then analyze how long the loss rate remained in the same category. Figure 2.9(a) plots the weighted CDF for four different loss series associated with each trace in $\mathcal{W}_1$ and $\mathcal{W}_2$: the loss episode rate computed over 1-minute intervals, the raw packet loss rate over 1-minute intervals, and the same but com-



(a) Results for loss categories summarized in Table 2.3.



(b) Results for loss categories summarized in Table 2.5.

Figure 2.9: Operational constancy for packet loss and loss episodes, conditioned on the constancy lasting 50 minutes or less ($\mathcal{W}_1$: left; $\mathcal{W}_2$: right).

Table 2.4: Probabilities of observing a constancy interval of 50 or more minutes for loss categories summarized in Table 2.3.

data set	1 min. episodes	1 min. packets	10 sec. episodes	10 sec. packets
$\mathcal{W}_1$	71%	58%	25%	23%
$\mathcal{W}_2$	71%	56%	26%	24%

Table 2.5: Loss categories with a different set of cutpoints.

Range	0–0.25%	0.25–1.25%	1.25–3.5%	3.5–7.5%	7.5–15%	15+%

Table 2.6: Probabilities of observing a constancy interval of 50 or more minutes for loss categories summarized in Table 2.5.

data set	1 min. episodes	1 min. packets	10 sec. episodes	10 sec. packets
$\mathcal{W}_1$	62%	55%	17%	17%
$\mathcal{W}_2$	59%	43%	19%	19%

puted over 10-second intervals. The CDF is weighted by the size of the constancy interval, as mentioned above; thus, we interpret the plot as showing the unconditional probability that at any given moment we would find ourselves in a constancy interval of duration T or less. For example, in $\mathcal{W}_1$ about 50% of the time we will find ourselves in a constancy interval of 10 min. or less, if what we care about is the constancy of loss episodes computed over minute-long intervals (solid line in the top left plot of Figure 2.9).

An important point is that we truncated the plot to only show the distribution of intervals 50 minutes or less. We characterize longer intervals separately, as these reflect entire datasets that were operationally steady. Since our datasets spanned at most one hour, constancy over the whole dataset provides a lower bound on the duration of constancy, rather than an exact value, and hence differs from the distributions in Figure 2.9(a).

For the four loss series, the corresponding probabilities of observing a constancy interval of 50 or more minutes are summarized in Table 2.4. Thus, if we only care about constancy of loss viewed over 1-minute periods, then about two-thirds (56–

71%) of the time, we will find we are in a constancy period of at least an hour in duration—it could be quite a bit longer, as our measurements limited us to observing at most an hour of constancy.

We also see that the key difference between the 10 sec. and 1 min. results is the likelihood of being in a period of long constancy: it takes only a single 10-second change in loss rate to interrupt the hour-long interval, much more likely than a single 1-minute change. If we condition on being in a shorter period of constancy, then we find very similar curves. In particular, if we are not in a period of long-lived constancy, then, per the plot, we find that about half the time we are in a 10-minute interval or shorter, and there is not a great deal of difference in the duration of constancy, regardless of whether we consider one-minute or 10-second constancy, or loss runs or loss episodes.

Finally, we repeated this assessment using a set of cutpoints for the loss categories that fell in the middle of the cutpoints in Table 2.3 (see Table 2.5), to test for possible binning effects in which some traces straddle a particular loss boundary. As shown in Figure 2.9(b) and Table 2.6, the results are very similar to those in Figure 2.9(a) and Table 2.4.

2.3.6 Comparing mathematical and operational constancy

We now briefly assess the degree to which we find that the notion of mathematical constancy of loss coincides with the notion of operational constancy of loss. While there are many dimensions in which we could undertake such an assessment, we aim here to only explore the coarse-grained relationship.

We begin by categorizing each trace as either "steady" or "not steady," where the distinction between the two concerns whether the trace has a 20-minute region

Table 2.7: Comparison between mathematical loss constancy and operational loss constancy evaluated using loss computed over 1 min.

Category	$\overline{M}\overline{O}$	$M\overline{O}$	$\overline{M}O$	MO
Proportion	6–9%	6–15%	2–5%	74–83%

Table 2.8: Comparison between mathematical loss constancy and operational loss constancy evaluated using loss computed over 10 sec.

Category	$\overline{M}\overline{O}$	$M\overline{O}$	$\overline{M}O$	MO
Proportion	11%	37–45%	0.1%	44–52%

of constancy; i.e., for mathematical constancy, a 20-minute CFR for the rate of the loss episode process, and for operational constancy, a 20-minute period during which the loss rate did not stray outside one of the particular regions. We then assess what proportion of the traces were neither mathematically nor operationally steady ($\overline{M}\overline{O}$), mathematically but not operationally ($M\overline{O}$), vice versa ($\overline{M}O$), and both (MO). Note that the choice of 20 minute is not arbitrary—it strikes as a reasonable balance between the need to have a sufficient number of "steady" traces and the requirement of accurately reflecting the constancy level of traces (i.e. only traces with a high level of constancy are categorized as "steady").

For operational constancy evaluated using loss computed over 1 min, the figures are summarized in Table 2.7. (The minor variation in the figures depends on whether for operational constancy we look at loss rate or loss episode rate, and whether we use the first or the second set of loss categories as discussed at the end of Section 2.3.5.) Clearly, the notions of mathematical and operational constancy mostly coincide.

However, if we instead evaluate operational constancy using loss rates computed over 10 sec. intervals, the figures are significantly different. This is evident from Table 2.8. We can summarize the difference as: *Operational constancy of packet loss coincides with mathematical constancy on large time scales such as viewing how loss changes from one minute to the next; but not nearly so well on medium time scales such as looking at 10-second intervals.*

2.3.7 Predictive constancy of loss rate

The last notion of packet loss constancy we explore is that of *predictive* constancy, i.e., to what degree can an estimator predict future loss events?

There are a number of different loss-related events we could be interested in predicting. Here, we confine ourselves to predicting the length of the next loss-free run. We chose this event for two reasons: first, we do not have to bin the time series (which predicting loss rate over the next T seconds would require); and second, there are known applications for such predictions, such as TFRC [19].

The next question is what type of estimator to use. We assess three different types popular in the literature: moving average (MA), exponentially-weighted moving average (EWMA) such as used by TCP [26], and the S-shaped moving average estimator (SMA) used by TFRC. This last type is a class of weighted moving average estimators that give higher weights to more recent samples; we use the same subclass as as in TFRC, which gives equal weight to the most recent half of the window, and linearly decayed weights for the earlier half; see [19] for discussion.

For each of these estimators there is a parameter that governs the amount of memory of past events used by the estimator. For MA and SMA, we used window sizes of $2, 4, 8, 16, 32$; and for EWMA, $\alpha = 0.5, 0.25, 0.125$, and 0.01, where $\alpha = 0.5$

corresponds to weighting each new sample equally to the cumulative memory of previous samples, and $\alpha = 0.01$ weights the previous samples 99 times as much as each new sample.

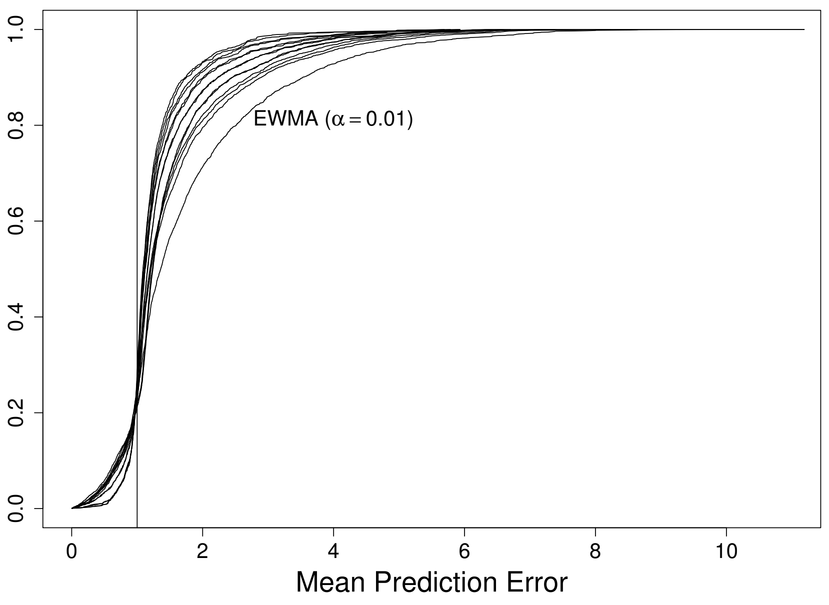
Once we have defined what estimator to use, we next have to decide how to assess how well it performed. To do so, we compute:

$$\text{prediction error} = E\left[\ \left|\log\left(\frac{\text{predicted value}}{\text{actual value}}\right)\right|\ \right]$$

where the expectation, which is computed over each of the events (loss-free runs) in a trace, reflects the ratio by which the estimator typically misses the target. We then compute CDFs that show the range of how well a given estimator performs over all of the traces.

Figure 2.10 shows the resulting CDFs, computed for all traces (top plot) and for all CFRs within the traces (bottom plot). The vertical line in each plot reflects a prediction error of 1, corresponding to overestimating or underestimating by a factor of e. (It turns out that the best one can achieve, on average, for predicting IID exponential random variables is a prediction error of 1.02 [17].) We have plotted CDFs for all of the different estimators and sets of parameters, and the plot does not distinguish between them because the main point to consider is that virtually all of the estimators perform about the same—*the parameters do not matter, nor does the averaging scheme.*

We interpret this as reflecting that the process does not have significant structure to its short-range correlations that can be exploited better by particular types of predictors or window sizes; all that the estimators are doing is tracking the mean of the process, which varies more slowly than do the windows. There are two exceptions, however. First, in the top plot, the CDF markedly below all the others corresponds to EWMA with $\alpha = 0.01$. That estimator has a lengthy memory (on



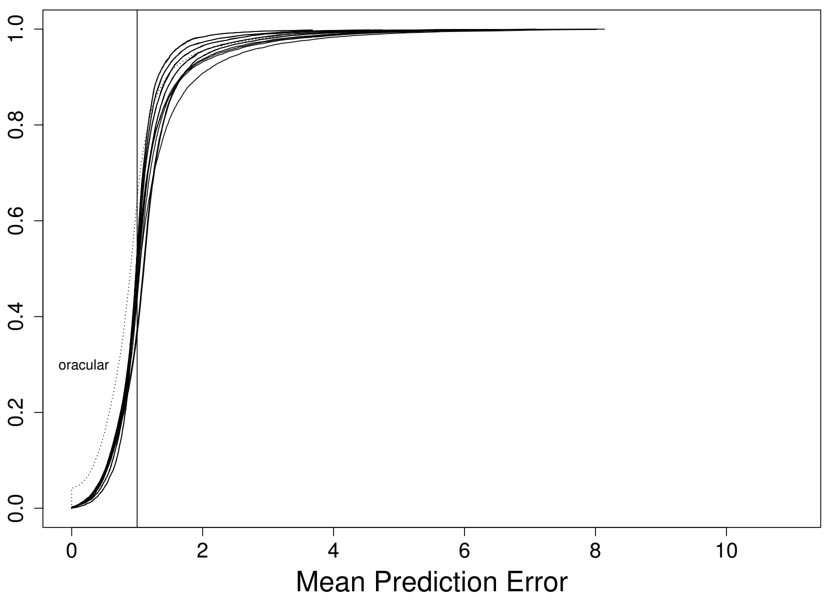


Figure 2.10: CDFs of the mean error for a large number of loss predictors, computed over entire traces (top) or change-free regions (bottom).

the order of 100 packets), and accordingly cannot adapt to rapid fluctuations in the loss process. In addition, that estimator will do particularly poorly during a transition between two CFRs, because it will remember the behavior in the older CFR for much longer than the other estimators. We see that in the lower plot, it fares better, because that plot does not include transitions between CFRs.

Also, in the second plot we have added an “oracular” estimator (dotted). This estimator knows the mean loss-free length during the CFR, and always predicts that value. We can see that it does noticeably better than the other estimators about half the time, and comparable the other half. A significant element of its improved performance is that the lower plot is heavily skewed to favoring estimators that do well over traces that are highly non-steady (many CFRs), because each of the CFRs will contribute a point to the CDF. The success of the oracle also suggests that it might be a good general strategy to construct estimators that include an explicit decision whether to restart the estimator, so they can adapt to level shifts in a nimble fashion.

Finally, we repeated the analysis after applying a random shuffle to the traces to remove their correlational structure. Figure 2.11 shows the results for three estimators—EWMA with $\alpha = 0.01$, EWMA with $\alpha = 0.25$, and the oracular estimator. The results for the other estimators we tested are very similar to those for EWMA with $\alpha = 0.25$ and are omitted for plotting clarity. We see that random shuffling makes only a slight difference in the estimators’ performance, reducing the discrepancy between the EWMA with $\alpha = 0.01$ estimator and the others, and we find that the various estimators do only slightly worse than an oracular estimator applied to the now-IID time series.

We finish with a look at the relationship between how well we can predict loss

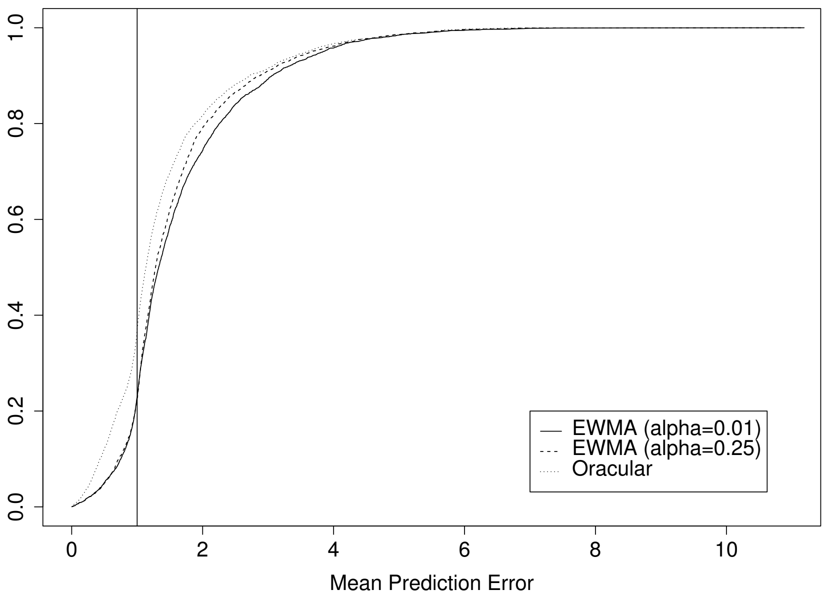


Figure 2.11: Effects of random shuffling on the estimators' performance.

versus the presence or absence of mathematical and/or operational constancy. As in Section 2.3.6, we aim only to understand the coarse-grained relationship, and again we consider a trace mathematically steady if it has a maximum CFR of at least 20 minutes, and operationally steady if it stays within a particular loss region for at least 20 minutes.

Partitioning the lossy ($\geq 1\%$ loss) traces on that criteria, using EWMA with $\alpha = 0.25$ we attain the predictor error CDFs shown in Figure 2.12. We see that the quality of the predictor is virtually unchanged if we have neither mathematical nor operational constancy, or just one of them. But if we have both, then the predictor's performance is *worse*. This is because in this regime the loss episode process resembles an IID process without significant short-term variations, and the recent samples seen by the estimator provide no help in predicting the next event. In addition, note that if we look at all traces rather than just the lossy traces,

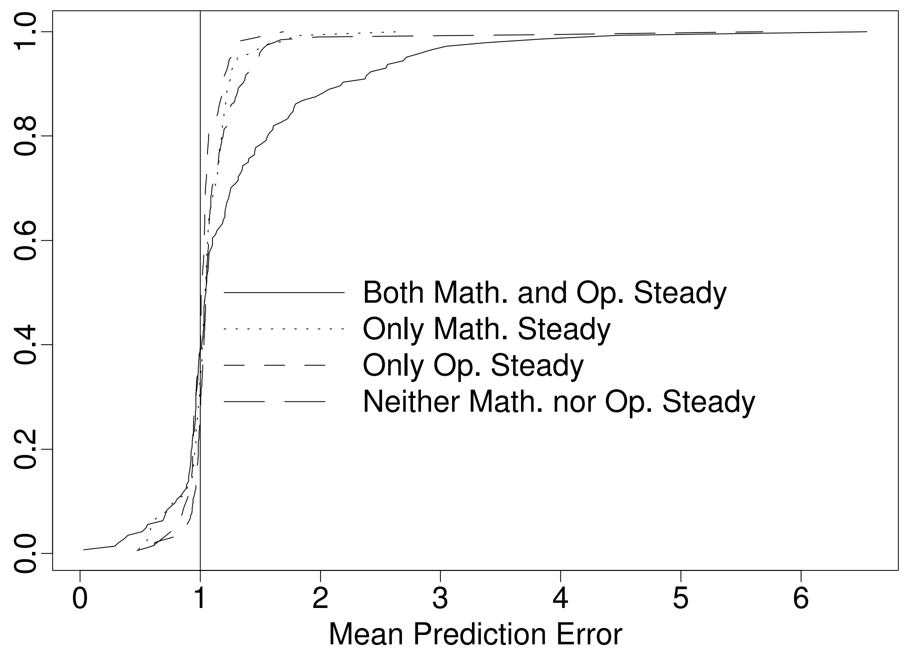


Figure 2.12: CDFs of the mean error for EWMA ($\alpha = 0.25$) estimator computed over sets of lossy traces with different types of constancy.

the estimators again do worse, because for the type of event we are predicting (interval until the next loss episode), traces with low loss levels provide very few samples to the estimator. However, low loss is also a condition under which we generally do not care about the precision of the estimator, since loss events will be quite rare. In summary, *predictors do equally well whether or not we have other forms of constancy, unless we have constancy resembling an IID process with little short-term variation.*

2.4 Delay constancy

We next turn to exploring the types of constancy associated with packet delays. Mukherjee found that packet delay along several Internet paths was well-modeled using a shifted gamma distribution, but the parameters of the distribution varied from path to path and on time scales of hours [45]. Similarly, Claffy and colleagues found that one-way delays measured along four Internet paths exhibited clear level shifts and non-constancies over the course of a day [12].

For our analysis, we again use the `zing` Poisson packet streams measured on the NIMI hosts. Because the NIMI hosts lack synchronized clocks, we confine our analysis to those datasets with bidirectional packet streams. These are generated by `zing` on host A sending "request" packets host B, and the `zing` on host B immediately responding to each of these by sending back matching "reply" packets, facilitating round-trip measurement at host A. The delay in `zing`'s response is short, usually taking 100–200 μsec, occasionally rising to a few ms [55].

2.4.1 Delay "spikes"

The data totaled 130M RTT measurements made between 613 distinct pairs of hosts. In analyzing it, the first phenomenon we had to deal with is the presence of delay *spikes.* These are intervals (often quite short) of highly elevated RTTs. They are rare, but if unaddressed can seriously skew our analysis due to their magnitude. Figure 2.13 conveys the size and prevalence of spikes. For each trace, we computed the median of all of the RTT measurements, and then normalized each RTT measurement by dividing it by the median. This allows us to then plot all of the measurements together to assess, in high level terms, the magnitude of RTT variation present in the data. The plot shows the complementary distribution of the RTT-to-median ratio; this style of plot emphasizes the upper tail. For reference we have drawn lines reflecting a ratio of 10:1 (vertical) and a probability of 10^{-3} (horizontal). Clearly, there are a significant number of very large RTTs, though not so many that we would consider them anything other than an extreme upper-tail phenomenon.

To proceed with separating spikes from regular RTT behavior, we need to devise a definition for categorizing an RTT measurement as one or the other. We were unable to find a crisp modality to exploit—the only one present in the plot is for ratios above or below 100:1, but that cutoff point omits many spikes that we found visually—so we settled on the following imperfect procedure: for each new RTT measurement R', we compared it to the previous non-spike measurement, R. If $R' \geq \max(k \cdot R, 250\text{ms})$, then we consider the new measurement a spike; otherwise, we set $R \leftarrow R'$ and continue to the next measurement.[1] We then

[1] We found the 250 ms lower bound necessary for applying the classifier to traces with quite low RTTs.

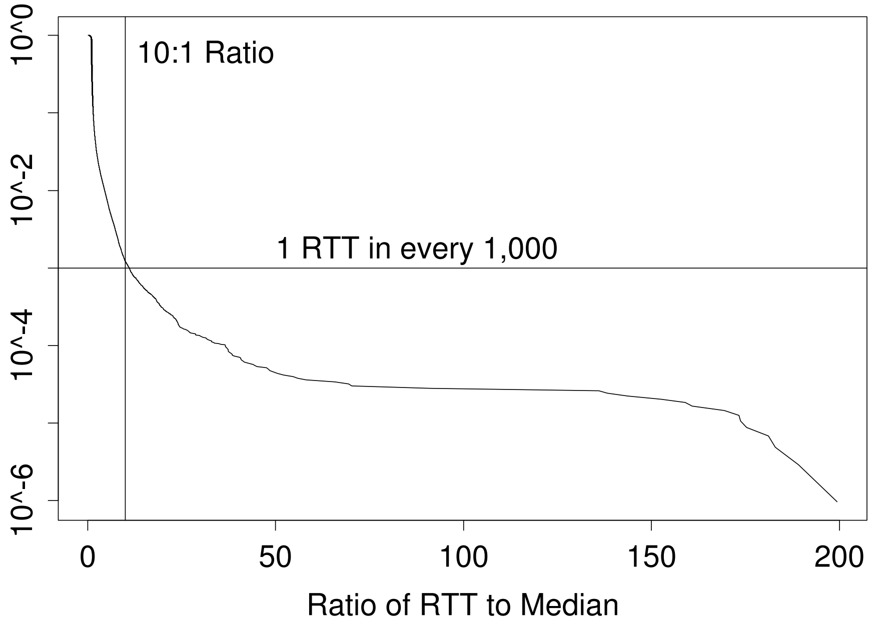


Figure 2.13: Complementary distribution of the ratio of RTT samples to the median of their traces, computed for $\mathcal{W}_2$.

applied this classification for $k = 2$ and $k = 4$. Doing so revealed two anomalies: a high latency path plagued by rapid RTT fluctuations ranging from 200 ms to 1 sec, and a pair of hosts that periodically jumped their clocks. With the anomalies removed, we find that $k = 2$ categorized $1.1 \cdot 10^{-3}$ of the $\mathcal{W}_1$ RTTs as spikes, and $k = 4$ categorized $3.4 \cdot 10^{-4}$.

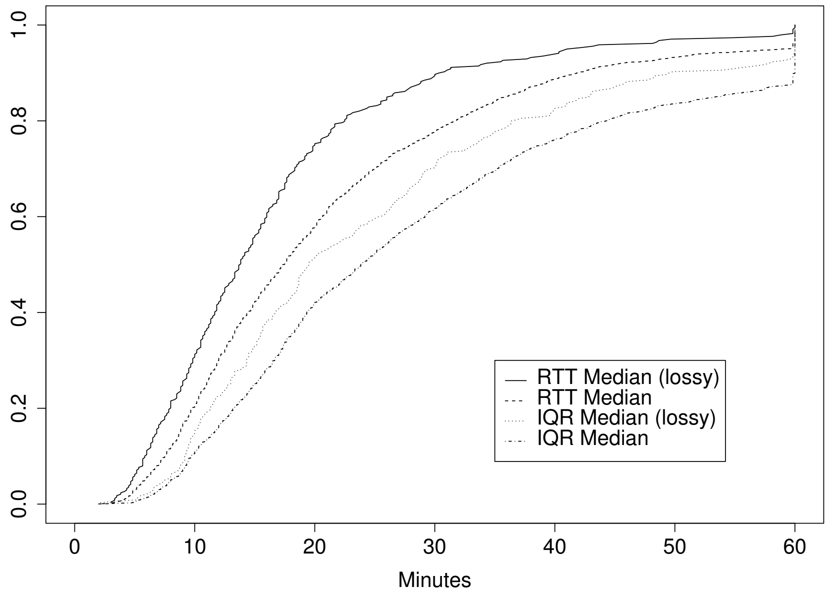
Once we had the definition in place, we could check it in terms of "yes, these are really outliers," as follows: for each trace we computed $\overline{x}$ and σ, the mean and standard deviation of the RTT measurements *with the spikes removed*. We then for each spike assessed how many σ it was above $\overline{x}$. For $\mathcal{W}_1$, the $k = 2$ definition leads to spikes that are typically (median) 16.9σ above the mean, with 80% being more than 5.6σ. For $k = 4$, these figures rise to 28σ and 6.6σ.

2.4.2 Constancy of the body of the RTT distribution

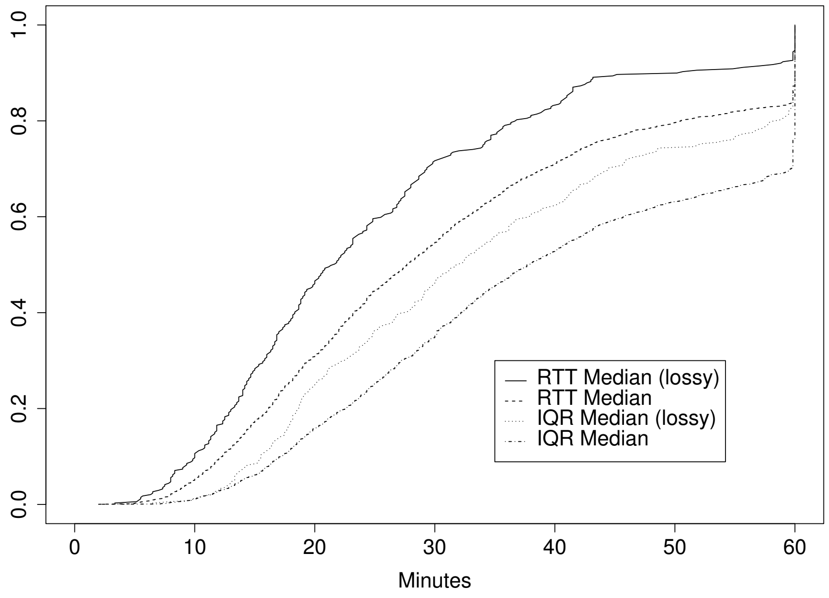
The degree to which RTT spikes are indeed outliers points up a need to assess the constancy of the body of the RTT distribution separate from that of the RTT spikes. We do so by applying change-point analysis to the median and inter-quartile range (IQR) of the distribution.[2]

Figure 2.14 shows CDFs of the size of the largest corresponding CFRs. We see that, overall, the median is less steady than the IQR (indeed, we find that IQR change-points appear to often be a subset of median change-points), and both distributions shift about 5 minutes to the left for lossy traces. The striking difference with Figure 2.5, though, is the lack of entire hours with no change-points.

[2]The IQR of a distribution is the distance between the 25th and 75th percentiles. It serves as a robust counterpart to standard deviation. For IQR change-points, we compute the IQR over ten-second intervals and look for a change in the median of that time series.



(a) Results for $\mathcal{C}P/RankOrder$.



(b) Results for $\mathcal{C}P/Bootstrap$.

Figure 2.14: CDF of largest CFR for median and IQR of packet RTT delays. "Lossy" is the same analysis restricted to traces for which the overall loss rate exceeded 1%.

Thus we find that *overall, delay is less steady than loss*, and that, while there's a wide range in the length of steady delay regions, in general delay appears well described as steady on time scales of 10–30 minutes. We can also test the median and IQR (computed over 10-second intervals) for independence within each CFR. Using the Box-Ljung test for up to 6 lags, we find good agreement (90–92%) with independence.

2.4.3 Constancy of RTT spikes

Having characterized the constancy of the packet delay distribution's body, we now turn to the constancy of the RTT spike process. Analogous to our approach for packet loss, we group consecutive spikes into spike episodes, noting that in general the episodes are quite short lived: for example, the median duration of a spike episode (using $k = 2$) in $\mathcal{W}_1$ was 150 ms, and the mean 275 ms.

Upon applying change-point detection to the spike episode process, we find spike episodes even more steady than loss episodes: the process is steady across the entire hour 75-83% of the time for $k = 2$ spikes, and 90% of the time for $k = 4$ spikes (see Table 2.9). In addition, we find the interarrivals between spikes are well-modeled as IID exponential, i.e., Poisson.

Table 2.9: Probabilities of having a spike episode process that is change-free over the entire hour.

Test	$k = 2$ spikes	$k = 4$ spikes
$\mathcal{CP}/RankOrder$	75%	90%
$\mathcal{CP}/Bootstrap$	83%	90%

2.4.4 Operational constancy of RTT

Similar to our analysis for loss (Section 2.3.5), we assess the operational constancy of RTTs by partitioning the delays into a set of categories and then assessing the duration of regions over which the measured RTT stays within a single category.

Different applications can have quite different views as to what constitutes good, fair, poor, etc., delay. To have concrete categories, we used ITU Recommendation G.114 [25], which defines three regions: 0–150 ms ("Acceptable for most user applications"), 150–400 ms ("Acceptable provided that Administrations are aware of the transmission time impact on the transmission quality of user applications"), 400+ ms ("Unacceptable for general network planning purposes"). Because these recommendations are for one-way delays and we are analyzing RTTs, we doubled them to form RTT categories, and then sub-divided 0–300 ms into 0–100 ms, 100–200 ms, and 200–300 ms, to allow a somewhat finer-grained assessment.

We find that more than half of the traces have maximum constancy intervals under 10 min, and 80% are under 20 min. We found virtually no difference whether or not we left RTT spikes in the traces (since they are rare), or when we tested a "shifted" version of the categories similar to the shifted version of loss rates discussed in Section 2.3.5. Thus, *not only are packet delays not mathematically steady, they also are not operationally steady.*

2.4.5 Predictive constancy of delay

We finish our assessment of different types of delay constancy with a brief look at the efficacy of predicting future RTT values. We again use the families of estimators discussed in Section 2.3.7. The events they process are RTT measurements, and our assessment concerns how well they predict the next measurement. Fig-

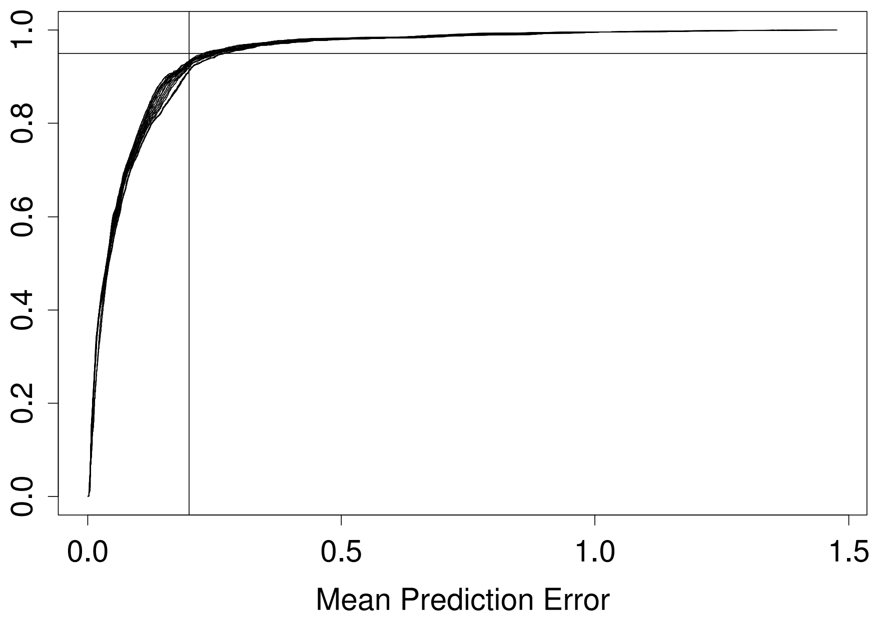


Figure 2.15: CDFs of the mean error for a large number of delay predictors.

ure 2.15 shows that the estimators again all perform virtually identically, and that their performance is very good: the vertical line on the plot marks a mean prediction error of 0.2, which corresponds to estimating the next value within a factor of $e^{0.2} \approx 22\%$, and the horizontal line marks 95% of the distribution. We attain virtually identical results whether or not we include RTT spikes in the measurements. Thus, we find that, in contrast with loss (Figure 2.10), *in general, delay is highly predictable.* Of course, for some applications, the consequence of mispredicting delay can be significant (e.g., a bad TCP retransmission timeout); we are not blithely asserting that applications will find highly predictable those facets of delay that they particularly care about, only that delay in general is highly predictable.

2.5 Throughput constancy

The last facet of Internet path constancy we study is end-to-end throughput. Compared to loss and delay, throughput is a higher-level path property, a product of the first two plus the dynamics of the transport protocol used. In addition, applications have a wide range of throughput requirements. To keep our analysis tractable, we confine ourselves to a simple notion of throughput constancy, namely the minute-to-minute variations observed in 1 MB TCP transfers. The data we analyzed consisted of 169 runs of 5 hours each, comprising a total of 49,000 connections measured along 145 distinct Internet paths.

Based on a very large packet-level trace collected at a single busy Web server, [8] found that the throughput of Web transfers exhibited significant temporal (several minutes) and spatial stability despite wide variations in terms of end-host location and time of day. Their study differs from ours in that the server was a single site, there were many more clients, and the analysis focused on the throughput of Web transfers, which are usually much shorter than our transfers. In other previous work, Paxson found that for a measure of available bandwidth derived from timing patterns in TCP connections, the predictive power of the estimator was fairly good for time periods up to several hours [54].

2.5.1 Throughput constancy analysis

Figure 2.16 shows some of the different types of throughput dynamics we observed. The top shows a five hour dataset for which the entire run is well-modeled as IID (see further discussion below). The next plot shows a clear level shift from one throughput value to another. If the trace is split at the point of the shift, then

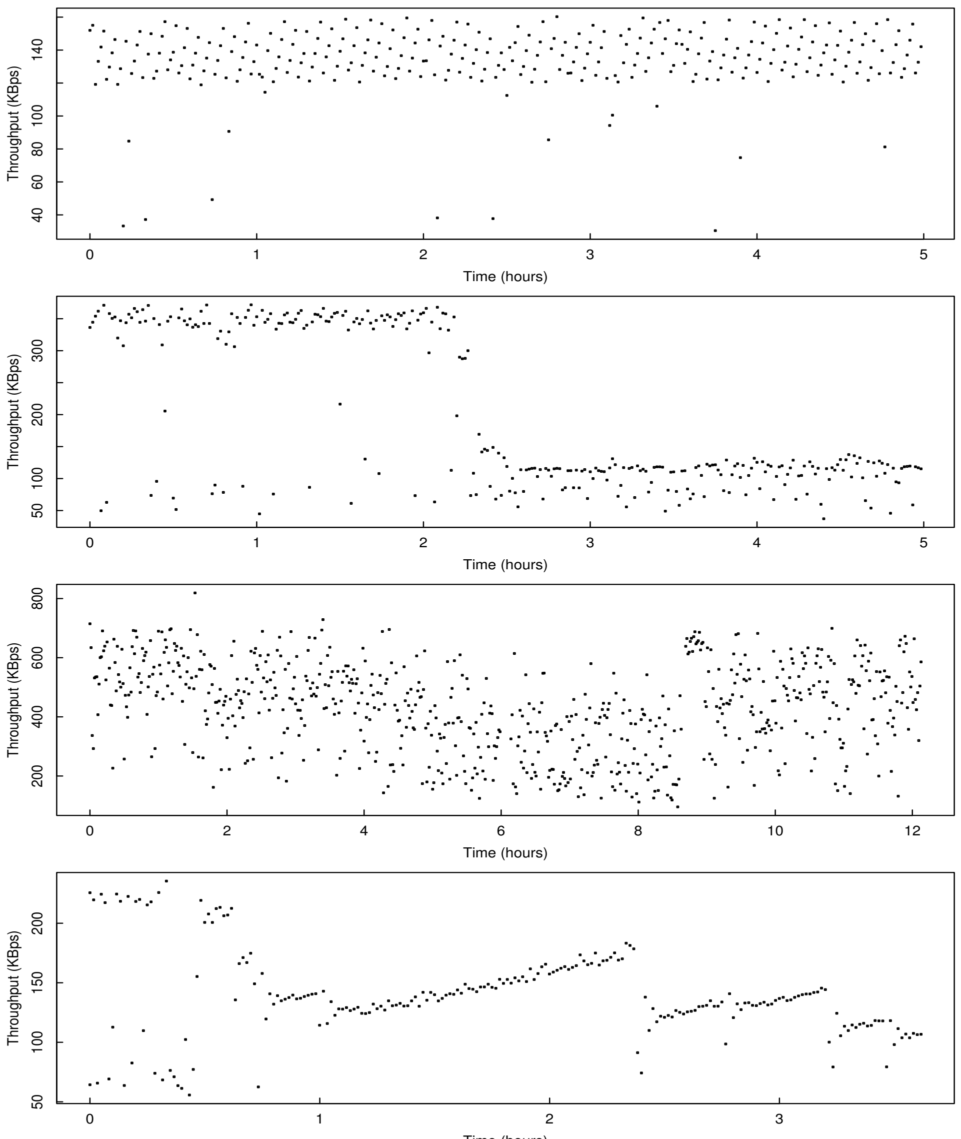


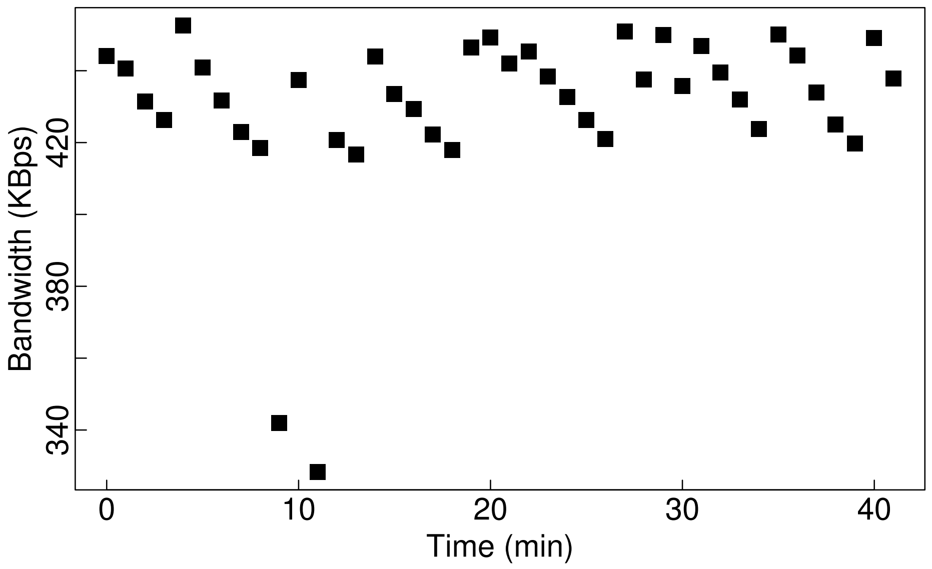
Figure 2.16: Different throughput patterns: IID, level shift, mess, trend.

both halves are well-modeled as IID, though without this the whole trace is not. The third plot shows a "mess"—throughput figures vary by a factor of five, with little apparent pattern other than a dip between hours 6 and 8. The final plot shows quite baffling behavior: a slow but steady climb in throughput from 120 KBps to 170 KBps over the course of more than an hour, followed by an abrupt return to 120 KBps, and another slow but steady rise.

Upon inspection, the top plot shows a striking pattern during the first two hours: a series of downward sloping lines that chart throughput varying from 160 KBps to 120 KBps. To investigate this behavior, we conducted additional measurements in which we used a packet filter to trace the individual packets of a series of 1 MB TCP transfers. The top half of Figure 2.17 shows a similar sawtooth pattern from such a trace, with throughput varying from 420–450 KBps.

The bottom half of the figure reveals the explanation for the pattern. When a TCP connection begins, after the SYN handshake the sender's congestion window is set to a single packet to begin "slow start." It sends this packet, but when it arrives, the receiver does not immediately acknowledge it but instead implements TCP's delayed acknowledgment mechanism, whereby it waits on a timer for additional data to arrive (though in this case, none can, since the congestion window does not permit it) prior to acknowledging. While waiting on the timer, the connection is completely stalled, because an ACK is required to advance the flow control and congestion windows to permit new data to be sent. Thus, each connection incurs a delay simply due to waiting on the delayed acknowledgment timer. Furthermore, it only incurs this penalty once, since as soon as two or more packets are in flight, the receiver will generate ACKs without any additional delay.

We have accordingly plotted the duration of the first delayed acknowledgment,



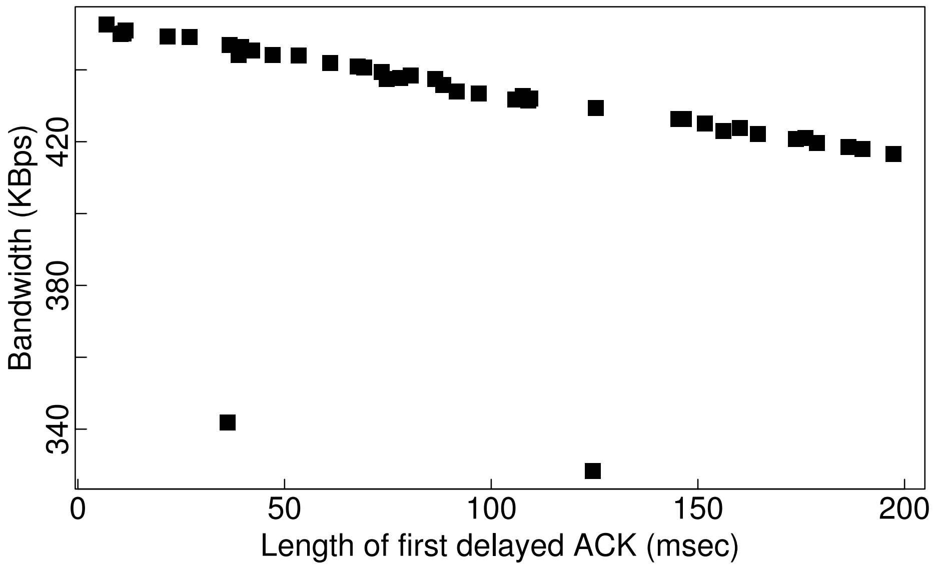


Figure 2.17: Throughput periodicity (top) and relationship between throughput and length of first delayed acknowledgment (bottom).

i.e., the incurred penalty, along the X-axis, and the total throughput attained for the 1 MB transfer along the Y-axis. That the different measurements fall on a clear line demonstrates that the difference in measured throughput is entirely explained by the timer penalty—it is the only source of variability for these connections! (The two lower throughput figures reflect connections that incurred a retransmission.) Finally, the reason that the sawtooth pattern occurs is due to the use in many TCPs, including those in the NIMI infrastructure, of a "heartbeat" timer that chimes independently from the exact 200 msec. timer interval the TCP would like. As the connection's initiation time moves in phase relative to the heartbeat, an increasingly short timer interval results, diminishing to 0 msec, until finally the interval wraps in phase and the delay penalty returns to 200 msec.

Understanding this effect is important, as otherwise we could erroneously conclude that there are complicated network dynamics that lead to various non-constancies in our measurements. Unfortunately, this simple explanation does not suffice to explain the bottom plot in Figure 2.16. There, the difference in time between the connections at the low end of the throughput ramp and the high end is 2.2 sec, too much to be explained by a single timer.

2.5.2 Mathematical constancy of throughput

We applied change-point analysis to the mean of the series of per-minute throughput measurements in each trace. Figure 2.18 and Figure 2.19 show the CFRs for traces with different throughput dynamics (Figure 2.16). We see the major change-points reported by both tests have good agreement with visually perceived change-points. We also note that $\mathcal{CP}/Bootstrap$ generally only finds a subset of the change-points reported by $\mathcal{CP}/RankOrder$. For example, it finds no change-point

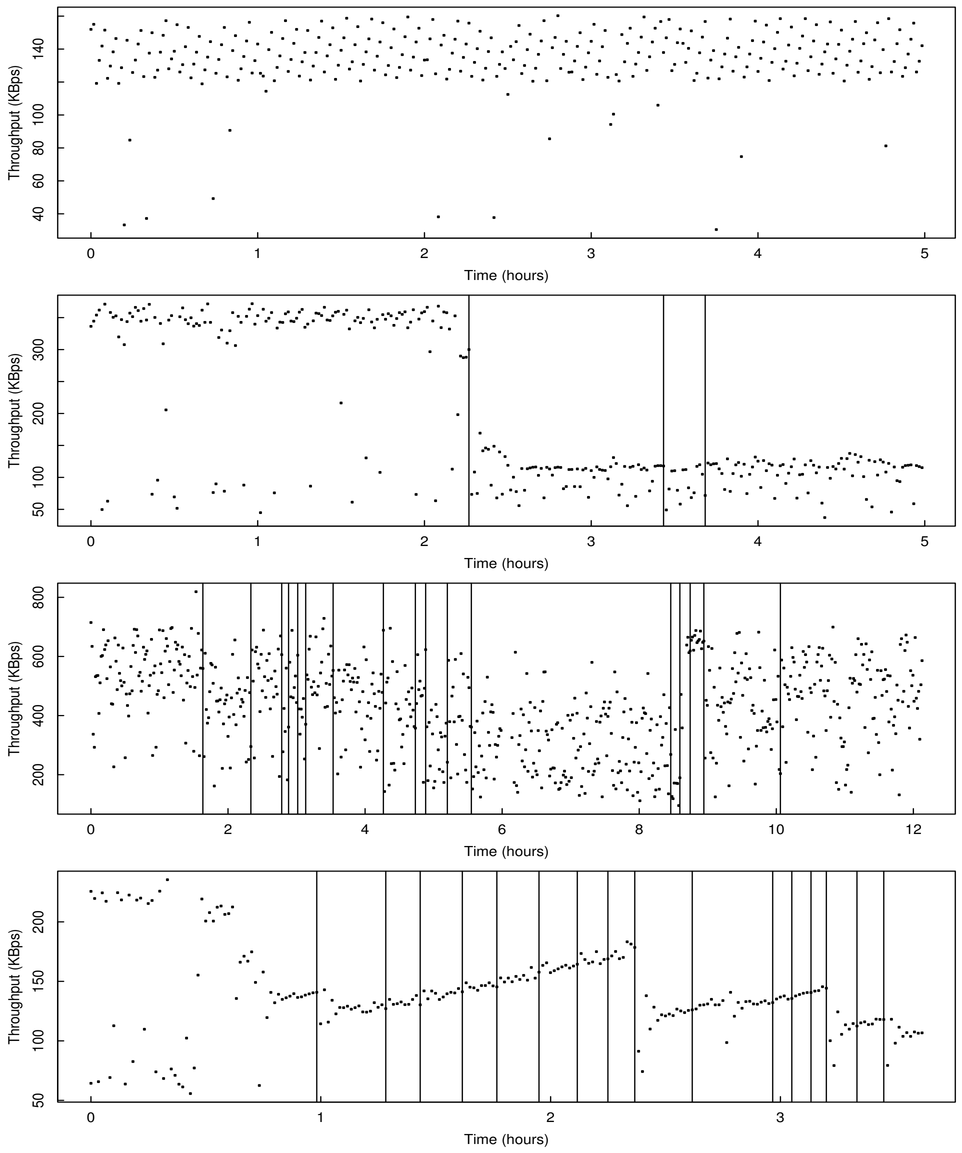


Figure 2.18: Change-free regions (separated by vertical lines) reported by test $\mathcal{C}P/RankOrder$ for traces in Figure 2.16.

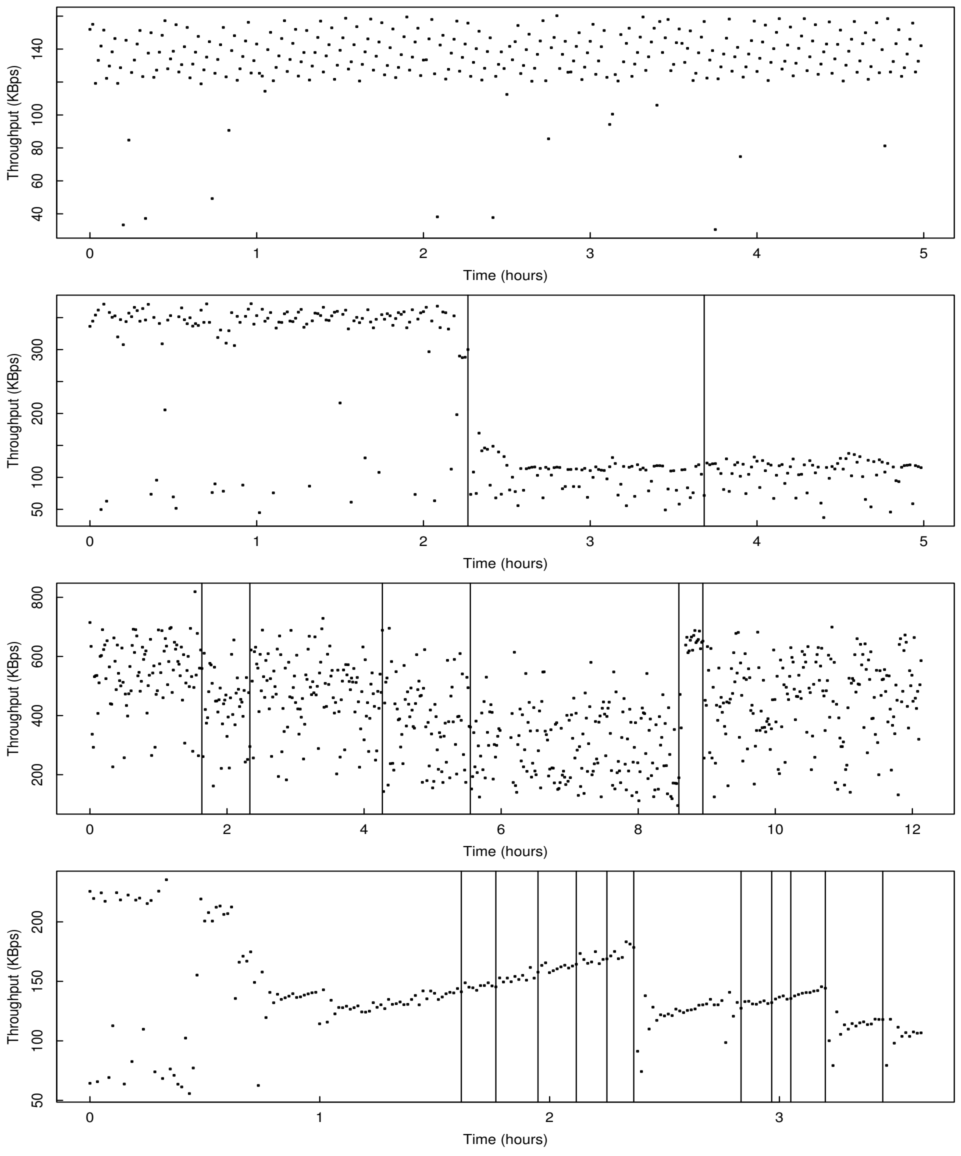


Figure 2.19: Change-free regions (separated by vertical lines) reported by test $\mathcal{CP}/Bootstrap$ for traces in Figure 2.16.

for the first one and a half hours of the trace that exhibits slow but steady trend, whereas $\mathcal{CP}/RankOrder$ identifies three change-points. This is because, as shown in Appendix A.1, $\mathcal{CP}/RankOrder$ is more sensitive to small changes especially when the sample timeseries is relatively short. Consequently, $\mathcal{CP}/RankOrder$ is less likely to miss actual change-points at later stage of the recursion in the extended tests for detecting multiple change-points. However, this comes at the expense of sometimes finding extraneous change-points. So the actual level of constancy falls somewhere in between.

Figure 2.20 shows the cumulative distribution of the maximum CFR and the weighted average of the duration of the CFRs (per the discussion of Figure 2.5 previously). We see that few traces are steady over the entire 5-hour period, and for 60-70%, the largest CFR is 2.5 hours long or less. The weighted averages are shifted over about 45 minutes; over half of the time we find ourselves in a change-free region of under 1.5 hours duration.

On the other hand, throughput does not wildly fluctuate minute-by-minute: only 10% of the time do we find ourselves in a CFR of under 20 minutes duration. Similarly, the median number of change-points in a trace is 8. Finally, within CFRs, we find that the individual throughput measurements are well modeled as IID, 92% passing the Box-Ljung test for autocorrelation up to 6 lags; over entire traces, however, this figure falls to 24%.

2.5.3 Operational constancy of throughput

We adopt a simple notion of operational throughput constancy, namely whether the observed bandwidth stays in a region for which the ratio between the maximum and minimum observed values is less than a factor of ρ. Figure 2.21 shows the

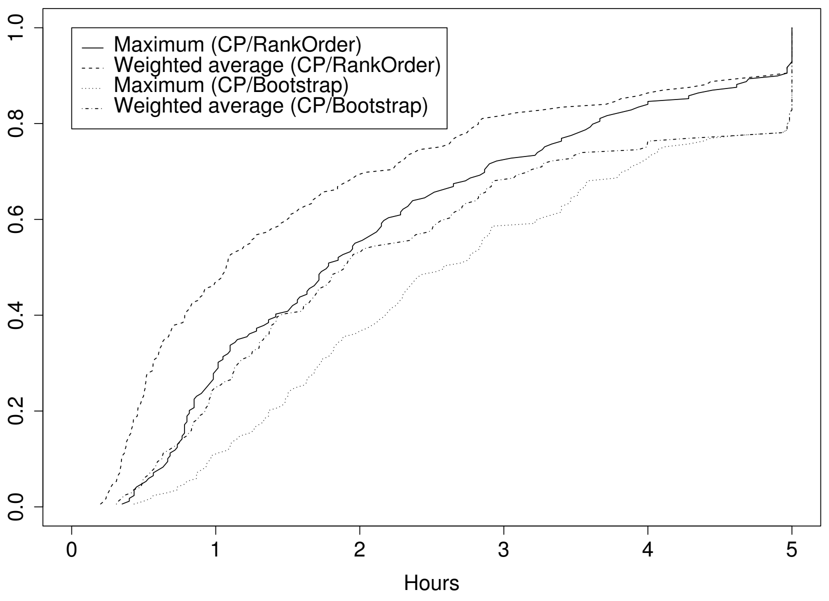


Figure 2.20: CDF of maximum and weighted average CFRs for throughput achieved transferring 1 MB using TCP.

distribution of the size of the maximum steady regions, for $\rho = 1.2$ through $\rho = 10$. We see that if our operational requirement is for bandwidth not to vary by more than 20% peak-to-peak, then we will only have a few minutes of constancy, but as ρ increases, so too does the maximal constancy, fairly steadily; for peak-to-peak variation of a factor of 3, it is often several hours.

We also find that, due to the wide range in operational constancy as we vary ρ, there is no simple relationship between the mathematical and operational constancy of throughput. For example, if we classify a trace as operationally steady if it has a maximum CFR of at least 2 hours, then we get completely different figures for $\rho = 1.2$ and $\rho = 10$; see Table 2.10 for details.

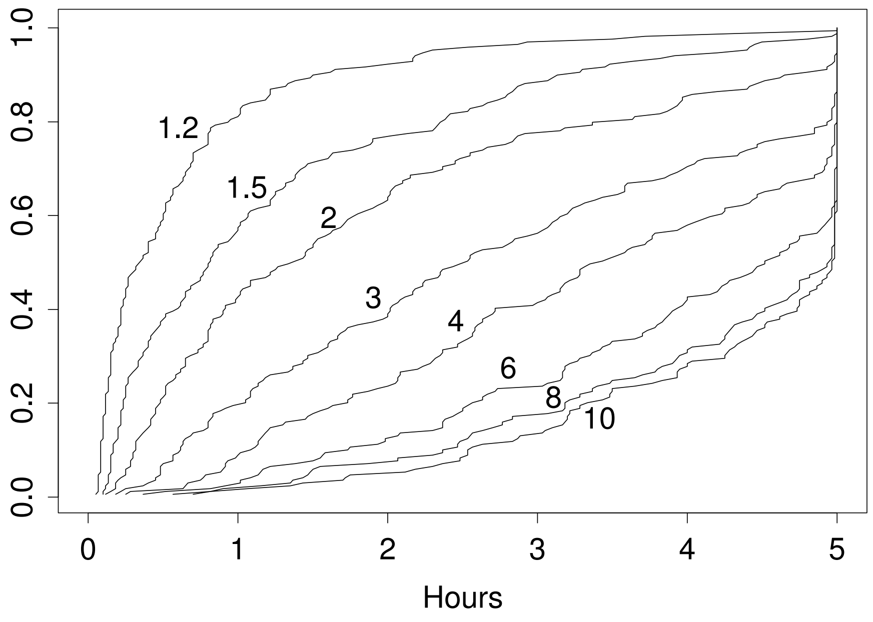


Figure 2.21: Distribution of maximum operational constancy regions for $\rho = 1.2$ (leftmost), …, $\rho = 10$ (rightmost).

Table 2.10: Comparison between mathematical throughput constancy and operational throughput constancy evaluated using different ρ.

Category	$\overline{M}\overline{O}$	$M\overline{O}$	$\overline{M}O$	MO
Proportion ($\rho = 1.2$)	53%	39%	2.4%	5.9%
Proportion ($\rho = 10$)	3.6%	1.2%	51.5%	43.8%

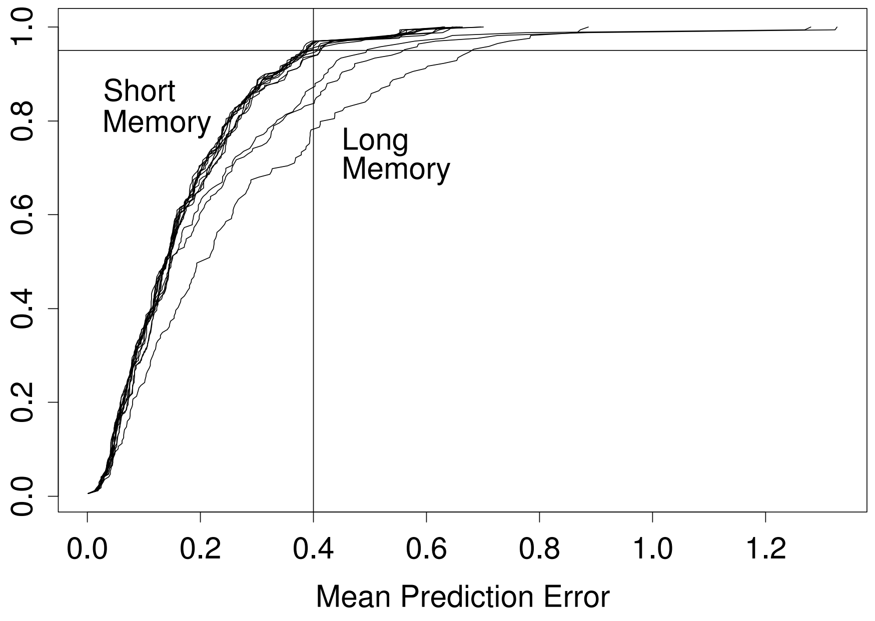


Figure 2.22: CDFs of the mean error for a large number of throughput predictors.

2.5.4 Predictive constancy of throughput

We finish our look at different types of throughput constancy with a look at how well an estimator can predict the next observed throughput measurement. Figure 2.22 shows how the families of estimators discussed in Section 2.3.7 performed in estimating the next throughput value over each 5-hour trace in its entirety. Almost all of the estimators perform equally well, with 95% of their estimates (horizontal line) yielding an error of 0.4 (vertical line) or lower, corresponding to estimating the next value within a factor of $e^{0.4} \approx 50\%$. However, three estimators do poorly: EWMA with $\alpha = 0.01$, and MA and SMA with windows of 128. These reflect estimators with long memory, as indicated on the plot (the other estimators had windows of 16 or less, or $\alpha \geq 0.125$), indicating that when predicting throughput, remembering observations from a number of minutes in the past is fine, but remembering for more than an hour can mislead the estimator. Finally, we note that for traces that are mathematically steady (maximum CFR $\geq$ 1 hour), the short-memory estimators do nearly twice as well (half the mean error) as they do on all the traces. (We do not attempt a comparison between prediction and operational constancy, since for throughput there is such a wide range of operational constancy depending on the parameter ρ.)

2.6 Summary

Applications and protocols are becoming more *adaptive* and *network-conscious.* Network operators and algorithms are increasingly relying on measurements to assess current conditions. Mathematical models are playing a larger role in the discussions of Internet traffic characteristics. For each of these developments, one of the key issues is the constancy of the relevant Internet properties. Yet each involves a quite different notion of constancy. We have discussed how mathematical, operational, and predictive constancy sometimes overlap, and sometimes differ substantially. That they can differ significantly highlights how it remains essential to be clear which notion of constancy is relevant to the task at hand.

In this chapter we have attempted to shed light on the current constancy properties of four key Internet path properties: routes, loss, delay, and throughput. One surprise in our findings is that many of the processes are well-modeled as IID, once we identify change-points in the process's median (loss, throughput) and aggregate fine-grained phenomena into episodes (loss runs, delay spikes). However, IID models are a mixed blessing; they are very tractable, but IID processes are very hard to predict.

The need to refine the analysis by looking for change-points and identifying episodes illustrates how important it is to find the right model. For example, while the loss process itself is both correlated and non-steady, when reduced to the loss episode process, the IID nature of the data becomes evident. This illustrates the importance of considering the constancy of a path property not as a fundamental property in its own right, but only as having meaning in the context of a model, or an operational or protocol need.

Another general finding is that almost all of the different classes of predictors frequently used in networking (moving average, EWMA, S-shaped moving average) produced very similar error levels. Sometimes the predictors performed well, such as when predicting RTTs, and sometimes poorly, because of the IID nature of the data (loss, throughput).

Finally, the answer to the question "how steady is the Internet?" depends greatly on the aspect of constancy and the dataset under consideration. However, it appears that for all three aspects of constancy, and all four quantities we investigated, one can generally count on constancy on at least the time scale of minutes.

Chapter 3

The Use and Performance of Content Distribution Networks

The bulk of traffic on the Internet is Web-related and a few thousand Web sites receive a significant fraction of the request traffic. A major concern of users on the Web is access latency. Changes in the HTTP protocol [32] have enabled access latency reduction via improved caching, longer-lived HTTP connections, and the ability to download selective portions of a resource. However, for sites that receive a very large number of requests daily, these changes are not sufficient to handle the rate of requests while delivering acceptable performance.

Caching, an important element of Web performance, aims to diminish the load on "origin" servers (on which Web resources either reside or are generated), eliminate redundant data traversing the network, and reduce user-perceived latency. The effectiveness of traditional caching is limited for a variety of reasons, such as diversity of resource access, increasing dynamic content, and concerns about consistency of cached responses.

The chief aim of caching is to move the content closer to the user. A *Content*

Distribution Network (CDN) consists of a *collection* of (non-origin) servers that attempt to offload work from origin servers by delivering content on their behalf. The servers belonging to a CDN may be located at the same site as the origin server, or at different locations around the network, with some or all of the origin server's content replicated amongst the CDN servers. For each request, the CDN attempts to locate a CDN server close to the client to serve the request, where the notion of "close" could include geographical, topological, or latency considerations. With content distribution, the origin servers have control over the content and can make separate arrangements with servers that distribute content on their behalf.

In this chapter, we provide background on content distribution techniques and then examine CDNs from three perspectives: 1) How are CDNs being used to serve content from origin servers? What is the nature of content being offloaded by origin servers to CDNs? 2) How are CDNs using existing protocols (DNS, TCP, HTTP) to serve content and does the CDN use of these protocols differ from that of origin servers? 3) How are CDNs performing in serving content both relative to origin servers and amongst themselves?

Our study uses a large amount of real data on both static and streaming content, takes into account all the relevant protocols implemented in different flavors, and examines issues related to protocol compliance. Our study shows that some of the widespread practices do not necessarily improve performance, and interactions across certain layers merit closer attention. The rest of the chapter is organized as follows. Section 3.1 provides background on various CDN techniques. Section 3.2 identifies major performance components for CDN-served content. Section 3.3 examines how CDNs are used today. Section 3.4 discusses protocol issues related to DNS, different TCP stacks, and levels of compliance to the HTTP protocol.

Section 3.5 describes the methodology of a performance study of CDNs with Section 3.6 detailing the results of this study. Section 3.7 summarizes the work.

3.1 Background on CDN techniques

As background on CDNs, we identify six commonly used content distribution techniques, used both in isolation and in combination with each other. Although our study is *client-centric* and relies only on what a Web client can itself measure, it is still valuable to understand the range of existing approaches. The differences in the approaches stem from how a CDN solution selects an appropriate server amongst its collection to deliver the content. Our study only examines the last two techniques since they are the commonly used ones and amenable to a large-scale, client-centric study.

3.1.1 Manual hyperlink selection

In this simple scheme, the origin site embeds in the Web page itself multiple URLs pointing to individual CDN servers. The client manually selects the link to the CDN server they want to use.

3.1.2 HTTP redirection

Some of the HTTP protocol's redirection class (3xx) response codes can be used to redirect the user agent to take alternative action, such as automatically following the link provided in the `Location` response header. Clients are redirected to a CDN server or the origin server using HTTP response codes such as `302 Found`, or `307 Temporary Redirect`. This technique requires two HTTP-level operations.

3.1.3 Layer 7 switching

Layer 7 switching, often termed Web switching, is a technique that allows a Web switch to transparently redirect client requests to different CDN servers or to the origin server based on the content of the client's request as well as the client's network address and port information. With Layer 7 switching, the client HTTP transaction sequence is as follows.

1. The client makes a request.
2. A Web switch using a virtual IP address corresponding to that of the host-name in the requested URL receives the request.
3. Upon the receipt of a TCP SYN packet, the Web switch replies to the client with a TCP SYN-ACK purportedly from the origin server. It then waits to see the first TCP *data* packet, in order to examine its HTTP header and URL information. If the request is split across multiple TCP packets, the switch must examine all of those packets. The switch then selects the best server or cache (possibly the origin server) to satisfy the request.
4. The Web switch uses one of the following two approaches to transparently redirect client requests to the chosen server.

   TCP splicing. An additional TCP connection is created between the switch and the chosen back-end server. The client request is passed to the server through this connection, and the response received by the switch from the server on the connection is transferred through the original connection back to the client. Once both TCP connections are established, the switch can set up appropriate network/port address translations and sequence number modifications to "splice" the two TCP connections, and from this point on all packets can be forwarded by the switch using custom hardware without intervention by switch controllers.

TCP endpoint migration. The Web switch creates a simplex TCP connection to the selected server and signals the client IP address, port number, and sequence number information to the server. The server can then respond to the client directly, thus removing the switch from the data path of the server response. However, the switch still needs to forward the client packets to the server. Again, such packet forwarding can be done through custom hardware.

3.1.4 Layer 4 switching

In Layer 4 switching, the switch selects the best back-end server based only on network address and port information. It thus does not need to wait for any TCP data packets. Because Layer 4 switches are content blind, they primarily help in load balancing. The client HTTP transaction sequence in Layer 4 switching begins with the same first two steps as Layer 7 switching (client's request received by Web switch). The remaining steps differ:

3. Upon the receipt of a TCP SYN packet belonging to a new session, the switch selects a server to satisfy the request and sets up NAT [66] so that the real server will transparently receive the subsequent packets. The NAT continues until the TCP connection terminates.

4. Similarly, the switch intercepts packets traveling from the real server to the client and performs the reversed address translations.

3.1.5 URL rewriting

An origin server can rewrite URL links as part of dynamically generating pages to redirect clients to different content servers. The Clearway CDN company [13], for example, can identify objects on customer origin server sites that are likely to gain from replication and push them to the CDN mirror servers. At resource access time, the page is dynamically rewritten with the IP address of one of the mirror servers, thus avoiding the need for a DNS lookup, so that the client can quickly fetch the references to replicated objects.

3.1.6 DNS redirection

DNS Redirection is the principal focus of our study.

The HTTP transaction sequence for DNS redirection is as follows:

1. The user requests a resource by typing in a URL or clicking on a link from a browser; or the request may be automatically generated as a result of the resource being embedded in a container document. The client may not know that the resource is actually served by a CDN.

2. The browser sends a DNS query to the local DNS server to obtain the IP address of the content server in the URL.

3. If it does not have the address mapping already in its cache, the local DNS server sends a query to the authoritative DNS server of the content server, which is generally provided by the CDN company.

4. The authoritative DNS server selects the best CDN server for the client based on the IP address of the client's local DNS server (not on the client's

IP address [61]) and perhaps a number of factors such as the availability of resources and network conditions. The authoritative DNS server replies with the IP address of its current choice of the best CDN server(s). Generally the reply has a low time-to-live (TTL) so that the CDN can change the mapping quickly to facilitate load balancing between its servers.

5. The client contacts the returned CDN server and uses HTTP to retrieve the content from it.

If the CDN server does not have the requested content, then it must either fail or first retrieve the content from the origin site. Alternatively, if the CDN server is overloaded it can redirect the client to a different CDN server or the origin server itself, via HTTP-level redirection or by generating dynamic pages with rewritten URLs pointing to the new server.

There are three broad flavors of DNS redirection CDNs: the full site's content can be delivered by the CDN, parts of a site (typically, images) are delivered, or redirection occurs after the URL is rewritten. With full-site content delivery, the origin server is largely hidden except to the CDN; the origin site modifies its DNS zone file (a zone is a subtree of the DNS hierarchy that is separately administered) to reflect the authoritative DNS server provided by the CDN company. Adero [2], NetCaching [47], and Unitech Networks' IntelliDNS [24] are examples of CDNs delivering full content.

The most popular flavor of CDNs is partial content delivery—the origin site modifies the embedded URLs for objects to be served by the CDN so that the host names in the URLs are resolved by the CDN's DNS server. The actual syntax of the rewritten URL varies with the CDN. For example, Speedera changes `www.foo.com/bar.gif` to `foo.speedera.net/www.foo.com/bar.gif`. The host name in the modified URL can be in the same domain as the origin site or in a different domain. In the former case, besides modifying embedded URLs, the origin site also needs to modify its zone file. Akamai [3], Digital Island [16], Mirror Image [43], Solidspeed [64], and Speedera [65] are examples of CDN companies delivering partial content.

Fasttide [18] is an example of a CDN-brokering company that combines URL-rewriting with DNS redirection. Upon receiving a client request, Fasttide rewrites the embedded URLs to identify a particular server name served by one of the CDNs from which Fasttide re-sells service (e.g. Adero and Digital Island). That CDN then uses DNS-redirection to map the server name to different IP addresses.

Table 3.1 lists the set of CDNs that we came across in our study. The list is by no means complete, but does include the most prominent set of CDN companies.

Table 3.1: Referenced content distribution networks and their URLs

CDN	URL
Adero	www.adero.com
Akamai	www.akamai.com
Clearway	www.clearway.com
Digital Island	www.digitalisland.com
Exodus	www.exodus.com
FastTide	www.fasttide.com
Intel	www.intelonline.com
Mirror Image	www.mirror-image.com
Navisite	www.navisite.com
NetCaching	www.netcaching.com
Solidspeed	www.solidspeed.com
Speedera	www.speedera.com
IntelliDNS	www.unitechnetworks.com
Yahoo!Broadcast	business.broadcast.com

3.2 Performance components

From a user's perspective, the most important metric of Web performance is the time to download a Web page. For CDNs, we can break down the total user-perceived latency into a number of factors, as follows:

3.2.1 Caching

The presence and effectiveness of a browser cache, proxy (chain) cache, and surrogate cache (a gateway delegated to operate on behalf of an origin server) all affect performance, as does the prevalence of non-cacheable content and how the CDN handles it. Since image content served by CDNs can be cached at a browser or proxy, the varying levels of caching effectiveness can have a significant impact on the relative benefit or cost of a CDN. Most CDNs cache static content, which is the initial focus of our benchmark. However, since the amount of dynamic content and streaming media is growing on the Web, it is important to understand CDN performance for non-cacheable and streaming content.

CDNs use one of the following four methods to deal with requests for non-cacheable content:

- Prevent clients from sending requests for non-cacheable content to CDN servers by only modifying embedded URLs for cacheable content.
- Use Layer 7 switching to forward the request to the origin server to serve directly.
- Open a separate HTTP connection to forward the request to the origin server, obtain the result, and then send it back to the client.

- Redirect clients to the origin server either at HTTP level or by generating dynamic pages with rewritten URLs pointing to the origin server.

Clearly, these methods can perform differently in terms of user-perceived latency. To quantify the performance, we could send HTTP GET requests for non-existent URLs to a CDN server and measure the time to get a negative result. The response for such requests should be an error message, and largely similar, in terms of time, to requests for non-cacheable content.

3.2.2 DNS lookup overhead

The overhead of the DNS mechanism may include DNS lookups of a proxy, the origin server site, and a CDN server, with possible additional lookups all the way to a root domain name server in the worst case. Because in our experience CDNs often use shorter DNS TTLs than origin sites, this lookup will be needed more frequently. If only the embedded URLs are served by the CDN, clients need to do a DNS lookup for the origin site as well as the CDN server.

3.2.3 Protocol issues

The TCP connection setup/tear-down overhead can be amortized among multiple transfers when using persistent connections (with or without pipelining). Use of pipelining over a single persistent connection to retrieve a resource and all its embedded resources appears to give the best performance [34, 48]. Another important factor is the HTTP protocol compliance [31] of the server (both origin server and CDN server). Absence of compliance can lead to performance variations and other unexpected problems. Different TCP implementations can also significantly affect performance. There has been earlier work examining the impact of TCP on Web performance [8]. Recently, a tool has been created to examine the actual TCP implementation used by a Web server [50]. The overhead for TCP connection setup/tear-down varies depending on the HTTP protocol versions and options.

3.2.4 Server issues

We can estimate server load based on the time between when the client sends its request and when the first byte of the reply appears, and compare this time with the round trip time seen for the connection establishment. This technique assumes that the server's kernel will rapidly process an incoming connection request, but that the server's ability to start replying to a request will diminish as the load on the server grows. The load may also be affected by special work the server may have to do to generate the response. In addition, we can repeat our benchmark experiments at different times of the day and week to take into account different network and server load conditions. Among the other factors we need to consider are download performance for different object sizes, dynamic and static pages with similar sizes (to quantify the extra delay for generating dynamic pages), and the effect of the server's send socket size. We ideally would like to measure how often server redirection (either HTTP-level or based on dynamic URL rewriting) occurs and how much overhead it incurs.

3.2.5 CDN use by origin sites

Finally, for partial-site content delivery, the relative percentage and size of the origin site's content served by the CDN will set an upper bound on the possible performance improvement attainable by using the CDN. For full-site content delivery, the amount of dynamic content will affect the degree to which the CDN will be able to serve content without contacting the origin site.

3.3 Use of content distribution networks

The first part of our study examined how CDNs are being used to serve content in the Web and the nature of the content served. In [34] it was reported that only 1-2% of approximately 670 popular Web sites were employing CDNs to serve content based on data gathered in November, 1999. As a follow-up to this study we compiled two lists of popular sites for determining the use of CDNs by popular Web sites. The first list, "HOTMM127," contained 127 sites obtained by obtaining the Media Metrix top 50 list [41] and the 100hot.com list [1]. The second list, "URL588-MM500," was larger, containing 1,030 sites obtained by combining the list of servers used in [31] and the Media Metrix top 500 list [42]. Home pages from sites on each of these two lists were retrieved on a daily basis during November and December 2000 for 60 days.

In analyzing the home pages and their embedded images we found that 39 (31%) of the HOTMM127 sites and 177 (17%) of the URL588-MM500 sites used a CDN to serve some of the content on the page. These results indicate a clear increase in the number and percentage of popular origin sites using CDNs to serve content in comparison with the results in [34]. CDN-served content was identified by the presence of a CDN provider name in the server portion of a URL. We also used the output of the *dig* (Domain Information Groper) utility, which does a DNS lookup, to look for a CDN provider as the authoritative name server for other server names we encountered. The second heuristic is necessary because a lot of origin sites choose to protect their brand names by avoiding any CDN provider name in the URL. Of the 39 HOTMM127 sites using CDNs, 37 used Akamai and two used Digital Island. Of the 177 URL588-MM500 sites using CDNs, 165 used Akamai, 20 used Digital Island and one used Adero. Some of these sites used more than one CDN.

We also wanted to examine the content served by CDNs that were *not* well-represented in the list of popular sites. We therefore created a list of 58 Web sites ("CDN58") believed to be served by a CDN provider other than Akamai and gathered data for a few weeks in December, 2000 using the same methodology. This list included 10 sites using Adero, 13 using Digital Island, 11 using Solidspeed and 24 using Speedera.

3.3.1 Change characteristics of CDN-served content

To better understand the characteristics of CDN-served content we used the results of our daily crawl of popular Web sites over a 60 day period during November and December 2000 to examine the rate of change of the content. We analyzed the change characteristics of this content from two perspectives: 1) how frequently the set of URLs served by a CDN changes; and 2) how frequently the same URL served by a CDN changes. The results of this analysis are shown in Table 3.2.

The results show that the set of URLs served by CDNs changes little for each of the three sets of home pages. CDN-served objects have a 86-94% chance of being previously seen. In a small number of cases, a new URL with the same contents (based on the same MD5 checksum) as a previously seen URL was found. For images, which constitute almost all of the CDN-served content, the content of a URL changes little—less than one percent based on changes in the MD5 checksum.

Table 3.2: Daily change characteristics of CDN-served objects from home pages of given sites.

Statistics	HOTMM127	URL588-MM500	CDN58
Objects (1000s)	24.9	75.0	15.6
Prev. Seen URL (%)	89	86	94
New URL with Prev. Seen MD5 (%)	1	4	0
New URL with No-Cache (%)	0	1	2
Seen URL with Missing/Changed LModtime (%)	2	2	2
Seen URL with Changed MD5 (%)	0.2	0.3	0.3

The change frequency is a bit higher when considering cases of an HTTP `no-cache` directive (used to bypass the cache and fetch a resource directly from the origin server, 0-2%) or a missing or changed lmodtime (last modification time, 2%). These results indicate that these CDNs are serving little, if any, dynamically generated content that is actually changing on each access.

3.3.2 Nature of HTTP-requested CDN content

The results from our periodic crawl of Web sites provides one perspective on the CDN-served content at these sites. However, they do not directly measure the nature of CDN-served content that has been served based on user HTTP requests. To analyze the CDN-served content served due to user requests we extracted data from two large proxy log sets—the proxy log traces from nine NLANR sites (listed in Table 3.3) recorded over the course of a week in January 2001 [49] and the traces from three sites of a large manufacturing company recorded over the course of a week in September 2000.

The NLANR traces consist of 33 million accesses from 5,023 client IP addresses. The company traces consist of 114 million accesses from 155,000 client IP addresses. These proxy logs were chosen because they are timely and represent two large and distinct user groups. We can summarize the sets of logs as follows.

- Images account for 96-98% of the CDN-served objects, but only 40-60% of the CDN-served bytes.
- Among the CDNs, Akamai serves over 85-98% of the CDN-served objects in the proxy site logs and a comparable range of the CDN-served bytes.
- Focusing on images, which predominate the CDN-served object requests, the

Table 3.3: NLANR cache server names and locations.

Server Name	Location
pb.us.ircache.net	Pittsburgh, PA
uc.us.ircache.net	Urbana-Champaign, IL
bo.us.ircache.net	Boulder, CO
sv.us.ircache.net	Silicon Valley, CA (FIX-West)
sd.us.ircache.net	San Diego, CA
pa.us.ircache.net	Palo Alto, CA
sj.us.ircache.net	San Jose, CA (MAE-West)
rtp.us.ircache.net	Research Triangle Park, NC
startap.us.ircache.net	STARTAP, the international connection point in Chicago, IL

logged cache hit rates of CDN-served images ranges from 30-80% while the cache hit rates ranges from 25-60% for non-CDN-served images. Cache hit rates are generally 20-30% higher for CDN-served content when comparing the two hit rates from the same proxy site. These results indicate some correlation between frequently requested and CDN-served image content.

3.3.3 Emerging content: streaming media

In the early days of the Web, content was mostly text. This changed in a few years to be mostly images. Recently, the content mix has again changed to include streaming media. Our study and another recent study [10] show that although streaming media is a small fraction of the number of resources, they can contribute to a significant fraction of the bytes. The recent dramatic increase in popularity of peer-to-peer networks (e.g., Napster [46], Gnutella [22]) has led to a significant increase in streaming content. Delivering large streaming media resources raises

Table 3.4: Typical streaming media content types and filename suffixes.

Content Type	Typical Filename Suffixes
Macintosh audio format	*aiff*
Microsoft audio streaming format	*asf asx*
Microsoft video	*avi*
MIDI music data	*midi*
QuickTime video	*mov*
MPEG-2 audio	*mp2*
MPEG-3 audio	*mp3 m3u*
MPEG audio/video	*mpa mpe mpeg mpg mpv*
RealAudio	*ra ram*
RealAudio plugin	*rm rpm*
Shock Wave Flash	*swf*
Microsoft audio	*wav*

Table 3.5: Suffixes for popular streaming media redirection files.

Suffix	Description
asx	Redirection file for Microsoft media
m3u	Redirection file for MP3 formats
ram	Redirection file for Real media

new performance challenges. HTTP is used as the primary protocol to access such content though the transfer of the content itself may be over other protocols. In the two proxy log sets we found less than one percent of CDN-served objects were for streaming media, but these objects accounted for 14-20% of the bytes served by CDNs over HTTP.

To look at the amount of streaming media content available, we identified twenty suffixes of Web resources typically used to name streaming content—*aiff, asf, asx, avi, midi, mov, mp2, mp3, m3u, mpa, mpe, mpeg, mpg, mpv, ra, ram, rm, rpm, swf, wav.* The corresponding content types are summarized in Table 3.4.

Using the advanced search features of the popular search engine AltaVista [4, 5], such as locating Web pages that *refer* to pages that contain any of the suffixes, narrowing the search to each day in the last three years, and specifying the number of matched results to be fetched, we created a seed list of 1,837,339 URLs. We then extracted two sets of URLs: 286,493 URLs belonging to 1,030 popular Web sites (from the URL588-MM500 list) and another set of 286,493 random URLs from the remaining collection. The combined 572,986 URLs were requested using an HTTP client program and embedded links that matched any of the suffixes were recorded. For three widely used suffixes among streaming media objects (summarized in Table 3.5), we extracted the referenced streaming media object(s).

A significant fraction of the URLs were not available (i.e., HTTP `404 Not Found` error message was returned). Of the potential total of 999,030 streaming media objects in the successfully fetched URLs, we were able to obtain the headers for 576,757 objects. These 576,757 objects were accessible via one of HTTP, PNM (Progressive Networks Media, a proprietary protocol of Real Networks), MMS (Microsoft Media Services), and RTSP (Real Time Streaming Protocol [62]).

Depending on the access protocol, we used appropriate methods to gather meta-information for the object. For HTTP, we used `HEAD`, for RTSP we used `DESCRIBE`, and for `MMS` files we used an instrumented version of ASFRecorder [7] to obtain the ASF header information. ASF (Advanced Streaming Format) is a compressed file format for storing audio and video information. We were unable to obtain meta-information for the PNM-served objects due to the proprietary nature of the protocol. Results from our data are shown in Table 3.6 with HTTP-served results from the NLANR and large manufacturing company (LMC) proxy log sets shown for comparison. The table shows that streaming media objects served by CDNs are much more likely to use MMS than the set of non-CDN served objects. Table 3.7 summarizes the number of streaming media objects accessible via different protocols (HTTP, MMS, PNM, RTSP) from popular and less-popular Web sites. We see there was no appreciable difference among the streaming objects accessed using HTTP or MMS between the popular and less-popular sites. However, objects accessible via PNM and RTSP occurred ten times less often in the set of objects extracted from popular Web sites than from the less-popular Web sites.

3.4 Protocol issues

The second part of our study examined the impact of various protocols involved in Web transfers—DNS, TCP, and HTTP. Where appropriate, we contrast their use at both origin Web server and CDN server sites. First, we examine the Time to Live values assigned by DNS servers of CDN and origin Web servers. Next, we check TCP behavior at popular Web server sites and at CDN sites to examine variance

Table 3.6: Number and percentage of streaming media objects and bytes served with each protocol.

Data source	Protocol	Non-CDN-Served Objects	Non-CDN-Served GBytes	CDN-Served Objects	CDN-Served GBytes
Crawl data	HTTP	502,550 (88%)	637.9	874 (22%)	5.1
	PNM	44,279 (8%)	–	309 (8%)	–
	MMS	20,406 (4%)	38.8	2,146 (55%)	7.3
	RTSP	5,588 (1%)	45.7	605 (15%)	1.0
NLANR	HTTP	168,686	4.3	9,147	2.2
LMC	HTTP	177,847	41.7	24,181	17.2

Table 3.7: Number of objects accessible at given sites via different protocols.

	Number of Accessible Objects	
Protocol	Popular sites	Less popular sites
HTTP	243,187	260,237
MMS	12,764	9,788
PNM	4,105	40,483
RTSP	363	5,820

in transport techniques. Finally, we examine the level of HTTP compliance in Web server implementations at CDN sites and compare against origin server sites. These protocol issues affect performance in less obvious ways than the content characteristics discussed in Section 3.3, but we bring these issues in as appropriate in looking at CDN performance results in Section 3.6.

3.4.1 Use of DNS

A widely used technique to ensure users requests are transparently redirected to CDN servers is DNS redirection, as discussed in Section 3.1.6. In order to balance client requests to CDN servers and to avoid flash crowds (whereby a sudden flurry of activity is noticed at a particular Web site due to the occurrence of a major event), CDN servers balance load within their collection of servers. A common technique used by CDNs for load balancing is assigning small DNS TTLs for the IP addresses they return forcing clients to perform frequent DNS lookups. This approach gives CDNs more control over which of their servers clients can use. We observed authoritative DNS TTLs of 10 seconds for Adero, 20 seconds for Akamai, Digital Island and Solidspeed, and 120 seconds for Speedera (See Table 3.8 for more details). Mirror Image uses a larger DNS TTL of one hour, presumably because they have fewer IP addresses from which to choose for a client, but each address is handled by a cluster of servers. Comparing these authoritative DNS TTLs to those used by a selected set of popular origin sites we find the origin site DNS TTLs ranged from 15 minutes for `cnn.com` to six hours for `espn.com`, except for a one minute DNS TTL for `bloomberg.com`.

Table 3.8: DNS TTL times (sec.)

Site	Mean	Stdev	10%	25%	Median	75%	90%
Adero	9	1	10	10	10	10	10
Akamai	19	1	20	20	20	20	20
Digisle	29	4	30	30	30	30	30
Solidspeed	19	1	20	20	20	20	20
Speedera	117	13	120	120	120	120	120
Mirrorimage	2,046	1,282	221	848	1,988	3,600	3,600
Fasttide	146	495	6	19	30	30	230
Popular US origin	5,839	16,691	20	30	279	1,545	14,889
Popular Intl. origin	5,212	6,469	300	694	2,315	7,851	14,273

3.4.2 TCP stack flavors

To examine the TCP behavior of Web servers at both origin and CDN sites, we used the TCP Behavior Inference Tool (TBIT [50, 51]). TBIT uses active probing techniques to characterize the TCP behavior of a host. We examined the TCP flavors in the various CDN servers in our study, as well as a list of 1,030 popular Web servers (denoted as "URL588-MM500") obtained by combining the list of servers used in [31] and the Media Metrix top 500 list [42].

As shown in Table 3.9, several CDNs have a slightly higher initial window size compared to the popular Web servers (although within the acceptable parameters of RFC 2414). Table 3.10 summarizes the various CDN and popular Web servers' support for ECN (Explicit Congestion Notification [58]), delayed acknowledgement, and SACK (Selective Acknowledgement [39]). As we can see, all the CDN servers we tested are ECN-capable, which is much higher than the fraction of ECN-capable origin sites. Meanwhile, there is a much higher percentage of SACK support among CDN sites, which can be useful in cases of heavy congestion. However, unlike a reasonable fraction of the origin server sites, none of the CDNs implement delayed acknowledgment. Delayed ACKs reduce acknowledgment traffic by piggybacking the ACK on data packets and alternating the ACKs, but delayed-ACK timers may degrade Web performance in a visible manner. CDNs serving small static images are likely to benefit by turning off delayed ACKs. Table 3.11 shows the congestion control algorithms used by different CDN and Web servers. Unlike an appreciable fraction of origin sites, few CDNs servers' TCP stacks implement TCP-Tahoe without fast retransmission (TahoeNoFR). Use of TahoeNoFR in origin sites indicates that those sites are less likely to time out in presence of packet loss and the absence of TahoeNoFR flavor of TCP in most CDNs is a positive sign.

Table 3.9: Initial congestion window size of various CDN and popular Web servers.

	Initial Congestion Window Size (pkts)				
Site	1	2	3	4	5 or more
Adero	90.5%	9.5%	0	0	0
Akamai	0	0.7%	99.3%	0	0
Clearway	0	100%	0	0	0
Digisle	0	1.1%	98.9%	0	0
Solidspeed	0	0	100%	0	0
Speedera	0	100%	0	0	0
URL588-MM500	27.6%	72.7%	1%	1.2%	2.5%

3.4.3 HTTP protocol compliance

A recent study examining the performance difference between various protocol options [34] showed that the absence of certain HTTP features had a direct impact on user-perceived latency. Since CDNs deliver content on behalf of the origin sites, it is important to examine the level of protocol compliance in their servers.

Table 3.10: Various CDN and popular Web servers' support for ECN, Delayed ACK, and SACK.

Site	ECN-capable	DelAck-enabled	Sack-capable
Adero	100%	0	88.1%
Akamai	100%	0	100%
Clearway	100%	0	100%
Digisle	100%	0	51.1%
Solidspeed	100%	0	100%
Speedera	100%	0	100%
URL588-MM500	76.2%	33.9%	28.6%

Table 3.11: Congestion control algorithms used by various CDN and popular Web servers.

	Congestion Control Algorithm			
Site	NewReno	Reno	Tahoe	TahoeNoFR
Adero	58.3%	36.1%	5.6%	0
Akamai	91.9%	0.7%	7.2%	0
Clearway	83.3%	0	16.7%	0
Digisle	95.6%	0	4.4%	0
Solidspeed	100%	0	0	0
Speedera	70.3%	0	29.7%	0
URL588-MM500	11.6%	33.9%	3.4%	51.1%

A recent examination of the compliance-level of implementations of the Web server at several popular Web sites showed a lack of compliance with several of the key features of HTTP/1.1 [31]. Here we studied prominent CDNs to examine any significant variance in protocol compliance in their servers as compared to origin servers. We began by generating a list of IP addresses for the various CDNs. Using a customer's Web page of each CDN, we extracted a CDN server name and then resolved it against a large list of domain name servers (over 28,000 servers in 146 countries) using *dig*. To validate the addresses we fetched a single image (known to be served by the CDN) from all the returned IP addresses to correct potentially incorrect information obtained from DNS servers. We clustered the CDN server addresses using the network-aware clustering technique outlined in [33] and randomly picked an IP address from each cluster. This approach reduces the traffic for our compliance tests and the set of addresses chosen gives a first order indication of the coverage of the CDN companies.

With the list of IP addresses of CDN servers, we tested the compliance of CDN servers over a period of several months (between October 2000 and January 2001) in a manner similar to [31]. Most of the CDNs passed the basic tests involving the `GET` and `HEAD` methods, persistent connections, pipelining, and range requests, although Speedera was a notable exception and does not support pipelining (it does pass the other tests). Overall, the compliance level of CDN servers was comparable to that of origin servers. Some of the CDNs, such as Akamai and Digital Island, still report to be running HTTP/1.0, but they appear to implement many of the HTTP/1.1 features.

3.5 Performance study of static image delivery

The third aspect of our study of CDNs examined their performance in serving static images, the predominant content type served by CDNs. The performance of CDNs can be measured in many ways—how many requests are offloaded from origin servers, their impact on client-perceived latency and their ability to efficiently load balance requests amongst a set of CDN servers. Access to CDN log data is needed to measure the actual number of requests offloaded by CDN servers, but the other two performance indicators can be measured through an active measurement study.

The study focuses on the client-perceived performance of CDNs using DNS redirection and URL rewriting. Little work has been done on measuring the performance of CDNs. One piece of work briefly examined how content distribution servers improved latency when compared to throughput from the origin servers [34]. Johnson et al. [30] assessed the degree to which two different CDNs optimally redirected requests among their mirrors. By studying three clients downloading a single 3-4KB image they found that the CDNs appeared to use the DNS mechanisms not to necessarily select optimal servers, but to avoid selection of bad servers, though it is hard to know how to generalize their study given its limited scope.

Our study evaluates response time performance of CDNs in delivering content to a set of client sites. Because the study is based on client-side measurements, it can be used to better understand performance issues for CDNs using techniques visible at the client, such as DNS redirection. The study could be used by content providers seeking to evaluate potential performance improvements by contracting with a CDN; the content provider could perform the study from across-section of their own customer sites to better understand which CDN will provide better response time relative to servers at the content provider site.

We begin by outlining our methodology starting with the construction of a canonical page used in the study. We then describe our experiment, including the measurement infrastructure, clients, CDNs and origin sites used in the study.

3.5.1 Methodology

This performance study concentrates on the delivery of static content—specifically, images—to Web clients. The primary performance measure used for the assessment is the client-perceived response latency for locating a specific content distribution server using the DNS and then downloading a set of images from the CDN server.

The performance study is appropriate for a number of reasons. First, as shown in Section 3.3, the distribution of static content in the form of images is a common feature shared by many CDNs. Second, a primary purpose of CDNs is to move content closer to end users, thereby reducing the latencies for users to retrieve the content. Third, our methodology tests both the additional delay and the effectiveness of CDNs using DNS to direct requests away from loaded servers. Fourth, our methodology can be applied to CDNs without bias. Finally, the methodology can be applied to origin sites to create a baseline to assess the relative performance of CDNs.

3.5.2 Content for study

We began this part of our study by determining realistic distributions for the number and sizes of embedded images expected on a Web page. Our motivation was to construct a "canonical page" that reflects these distributions for static images as typically served by CDNs. For each CDN we then construct a list of image URLs currently served by the CDN that closely match in size those on the canonical page. By doing so, we can retrieve from different CDNs a set of items similar to those we retrieve from other CDNs, and likewise similar to those we would find a CDN serving if it were used to serve content for a popular Web page.

We first gathered image data from the home pages and their immediate descendents of the top popular Web sites as identified by MediaMetrix [41, 42] and 100hot.com [1]. We also gathered data on images known to be served by the different CDNs; these formed the pool from which we selected the set of images to construct each CDN's canonical page. Table 3.12 shows the resulting median and mean image sizes for when we originally gathered the data in September 2000 and when we gathered the data again in January 2001. The four CDNs shown are those for which we gathered data in both time periods.

Figure 3.1 shows the size distribution of images served by various CDNs and popular Web sites in January 2001. The results for September 2000 are highly similar. We see that the distribution of the image sizes yields similar distributions for all sets. In addition, the size distribution for all embedded images is similar to the size distribution for images served by CDNs.

Similarly, we studied the number of embedded images on these Web pages, with median and mean results also shown in Table 3.12 for both periods of data collection. Pages with content served by Speedera show smaller median values for the number of embedded images, with a fair amount of consistency for the other sets of pages. In developing a canonical page of images for our study we decided on a page with 18 embedded images using the empirical distribution of sizes for embedded images from the popular Web sites to randomly generate the size distribution of these images.

Because the percentage of embedded images on a page being served by a CDN

Table 3.12: Statistics for the size and number of embedded images on given Web pages

	Sept. 2000				
Source of Pages	Image Byte Size		Images on Page		No.
(Number of Sites)	Med.	Mean	Med.	Mean	Pages
Hot100 Sites (100)	808	2,231	17	19.3	5,796
MM500 Sites (495)	765	2,201	18	20.1	23,295
Adero-Served (10)	1,085	2,435	19	18.0	362
Akamai-Served (261)	794	2,289	19	21.1	17,270
Digisle-Served (13)	762	1,463	20	13.5	665
Speedera-Served (24)	815	1,859	9	10.6	1,297

	Jan. 2001				
Source of Pages	Image Byte Size		Images on Page		No.
(Number of Sites)	Med.	Mean	Med.	Mean	Pages
Hot100 Sites (100)	758	2,756	18	20.3	5,804
MM500 Sites (495)	706	2,276	19	21.1	23,023
Adero-Served (10)	622	2,233	10	15.9	268
Akamai-Served (261)	642	2,455	20	23.5	7,506
Digisle-Served (13)	847	2,368	22	21.9	559
Speedera-Served (24)	860	2,199	10	11.7	1,073

Table 3.13: Sizes of 54 images used in the performance tests.

Index	Size (bytes)	Index	Size (bytes)	Index	Size (bytes)
1	49	19	9,776	37	61
2	1,836	20	3,020	38	11,824
3	54	21	384	39	9,753
4	2,291	22	2,425	40	3,541
5	1,272	23	354	41	1,428
6	6,635	24	430	42	880
7	78	25	788	43	82
8	6,840	26	5,732	44	9,429
9	117	27	93	45	124
10	2,175	28	12,384	46	1,118
11	912	29	160	47	2,282
12	462	30	417	48	115
13	12,902	31	571	49	91
14	2,182	32	85	50	59
15	36	33	3,526	51	3,927
16	35	34	641	52	12,705
17	2,209	35	3,451	53	46
18	307	36	334	54	291

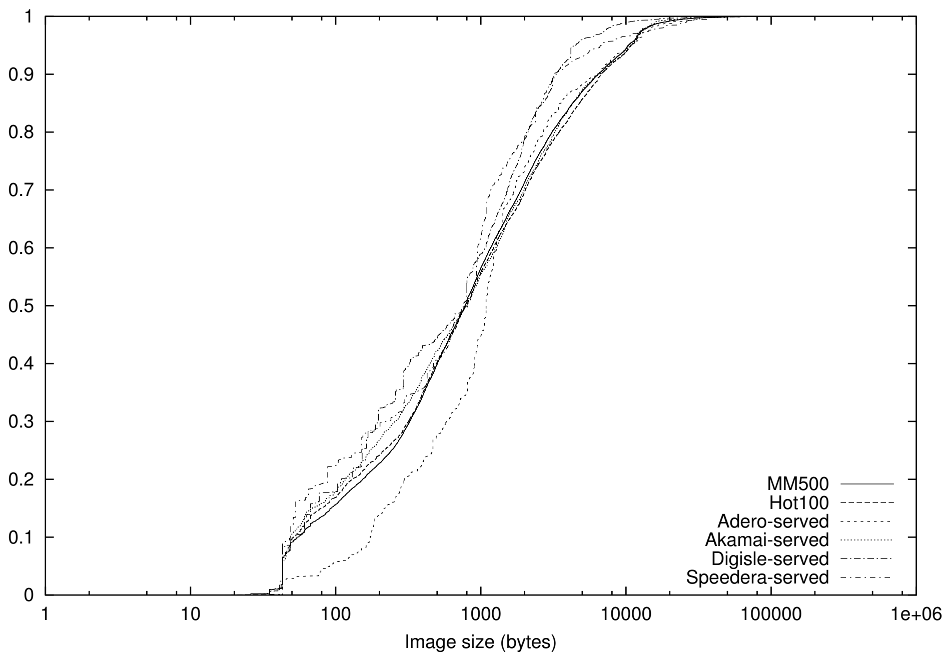


Figure 3.1: Size distribution of images served by CDNs and popular Web sites (Jan. 2001)

varies, we also examine variations on our canonical page of 18 images. To do so we have drawn a set of 54 image sizes from the distribution for testing the downloading of a large number of images. Some pages contain fewer than 18 images or images may be cached as indicated by results from Section 3.3. Therefore in our tests we actually download 54 images, but record the intermediate measurements for downloading 6, 12 and 18 images. All results reported in this paper are for downloading the first 18 images unless otherwise specified. Table 3.13 lists the sizes of all 54 images in the order they are retrieved in our tests.

3.5.3 Study description

We describe the particulars of matching images actually served by a CDN or origin site with those in our canonical page in Section 3.5.5. The basic algorithm, which mimics the steps taken by a user agent, to retrieve a set of images from a server is as follows:

1. For CDNs using only DNS redirection, perform a DNS lookup of the server name to obtain an IP address for the server. Record the time taken using *dig*. For CDNs using URL rewriting, we first go to the origin site that is the source of the CDN-served images to determine the current CDN server from which to download the images. If this server is an IP address, then use that address, otherwise do a DNS lookup for the IP address.

2. Retrieve all images from the server at the given IP address. We use *httperf* [44], modified to allow specification of a specific IP address, for all Web object retrievals. This step is not timed, rather, it is intended to ensure that all images have been retrieved and cached by the CDN server if not already present. We want this instance of our study to measure delays for downloading content from the CDN server to the client, and not to unknowingly include delays for a CDN server to retrieve contents from the origin site.

3. Retrieve all images from the server at the given IP address using a separate TCP connection for each image with up to four images being retrieved in parallel. Measure the delay to establish the connection, to receive the first byte of the reply for each image request sent, and to retrieve the remaining bytes of each image. In addition to this HTTP/1.0 style of connections, we also test two retrievals approaches based on HTTP/1.1. In both HTTP/1.1 approaches we use up to two persistent connections to a server. In one test we use serialized requests over these persistent connections and in the other test we use pipelined requests. As reported in Section 3.4, not all CDNs supported

these options, but results are shown for the CDNs that do support them.

This basic methodology is repeated on a periodic basis over the course of a day so that time–of-day effects will tend to average out. We repeat the test every 30 minutes (with up to 10 minutes of jitter to avoid synchronization effects) for each client to each CDN and each origin site in our test set.

3.5.4 Client sites for study

The methodology is defined independent of particular clients, CDNs, or origin sites. We exercised the methodology from a collection of two dozen worldwide sites that comprise part of the NIMI measurement infrastructure [57]. NIMI consists of a number of widely deployed measurement "platforms" that accept authenticated requests to schedule measurements (in our case, scripts running *httperf*) for some future time, perform the measurements at the indicated time, and send back the results. The number of available NIMI platforms gives us good breadth in our client test base, though one limitation is that NIMI is currently heavy on U.S. university and government laboratory sites—particularly on the U.S. East and West coasts.

3.5.5 Content distribution networks and origin sites for study

We began by creating an instance of the canonical page using images served by that CDN. To find images served by the CDN, we used the results from the background study described in Section 3.5.2. Table 3.14 shows the source of the images for each of the six CDNs we evaluated in January 2001. Using this source of images, we found the image closest in size to each of the 54 images in our complete set of image sizes. The table shows the average size difference in bytes between the images served by the CDN and those image sizes on the list. The relatively small averages indicate success in matching actual images with target image sizes. As a means for comparison, we also created instances of the canonical page for some popular U.S. and international origin sites from images being served by those sites.

Table 3.14: Instantiation of canonical page for CDNs and origin sites

	Site	Source of Images	Avg. size diff. (bytes)
CDN	Adero	images.mothernature.com	32.8
	Akamai	ivillage.com(a820.g.akamai.net)	37.8
	Clearway	nothingness.org	11.6
	Digisle	fp.cnbc.com	49.8
	Fasttide	www.itat.com	78.0
	Speedera	yack.speedera.net	9.6
US	amazon.com	www.amazon.com	7.6
	bloomberg.com	www.bloomberg.com	29.9
	cnn.com	www.cnn.com	2.9
	espn.com	espn.go.com	1.8
	mtv.com	www.mtv.com	2.1
	nasa.gov	www.hq.nasa.gov	6.6
	playboy.com	www.playboy.com	1.2
	sony.com	www.spe.sony.com	12.0
	yahoo.com	us.yimg.com	10.4
International	UK	www.bbc.co.uk	2.0
	Korea	image.hanmail.net	7.3
	UK	www.msn.co.uk	19.9
	Australia	www.telstra.com.au	11.0
	Brazil	www.uol.com.br	1.2
	Japan	st6.yahoo.co.jp	10.5

3.6 Results of static image performance study

In the study, we took measurements from the NIMI clients on three separate weekdays in September, 2000, using only the HTTP/1.0 protocol with parallel retrievals. Four CDNs employing the DNS redirection technique were tested. Later, we took three measurements from the NIMI clients on three separate weekdays in January 2001, using the three separate retrievals methods as described in Section 3.5. In the January tests, we also add two more CDNs to our study—Clearway and Fasttide, which both use the URL rewriting technique. Our initial set of test CDNs only used DNS redirection. The three datasets each in September 2000 ($\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$) and January 2001 ($\mathcal{J}_1$, $\mathcal{J}_2$, $\mathcal{J}_3$) are summarized in Table 3.15. All runs started shortly after 4AM EDT and ran every 30 minutes until 1AM or 2AM the following day. The datasets ranged from 24 to 25 NIMI clients returning measurement results. The primary results in this chapter are from $\mathcal{J}_3$, the last of the three January 2001 datasets, which included results from 19 U.S.-based clients. We focus on U.S. clients because they are more representative of the geographic United States relative to the sparse representation of the international NIMI clients. Comparisons with results from $\mathcal{S}_1$, the first of the three September 2000 datasets, are made as appropriate. Mostly, the three experiments from each timeframe show consistent results, but we note differences as appropriate.

3.6.1 Response time results

The first set of results examine the DNS lookup time to obtain the IP address of a specific CDN server to download the 18 images from the canonical test page for each of the CDNs and origin sites in our study set. The completion time seen by

Table 3.15: Summary of datasets collected in the CDN study.

Dataset	Date	# successful retrievals
$\mathcal{S}_1$	09/12/2000	17,995
$\mathcal{S}_2$	09/20/2000	15,369
$\mathcal{S}_3$	09/21/2000	16,595
$\mathcal{J}_1$	01/03/2001	90,561
$\mathcal{J}_2$	01/10/2001	50,865
$\mathcal{J}_3$	01/19/2001	71,529

a client is the sum of these two components. Table 3.16 and Table 3.17 show the results of this study for each CDN for a September 2000 and January 2001 test. The results shown are for downloads done with parallel HTTP/1.0 requests. The tables also show combined results for the set of U.S. and international origin sites.

Table 3.16: DNS lookup times for images 1-18 from U.S. clients to CDNs and origin sites.

	DNS Lookup Time (sec.)					
CDN/	Sept. 2000: $\mathcal{S}_1$			Jan. 2001: $\mathcal{J}_3$		
Origin Site	Mean	Med.	90%	Mean	Med.	90%
Adero	5.68	0.14	13.44	4.26	0.15	8.53
Akamai	0.22	0.04	0.20	0.10	0.03	0.17
Clearway		–		0.00	0.00	0.00
Digisle	0.18	0.04	0.14	0.12	0.04	0.15
Fasttide		–		0.56	0.11	0.51
Speedera	0.14	0.03	0.14	0.15	0.04	0.17
U.S. Origin	0.13	0.00	0.14	0.33	0.03	0.20
Intl Origin	0.51	0.00	0.63	0.46	0.00	0.41

The mean, median, and 90th percentile results show that in September 2000 most CDNs provided better download performance for the U.S. clients than did the

Table 3.17: Download times (sec.) for images 1-18 from U.S. clients to CDNs and origin sites with parallel HTTP/1.0 requests.

CDN/	Parallel-1.0 Download Time (sec.)					
	Sept. 2000: $\mathcal{S}_1$			Jan. 2001: $\mathcal{J}_3$		
Origin Site	Mean	Med.	90%	Mean	Med.	90%
Adero	1.66	1.27	3.04	1.16	1.02	1.77
Akamai	2.40	0.81	4.79	1.06	0.34	3.01
Clearway		–		1.26	0.85	2.98
Digisle	1.35	0.43	3.08	1.15	0.50	1.80
Fasttide		–		1.55	0.96	3.37
Speedera	2.70	2.92	4.35	0.57	0.20	1.14
U.S. Origin	2.66	1.25	5.38	3.40	1.06	4.90
Intl Origin	5.67	3.70	11.81	3.62	3.12	5.55

U.S. origin sites, and that in January 2001 all CDNs provided substantially better download performance. Download results for Speedera changed dramatically as it performed the worst in the September test, but the best in the January test. This performance improvement corresponded with a large increase in the number of CDN servers deployed by Speedera. The overall download time results are relatively consistent over the three NIMI tests within each test period.

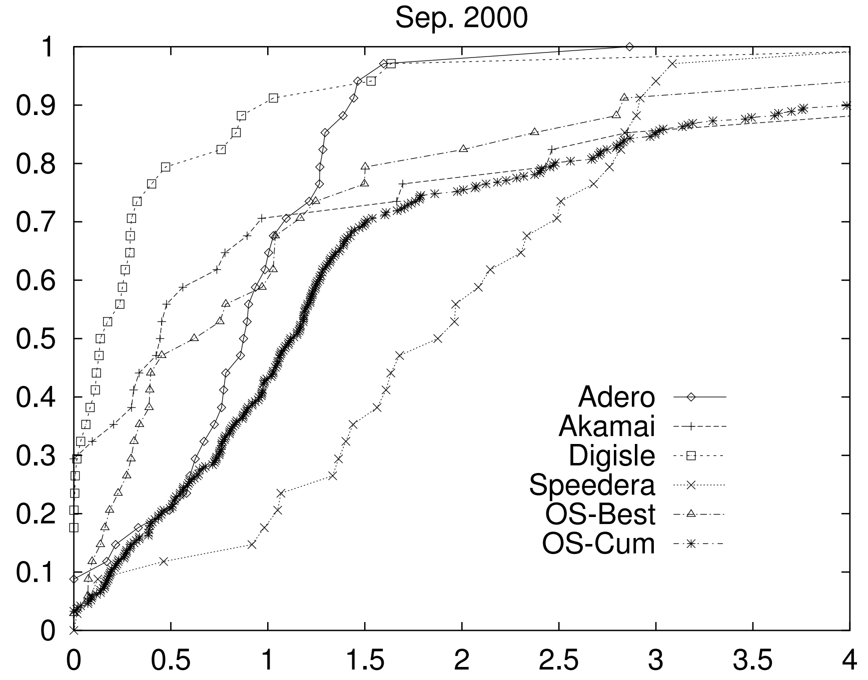
Clearway incurs no DNS lookup time because load balancing amongst servers is done by rewriting URLs to directly include server IP addresses. Particularly striking in the DNS results are the large times for Adero with a mean of over five seconds in September and four seconds in January. Further investigation shows that relatively short DNS TTLs not only for the first-level name server of Adero (10 sec.), but also for upper-level name servers for Adero and `images.mothernature.com` (30 min. to 3 hrs.) leads to potentially four non-cached DNS lookups, each of which may introduce a timeout of 5 seconds. Approximately 25% of the DNS lookups

Table 3.18: Download times (sec.) for images 1-18 from U.S. clients to CDNs and origin sites with serial and pipelined HTTP/1.1 requests.

CDN/ Origin Site	Jan. 2001 ($\mathcal{J}_3$) Serial-1.1 Download Time (sec.) Mean	Med.	90%	Pipeline-1.1 Download Time (sec.) Mean	Med.	90%
Adero	0.87	0.81	1.48	0.88	0.75	1.67
Akamai	0.61	0.24	1.36	0.47	0.16	1.07
Clearway	0.96	0.87	1.61	0.55	0.46	0.81
Digisle	1.13	0.36	1.42	0.51	0.20	0.66
Fasttide	1.05	0.85	2.18	0.75	0.55	1.55
Speedera	0.62	0.38	1.28	no support		
U.S. Origin	1.96	1.11	3.84	partial support		
Intl Origin	3.76	3.50	5.72	partial support		

took more than five seconds in the January test. The high DNS lookup times for Adero were consistent in all NIMI tests during both the September and January tests.

In the January test we also examined the download times using HTTP/1.1 persistent connections. As indicated in Section 3.4, all CDNs support persistent connections even though Akamai and Digital Island servers only report to support only HTTP/1.0. The results in Table 3.18 show that all CDNs, except for Speedera with pipelining, successfully supported both serial and pipelined requests. Approximately 50% of the U.S. and international server sites supported pipelining. The download results in Table 3.18 can be compared with those results in Table 3.17 and show that use of persistent connections yields better results than parallel-1.0 requests for all CDNs. Akamai provides the best overall download times using persistent connections.



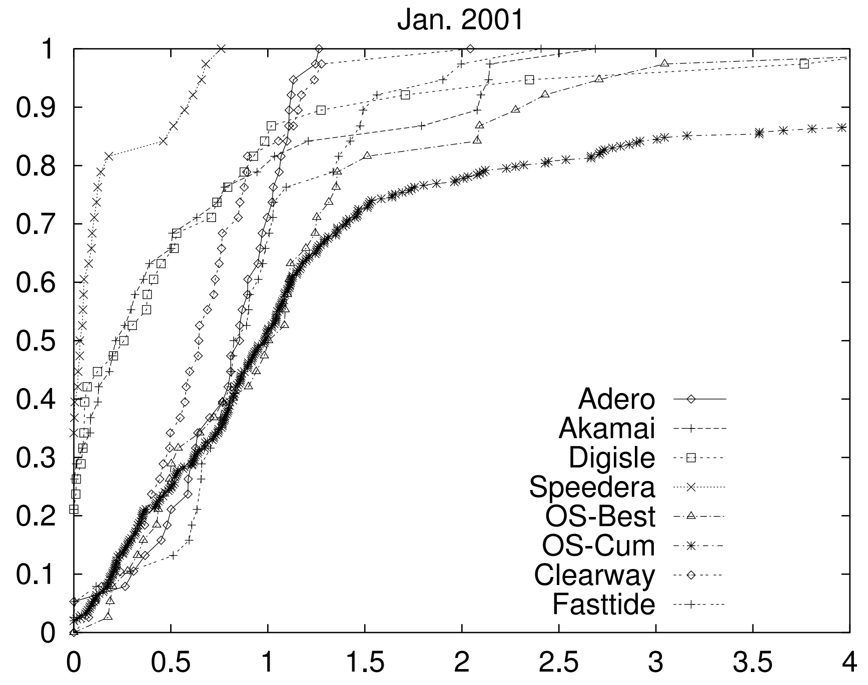
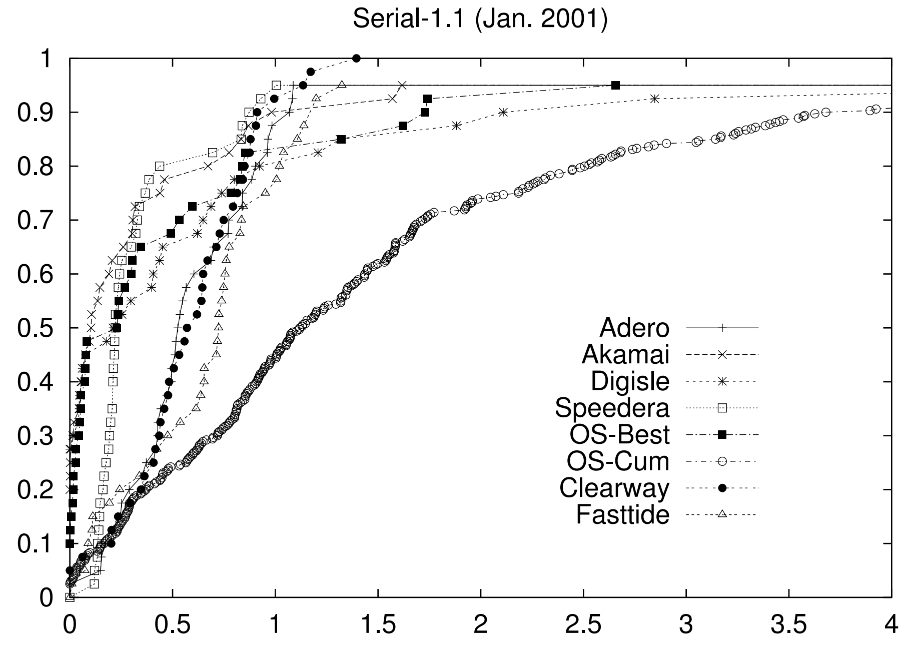


Figure 3.2: CDF of difference between CDN mean/median and best CDN mean/median for each client (sec.) with Parallel-1.0 requests (Sept. 2000: $\mathcal{S}_1$ and Jan. 2001: $\mathcal{J}_3$) .



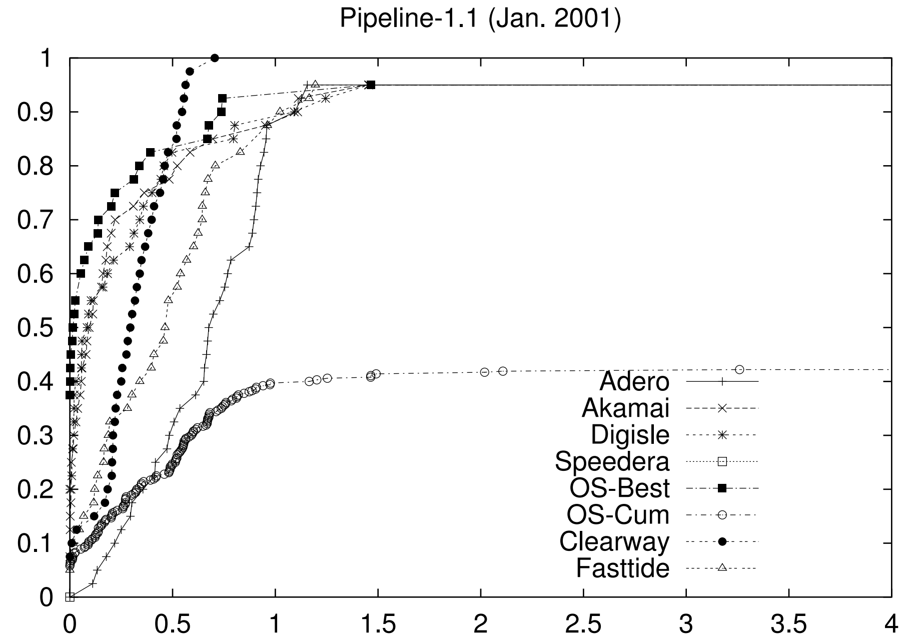


Figure 3.3: CDF of difference between CDN mean/median and best CDN mean/median for each client (sec.) with Serial-1.1 and Pipeline-1.1 requests (Jan. 2001: $\mathcal{J}_3$) .

We also examined performance for individual clients by using both mean and median download times (ignoring DNS delay) for each of the CDNs at each of the U.S. clients used during that test. Results in Figure 3.2 show a cumulative distribution function for the difference between a CDN's performance and the best performing CDN at each client. Results are shown for both test periods. In September, the performance of Digital Island was the best of the CDNs for over 30% of the clients and was within 0.5 seconds of the best for over 80% of the clients. Akamai exhibited the most variation, with over 70% the results either the best or within one second of the best, but also over 10% of its mean and median results more than four seconds slower than the best. The results also show that each CDN provided the best mean or median results for at least one client. Most of the CDNs performed better than the best performing origin server (OS-Best) and the cumulative performance of all origin servers (OS-Cum).

In the January results of Figure 3.2, Speedera clearly shows the best results and all CDNs perform relatively better than OS-Cum. Further examination of results for serial-1.1 (Figure 3.3) show that Akamai and Speedera both perform well with each having mean and median download times within one second of the best for over 90% of the clients. Akamai was the best CDN performer for the pipeline-1.1 results (Figure 3.3) with all CDNs that support pipeline-1.1 performing better than the cumulative origin server performance. Using pipeline-1.1, Clearway provides consistently good results with all Clearway clients experiencing mean/median downloads within 0.7 seconds of the best performer at each client.

We also examined variation in results due to the number of images to be downloaded. A summary of these results (from January 2001 test) are given in Table 3.19, which shows the range of mean download times for CDNs for a given

Table 3.19: Mean download performance (sec.) from CDN and origin servers for different numbers of images and protocol options (Jan. 2001: $\mathcal{J}_3$).

Site	Protocol Option	Number of Downloaded Images			
		6	12	18	54
CDN	Parallel-1.0	0.26–0.76	0.40–1.23	0.58–1.53	1.49–3.31
	Serial-1.1	0.27–0.53	0.42–0.81	0.61–1.13	1.46–2.52
	Pipeline-1.1	0.26–0.50	0.37–0.67	0.47–0.88	1.09–2.04
US origin	Parallel-1.0	1.63	2.45	3.40	8.42
	Serial-1.1	1.06	1.46	1.96	4.87
	Pipeline-1.1	0.36	0.50	0.67	2.44

number of images to download and each protocol option. The results show close to linear correspondence between the number of images and the download performance for each of the protocol options. Reducing the number of images and using the HTTP/1.1 protocol options both reduce the range variation among the CDNs. In these cases the DNS lookup performance becomes a bigger contributor to the overall response time. The mean download times from origin servers are also shown in Table 3.19. They are clearly higher than the CDN range. These results indicate CDNs offer better overall performance than this set of origin servers for this set of clients. The results indicate that caching of these images, which reduces the number needing be retrieved, reduces, but does not eliminate, the performance difference.

3.6.2 CDN server use

The number of servers available to a CDN may impact its performance. We analyzed the number of distinct IP addresses returned to our clients in response to DNS queries or URL rewriting. Table 3.20 shows the mean, median, and maximum number of IP addresses used for a CDN on a per-client basis for both the September and January tests. It also shows the total number of distinct server IP addresses used by the collective set of clients in each test.

Table 3.20: Number of distinct IP addresses returned to a client.

	Sept. 2000: $\mathcal{S}_1$					Jan. 2001: $\mathcal{J}_3$				
CDN	mean	med.	min	max	total	mean	med.	min	max	total
Adero	4.6	5	1	9	13	4.8	5	2	8	11
Akamai	5.8	2	2	17	65	8.5	8	2	19	103
Clearway			–			5.6	6	5	6	6
Digisle	2.7	3	1	5	24	3.4	4	1	6	24
Fasttide			–			8.7	9	3	11	23
Speedera	1.5	1	1	3	3	10.3	10	5	26	83

The results show a significant change in the number of servers discovered for Speedera in the two test sets corresponding to Speedera's improved performance. The size of Akamai's discovered network also grew while the size of the network for Adero and Digital Island was relatively static.

3.6.3 DNS load balancing

In Section 3.4 we observed that CDNs using DNS redirection assigned relatively small DNS TTLs to minimize caching of DNS mappings and allow for DNS load balancing. We examined the effectiveness of this load balancing in yielding better download and completion times for clients. We modified our basic testing structure so that each client does a DNS lookup and stores a "fixed" IP address for each CDN server. This fixed address was actually looked up every eight hours, asynchronously to our other testing. We compared how often the fixed IP address was the same as the new one obtained from the DNS for the test each half-hour (if so, then the lookup was unnecessary overhead). If the IP addresses were different, we did a separate preload (step 2 in Section 3.5.3) and download (step 3) from the fixed IP address and compared the download results obtained from the two separate servers, in order to assess just what the redirect gained. We show a summary of these results for parallel-1.0 requests in Figure 3.4 for both the September 2000 and January 2001 tests. Results for the HTTP/1.1 protocol options in January are similar in nature.

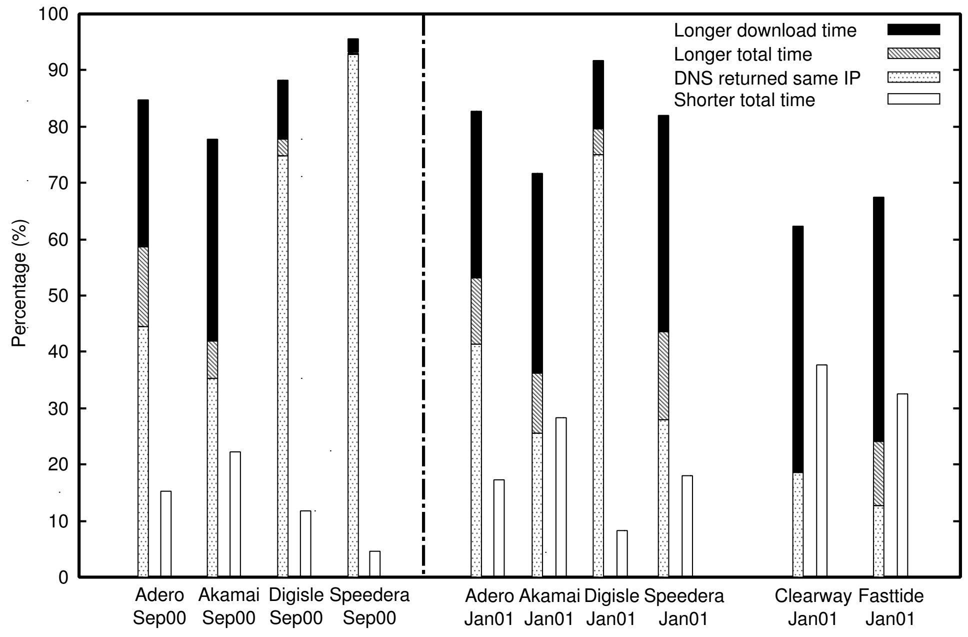


Figure 3.4: Performance comparison of new vs. fixed IP address (Sept. 2000: $\mathcal{S}_1$ and Jan. 2001: $\mathcal{J}_3$).

The results for each CDN are broken into four categories. The first three are plotted together, and represent cases in which the extra DNS lookup had no positive benefit, and in general increased the response time; the last represents the case where the redirection was clearly beneficial and is plotted in a separate column. In the Sep. 2000 test the first category (fixed and new IP address are the same) accounts for 30–40% of the cases for Akamai to over 90% of the cases for Speedera. This category accounts for 15% (Fasttide) to 70% (Digital Island) in the January test. In these cases, the download times would be identical for the fixed and new IP address, but DNS lookup costs are incurred for the new IP address, increasing the overall completion time. The second category represents cases where the combined DNS and download time for the new IP address are larger than the download time for the fixed IP address, but the download time by itself is not. Thus, we lost performance in this case, but only when we factor in the DNS overhead. This category represents up to 10% of the cases for both tests. The third category of comparison occurs when the download time (irrespective of DNS costs) for the new IP address is larger than for the fixed IP address. This category represents a clear loss of performance, even if we do not consider the cost of the DNS lookup. Akamai has the most cases in this category in both tests (30-40%).

The last category, plotted separately, shows the percentage of cases where the overall completion time is better for the new IP address over the download time of the fixed IP address. These cases show where the time to do a DNS lookup is warranted in terms of better overall response time for the client. In September, Speedera had 5% of its cases in this category with about 20% for Akamai. In January, Akamai, Clearway and Fasttide were in the 30-40% range.

Table 3.21 shows the mean, median and 90th percentile values for the new download, new completion (including DNS lookup), and fixed download times for the parallel-1.0 requests in both January 2001 and September 2000. The results are mixed as to whether the average download times for the new IP address are better than for the fixed IP address. However, if we compare the completion time for the new IP address with the download time for the fixed IP address, which

Table 3.21: Parallel-1.0 performance (sec.) for server at new and fixed IP addresses.

	Sept. 2000: $\mathcal{S}_1$								
	New Download Time			New Completion Time			Fixed IP Download Time		
CDN	Mean	Med.	90%	Mean	Med.	90%	Mean	Med.	90%
Adero	1.83	1.35	3.20	7.37	2.38	15.20	1.84	1.32	3.39
Akamai	2.84	0.95	5.93	3.08	1.11	6.61	2.72	0.91	6.45
Clearway		–			–			–	
Digisle	1.58	0.52	3.19	1.79	0.59	3.72	1.78	0.55	3.45
Fasttide		–			–			–	
Speedera	2.93	2.95	4.92	3.08	3.04	5.16	3.04	3.01	4.99

	Jan. 2001: $\mathcal{J}_3$								
	New Download Time			New Completion Time			Fixed IP Download Time		
CDN	Mean	Med.	90%	Mean	Med.	90%	Mean	Med.	90%
Adero	1.15	1.02	1.73	5.40	1.39	9.60	1.09	0.51	1.60
Akamai	1.06	0.34	3.01	1.15	0.39	3.05	1.00	0.41	3.00
Clearway	1.19	0.84	2.94	1.19	0.84	2.94	1.16	0.76	3.07
Digisle	1.19	0.47	1.83	1.31	0.52	2.30	1.21	0.43	1.70
Fasttide	1.58	0.96	3.37	2.10	1.19	4.72	1.46	0.91	3.25
Speedera	0.57	0.20	1.18	0.72	0.26	1.53	0.53	0.18	1.01

incurs no DNS cost, then we see that for all CDNs, except median Akamai times in January 2001, the fixed IP address performs better. Furthermore, the 90% percentile results indicate that the DNS lookup is not improving the worst case download times. Lowering the bound on worst case results is another argument for using small DNS TTLs, but only in the case of Clearway are the 90% download results better for the new versus the fixed IP address. Inclusion of the DNS lookup times only increases the difference in results. For the two HTTP/1.1 protocol options the only case where the completion time for the new IP address is better than the fixed download time is for median serial-1.1 results of Speedera.

We performed a similar study and analysis using the previous IP address returned in our study rather than a fixed IP address. In this study, the previous server was obtained in a DNS lookup that occurred 30 minutes ago rather than up to eight hours ago. In the September test, for all of the CDNs, except Digital Island, the download time using the new IP address was actually worse than the download time using the previous IP address and for all of the CDNs, the completion time with the new IP address was worse than the download time with the previous IP address. In the January test, the completion using the new IP address was always worse than the download time using the previous IP address except for the Clearway median results under serial-1.1. Results from other test sets were similar.

These results indicate that use of a small DNS TTL by the CDNs, which forces a DNS lookup in the critical path of resource retrieval, does not generally result in better server choices being made relative to client response time. While CDNs may decide small DNS TTLs are needed for other reasons, our results indicate that improved client response time is generally not one of them.

3.7 Summary

This chapter provides a discussion and analysis of CDNs. We have used multiple sources of data: active measurements obtained via repeated crawls over a period of time and passive measurements representing large number of users from different organizations. We have analyzed the various protocols used by CDNs in terms of compliance and by implementation flavors. We have also analyzed content types commensurate with traffic patterns on the Web.

Results from a long-term crawl of popular Web sites show that the use of CDNs to offload content from origin servers has increased significantly over the past year. CDNs are being used by origin servers to serve roughly the same type of relatively static content. Using Web proxy logs we found that requested objects served by CDNs are largely images with a 20-30% higher cache-hit rate than for non-CDN-served images at the proxy cache. CDN-served streaming media objects requested in the proxy logs constitute less than one percent of objects, but 14-20% of bytes. Streaming media objects are served by the HTTP protocol, but also with PNM, MMS and much less frequently via RTSP. HTTP was the dominant protocol for serving streaming media content for non-CDN servers, but MMS was used for the majority of CDN-served objects.

In examining the use of the DNS, TCP and HTTP protocols by CDNs relative to their use by origin servers we found that CDNs using DNS redirection assign small authoritative DNS TTLs to balance load amongst their collection of servers. These DNS TTLs are generally much smaller than the DNS TTL of popular origin servers. CDN servers' TCP stacks did not use Tahoe without fast retransmissions in contrast to origin servers where an appreciable fraction did. CDNs did not use delayed acknowledgments though 30-40% of the origin server sites we tested did.

We found similar levels of HTTP/1.1 compliance between origin servers and CDNs.

In a performance study from a set of two dozen client sites over a period of many months, each CDN yielded the best performance for at least one client when considering mean and median download time as measures of comparison. Some CDNs generally provide significantly better results than others when we examine results from the entire set of clients. In looking at relative performance with a selected set of popular U.S. origin servers, we found that in our initial set of tests, the CDNs performed better, but not significantly better than the origin servers. In the latter set of tests the CDNs performed significantly better for downloading small to large numbers of images. We also studied the performance effect caused by CDNs using small DNS TTL values to load balance amongst a set of servers. We compared the download time for the set of images from the server returned for a DNS lookup with the download time for a fixed server of the CDN. We also looked at results using the previous server returned by a DNS lookup. In most cases we found that the download time for the newly obtained server was not better than for the fixed or previous server. In almost all cases, for all CDNs and all HTTP protocol retrieval options, we found that when factoring in the time for the DNS lookup, response time was actually better using the previous or fixed server. This result indicates that client response time is generally not improved with small DNS TTLs.

Chapter 4

Conclusions and Future Work

In this chapter, we summarize our contributions, and point out directions for future research, both in addressing the limitations of this work, and in pursuing new avenues.

4.1 Contributions

We endeavored in this thesis to characterize the end-to-end Internet performance in general, meaningful ways. The Internet's great diversity has made this undertaking immensely challenging.

To address the challenges, we have measured and analyzed data about a wide range of network functions, ranging from low-level end-to-end Internet path properties, to user-perceived performance of large-scale Internet systems. Our results have deepened our understanding of today's Internet performance and can provide useful guidelines for better designing and provisioning the future network. At the core of our measurements and analysis is a set of concepts and tools that we have developed to facilitate our understanding of the Internet behavior. Compared to the detailed results from our measurements, these fundamental concepts and tools might prove longer-lived—they can be applied to analyze future Internet behavior which may be arguably quite a bit different from today's Internet behavior due to changing traffic.

Below we describe some of the most important lessons we have learned in more details.

1. Throughout our study, we devote substantial effort to cope with various measurement and analysis difficulties caused by the enormous diversity of the Internet. Our experience suggests some effective strategies that can be used by future studies to tackle such diversity—repeating the study over a relatively long period of time, measuring a sufficiently diverse set of Internet paths, and taking into account multiple protocol layers.

2. In today's networking practices, constancy is often either completely ignored or taken for granted. In our constancy study, we have identified and carefully distinguished between three different notions of constancy: mathematical, operational, and predictive. Our results have demonstrated how these notions sometimes overlap, and sometimes differ substantially. That they can differ substantially suggests that making inappropriate assumptions of constancy properties can potentially lead to misleading conclusions. This highlights how it remains essential to be clear which notion of constancy is relevant to the task at hand.

3. We have found that many of the processes are well-modeled as IID, once we identify change-points in the process's median and aggregate fine-grained phenomena into episodes. This is very surprising especially given the tremen-

dous diversity and complexity of Internet behavior. Of course, IID models are a mixed blessing; they are very tractable analytically, but IID processes are very hard to predict.

Note that in many cases the underlying constancy property comes to the surface only after we refine the analysis by identifying change-points and considering episodes. For example, the loss process itself is both correlated and non-steady, but when reduced to the loss episode process, the IID nature of the data becomes evident. This suggests we should consider the constancy of a path property not as a fundamental property in its own right, but only as having meaning in the context of a model, or an operational or protocol need.

4. Another general finding is that almost all of the different classes of predictors frequently used in networking (moving average, EWMA, S-shaped moving average) produced very similar error levels. The parameters don't matter, nor does the particular averaging scheme. Sometimes the predictors performed well, such as when predicting RTTs, and sometimes poorly, because of the IID nature of the data (loss, throughput). This suggests that for many network-aware applications, what really matters is whether or not they adapt and on what basic time scale, not how they adapt.

5. Meanwhile, while the exact level of constancy depends greatly on the aspect of constancy and the dataset under consideration, it appears that for all three aspects of constancy, and all four quantities we investigated, one can generally count on constancy on at least the time scale of minutes.

6. In our study, we devote considerable attention to detect pathologies and

modal behavior in the data, and have indeed found a number of them. For example, we have uncovered routing loops, outages, periodicity, packet replication, a sawtooth pattern in throughput data. These pathological behavior, if left unaddressed, potentially could seriously skew our analysis. This illustrates the importance of sanity checking in dealing with the tremendous diversity of Internet behavior.

7. We have provided a timely discussion and analysis of the role of content distribution networks. To the best of our knowledge, this is the first serious effort on assessing CDNs. Our study uses a large amount of real data on both static and streaming content, takes into account all the relevant protocols implemented in different flavors, and examines issues related to protocol compliance and performance. We believe, our experience provides a baseline on an effective way to evaluate large-scale Internet systems and applications.

8. Among our results, we have found significant performance variation among different CDNs. However, caching and the use of advanced HTTP/1.1 features tend to significantly reduce the performance difference both among different CDNs, and between CDNs and origin sites. Meanwhile, a common technique—DNS-based redirection—in general does not improve client download time. These findings highlight the importance of measurement-based study in the development of future Internet technologies.

4.2 Future research

There are several directions of future work suggested by our research. Some of them are extensions of our work, while others explore new avenues.

First, we would like to better understand the CDN performance. A natural follow up of our study is to expand our methodology to study the performance of CDNs in serving streaming content. We are also interested in assessing the CDN performance in the presence of flash crowds. CDNs handle flash crowds that develop because of a sudden news event, or well-known events such as large sporting events or webcasts. It is hard in practice to test CDNs performance during a flash crowd given the rarity of their occurrence and the disruptive nature of such a test, but a limited test during flash crowd situations is plausible.

For the constancy study, we want to refine our change-point analysis framework to remove some of the bias exhibited by the current algorithm. Meanwhile, we would like to investigate how the constancy behavior we identified in this study relates to other fundamental aspects of Internet dynamics such as the long range dependence (LRD) phenomenon [37].

Finally, we would like to use the insights obtained in our study to improve the performance of Internet applications. For instance, we are interested in developing effective algorithms to estimate the network conditions for adaptive applications. We want to come up with more effective CDN redirection techniques. We are also intrigued by the server selection and replica placement problems in content distribution networks.

Appendix A

Statistical Methodology

In this appendix we discuss the three main statistical techniques we use in our analysis—tests for change-points, independence, and exponential interarrivals.

A.1 Testing for change-points

We apply two different tests, $\mathcal{C}P/RankOrder$ and $\mathcal{C}P/Bootstrap$, to detect changes in the median. As noted in the beginning of Chapter 2, neither test is perfect. The first test—$\mathcal{C}P/RankOrder$—is *biased* towards sometimes finding extraneous change-points. The effect of the bias is to underestimate the duration of steady regions in our datasets. The second test—$\mathcal{C}P/Bootstrap$—does not have the bias. However, it is *less sensitive* to small changes. To accommodate the imperfection, we apply both tests so that we can give some bound on the size of steady regions. Below we describe these tests in detail and then use Monte Carlo simulation to evaluate their accuracy.

Both $\mathcal{C}P/RankOrder$ and $\mathcal{C}P/Bootstrap$ detect change-points in a two step approach: first identifying a candidate change-point, then applying a statistical test to determine whether it is significant. The combined approach [36, 68] uses an analysis of ranks in order to detect changes in the median [63]. Being based on ranks, the method is resistant, i.e., tolerant to the presence of outliers. Furthermore, the hypotheses underlying these test are quite weak; equality of variances is not required.

Consider first a set of n values $(x_i)_{i=1,\ldots,n}$ comprising a segment of a given time series. Construct the rank r_i of each x_i within the set, i.e., 1 for the smallest and n for the largest. Compute the cumulative rank sums $s_i = \sum_{j=1}^{i} r_i$. The basis of the test is that if no change point is present, the cumulative rank sums s_i should increase roughly linearly with i. Indeed, suppose we form the adjusted sum:

$$s'_i = |s_i - \overline{s}_i|$$

as the difference between s_i, and its presumed mean $\overline{s}_i = i(n+1)/2$ assuming no change-point to be present. Then s'_i should stay close to zero. If, however, a change-point is present, higher ranks should predominate in either the earlier or later part of the set, and hence s'_i will climb to a maximum before decreasing to zero at $i = n$. We identify the maximizing index i_0 for s'_i and i running over $\{1, \ldots, n\}$ as a candidate change-point.

In the second stage, to test equality of two sets $X^- = \{x_1, \ldots, x_{i_0-1}\}$ and $X^+ = \{x_{i_0+1}, \ldots, x_n\}$, $\mathcal{CP}$*/Bootstrap* uses the bootstrap analysis procedure outlined in [68], while $\mathcal{CP}$*/RankOrder* uses the Fligner-Policello Robust Rank-Order Test [63].

- *Bootstrap analysis* (used in $\mathcal{CP}$*/Bootstrap*). The bootstrap analysis procedure outlined in [68] uses S_{diff}, defined as $(\max s_i - \min s_i)$, to estimate the magnitude of the change at the candidate change-point. It determines the confidence level of change by testing how often the bootstrap difference S^0_{diff}

of a bootstrap sample $\{x_i^0\}$—a random permutation of $\{x_i\}$—is less than the original difference S_{diff}.

- *Fligner-Policello Robust Rank-Order Test* (used in $\mathcal{C}P/$*RankOrder*). The test statistic is constructed as follows. For $x \in X^+$ define r_x^+ as the rank of x in $X^+ \cup X^-$ minus the rank of x in X^+, with rank ties handled by assigning the average rank to all members of a tie set. Define rank mean $r^+ = \sum_{x \in X^+} r_x^+ / n^+$ where $n^+ = \#X^+$, and sums of squared differences $v^+ = \sum_{x \in X^+} (r_x^+ - r^+)^2$. Define n^-, r^-, and v^- symmetrically. Then the test statistic:
$$z = \frac{n^+ r^+ - n^- r^-}{2\sqrt{r^+ r^- + v^+ + v^-}}$$
has, asymptotically as $n \to \infty$, a standard normal distribution. Thus we can associate a significance level with the candidate change point i_0 in the usual manner. By choosing a significance level ℓ (we use 5% throughout this thesis) we specify our acceptable probability ℓ of incorrectly rejecting the null hypothesis. The test accepts the null hypothesis (in a two-sided test) if $F(|z|) < 1 - \ell/2$ where F is the cumulative distribution function of the standard normal distribution. (However, note that the large n asymptotic is not sufficiently accurate when i_0 and $n - i_0 \leq 12$; in this case Table K in Appendix I of [63] should be used.) In some cases we shall use this test on binary data, in which case it reduces to a test of the equality of the expectations corresponding to binary states on either side of the candidate change-point.

The above can be extended to the identification of multiple change points, as follows [36, 68]. First, choose a significance level. Second, apply the above method recursively to the two segments $\{1, \ldots, i_0\}$ and $\{i_0+1, \ldots, n\}$ until no more change points are found at the chosen significance level. Third, apply backward elimination to reinspect the set of candidate change points in order to eliminate false detections, as follows. Let there be m change-point candidates $j_1 < \cdots < j_m$. Let j_0 and j_{m+1} be 0 and n respectively. Starting with the first identified candidate, call it j_{k_0} $(1 \leq k_0 \leq m)$, reinspect for change-points on the set $\{j_{k_0-1} + 1, \ldots, j_{k_0+1}\}$, and adjust or delete non-significant change-points. Repeat for all candidates in order of identification. Repeat backward elimination until the number of change-points is stable. By reestimating each change-point using only the data between the two surrounding change-points, backward elimination avoids the contamination caused by the presence of multiple change-points at the time of recursion and consequently helps to reduce the rate of false detections.

Below we use Monte-Carlo simulations to assess the accuracy of both tests. Table A.1 shows the false positive ratios (i.e., how often the tests falsely identify a change-point) on IID random samples drawn from a Gaussian distribution. All the ratios are computed over 50,000 simulations. We use 5% significance level for both tests. Therefore, if the test has no bias, we expect to observe a false positive ratio of around 5%. This is indeed the case for $\mathcal{CP}$/*Bootstrap*. In contrast, $\mathcal{CP}$/*RankOrder* falsely detects a change-point nearly half of the time. This suggests that $\mathcal{CP}$/*RankOrder* is biased towards finding extraneous change-points. The main reason for this bias is that we are applying the Fligner-Policello Robust Rank-Order Test to the point of maximum deviation instead of a random location. Such preconditioning changes the asymptotic distribution of the test statistic, and makes the test overly sensitive. $\mathcal{CP}$/*Bootstrap* does not have this bias, because it is explicitly designed to be testing the maximum deviation via bootstrapping.

Table A.1: False positive ratios of $\mathcal{C}P/RankOrder$ and $\mathcal{C}P/Bootstrap$ on IID samples with different lengths (N).

Test	$N = 100$	$N = 1,000$
$\mathcal{C}P/RankOrder$	42.4%	43.4%
$\mathcal{C}P/Bootstrap$	4.9%	5.0%

Table A.2: Accuracy of $\mathcal{C}P/RankOrder$ and $\mathcal{C}P/Bootstrap$ on samples with a single change-point (sample length=100).

		$\mathcal{C}P/RankOrder$			$\mathcal{C}P/Bootstrap$		
	Actual		Detected C.P.			Detected C.P.	
Δ	C.P.	FN	mean	stdev	FN	mean	stdev
0.5σ	25	19.8%	36.3	16.2	73.9%	35.3	12.7
	50	13.5%	50.0	10.7	53.9%	50.0	7.8
	75	19.7%	63.8	16.1	74.0%	64.9	12.5
σ	25	0.4%	30.1	8.4	16.5%	30.1	7.8
	50	0.1%	50.0	4.2	2.1%	50.0	4.1
	75	0.4%	69.8	8.4	16.4%	69.9	7.8
1.5σ	25	0.0%	28.0	5.1	0.3%	28.0	5.1
	50	0.0%	50.0	2.1	0.0%	50.0	2.1
	75	0.0%	72.0	5.0	0.2%	72.1	5.0
2σ	25	0.0%	27.1	3.6	0.0%	27.1	3.6
	50	0.0%	50.0	1.2	0.0%	50.0	1.2
	75	0.0%	73.0	3.5	0.0%	73.0	3.5
3σ	25	0.0%	26.4	2.4	0.0%	26.4	2.4
	50	0.0%	50.0	0.5	0.0%	50.0	0.5
	75	0.0%	73.6	2.5	0.0%	73.6	2.5

Table A.3: Accuracy of $\mathcal{C}P/RankOrder$ and $\mathcal{C}P/Bootstrap$ on samples with a single change-point (sample length=1000).

		$\mathcal{C}P/RankOrder$			$\mathcal{C}P/Bootstrap$		
	Actual		Detected C.P.			Detected C.P.	
Δ	C.P.	FN	mean	stdev	FN	mean	stdev
	250	0.0%	274.7	44.1	0.0%	274.7	44.1
0.5σ	500	0.0%	500.0	19.6	0.0%	500.0	19.6
	750	0.0%	724.8	44.5	0.0%	724.8	44.5
	250	0.0%	257.6	14.3	0.0%	257.6	14.3
σ	500	0.0%	500.0	5.1	0.0%	500.0	5.1
	750	0.0%	742.6	13.9	0.0%	742.6	13.9
	250	0.0%	253.8	7.1	0.0%	253.8	7.1
1.5σ	500	0.0%	500.0	2.3	0.0%	500.0	2.3
	750	0.0%	746.3	7.0	0.0%	746.3	7.0
	250	0.0%	252.4	4.4	0.0%	252.4	4.4
2σ	500	0.0%	500.0	1.2	0.0%	500.0	1.2
	750	0.0%	747.6	4.5	0.0%	747.6	4.5
	250	0.0%	251.6	2.9	0.0%	251.6	2.9
3σ	500	0.0%	500.0	0.5	0.0%	500.0	0.5
	750	0.0%	748.5	2.9	0.0%	748.5	2.9

Table A.4: Probabilities of falsely detecting multiple change-points on traces with a single change-point.

	Actual	$\mathcal{C}P/RankOrder$		$\mathcal{C}P/Bootstrap$	
Δ	C.P.	$N = 100$	$N = 1000$	$N = 100$	$N = 1000$
0.5δ	$0.25N$	47.1%	70.0%	2.1%	8.8%
	$0.50N$	52.4%	66.5%	3.3%	9.2%
	$0.75N$	46.1%	69.7%	1.9%	8.6%
δ	$0.25N$	62.8%	70.3%	5.4%	11.0%
	$0.50N$	61.7%	67.2%	6.5%	9.7%
	$0.75N$	62.5%	70.0%	5.6%	11.0%
1.5δ	$0.25N$	66.0%	70.3%	7.6%	10.2%
	$0.50N$	63.9%	68.0%	9.0%	9.7%
	$0.75N$	66.4%	70.5%	7.9%	10.5%
2δ	$0.25N$	67.5%	69.9%	8.9%	10.0%
	$0.50N$	66.2%	67.9%	8.9%	9.6%
	$0.75N$	67.0%	70.3%	8.7%	10.3%
3δ	$0.25N$	67.6%	69.2%	8.9%	9.7%
	$0.50N$	65.6%	68.0%	9.5%	9.6%
	$0.75N$	67.1%	69.4%	9.0%	9.9%

Table A.2 and Table A.3 show (*i*) the false negative ratios (i.e., how often the tests fail to detect an actual change-point), and (*ii*) how close the detected change-points are from the actual ones. The sample timeseries are drawn i.i.d. from a Gaussian distribution with a single shift in median (mean). We vary the magnitude of change (Δ) among 0.5σ, σ, 1.5σ, 2σ, and 3σ, where σ is the standard deviation of the Gaussian distribution; and vary the location of the actual change-point to be one of the quarter points. The length of the sample timeseries is kept at either 100 (for Table A.2) or 1,000 (for Table A.3). Again, all the statistics are computed over 50,000 simulations; and 5% significance level is used for both tests. We see that when the length of the sample timeseries is relatively small (100), $\mathcal{C}P/RankOrder$ is much more sensitive to small changes ($\Delta \leq \sigma$) than $\mathcal{C}P/Bootstrap$. Note that we often need to apply the test to such small samples, especially at later stages of the recursion. In all other scenarios, the two tests achieve the same good performance—there are virtually no false negatives and the reported change-points are very close to the actual ones.

We finish with a look at the accuracy of the extended tests for detecting multiple change-points. Table A.4 shows the probabilities of falsely detecting multiple change-points on traces with a single change-point. We see that $\mathcal{C}P/RankOrder$ falsely reports additional change-point(s) about two thirds of the time, while $\mathcal{C}P/Bootstrap$ does so only about 10% of the time. The oversensitivity of the extended $\mathcal{C}P/RankOrder$ test is apparently due to the bias of $\mathcal{C}P/RankOrder$.

A.2 Testing for independence

We assess independence using the Box-Ljung test [38]. For a time series with n elements, the Box-Ljung statistic Q_k is a weighted sum of squares of measured autocorrelations r_i from lags 1 up to k:

$$Q_k = n(n+2) \sum_{i=1}^{k} \frac{r_i^2}{n-i}.$$

Under the null hypothesis that the process comprises independent Gaussian random variables, the distribution of Q_k converges, for large n, to a χ^2 distribution with k degrees of freedom. Thus by comparing the test statistic Q_k with the $1 - \ell$ quantile of the appropriate χ^2 distribution, we can test whether the autocorrelations of the time series differ at significance level ℓ from those of independent Gaussian random variables. In fact, as remarked in [38], the test is relatively insensitive to departures from the Gaussian hypothesis in the underlying process. This is because the measured autocorrelations r_i are asymptotically Gaussian provided the marginal distribution of the underlying process has finite variance. (While infinite variance—heavy tails—abound in networking behavior, the time series we consider here are generally well bounded, and certainly have finite variance.)

A.3 Testing for exponential distributions

An exploratory test for an exponential distribution of inter-event times is to plot the log-complementary distribution function; for an exponential distribution this is linear with slope equal to the negative of the reciprocal of the mean. A statistical test is that of Anderson-Darling [6]. This test has been found to be more powerful than either the Kolmogorov-Smirnov or the χ^2 tests, i.e., its probability of correctly rejecting the null hypothesis (that the distribution is exponential) is greater; see [15]. This is, in part, due to the fact that the Anderson-Darling test employs the full empirical distribution (rather than binning, as in a χ^2 test), allowing it to give more weight to larger sample values whose presence can lead to a violation of the null hypothesis.

For a set of n rank-ordered inter-event times $t_1 < \cdots < t_n$, the appropriate Anderson-Darling statistic is:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)\left\{\log(1-e^{-t_i/\bar{t}}) - t_{n+1-i}/\bar{t}\right\}$$

where $\bar{t} = n^{-1}\sum_{i=1}^{n} t_i$ is the empirical mean inter-event time. We reject the null hypothesis at significance level ℓ if the test statistic exceeds the tabulated values appropriate for that level; see, e.g., Table 4.11 in [15]. We note the importance of using the table appropriate to the present case in which the mean is estimated from the sample, rather than being specified in advance. Moreover, the table explicitly takes into account the effect of a finite sample size n.

Bibliography

[1] 100hot.com, June 2000. `http://www.100hot.com/`.

[2] Adero, Jan. 2001. `http://www.adero.com/`.

[3] Akamai, Jan. 2001. `http://www.akamai.com/`.

[4] AltaVista, 2001. `http://www.altavista.com`.

[5] AltaVista advanced search cheat sheet, 2001.
`http://www.altavista.com/sites/help/adv_search/syntax`.

[6] T. W. Anderson and D. A. Darling. Asymptotic theory of certain goodness of fit criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23:193–212, 1952.

[7] ASFRecorder. `http://asfrecorder.virtualave.net/`.

[8] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. TCP Behavior of a Busy Internet Server: Analysis and Improvements. In *Proceedings of IEEE Infocom 1998 Conference*, Mar. 1998.

[9] J.-C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proceedings of ACM SIGCOMM '1993*, Sept. 1993.

[10] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy. Measurement and analysis of a streaming-media workload. In *Proceedings of Usenix Symposium on Internet Technologies and Systems (USITS '2001)*, San Francisco, California, USA, Mar. 2001.

[11] K. Claffy, H.-W. Braun, and G. Polyzos. Tracking long-term growth of the NSFNET. *Communications of the ACM*, 37(8):34–45, Aug. 1994.

[12] K. Claffy, G. Polyzos, and H.-W. Braun. Measurement considerations for assessing unidirectional latencies. *Internetworking: Research and Experience*, 4(3), Sept. 1993.

[13] Clearway, Jan. 2001. `http://www.clearway.com/`.

[14] K. G. Coffman and A. M. Odlyzko. Internet growth: Is there a “Moore’s Law” for data traffic, 2000.

[15] R. B. D’Agostino and M. A. Stephens, editors. *Goodness-of-Fit Techniques*. Marcel Dekker, New York, 1986.

[16] Digital Island, Jan. 2001. `http://www.digitalisland.com/`.

[17] N. Duffield. Private communication, 2001.

[18] Fasttide, Jan. 2001. `http://www.fasttide.com/`.

[19] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proceedings of ACM SIGCOMM ’2000*, Aug. 2000.

[20] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. *IEEE/ACM Transactions on Networking*, 2(2):122–136, Apr. 1994.

[21] E. Gilbert. Capacity of a burst-noise channel. *Bell Systems Technical Journal*, 39(5):1253–1265, Sept. 1960.

[22] Gnutella, 2000. `http://gnutella.wego.com`.

[23] J. Gwertzman and M. Seltzer. World-Wide Web cache consistency. In *Proceedings of the 1996 Usenix Technical Conference*, Jan. 1996.

[24] Unitech Networks’ IntelliDNS, Jan. 2001.
`http://www.unitechnetworks.com/IntelliDNS/`.

[25] One-way transmission time. ITU Recommendation G.114, International Telecommunication Union, Feb. 1996.

[26] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM ’88*, Aug. 1988.

[27] V. Jacobson. traceroute, 1989.
`ftp://ftp.ee.lbl.gov/traceroute.tar.Z`.

[28] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323, IETF, May 1992.

[29] W. Jiang and H. Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proceedings of NOSSDAV ’2000*, June 2000.

[30] K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek. The measured performance of content distribution networks. In *Proceedings of the Fifth International Web Caching Workshop and Content Delivery Workshop*, Lisbon, Portugal, May 2000. `http://www.terena.nl/conf/wcw/Proceedings/S4/S4-1.pdf`.

[31] B. Krishnamurthy and M. Arlitt. PRO-COW: Protocol Compliance on the Web—A Longitudinal Study. In *Proceedings of Usenix Symposium on Internet Technologies and Systems (USITS '2001)*, Mar. 2001. `http://www.research.att.com/~bala/papers/usits01.ps.gz`.

[32] B. Krishnamurthy, J. C. Mogul, and D. M. Kristol. Key Differences between HTTP/1.0 and HTTP/1.1. In *Proceedings of the Eighth International World Wide Web Conference*, Toronto, Canada, May 1999. `http://www.research.att.com/~bala/papers/h0vh1.html`.

[33] B. Krishnamurthy and J. Wang. On network-aware clustering of web clients. In *Proceedings of ACM SIGCOMM '2000*, Aug. 2000. `http://www.research.att.com/~bala/papers/sigcomm2k.ps`.

[34] B. Krishnamurthy and C. Wills. Analyzing factors that influence end-to-end Web performance. In *Proceedings of the Ninth World Wide Web Conference*, Amsterdam, Sweden, May 2000.

[35] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001. To appear.

[36] J. Lanzante. Resistant, robust and non-parametric techniques for the analysis of climate data: theory and examples, including applications to historical radiosonde station data. *International Journal of Climatology*, 16(11):1197–1226, 1996.

[37] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transaction on Networking*, 2(1):1–15, Feb. 1994.

[38] G. Ljung and G. Box. On a measure of lack of fit in time series models. *Biometrika*, 65:297–303, 1978.

[39] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgement Options. RFC 2018, Apr. 1996.

[40] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proceedings of ACM SIGCOMM '96*, Aug. 1996.

[41] Media Metrix top 50 list for June 2000. `http://www.mediametrix.com/press/releases/20000720a.jsp`.

[42] Media Metrix top 500 list for May 2000.
`http://www.mediametrix.com/data/top500.jsp`.

[43] Mirror Image, Jan. 2001. `http://www.mirror-image.com/`.

[44] D. Mosberger and T. Jin. httperf—a tool for measuring Web server performance. In *Proceedings of the First Workshop on Internet Server Performance (WISP '98)*, Madison, Wisconsin, USA, June 1998.
`http://www.hpl.hp.com/personal/David_Mosberger/httperf`.

[45] A. Mukherjee. On the dynamics and significance of low frequency components of Internet load. *Internetworking: Research and Experience*, 5:163–205, Dec. 1994.

[46] Napster, 1999. `http://www.napster.com`.

[47] NetCaching, Jan. 2001. `http://www.netcaching.com/`.

[48] H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. Lie, and C. Lilley. Network performance effects of HTTP/1.1, CSS1, and PNG. In *Proceedings of ACM SIGCOMM '97*, Sept. 1997.

[49] NLANR. Proxy cache log traces, 2001. `ftp://ircache.nlanr.net/Traces/`.

[50] J. Padhye and S. Floyd. TBIT, the TCP behavior identification tool, July 2000. `http://www.aciri.org/tbit`.

[51] J. Padhye and S. Floyd. On inferring TCP behavior. In *Proceedings of ACM SIGCOMM '2001*, Aug. 2001. To appear.

[52] V. Paxson. Growth trends in wide-area TCP connections. *IEEE Network*, 8(4):8–17, July/August 1994.

[53] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, Oct. 1997.

[54] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):277–292, June 1999.

[55] V. Paxson. Private communication, 2001.

[56] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.

[57] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large-scale Internet measurement. *IEEE Communications*, 36(8):48–54, Aug. 1998.
`ftp://ftp.ee.lbl.gov/papers/nimi-ieee-comm98.ps.gz`.

[58] K. K. Ramakrishnan and S. Floyd. A Proposal to add Explicit Congestion Notification (ECN) to IP. RFC 2481, Jan. 1999.

[59] J. Rice. *Mathematical Statistics and Data Analysis (second edition)*. Duxbury Press, 1995.

[60] H. Sanneck, G. Carle, and R. Koodli. A framework model for packet loss metrics based on loss run length. In *Proceedings of SPIE/ACM SIGMM Multimedia Computing and Networking Conference*, Jan. 2000.

[61] A. Shaikh, R. Tewari, and M. Agrawal. On the effectiveness of DNS-based server selection. In *Proceedings of the IEEE Infocom 2001 Conference*, Anchorage, Alaska, USA, Apr. 2001.

[62] H. Shulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326, IETF, Apr. 1998.

[63] S. Siegel and N. Castellan. *Non-parametric Statistics for the Behavioral Sciences*. McGraw–Hill, New York, 1988.

[64] Solidspeed, Jan. 2001. `http://www.solidspeed.com/`.

[65] Speedera, Jan. 2001. `http://www.speedera.com/`.

[66] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, IETF, Aug. 1999.

[67] W. R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.

[68] W. A. Taylor. Change-point analysis: A powerful new tool for detecting changes, 2000. `http://www.variation.com/cpa/tech/changepoint.html`.

[69] Telcordia Technologies. Internet Sizer.
`http://www.netsizer.com`.

[70] K. Thompson, G. Miller, and R. Wilder. Wide–area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–33, Nov. 1997.

[71] R. Wolff. Poisson arrivals see time averages. *Operations Research*, 30(2):223–231, 1982.

[72] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modeling of the temporal dependence in packet loss. In *Proceedings of IEEE INFOCOM '99*, Mar. 1999.

[73] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001. To appear.

[74] Y. Zhang, V. Paxson, and S. Shenker. The stationarity of Internet path properties. Technical report, ACIRI, May 2000.