Client-based Characterization and Analysis of End-to-End Internet Faults

Venkata N. Padmanabhan
Microsoft Research

Sriram Ramabhadran
UC San Diego

Jitendra Padhye
Microsoft Research

March 2005

Technical Report
MSR-TR-2005-29

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Client-based Characterization and Analysis of End-to-End Internet Faults

Venkata N. Padmanabhan
Microsoft Research

Sriram Ramabhadran
UC San Diego

Jitendra Padhye
Microsoft Research

***Abstract*— We present a client-based characterization of end-to-end Internet faults. Unlike prior studies of Internet faults that have focused on probing routers using tools such as traceroute and/or listening in on routing protocol messages, we consider a novel approach based on having clients *passively* observe *end-to-end transactions* that they are involved in. Observations from multiple clients are combined to arrive at a more complete picture of the extent and the likely cause of faults.**

We present the characterization of real faults observed by a heterogeneous collection of 134 client hosts, as they repeatedly downloaded content from a diverse set of 80 web sites, over a period of one month. We find a wide range in the failure rate of these transactions (e.g., 100% failure rate for certain client-server pairs). About 30% of transaction failures are due to DNS problems, with most of the rest being due to the inability of the client to be able to establish a TCP connection to the server. Also, by correlating failure observations across clients and servers, we find that client-side problems account for the overwhelming majority of DNS lookup failures whereas server-side problems are the dominant cause of TCP connection failures.

We believe that our findings suggest the promise of a novel approach to diagnosing end-to-end Internet faults based on leveraging the collective experience of a diverse set of end-hosts to overcome the opacity of the network. We briefly discuss the key challenges in realizing such a system.

I. Introduction

We present a client-based characterization of end-to-end Internet faults. Here "client" refers to end-hosts such as user desktop or home computers, and "end-to-end fault" refers to the failure of communication between the clients and other hosts such as servers.

Our goal is to characterize network problems in terms that are meaningful to end users (e.g., server problem, client site-specific problem etc.), by only using that information which is available by passive monitoring of end-to-end traffic at the client hosts.

Our analysis correlates information gathered at multiple clients to answer the following key questions:

- What is the frequency of end-to-end failures, and how does the frequency vary across the various categories of clients and servers in our study?
- When downloads from a server fail, what typically is the fraction and distribution of clients that are affected? Do failures tend to affect most or all clients across the board (indicating a server-side problem) or a smaller subset of clients (indicating a client-side problem)?
- In the case of client-side problems, do the affected clients tend to share attributes such as network location?

The data presented in this paper was gathered over a period of one month, during which a diverse set of 134 clients communicated with a target server set of 80 web servers. We recorded and analyzed failures of end-to-end transactions between these clients and servers.

Our main findings are as follows:

- The overall failure rate for a given server or a given client can be noticeable. Failure rates in excess of 2% are not uncommon. The failure rate varies considerably across servers and clients. About 30% of the failures can be traced to DNS problems, and most of the rest are due to the inability of the client to establish a TCP connection to the remote web server.
- By correlating failures observed across multiple clients and websites, we can categorize many of the failures either as *server-side* (i.e., affecting a significant fraction of accesses to the server from many clients) or *client-side* (i.e., affecting a significant fraction of accesses from a client to many sites) or otherwise. We find that client-side problems account for the overwhelming majority of DNS lookup failures whereas server-side problems are the dominant cause of TCP connection failures.
- In the case of multiple clients that are co-located (e.g., two planetlab nodes in the same university), there is often a good correlation in the faults that are flagged as client-side by our analysis. This gives us confidence in the validity of our analysis despite the difficulty of doing direct validation.

We believe that our findings indicate the promise of a novel paradigm for diagnosing end-to-end Internet faults based on leveraging the collective experience of a diverse set of end-hosts to overcome the opacity of the network.

There has been much prior work on characterizing Internet faults and developing tools to identify their their cause. We now briefly discuss how our approach is different from previous work. A detailed discussion appears in Section VI.

Our analysis is based on *passive* observation of *end-to-end transactions* at the clients. Thus we are in a position to observe end-to-end failures that are affected not only by the health of the IP-level client-server path but also that of intermediaries such as proxies and DNS servers. This is in contrast to prior work based on tools such as traceroute that has focused on actively probing just the IP-level path. Such active probing can incur considerable overhead when employed on a large scale and can be infeasible in environments where traceroute traffic is block by ISPs or intermediate entities such as firewalls.

Another key feature of our approach is that we combine observations from multiple end hosts to infer the nature of failures. While there has been previous work on correlating BGP observations gathered at multiple vantage points [12], [14], we believe that our work is the first one to correlate observations on end-to-end failures from multiple end hosts.

The rest of this paper is organized as follows. In the next section, we describe our analysis framework in detail. We then describe our experimental setup and methodology in Section III. This sets the stage for the presentation of our results and analyses in Section IV. In Section V, we briefly discuss the feasibility of a novel, end-host based Internet fault diagnosis system. In Section VI we survey previous work on the characterization and analysis of Internet failures, and discuss how our work relates to previous research. We present our conclusions in Section VII.

II. Fault Analysis Framework

In this section, we present a framework for client-based characterization and analysis of end-to-end Internet faults. We consider faults that can be detected via passive observation of end-to-end communication at clients and the inferences that can be made by combining observations made at multiple clients. Although our discussion here as well as our experiments focus on end-to-end communication in the form of web downloads, we believe that the framework presented here could be extended to other forms of end-to-end communication such as media streaming.

In the rest of this paper, we will use words "failure", "fault" and "problem" interchangeably. Note that "failure" does not imply a total inability to communicate, but rather just noticeably abnormal behavior (e.g., a failure rate of 15% that is much higher than the normal failure rate of say 1%).

A. Failure of Individual Transactions

We begin by discussing failures and failure modes of individual web downloads, or *transactions*, as observed at a client. We present a categorization of such locally-observable failures in the form of a *failure tree*, as shown in Figure 1.

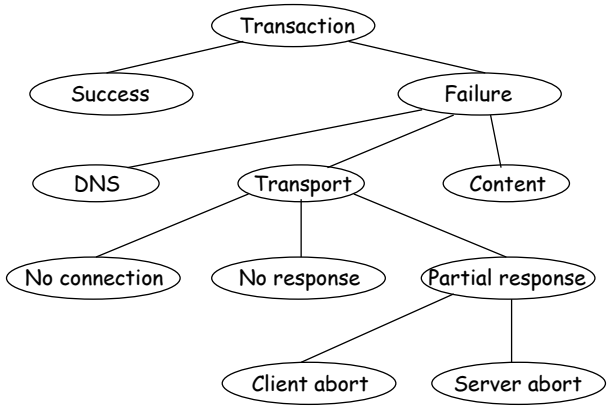

Fig. 1. A *failure tree* depicting the categorization of transport failures as observed at an individual client host.

A web transaction consists of a client resolving a web server name to the corresponding IP address, establishing a TCP connection to the server, and downloading the object of interest using the HTTP protocol. [1] A transaction fails when any of these steps fails. Furthermore, since these steps proceed in order and the client can tell which step, if any, has failed, there are three categories of failures observable at the client:

1) **DNS:** The website name cannot be resolved.
2) **Transport:** Name resolution is successful, but a failure is encountered when connecting to the server or downloading content. In the context of web downloads, this would correspond to a failure at the TCP connection level.
3) **Content:** The TCP connection is successful. However, the server does not supply the desired content and instead returns an error indication. The error could be for various reasons, including the desired content not being present at the server, the server being unable to serve the content because of load, or the client not being authorized to download the content.

Transport failures (specifically in the context of TCP) can in turn be categorized as follows:

1) **No Connection:** The client is unable to connect to the server. In the context of TCP, this corresponds to a failure of the SYN handshake. Such a failure can happen either because of a network connectivity problem or because the server is down.
2) **No response:** The client is able to establish a connection and send its request, but it does not receive any response. This can happen because of a failure of the server application[2] or because the server is overloaded. A lack of response could also be because of network connectivity issues, although the fact that the SYN handshake was successful makes it less likely that there was a total failure of connectivity. [3]
3) **Partial response:** The client receives only part of the server's response before the connection is terminated prematurely, either by the server or by the client. The premature termination could have happened either because of a failure of the server or client application, or simply because the connection was progressing so slowly (say because of route failure or network congestion) that the client timed out and closed the connection.

B. Correlating Failures Across Clients and Servers

Local observations at an individual client of its communication with a particular server may not always indicate the nature of the problem that is causing the failure, in particular,

[1] For the purposes of our discussion here, we treat the download of the multiple objects that comprise a web page as separate transactions. We do not consider HTTP 1.1 pipelining effects in this paper. In our experiments, we download only a single object (the "index" page) from each server, so there is little opportunity to reuse connections or pipeline multiple requests.

[2] For instance, a web server may issue an `accept()` call, thus establishing a connection, yet fail to fork off a new process or thread to service the request.

[3] Note, however, that a TCP SYN cookie device such as a Cisco Guard box can create such an effect.

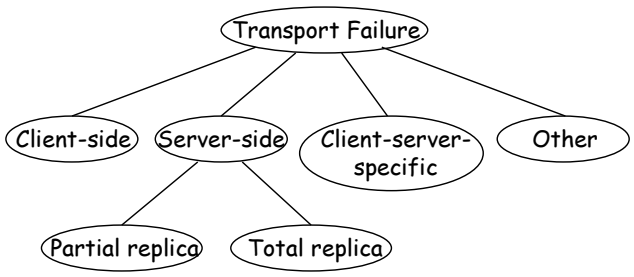

Fig. 2. A *failure tree* depicting the categorization of failures as inferred from correlating failure observations across client hosts. A similar classification can be done for DNS failures.

whether it is server-specific, client-specific, or otherwise. For example, in the case of a "no connection" failure, it is not clear whether the cause is a connectivity problem at the client end, or a server failure, or a problem in the interior of the network.

By correlating failure observations across clients and servers, we are better positioned to disambiguate the *likely* nature of failures. Rather than considering individual failures, we identify *failure episodes*, which correspond to periods with an abnormally high failure rate. Abnormal periods are identified by comparing with system-wide "normal" behavior across clients and servers. We defer a more detailed discussion of this issue to Section IV-C.

By combining failure observations across clients and servers, we are able to categorize failure episodes, as shown in Figure 2. Note that this categorization may be suggestive of the cause of failures or its location but does not indicate the root cause with certainty. Also, we do not claim that this categorization is unique, just that it is plausibly interesting and relevant from the viewpoint of end users.

1) **Client-side:** If a client is experiencing an abnormally high aggregate failure rate in its communication across many servers, we term the corresponding period as a *client-side failure episode* for the client in question. The fact that the client's communication across many servers is affected suggests a root cause at or close to the client. Note, however, that a network problem that affects a group of clients may also appear as a client-side problem from the viewpoint of an individual client.
   If the client's accesses are routed through a proxy, failures due to the proxy could also result in a client-side failure episode. We do not distinguish between proxy-related and other client-side failures in our analysis here.
2) **Server-side:** If a server is experiencing an abnormally high aggregate failure rate in its communication across many clients, we term the corresponding period as a *server-side failure episode* for the server in question. The fact that server's communication across many clients is affected suggests a root cause at or close to the server.
   For websites that have multiple replicas, a server-side failure episode could either affect all replicas (i.e., a *total* replica failure episode) or only a subset of the replicas (i.e., a *partial* replica failure episode). Note that "total" and "partial" only refer to the spatial extent of the failure episode across the replicas, not to total or partial failure of accesses to the website. So, for instance, an abnormally high failure rate of 20% that affects all replicas of a website would still be termed as a total replica failure episode.
   Since problems can arise independently at servers and clients, a server-side failure episode at a server could overlap in time with a client-side failure episode at a client.
3) **Client-server-specific** If a specific client-server pair is experiencing an abnormally high failure rate, but neither the client nor the server is experiencing an abnormally high failure rate in aggregate, then we term the corresponding period as a *client-server-specific failure episode*. The root cause of such a failure episode could be either at the client end or the server end or in the network in between.
4) **Other:** These are other periods when there may be failures but these are not significant enough in intensity for the period to be registered as abnormal for either the client, the server, or the client-server pair. Some of the failures during such periods could be due to transient problems that occur on a very short timescale.

III. Experimental Setup and Methodology

In this section, we present our experimental setup and methodology. We describe the set of clients and servers used in our experiments and also the tools and techniques used to perform measurements.

A. Overview

Our experiment was conducted during a one-month period: Jan 1-31, 2005. During this period, each client host repeatedly accessed a set of the URLs. The sequence of accesses was randomized to avoid any systematic bias. For each web page, we only downloaded the top-level "index" file, to limit network load.

For each download, we record several pieces of information at the *client*, including the DNS lookup time (or failure indication), download time (or failure indication), and a packet-level tcpdump trace of the entire transaction. All of these pieces of information can be easily obtained, with no additional network communication, in a real setting where clients monitor their own network activity.

Although the downloading activity of the clients in our experiment does constitute active measurement, our goal here is only to generate a synthetic workload that emulates, within the constraints of our experiment, clients that perform downloads in normal course. We only had a small number of clients in each "location", where "location" refers to campus, city, ISP network, etc. In contrast, in a real setting, with hundreds or thousands of participating clients at each location, clients may be able to meaningfully share information about their network experience without resorting to any "active" downloads. Previous findings in the context of web caching [27] point to the

potential for such sharing.[4] However, a more detailed analysis is needed to firmly establish the scope for such sharing, but we do not consider this issue in the present paper.

An alternative approach to avoid active downloads would be to gather packet-level traces of existing traffic, say at the border router of a busy campus (as in [23]). However, it is logistically challenging to set up such sniffers on the scale of dozens or hundreds of sites. Moreover, such sniffers may not yield the true picture of network health from the viewpoint of end-hosts. For example, there would be no record of DNS requests or TCP SYN attempts that failed due to a local fault, even before they reached a packet sniffer located say at the edge of a campus network.

B. Clients

We used 4 sets of clients in our experiments (summarized in Table I):

- **PlanetLab (PL):** We picked 95 PlanetLab nodes across 64 sites. Having multiple nodes at many of the sites enabled us to identify failures that were likely to be client-site-wide. 78 hosts were in the U.S. and Canada, and the remaining 17 were distributed across Asia and Europe. All nodes ran version 2.6.8 of Linux.
- **Dialup (DU):** We had 5 clients, all located in Seattle, dial into 26 PoPs of a large dialup ISP (anonymized for the purposes of review), spread across 9 cities in the U.S. The PoPs in each city were chosen so that the upstream ISP for each PoP was different. The clients dialed into the various PoPs in random order and then downloaded the URLs from the designated set also in random order. Thus although we only had 5 dialup clients, we effectively had 26 "virtual" clients, each of which connected to the Internet via a different path and hence provided a different perspective on the wide-area network. All nodes ran Microsoft Windows XP.
- **CorpNet (CN):** We had 5 nodes spread across 4 locations on the internal network of a large multinational corporation: 2 nodes in Seattle (SEA1 and SEA2) and 1 node each in San Francisco (SF), the U.K. (UK), and China (CHN). All external web requests from each of these 5 nodes was routed via a separate HTTP proxy. The proxy was located at the local site in all cases except CHN where it was located in Japan. In addition, we had another node in Seattle (SEAEXT) that was located outside the corporate firewall/proxy but shared the same WAN connectivity as SEA1 and SEA2. All the CN nodes ran various flavors of Microsoft Windows — 2000, XP, and 2003.
- **Broadband (BB):** We had 7 residential broadband clients (5 DSL and 2 cable modem) spread across 4 ISP networks (Roadrunner, SBC/Yahoo, Speakeasy, and Verizon) in 4 U.S. cities (Pittsburgh, San Diego, San Francisco, and Seattle). The access link speed for these hosts was 768/128 Kbps or better down/up.

[4]In fact, the potential for sharing transaction failure information might even be greater than it is for web caching, since the former does not require sharing at the content (e.g., file) level.

Our choice of a heterogeneous set of clients is motivated by the desire to obtain a broader understanding of Internet behavior than can be obtained from just the PlanetLab nodes, which are predominantly located at academic sites [9]. Although we had a total of $95 + 5 + 6 + 7 = 113$ client machines, the DU clients dialing in to 26 PoPs effectively gave us a total of 134 clients.

While the use of dialup clients might seem anachronistic, dialup is still a very significant and sometimes dominant access technology, both in the developing world and in the developed world (e.g., [6] indicates that 51% of U.S. home users were on dialup as of June 2004). Also, for the purposes of our study, the dialup clients provide visibility into *failures* observed on paths through commercial ISPs, and many of these failures are likely to be independent of the low speed of the dialup link.

Finally, since we were constrained to locate all the dialup clients in Seattle, there is the concern that the extra latency incurred in dialing into remote PoPs might skew the performance numbers. However, given our focus on failure rates rather than absolute performance numbers, this was not a significant concern.

C. Web Sites

We picked a set of 80 websites as the target for our download experiments. As indicated in Table II, we tried to ensure significant diversity among the web sites in terms of the geographic location, popularity, etc. (Popularity was determined based on the Alexa list [1].) Some of these sites were replicated or served via CDNs. We specifically included certain websites to enable correlation of failures across co-located websites (e.g., www.technion.ac.il and www.cs.technion.ac.il).

The number of sites chosen was constrained by the frequency with which each client could perform downloads, without generating excessive network traffic or triggering alarms. In our experiment, each client host accessed each website approximately 4 times an hour, which translates into $80 * 4 = 320$ downloads per hour from each client.

D. Tools

We used a set of off-the-shelf tools to do our measurements. In each measurement iteration, the URLs were sorted in random order. The procedure followed for each download was as follows:

1) Flush the local DNS cache.
2) Use `dig` [2] to measure the DNS lookup time.
3) Use `wget` [4] to download the URL ("index" file only).
4) Use `tcpdump` [3] or `windump` [5] to record a packet-level trace of the entire transaction.

There are also special steps for the DU and CN clients.

For the DU nodes, we pick a PoP to dial at random, connect to it, download all the URLs in random order, disconnect, and then pick another PoP to connect to. For each dialup session, we record the modem connection speed, as reported by the Windows OS.

Category	PlanetLab (PL)	Dialup (DU)	CorpNet (CN)	Broadband (BB)
# Clients	95	5 (26 PoPs)	5(+1)	7
Details	US-EDU (50), US/Canada ORG (19), US-COM (4), US-NET (5), Europe (13), Asia (4)	Boston(ILQ), Chicago(ILQ), Houston(ILQ), New York(IQU), Pittsburgh(ILQ), San Diego(ILQ), San Francisco(ILQ), Seattle(ILQ), Washington DC(IL)	San Francsico (1), Seattle (2+1), UK (1), China (1)	Pittsburgh (1), San Diego (2), Seattle (3), San Francisco (1)

TABLE I

DETAILS OF THE CLIENTS USED IN OUR EXPERIMENT. NOTE THAT THE DU CLIENTS WERE ALL LOCATED IN SEATTLE BUT DIALED INTO REMOTE POPS ACROSS 9 CITIES. EACH POP IN A CITY TYPICALLY HAD A DIFFERENT UPSTREAM ISP DRAWN FROM ICG(I), LEVEL3(L), QWEST(Q), AND UUNET(U). ALSO, ONE OF THE CN NODES IN SEATTLE IS OUTSIDE THE CORPORATE FIREWALL/PROXY BUT SHARES THE SAME WAN CONNECTIVITY AS THE OTHER TWO CN NODES IN SEATTLE.

US-EDU
1) berkeley.edu
2) washington.edu
3) cmu.edu
4) umn.edu
5) caltech.edu
6) nmt.edu
7) ufl.edu
8) mit.edu

US-POPULAR
9) amazon.com
10) microsoft.com
11) ebay.com
12) mapquest.com
13) cnn.com
14) cnnsi.com
15) webmd.com
16) espn.go.com
17) sportsline.com
18) expedia.com
19) orbitz.com
20) imdb.com
21) google.com
22) yahoo.com
23) games.yahoo.com
24) weather.yahoo.com
25) msn.com
26) passport.net
27) aol.com
28) nytimes.com
29) lycos.com
30) cnet.com

US-MISC
31) latimes.com
32) nfl.com
33) pbs.org
34) cisco.com
35) juniper.net
36) ibm.com
37) fastclick.com
38) advertising.com
39) slashdot.org
40) un.org
41) craigslist.org
42) state.gov
43) nih.gov
44) nasa.gov
45) mp3.com

INTL-EDU
46) iitb.ac.in
47) iitm.ac.in
48) technion.ac.il
49) cs.technion.ac.il
50) ucl.ac.uk
51) cs.ucl.ac.uk
52) cam.ac.uk
53) inria.fr
54) hku.hk
55) nus.edu.sg

INTL-POPULAR
56) amazon.co.uk
57) amazon.co.jp
58) bbc.co.uk
59) muenchen.de
60) terra.com
61) alibaba.com
62) wanadoo.fr
63) sohu.com
64) sina.com.hk
65) cosmos.com.mx
66) msn.com.tw
67) msn.co.in
68) google.co.uk
69) google.co.jp
70) sina.com.cn

INTL-MISC
71) lufthansa.com
72) english.pravda.ru
73) rediff.com
74) samachar.com
75) chinabroadcast.cn
76) nttdocomo.co.jp
77) sony.co.jp
78) brazzil.com
79) royal.gov.uk
80) direct.gov.uk

TABLE II

THE LIST OF 80 WEB SITES THAT WERE TARGETS OF OUR DOWNLOAD EXPERIMENT. FOR THE SAKE OF BREVITY, WE HAVE LEFT OUT THE "WWW" PREFIX FOR MOST OF THESE HOSTNAMES. WE ONLY DOWNLOADED THE TOP-LEVEL "INDEX" FILE AT EACH SITE. THE "POPULAR" CATEGORY REFERS TO SITES DRAWN FROM THE ALEXA TOP 500 LIST [1].

For the CN clients, we configure `wget` to issue requests with the "no-cache" cache-control header [15] set, which ensures that the response is received from the origin server. We do so to avoid having the proxy cache mask failures beyond the proxy. However, since the proxy rather than the CN client does name resolution, and there is no way for the client to force the local DNS cache at the proxy to be flushed, some DNS failures may be masked from the client.

We did not gather packet-level traces on the BB machines. Since these were peoples' home computers, there were concerns with both privacy issues and storage requirements. Also, while we did gather packet-level traces for the CN machines, these were not very interesting since they only revealed the dynamics of TCP connections to the local proxy.

E. Analysis

From the raw data recorded for each download, we obtain an indication of the success/failure of both the DNS lookup and the download, the DNS lookup time, the download time, and the failure code, if any, reported by `wget`. We store this information in a performance record, together with the client name, URL, server IP address, and time.

We also post-processed the `tcpdump/windump` packet traces using a simple tool called `TcpScope` that we have developed to estimate various TCP metrics. This tool is inspired by T-RAT [29], but is simpler and potentially more accurate, since we are capturing packets at the client (which is typically at the receiving end of the connection) rather than in the middle of the network. The specific TCP-level information we consider in this paper are:

1) **Cause of connection failure:** the connection-level failures were categorized as *no connection*, *no response*, or *partial response*, as discussed in Section II-A.
2) **Packet retransmission count:** the count of the total number of packets sent by the server and the number of retransmitted packets. We distinguish between packet retransmission and reordering by comparing the length of time by which a packet is out of order with the connection's RTT (which is also estimated based on the client-side packet trace). Since most web connections are short, we do not compute packet loss rate on a per connection basis. Rather we aggregate packet counts across multiple connections to compute the loss rate, as discussed in Section IV.

IV. EXPERIMENTAL RESULTS

In this section, we present our experimental findings. We present both "raw" statistics on failures and also inferences obtained by correlating failure information across clients and web sites. In our analysis here, we only consider *hard* failures, where a client was unable to access a web page regardless of the reason. The main advantage of this policy is that there is no ambiguity in what constitutes a hard failure. In future work, we plan to also consider *soft* failures, where the page download is successful but is much slower than normal.

We start in Section IV-A by presenting the statistics of transaction-level failures for the 4 categories of clients (PL, DU, CN, BB). We report the breakdown of these failures

by type: DNS, transport (i.e., connection-level), and content. In Section IV-B we present a more detailed analysis of connection-level failures from the viewpoint of the individual clients. We then turn to *correlating* failure observations across clients and servers with a view to classifying failure episodes as client-side, server-side, client-server-specific, or other. We present the correlation analysis separately for connection-level failures (Sections IV-C and IV-D) and DNS failures (Section IV-E). Finally, we present an analysis of client-server-specific failures in Section IV-F.

A. Transaction Failure Analysis

In this section, we present overall failure statistics for web transactions over the month-long data set. A *transaction* is an invocation of `wget` to download a URL.

1) Overall Transaction Failure Rate: We first compute the overall failure rate for each client over all its transactions with all servers. Figure 3 plots the CDF of the per-client failure rate thus computed. The median failure rate is 1.5% while the 95%-tile is 10%, which indicates that certain clients experience a significant rate of transaction failures over the one-month period. We defer discussion of the packet loss rate curve in Figure 6 until later in this section.

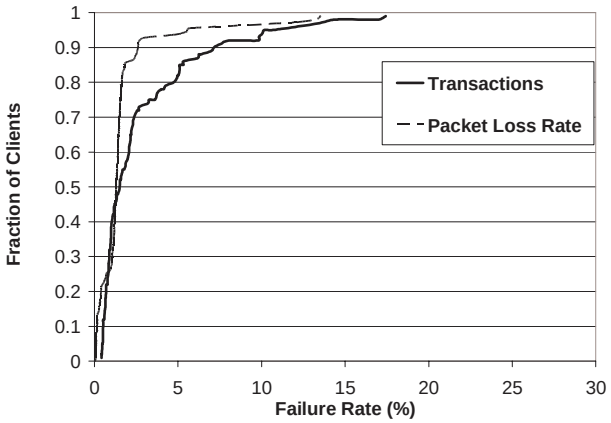

Fig. 3. CDF of transaction failure rate and packet loss rate for successful connections, across clients.

Figure 4 plots the mean transaction failure rate for each category of clients. It is interesting to note that the mean failure rate is lowest (0.69%) for the DU clients and significantly higher (2.76%) for the PL clients, despite the latter being connected to much higher-speed academic and research networks. This difference may be because the DU clients connect via a commercial dialup service (which presumably strives to provide a good quality of service) whereas the PL clients are part of the experimental PlanetLab network. Regardless, the main message here is that speed and reliability are not necessarily correlated.

2) Breakdown of Transaction Failures: Figure 5 plots the breakdown of transaction failures by type, for each category of clients (we exclude the CN clients, since these connect via a proxy that masks the true nature of the failure). The failure types are the ones depicted in the failure tree shown in Figure 1: DNS, transport (i.e., connection-level), and content. We find that for all categories of clients, connection-level failures dominate, accounting for 59-72% of all transaction failures. DNS failures account for most of the rest (26-40%). Content failures (i.e., HTTP-level failures) account for under 2% of the transaction failures in all cases.

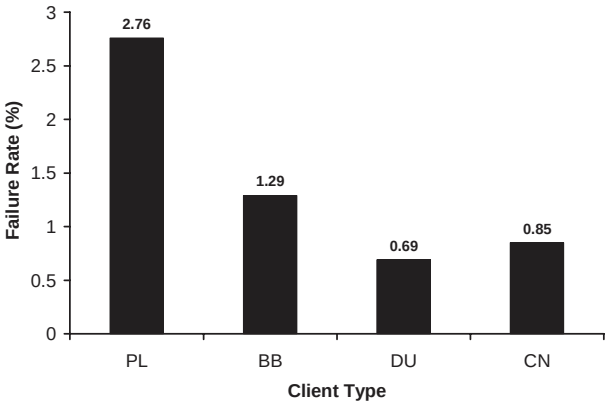

Fig. 4. Percentage of failed transactions for each category of clients.

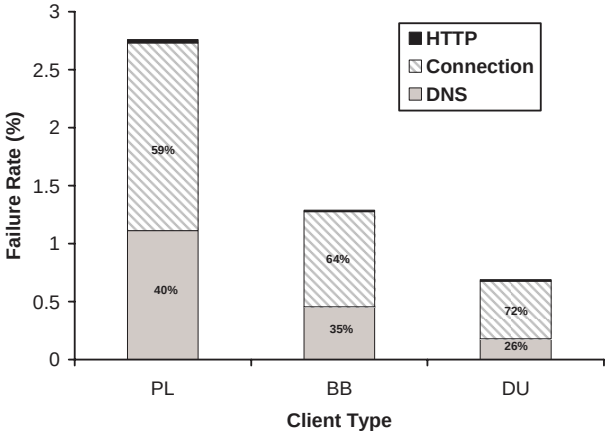

Fig. 5. Breakdown of transaction failures by type for each category of clients.

3) Packet Loss and Transaction Failures: Several previous measurement studies of Internet reliability have focused on the packet loss rate metric (e.g., [21], [30], [7]). To the extent that they use TCP to measure packet loss rate (e.g., [21], [7]), these studies have considered relatively long data transfers (e.g., at least 100 KB in [21]) to quantify packet loss rate. The question is how well packet loss rate of such "successful" connections correlates with the failure rate of end-to-end transactions.

Figure 3 plots the CDFs for both the transaction failure rate and the packet loss rate of successful connections. We also found the coefficient of correlation between transaction failure rate and packet loss rate across clients to be low: 0.19 (not shown in the figure). The main reasons for this lack of correlation are: (a) transactions can fail for reasons that have little to do with the end-to-end server-client path (e.g., DNS failures, as shown in Figure 5), (b) a transaction can succeed despite (possibly severe) packet loss, and (c)

estimating packet loss rate using TCP traffic is prone to bias, since failed connections that transfer no data (which are in fact quite significant, as discussed in Section IV-B.2) are ignored. Non-TCP-based techniques such as the `zing` tool [30] avoid this bias to an extent (though not entirely, since DNS failures are not accounted for), but using such tools requires control over both end-points and involves active measurements.

Thus we believe that it is important to study the failures of end-to-end transactions rather than only packet loss rate. In the following sections, we analyze the two dominant causes of transaction failures — connection failures and DNS failures — separately. The reason for analyzing these separately is that DNS resolution and TCP/HTTP connections typically involve distinct Internet components and possibly distinct network paths.

B. Connection Failure Analysis

In this section, we present a more detailed analysis of connection-level failures, which comprise a significant chunk of transaction failures (Figure 5).

1) Connection Failures vs. Transaction Failures: We first consider the relationship between transactions and TCP connections. As noted earlier, a *transaction* is an invocation of `wget` to download a URL. Like other web clients, `wget` could invoke multiple TCP connections in the context of a single transaction, in an attempt to mask failures. When DNS returns multiple IP addresses for a server and the TCP connection to one of them fails, `wget` tries to connect to the alternate addresses. Multiple connections also arise when the server returns an HTTP redirect response to the client.

Figure 6 compares the distributions of the transaction failure rate and the connection failure rate. The latter is computed over the individual TCP connections. A transaction could be successful despite the failure of one or more of its constituent TCP connections. On the other hand, a transaction could fail due to DNS or HTTP failures, which would not register as a TCP-level connection failure. Thus the transaction failure rate could be larger or smaller than the connection failure rate depending on which of these effects dominates. As Figure 6 shows, the transaction failure rate is generally larger than the connection failure rate, except in the tail of the distribution (90%-tile and above). However, when DNS and HTTP related failures are excluded, the residual transaction failure rate is no greater than the connection failure rate.

2) Breakdown of Connection Failures: We now analyze the TCP connection failures, which correspond to "transport" failures in the failure tree shown in Figure 1. The types of failure are "no connection" (TCP SYN exchange failed), "no response" (server did not return any bytes of response), and "partial response" (the server returned a partial response, but the connection was terminated prematurely). The breakdown is shown in Figure 7. We see that "no connection" failures dominate in the case of PL (79%) and DU (65%), and are significant in the case of BB (41%). The prevalence of "no connection" failures reinforces the point made in Section IV-A.3 of the unsuitability of TCP packet loss rate as an indicator of transaction failures. It is hard to incorporate information from a failed SYN exchange into an overall packet loss rate metric.

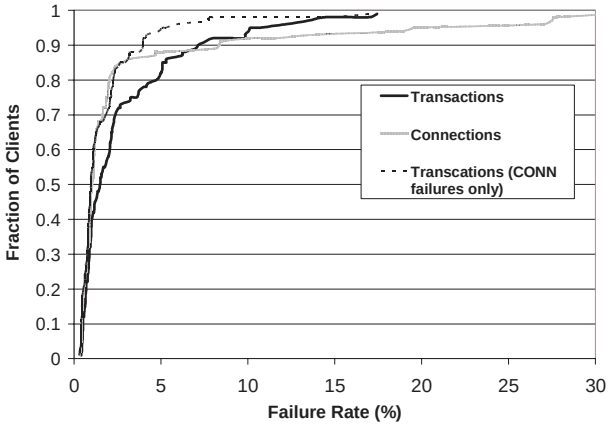

Fig. 6. CDF of the transaction failure rate (both the overall failure rate and the failure rate excluding transactions that failed due to DNS or HTTP causes) and the connection-level failure rate.

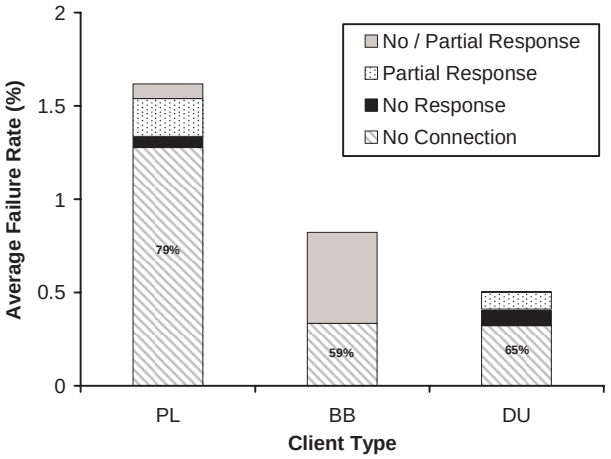

Fig. 7. Breakdown of TCP connection failures for each category of clients. The CN clients are excluded since they connect via a proxy, which masks the failures of its TCP connections to servers. The category marked "no/partial response" corresponds to cases where we lacked the tcpdump traces needed to disambiguate the precise cause of failure. Nevertheless, from the wget logs we are able to rule out "no connection" failures, leaving only "no response" and "partial response" as the possibilities.

Such extreme failures where no communication is possible between the client and the server suggest that either the server is down or that there is lack of connectivity between the client and server. In the latter case, it is unclear whether the root cause is at the client end or at the server end, or whether it is specific to the client-server pair in question. To shed more light on these and related questions, we now turn to correlating failure observations across clients and servers.

C. Correlation Analysis of Connection Failures

As discussed in Section II-B, we can obtain greater insight into the nature of failures by correlating failure observations across clients and servers. Specifically, we can determine whether failures are due to a client-side failure episode or

a server-side failure episode or otherwise. (Recall from Section II-B that we classify failure *episodes* rather than individual failures.) Our goal in this section is to apply such correlation analysis to connection-level failures.

1) Identifying Failure Episodes: The first step step in our analysis is identifying failure episodes, which as noted in Section II-B are periods of abnormally high failure rate at a client or a server. We identify abnormal periods by comparing with system-wide normal behavior. Abnormal periods for clients are identified by comparing with all clients and abnormal periods for servers are identified by comparing with all servers. The underlying assumption is that the system as a whole is mostly in the normal state (low failure rate or no failures at all), with abnormal behavior (high failure rate) being the exception.

We need to define the period over which failure rates are computed. There are two conflicting considerations here. We would like the period to be short enough to help identify failures that last say just a few minutes. For example, a 10-minute server outage might stand out on a 1-hour timescale but might be buried in the noise on a 1-day timescale. On the other hand, the period should be long enough for us to have a sufficient number of samples to be able to compute a meaningful failure rate (given the number of clients and servers, and the frequency of accesses in our experiment or in any practical system based on passively monitoring existing traffic). To balance both these considerations, we pick 1 hour as the episode duration. We are thus assured a few hundred accesses per client and per server in each episode. Also, the 1-hour duration allows us to identify relatively short-lived failures. The 1-hour episode duration also places minimal requirements on the degree of synchronization needed across the observations made at different clients.

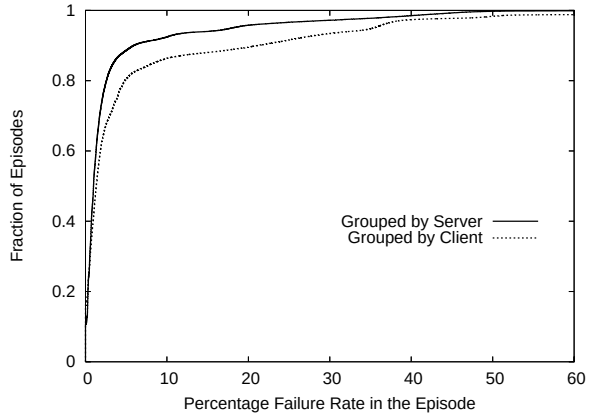

Fig. 8. CDF of the overall failure rate of 1-hour episodes across clients and servers.

To identify system-wide normal behavior, we consider the distribution of failure rate over the 31*24 = 744 episodes in our month-long trace. We do this separately for clients and for servers, as shown in Figure 8. For each 1-hour episode and each client, we compute the failure rate for all of the client's connections across all servers. This failure rate data yields the "client" CDF shown in Figure 8. Likewise for servers.

The CDFs in Figure 8 show a distinct knee separating portions of the CDF with very different slopes. To the left of the knee (i.e., the steep portion of the CDF) are the vast majority of episodes, all concentrated in a narrow range of low failure rates. We term this the "normal" range. To right of the knee (i.e., the flat portion of the CDF) are the episodes that experience a wide range of significantly higher failure rates. We term this the "abnormal" range. The episode failure rate, f, corresponding to the knee is used to separate the abnormal episodes (also termed as failure episodes) from the normal episodes.

In the analysis that follows, we experiment with two settings of the threshold f — 5% and 10% — the latter being more conservative.

Before proceeding with our analysis, we make one more point regarding our choice of the failure threshold, f. A threshold of 10% might appear to be too low, but in fact a failure rate of say 10% over a 1-hour episode is still very significant for a client or a server. Moreover, the failure might have actually been total (i.e., 100%) while it lasted, but the failure rate is lower when averaged over a full hour.

2) Classifying Failure Episodes: Using the threshold f, we can classify all 1-hour episodes for each client and for each server as either a failure (i.e., abnormal) episode or otherwise. Consider a particular pair of client C and server S. If connections between C and S experienced failures during that episode, the question is whether we can attribute them to a client-side problem or a server-side problem. Not that we are *not* trying to establish the cause of individual connection failures, but rather are only interested in establishing the *likely* cause of failures between C and S during that 1-hour episode.

If the episode in question was marked as a failure episode for C (based on the failure rate of C's connections to *all* servers) but *not* as a failure episode for S (based on the failure rate of S's connections to *all* clients), then we attribute the C–S connection failures in this episode to a client-side problem at C. (Recall from Section II-B that "client-side" does not necessarily mean that the root cause of the problem is close to the client, just that the problem is affecting a broad range of communication at the client.) Likewise, if it is a failure episode for S but not for C, we attribute the failures to a server-side problem at S.

However, if the episode is marked as a failure episode for both C and S, the question is which end we attribute the C–S failures to. This case is important to consider for the following reason. Assume that the server S and a handful of other servers happen to be experiencing problems that cause the accesses from a large number of clients to these servers to fail during the corresponding episode. Given the modest number of servers (80) in our experiment, it is possible that the failure of these handful of servers would cause a noticeable fraction of accesses from a large number of clients to fail during that episode. This failure rate could exceed our $f = 5\%$ threshold (or even the more conservative $f = 10\%$ threshold) at these clients. As a result these episodes would (incorrectly) be marked as a failure episode at the *clients*. Of course, it

would also be (correctly) marked as a failure episode at S and the other affected *servers*.

To correctly attribute the C–S failures during this episode to the server-side problem at S, we compare the overall failure rates for C and S for that episode, to see if one dominates the other. In the above example, the failure rate may be close to 100% for S but barely above the threshold f for C. So we can conclude that the C–S failures were *likely* due to a server-side problem.

In general, if both the C and S failure rates during an episode exceed the threshold f, but one of the failure rates is at least twice the other, we attribute the C–S failures during that episode to the dominating end (i.e., "*likely* client-side" or "*likely* server-side"). If neither dominates, we attribute the failures to *both* ends ("both" category).

Finally, if there are C–S failures during an episode that did not qualify as a failure episode at C or at S, we attribute the C–S failures to the "other" category. These correspond to failures that were not severe or widespread enough to register as a failure episode at either the client or the server end. Some of these are due to client-server-specific problems, i.e., failures that affect only a specific client-server pair (or more generally, a specific pair of client *site* and server), perhaps because of wide-area routing problems. Since we only have about 4 accesses per hour for any client-server pair in our experiment, we are not in a position to analyze such client-server-specific failures at the granularity of 1-hour episodes. We present a more coarse-grained analysis of such failures in Section IV-F.

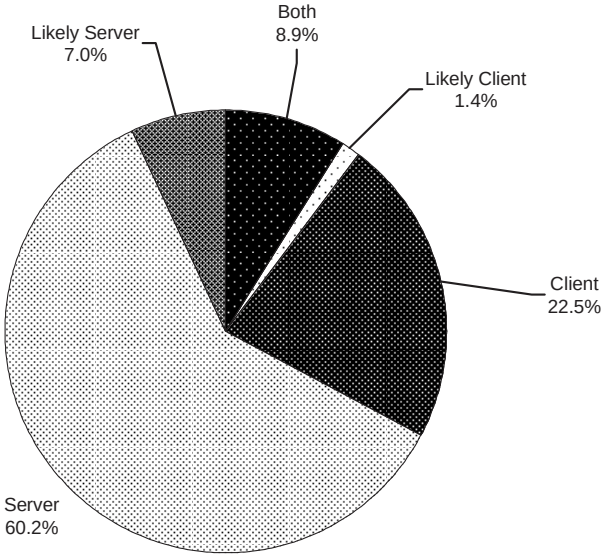

Fig. 9. Classification of failure triplets into the various categories. We use a threshold of $f = 5\%$ to identify client and server failure episodes. Only 61.7% of the failure triplets were classified as some flavor of client- or server-side failure episode.

We report our results by first counting the total number of <client,server,episode> combinations ("failure triplet" for short) such that the corresponding client-server pair experienced failure during the episode in question.[5] We then count the number of such failure triplets that could be classified as one of client-side, server-side, likely client-side, likely server-side, or both. The remainder go into the "other" category.

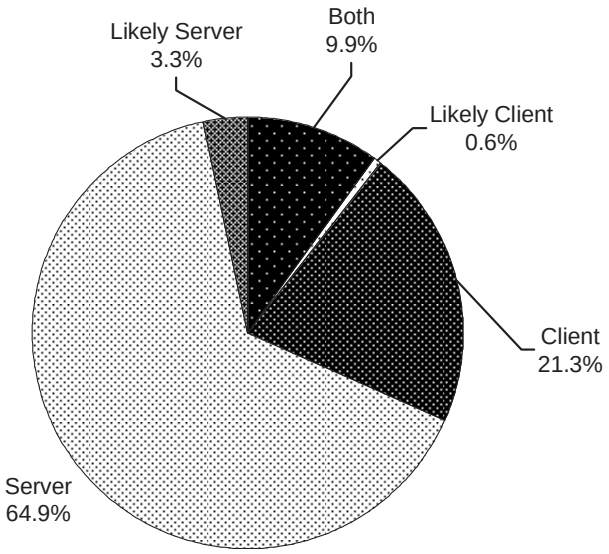

Fig. 10. Classification of failure triplets into the various categories. We use a threshold of $f = 10\%$ to identify client and server failure episodes. Only 45.2% of the failure triplets were classified as some flavor of client- or server-side failure episode.

Using the two settings of the threshold f — 5% and 10% — we were able to classify 61.7% and 45.2%, respectively, of the failure triplets. It is as expected that more failure triplets fall in the "other" category when we use the more conservative threshold of $f = 10\%$ to flag failure episodes.

Figures 9 and 10 show the breakdown of the classifiable failure triplets. We observe that there is a very good match in the breakdown across both figures, despite the different settings of the threshold, f. Also, the "client-side" and the "server-side" categories dominate the less certain ones.

Finally, the "server-side" category dominates the "client-side" one. In other words, at the level of connections, failures are much more likely due to server-side problems than client-side problems. There are a couple of reasons for this. First, a server *machine* going offline would cause a large number of clients to experience failures to that server. The corresponding event at the client end — the client machine being turned off — would not be noticed since the client would not be making any accesses during the corresponding period. This "bias" is just as well since in practice users would only care about failures that happen when they try to access servers and not about those that may happen when their machine is turned off.

A second issue is that since we are considering *connection*-level failures, that by definition excludes problems with DNS resolution, which would precede connection initiation. Client-side failures close to the last-mile might in fact cause a DNS

[5]Note that a failure triplet does *not* represent a client-server-specific failure. Rather, it represents a *general* client-side or server-side failure to which we ascribe the failures experienced between the corresponding client-server pair during the episode in question.

resolution failure, and hence not be registered as a connection-level failure. So it is important to note that our finding that server-side causes dominate connection-level failures might not reflect the situation for *transaction*-level failures. The correlation analysis of DNS failures that we present in Section IV-E sheds light on this other major cause of transaction failures.

3) Indirect Validation: It is difficult to directly validate our inferences of server-side and client-side failures, since we have little visibility into the network. Instead, we provide indirect evidence to support our interferences. We do this in two ways.

First, we consider how widespread the impact of server-side and client-side failures episodes is, i.e., what fraction of clients or servers is affected in such an episode. We would expect a server-side failure to impact a large fraction of the clients, and likewise expect a client-side failure to affect transactions to a large fraction of the servers. The results we present below (see #1) confirm this.

Second, we consider co-located clients (e.g., those on the same university campus) and determine the degree to which their client-side failure episodes are correlated. We would expect a significant degree of correlation, since many failures (though not all) might affect connectivity at the level of the subnet or even the entire campus. The results we present below (see #2) confirm this. We also perform a similar analysis for co-located servers, although we have far fewer instances of co-located servers in our data set.

#1: Spread of Server-side and Client-side Failures

We consider how widespread the impact of server-side and client-side failure episodes is. Ideally, we would like to answer this question by looking at how widespread the impact is within each failure episode. However, this is difficult to do because of sampling limitations, not just in our experiment but also in any practical system that is based on passive observation of existing traffic.

There are two sampling problems. A server-side problem could cause 100% failure for all client accesses during a short interval, say 10 minutes long. However, there would be no record of failure for clients that happened not to access the server in question during this short period. On the other hand, a server-side problem could last the entire hour but affect say only 20% of the transactions at random. While the underlying problem might be one that does not discriminate between clients accessing the server, there is a chance that some clients get lucky in the sense that none of their accesses to the server fail during the hour. So, in general, we are not in a position to definitively establish which clients *could* have been affected by the server-side problem.

In view of this difficulty, we only look at how widespread the impact of server-side and client-side failure episodes is over the entire month-long period.[6] That is, for each server, S, we consider all the failure triplets that it is part of and that were attributed to a server-side problem at S. (Recall from Section IV-C.2 that a failure triplet is the combination <client,server,episode>.) We are interested in how large the set of clients affected by these server-side episodes over the month-long experiment is. We quantify this "spread" by counting the fraction of all clients needed to account for 75% or 95% of the failure triplets involving the server, S. A similar "spread" metric can be computed for clients by considering how widespread the impact of client-side problem is across servers.

Server	Spread (75%-tile)	Spread (95%-tile)
sina.com.cn	38	59
craigslist.org	14	47
sohu.com	20	48
iitb.ac.in	38	66
technion.ac.il	37	68
cs.technion.ac.il	40	70
brazzil.com	44	67
ucl.ac.uk	51	80
muenchen.de	43	68
iitm.ac.in	44	69

TABLE III

THE LIST OF MOST FAILURE-PRONE SERVERS AND THE "SPREAD" QUANTIFYING HOW WIDESPREAD THE IMPACT OF THE CORRESPONDING SERVER-SIDE FAILURES IS.

Table III lists the spread for the most failure-prone servers. We find that the 75%-tile spread is typically over 35% and the 95%-tile spread is typically over 60%. This indicates that the failures that we flag as server-side typically impact a large fraction of the clients, as we would expect. This serves to indirectly validate the inferences made in Section IV-C.2.

We make one other observation regarding Table III. The most failure-prone servers are all located outside the U.S., with the exception of `craigslist.org`. Given that our client set is dominated by U.S.-based clients, it is hard to distinguish a network connectivity problem between the U.S. and the rest of the world from an actual server-side failure at a non-U.S.-based server that affects a large fraction of the clients.[7] In general, we do not have enough (or any) clients located close to many of the non-U.S.-based servers to be able to tell if such "local" clients were also affected by the apparent server-side failure. However, in some cases we were able to verify that the (small number of) clients located relatively close to the server were also affected at the same time that the U.S.-based clients were (e.g., clients in Korea experienced problems accessing `sina.com.cn` and clients in the U.K. experienced problems accessing `ucl.ac.uk`).

#2: Correlation Between Co-Located Clients and Servers

We first consider the extent to which client-side failure episodes are correlated across co-located clients. For each pair of co-located clients, we first determined the subset of episodes that were (separately) marked as a client-side episode for each client in the pair. We compute the "similarity" measure for the pair of clients as the ratio of the size of the intersection set (i.e., the client-side failure episodes in common) to the size of the union (i.e., episodes that are marked as a client-side

[6]Of course, this is not a perfect measure either since multiple distinct server-side problems during different episodes through the month could have affected different subsets of clients. The overall "spread" across clients might be large, but the spread during individual failure episodes could still be small.

[7]One could argue that this distinction does not matter from the viewpoint of the U.S.-based clients.

Number of co-located client pairs	35
Pairs with at least one client-side failure episode marked for at least one of the clients	25
Pairs with similarity $> 90\%$	6
Pairs with similarity in 50-90%	5
Pairs with similarity in 25-50%	1
Pairs with similarity $< 25\%$	13

TABLE IV

THE MEASURE OF THE SIMILARITY IN THE CLIENT-SIDE FAILURE EPISODES EXPERIENCED BY PAIRS OF CO-LOCATED CLIENTS.

Client pair	# client-side failure episodes in the union	Similarity
planet1.pittsburgh.intel-research.net planet2.pittsburgh.intel-research.net	377	94%
csplanetlab1.kaist.ac.kr csplanetlab3.kaist.ac.kr	2	100%
csplanetlab3.kaist.ac.kr csplanetlab4.kaist.ac.kr	2	100%
csplanetlab4.kaist.ac.kr csplanetlab1.kaist.ac.kr	2	100%
planetlab1.comet.columbia.edu planetlab2.comet.columbia.edu	31	0%
planetlab2.comet.columbia.edu planetlab3.comet.columbia.edu	37	81%
planetlab3.comet.columbia.edu planetlab1.comet.columbia.edu	31	0%

TABLE V

EXAMPLES OF CO-LOCATED CLIENTS AND THE DEGREE OF SIMILARITY IN THE CLIENT-SIDE FAILURE EPISODES THAT THEY EXPERIENCE.

failure episode for either or both clients).

We identify 35 pairs of co-located clients in our data set. These were mostly PL clients that were located on the same university campus network. But we also had two pairs of co-located BB nodes: a pair of nodes on the Roadrunner cable network in San Diego and a pair of nodes on the Verizon DSL network in Seattle.

Table IV shows the similarity measures across the client 35 pairs of co-located clients. Of these, 10 pairs did not experience any client-side failure episodes (and hence were in a sense 100% similar, although this is not very interesting). Of the remaining 25 pairs, about a quarter had very high similarity (over 90%) and another fifth had moderately high similarity (50-90%). However, a little more than half of the pairs had a low degree of similarity (under 25%). On closer examination, the overwhelming majority of these pairs experienced a very small number of client-side failure episodes through the month-long experiment (just 1-2 episodes, in some cases). Any mismatch (i.e., lack of sharing) in these rare failure episodes would result in a low similarity measure.

In general, a low degree of similarity in the client-side failure episodes for co-located clients could also arise because the failure was truly client-specific. One of the examples we consider next illustrates this point.

Table V lists a few examples of co-located clients that we studied: 3 nodes at Columbia University, 2 at Intel Research Pittsburgh, and 3 at KAIST. These three sets exhibit very different behavior.

The two nodes at Intel see a very large number of client-side failure episodes between them — 377 failure episodes out of a maximum possible count of $31 * 24 = 744$ over the month-long period. Moreover, there is a very high degree of similarity (94%) across the failure episodes experienced by these two co-located clients. On the other hand, the 3 nodes at KAIST experience a total of only 2 client-side failure episodes. However, both these episodes are shared by all 3 nodes.

The case of the Columbia clients is remarkably different. Two of the nodes — #2 and #3 — experience 31 and 36 failure episodes, respectively. The size of the union set is 37 and that of the intersection set is 30, yielding a high degree of similarity (81%). However, the behavior of the third Columbia node (#1) is very different. It only experiences 1 client-side failure episode and even this is not shared with either of the other two nodes. We do not have a satisfactory explanation for this significantly different behavior despite the 3 nodes being located on the same subnet at Columbia. We plan to approach the network administrators at Columbia to see if they could shed light on this issue.

In summary, we find that a little less than half the pairs of co-located clients shared 50% or more of their client-side failure episodes. Most of the rest were pairs that saw very few client-side failure episodes, making any similarity computation noisy.

We also repeated the similarity analysis for server-side failure episodes across co-located servers. Of the 4 such pairs in our data set, only 2 pairs experienced any server-side failure episodes. One of these pairs, `technion.ac.il` and `cs.technion.ac.il`, shares 67 out of a total of 78 server-side failure episodes, yielding a high similarity of 86%. The other pair, `ucl.ac.uk` and `cs.ucl.ac.uk`, has a similarity of only 6% — the former (`ucl`) experiences 49 server-side failure episodes, whereas the latter (`cs.ucl`) experiences only 3, all of which are shared with the former (`ucl`). We believe that more such pairs of co-located servers need to be studied before we can draw general conclusions.

D. Replicated Websites

As noted in Section II-B, websites could be *replicated*, in which case server-side problems could represent either *total* or *partial* replica failures. Total failures affect all replicas of a website, while partial failures affect only a subset of the replicas.

We repeat the correlation analysis at the granularity of server replicas with the view of further classifying the failure episodes in which the website experienced abnormal failure rate. Our goal is to classify the episodes that were earlier marked as *server-side*, as either total or partial, as shown in the failure tree in Figure 1.

We identify the set of replicas for a server S by considering all distinct IP addresses to which connections were attempted by any client while downloading content from S. To make our analysis meaningful, only IP addresses that account for at least 10% of the total number of connections to S are considered to be replicas. As a result, out of the 80 websites used in our experiments, 6 had zero replicas, 42 had exactly one replica and 32 had multiple replicas. The 6 websites with zero replicas

are basically those served by CDNs like Akamai, where the number of distinct IP addresses is very large, so that none of the IP addresses qualify to be counted as a replica per our definition above.

We found that 62% of the failure episodes marked as *server-side* belonged to the 32 servers that had multiple replicas. Of these episodes, an overwhelming majority of 85% were *total* replica failures, which means that *all* replicas of the website were experiencing more than $f = 10\%$ failures during that episode. This is a somewhat surprising finding. However, more detailed analysis shows that almost all of the total replica failures are due to websites whose replicas are on the same subnet, and hence are prone to correlated failures.

E. Correlation Analysis of DNS Failures

In a manner similar to Section IV-C, we identify and classify failure episodes due to DNS problems as client-side, server-side, or otherwise. (In this context, the "server" referred to by "server-side" is the website whose name clients are unable to resolve. The root cause, however, lies in the DNS system, not the website's server(s) per se.) As before, we use a threshold of $f = 10\%$ over a 1-hour time scale to identify failure episodes. The results are presented in Figure 11. Over 87% of the failure triplets were classified as some flavor of client- or server-side failure episode. In contrast to connection failures, the vast majority of classifiable DNS failure triplets are due to client-side problems. This is reasonable since, for instance, if the local DNS server were down, a client would experience close to 100% failure, no matter what the "server" (i.e., the name it is trying to resolve) is. In addition, loss of connectivity at a client site would often manifest itself as DNS lookup failures rather than connection problems. On the other hand, problems at the server end, such as an authoritative name server for a website going down, can be masked to some extent by redundancy and caching in the DNS hierarchy.

We also performed indirect validation of our results, in a manner similar to Section IV-C. We found that for there was 100% similarity in the failure episodes marked as *server-side* for co-located servers. Results for co-located clients are similar to those shown in Section IV-C. We omit further discussion due to lack of space.

F. Client-Server-Specific Failures

As discussed in Section IV-C.2, the low rate of accesses between any client-server pair makes it difficult for us to analyze failures that are specific to a client-server pair, at the granularity of one hour. Here we present an aggerated view of such failures over the entire one-month period, and point out some interesting cases.

We compute the connection-level failure rate for each client-server pair over the entire month. The median connection failure rate across all client-server pairs was 0.3%, and the 95%-tile was 5%.

Despite these seemingly low overall failure rates, we noticed that 36 client-server pairs (out of a total of 134 * 80 = 10720) were unable to communicate at all during the entire month.

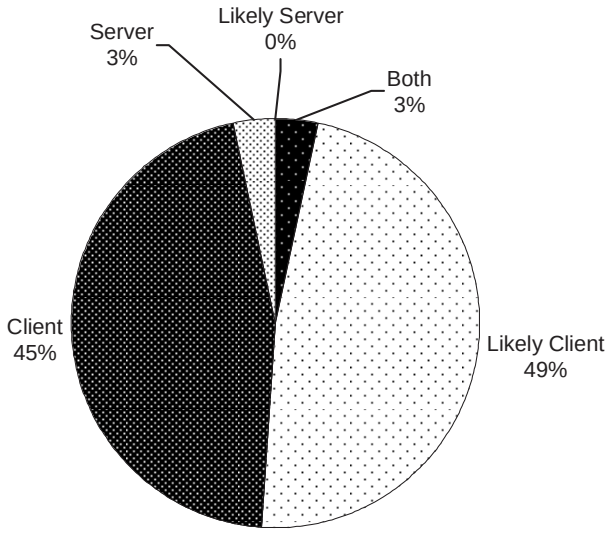

Fig. 11. Classification of failure triplets for DNS failures into the various categories. We use a threshold of $f = 10\%$ to identify client and server failure episodes. Over 87% of the failure triplets were classified as some flavor of client- or server-side failure episode.

Client	Server
planetlab2.isi.jhu.edu	criagslist.org
planetlab2.cnds.jhu.edu	craigslist.org
planetlab1.uc.edu	msn.com.tw
planetlab2.uc.edu	msn.com.tw
planetlab-1.cmcl.cs.cmu.edu	msn.com.tw
planetlab-2.cmcl.cs.cmu.edu	msn.com.tw
planet1.scs.cs.nyu.edu	sohu.com, sina.com.hk
planet2.scs.cs.nyu.edu	sohu.com, sina.com.hk, craigslist.org
grouse.hpl.hp.com	sohu.com sina.com.hk
pli1-pa-4.hpl.hp.com	sohu.com, sina.com.hk
planetlab1.di.unito.it	cs.technion.il
planetlab2.di.unito.it	cs.technion.il, sohu.com, sina.com.hk
planetlab2.cs.duke.edu	criagslist.org, nus.edu.sg
planetlab1.cs.northwestern.edu	mp3.com
planetlab3.xeno.cl.cam.ac.uk	cam.ac.uk

TABLE VI

SOME CLIENT-SERVER PAIRS WITH 100% CONNECTION FAILURE RATE.

Table VI lists a sample of such client-server pairs. A majority of these involve the servers `sohu.com` and `msn.com.tw`. However, note also that several US-based clients are not able to access the popular `craigslist.org` website. Furthermore, there is generally a good correlation between such total failures experienced by co-located clients.

We also point out some of the other interesting cases. The PlanetLab client at Northwestern University could not access `mp3.com`. Since the client has a very low failure rate otherwise and suffers no *client-side* failure episodes, the failure in accessing `mp3.com` could be due to university policy blocking accesses to this site. A similar pattern of failures happens with the two clients at CMU, which were unable to access `msn.com.tw`, despite having a very low failure rate otherwise and not suffering any client-side failure episodes. Another such example is the PlanetLab node at Cambridge University (UK), which was never able to access the university's own website.

We note that such seemingly "permanent" failures do tend

to get fixed over a period of time. For instance, in a separate data set gathered earlier in the project (October 2004), we had noticed that the CorpNet client located in China was never able to access `nmt.edu` over a period of several weeks. In the current data set, however, this client has no problems accessing the website.

We conduct a similar analysis of client-server-specific failures for DNS lookups. The median DNS failure rate across all client-server pairs is 0%, while the 95%-tile is 5%. While these are comparable to the connection failure rates, we did not find any client-server pairs with a 100% DNS lookup failure rate. However, the two broadband clients in Seattle as well as two dialup clients dialing into Seattle area POPs belonging to ICG and Level 3 failed to resolve `cs.technion.il` in over 70% of their attempts.

These examples point out the importance of analyzing client-server-specific failures more closely. We plan to do so in future work, by conducting a modified experiment where more frequent accesses are made between such problematic client-server pairs.

V. Discussion

Our results indicate the promise of a new paradigm for diagnosing Internet faults using failure observations gathered passively at multiple clients, an approach we outlined in a recent position paper describing the *NetProfiler* system [20]. We believe that such a cooperative, end-host-based approach to network diagnosis will become more important, as the Internet evolves towards becoming increasingly more opaque, with the growing deployment of firewalls, proxies and other "middle boxes".

Nevertheless, the feasibility of a client-based, cooperative fault diagnosis system hinges on several issues. First, there needs to be sufficient overlap between the access patterns of clients, to permit meaningful sharing of information and correlation analysis. As noted in Section III-A, previous findings in the context of web caching [27] point to the potential for such sharing. Furthermore, the potential for sharing fault information is increased because (a) sharing is not predicated on overlapping accesses to the same content (e.g., same files), and (b) even clients in different locations could usefully share fault information pertaining to a common set of servers they may have accessed. However, a more detailed analysis is needed to firmly establish the feasibility of such sharing.

Second, to be able to share failure observations with other end hosts, a client must have network connectivity. This requirement might sound paradoxial for a system designed to diagnose network faults. However, note that in many cases, an end host might see abnormally high failure rates only to certain servers or certain parts of the Internet. In such situations, it would be still be possible to compare observations with some or all other end hosts, say over a peer-to-peer network. In addition, even if a client is unable to communicate with any other end host, it could record its failure observations, and later share it with its peers. Such "post-mortem" analysis could help users benchmark the quality of their network experience against that of others over the long term, helping drive such decisions as upgrading service or switching ISPs.

Third, there are key security issues that confront any cooperative network diagnosis system. In particular, there are two interrelated issues: protecting the privacy of users and ensuring the integrity of the information that is shared. There is an inherent tradeoff between these goals, which makes it hard to achieve both to the fullest extent. In our position paper [20], we have outlined strategies for partially achieving these security goals. We defer the design of a complete solution to future work.

VI. Related Work

Several measurement studies of Internet performance and failures have appeared in the literature. The studies that are most closely related to our work are [11], [28], [16]. Several researchers have focused on individual facets of the performance of the wide-area Internet, such as DNS [18] or routing [14]. In this section, we will briefly discuss some of this work and contrast it with our approach. One overall observation is that previous studies have focused on hosts in academic and research networks. Such hosts have been at one or both ends of the experiments reported in these previous studies. In our study, we also consider hosts on commercial dialup and broadband ISP networks, and on a corporate network.

In [11] the authors analyze failures observed in wide-area networks. Their goal is to evaluate techniques such as caching and pre-fetching that could mitigate the impact of wide-area failures. They determine the location, duration, and rate of failures using traceroute measurements from previous studies. However, they do not consider failures beyond the IP-level path (e.g., DNS lookup failures). In contrast, our study relies on client-based passive observation of entire end-to-end transactions and correlation analysis to determine the location of failures.

In [28], a system called PlanetSeer is proposed to monitor and characterize path failures in the wide-area Internet. PlanetSeer monitors incoming and outgoing connections from of a set of PlanetLab nodes that serve as a distributed web proxy cache. Traffic anomalies are detected by passive monitoring. Once an anomaly is detected, PlanetSeer uses active probing (traceroute) from multiple nodes to further analyze and obtain a fine-grained view of the anomaly. However, as noted in Section I, the use of traceroute is problematic, although PlanetSeer avoids some of these problems because it is a *server-based* system rather than a client-based one. In contrast, we rely entirely on *passive* observation of traffic at *clients*. Furthermore, we consider all components, including DNS and web proxies (if any), that might be responsible for the failure of end-to-end transactions, rather than just the wide-area IP-level path.

In [16] the authors study path failures in the wide-area Internet. Their measurements show that most failures occur close to clients, and they develop a technique called one-hop source routing to route around failures that occur further away

from the client. One-hop source routing is much simpler than the re-routing techniques proposed previously [8], [24]. This work shares many of the characteristics of PlanetSeer that are in contrast to our work: the use of traceroutes for active probing and a focus on failures in just the IP-level path.

During the 1990s, Paxson conducted pioneering studies of Internet routing [21] and end-to-end TCP dynamics [22]. These studies, which pre-dated the PlanetLab infrastructure, were conducted using about 35 nodes, most of them on academic or research networks, that exchanged data using TCP and did traceroutes to each other repeatedly. Many of the findings (e.g., those pertaining to routing anomalies) that have inspired later work. However, the methodology used by Paxson is not suitable for our purposes, since it assumes control over both end-points, depends on traceroutes, and focuses on just the IP-level path.

SPAND [25] is a system for sharing performance information among end hosts belonging to a single subnet or site. The performance reports that are stored in a central database are used by end-hosts for performance prediction and mirror selection. While our work also focuses on sharing information across end-hosts, we differ from SPAND in a couple of key ways: (a) our goal is characterizing wide-area Internet *failures* rather than performance prediction, and (b) we combine observations made by a widely distributed set of end-hosts rather than just hosts in the same subnet or site.

NETI@home [26] is a system to gather detailed network performance information from end hosts. The goal is to compile this information and make it publically available. It is not clear, however, how this information is analyzed and whether it is being used to diagnose network problems.

Several researchers have studied individual facets of performance of the wide-area Internet. [18] reports on the performance of DNS queries originating from two sites: KAIST and MIT. A key finding is that about 36% of DNS lookups returned either no answer or an error. However, with data from just two sites, it is difficult to quantify the correlation of failures across sites. Studies such as [12], [14] have correlated BGP information from routers in a diverse set of locations to locate the source of Internet routing instabilities. Other studies [17] have considered the failure of intra-domain routing protocols such as IS-IS. In [29] the authors studied the performance of wide-area TCP connections by observing TCP flows in the middle of the network. However, none of these studies consider the impact of these problems on the performance of the end-to-end transaction. Furthermore, all of these studies are based on data gathered from the interior of the network, whereas our study is based on the end-host view.

Finally, our work bears some similarity to the work on network tomography [10], in terms of combining observations made on multiple network paths to deduce the internal state of the network. The key difference, however, is that tomography techniques are based on the analysis of fine-grained packet-level correlations, and therefore have typically involved active probing using multicast [10] or unicast [13] packets. Even the work on more coarse-grained, passive tomography [19] depends on active probing to discover the network topology.

In summary, our work is distinguished from previous work in terms of our focus on characterizing a broad range of Internet faults based on *passive*, *client-based* observations of *end-to-end* transactions, using *correlation* (or lack thereof) between data gathered at multiple sites.

VII. Conclusion

We have presented a client-based characterization of end-to-end Internet faults. Unlike many prior studies of Internet faults, our approach is based on correlating passive observations of end-to-end faults across a distributed set of clients.

We gathered failure data over a period of one month using a heterogeneous collection of 134 client hosts downloading content from a diverse set of 80 websites. We found a wide range in the failure rate of these transactions.

We have presented a correlation methodology to analyze failure data gathered at different clients. Using this methodology, we were able to classify many of the failures as being likely due to server-side or client-side problems. Over 30% of the failures seen in our data were caused by DNS problems, with most of the rest being due to the inability of the client to establish a TCP connection to the remote web server. The vast majority of DNS failures were inferred as being due to client-side problems while a majority of connection problems were inferred as being due to server-side problems. While direct validation of these inferences is difficult to do, we were able to do some indirect validation.

Our findings indicate the promise of a cooperative, client-based approach to monitoring and diagnosing Internet faults. Clients make observations on the failure of end-to-end transactions that they are involved in, and information from multiple clients is combined to arrive at a more complete picture of the extent and the likely cause of faults. We are currently building such a system.

Acknowledgements

We thank PlanetLab as well as the following individuals who provided us with access to the client hosts used in this study: Ramana Kompella, Karthik Lakshminarayanan, Yunxin Liu, Ant Rowstron, Rajesh Rao, Gurdev Sethi, Qian Zhang, and Lidong Zhou.

References

[1] Alexa Web Top 500. http://www.alexa.com/site/ds/top_500.

[2] `dig` tool. http://www.dns.net/dnsrd/tools.html#tool-dig.

[3] `tcpdump` tool. http://www.tcpdump.org/.

[4] `wget` tool. http://www.gnu.org/software/wget/wget.html.

[5] `windump` tool. http://windump.polito.it/.

[6] U.S. Broadband Penetration Report, June 2004. http://www.websiteoptimization.com/bw/0406/.

[7] M. Allman, W. Eddy, and S. Ostermann. Estimating Loss Rates With TCP. *Performance Evaluation Review*, Dec. 2003.

[8] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.

[9] S. Banerjee, M. Pias, and T. Griffin. The Interdomain Connectivity of PlanetLab Nodes. *PAM*, April 2004.

[10] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, November 1999.

[11] B. Chandra, M. Dahlin, L. Gao, and A. Nayate. End-to-end WAN Sevice Availability. *IEEE/ACM Transactions on Networking*, April 2003.
[12] D.-F. Chang, R. Govindan, and J. Heidemann. The Temporal and Toplogical Characteristics of BGP Path Changes. In *ICNP*, November 2003.
[13] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley. Inferring Link Loss using Striped Unicast Probes. In *INFOCOM*, 2001.
[14] A. Feldmann, O. Maennel, Z. Mao, A. Berger, and B. Maggs. Locating internet routing instabilities. In *SIGCOMM*, September 2004.
[15] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.
[16] K. Gummadi, H. Madhyastha, S. Gribble, H. Levy, and D. J. Wetherall. Improving the Reliability of Internet Paths with One-hop Source Routing. In *OSDI*, 2004.
[17] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *IMW*, 2002.
[18] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. DNS Performance and the Effectiveness of Caching. *IEEE/ACM Transactions on Networking*, 10(5), October 2002.
[19] V. N. Padmanabhan, L. Qiu, and H. Wang. Server-based Inference of Internet Link Lossiness. In *INFOCOM*, 2003.
[20] V. N. Padmanabhan, S. Ramabhadran, and J. Padhye. NetProfiler: Profiling Wide-Area Networks Using Peer Cooperation. In *IPTPS*, February 2005.
[21] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, October 1997.
[22] V. Paxson. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3):139–152, June 1999.
[23] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, Jun. 1995.
[24] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: a case for informed internet routing and transport. 1999.
[25] S. Seshan, M. Stemm, and R. H. Katz. Spand: Shared passive network performance discovery. In *USITS*, 1997.
[26] C. R. Simpson and G. F. Riley. NETI@home: A Distributed Approach to Collecting End-to-End Network Performance Measurements. *PAM*, April 2004.
[27] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the Scale and Performance of Cooperative Web Proxy Caching. In *SOSP*, 1999.
[28] M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services. In *OSDI*, 2004.
[29] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *SIGCOMM*, August 2002.
[30] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. In *IMW*, 2001.