Measuring Interactions Between Transport Protocols and Middleboxes

Alberto Medina, Mark Allman, Sally Floyd
ICSI Center for Internet Research
{medina,mallman,floyd}@icir.org

ABSTRACT

In this paper we explore the current network environment with respect to how the network's evolution ultimately impacts end-to-end protocols. The traditional end-to-end assumptions about the Internet are increasingly challenged by the introduction of intermediary network elements (middleboxes) that intentionally or unintentionally prevent or alter the behavior of end-to-end communications. This paper provides measurement results showing the impact of the current network environment on a number of traditional and proposed protocol mechanisms (e.g., Path MTU Discovery, Explicit Congestion Notification, etc.). We present results of measurements taken using an active measurement framework to study web servers. We analyze our results to gain further understanding of the differences between the behavior of the Internet in theory versus the behavior we observed through measurements. In addition, these measurements can be used to guide the definition of more realistic Internet modeling scenarios.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.3 [**Computer-Communication Networks**]: Network Operations; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks; C.2.6 [**Computer-Communication Networks**]: Internetworking

General Terms

Measurement, Design, Reliability, Standardization, Verification

Keywords

TCP, middleboxes, Internet, evolution

1. INTRODUCTION

While the Internet's architecture, protocols and applications are constantly evolving, there is often *competing evolution* between various network entities. This competing evolution can impact performance and robustness, and even halt communications in some cases. For instance, [23] shows that when setting up a TCP connection to a web server, attempting to negotiate the use of Explicit Congestion Notification (ECN) [27] interfered with connection establishment for over 8% of the web servers tested in 2000. For such web servers, the client can only establish a TCP connection by re-attempting the connection without negotiating ECN usage. The connection failures in the presence of ECN negotiation were caused by firewalls configured to interpret the attempt to negotiate ECN as the signature of a port-scanning tool [10]. On the one hand, these firewalls can be seen as incorrectly associating new functionality with one of the first appearances of that new functionality in an undesirable application. On the other hand, the firewalls can also be seen as doing their job of blocking unwanted traffic. This example shows the fundamental problem of different evolution paths that can cross to the detriment of smooth traffic flow on the Internet.

In this paper, we investigate the evolution of TCP [26], the Internet's most heavily used transport protocol, in the context of ongoing changes to the Internet's basic architecture. In particular, we study the ways in which so-called "middleboxes" (firewalls, NATs, proxies, etc.) — which change the Internet's basic *end-to-end principle* [28] — impact TCP. We seek to elucidate unexpected interactions between layers and ways in which the Internet differs from its textbook description, including the difficulties various real-world "gotchas" impose on the evolution of TCP (and end-to-end protocols in general). The measurements presented in this paper also serve as lessons for efforts that wish to further evolve end-to-end protocols and the Internet architecture.

In the study presented in this paper, we use active measurements to assess the capabilities supported by web servers (the primary data senders in web transactions) and their behavior in the context of the current Internet architecture on which they communicate. The remainder of this paper is organized as follows. Section 2 describes related work on measurement studies of transport protocols. Section 3 describes the tools and methodology we use in our study. Section 4 explores interactions between middleboxes and transport protocols. Section 5 discusses additional results. Finally, Section 6 presents our conclusions, and discusses open questions and future work.

2. RELATED WORK

This paper uses and extends the methodology from [23] on the TCP Behavior Inference Tool (TBIT). TBIT, the measurement tool used in our work, follows an earlier history of active probing of TCP. For instance, [8] treats TCP implementations as black boxes, observing how they react to external stimuli, and studying specific TCP implementations in order to assess the adherence to the specification.

There is also a considerable body of work on passive tests of

IMC'04, October 25–27, 2004, Taormina, Sicily, Italy.

TCP based on the analysis of packet traces. [24] outlines *tcpanaly*, a tool for analyzing a TCP implementation's behavior by inspecting sender and receiver packet traces of TCP connections run between pairs of hosts, while [25] outlines observed packet dynamics based on *tcpanaly*'s analysis. Finally, [3] assesses the properties of web clients using packet traces of TCP connections to a particular web server.

In addition, there is some research in the literature on the effect of middleboxes on transport protocol performance (e.g., [4]). We do not discuss the body of research on general architectural evaluations of middleboxes, or on the effect of middleboxes on DNS, BGP, and the like. Rather, the study presented in this paper focuses on interactions between middleboxes and transport protocols.

Finally, there is a large body of literature on active and passive approaches for estimating end-to-end network path properties using TCP [24, 6, 11]. In this paper we do not discuss TCP-based tests for estimating path properties such as loss rates, available or bottleneck bandwidth and durations of congestion episodes. Also prevalent in the literature, yet out of scope for the current effort, is the body of work based on passive measurements of traffic on a particular link to determine the breakdown of the traffic in terms of round-trip times, application layer protocols, transfer sizes, etc.

3. MEASUREMENTS: TOOLS AND DATA

As discussed above, we employ active measurements in this study. Specifically, we use a measurement tool called TBIT [23], to conduct active measurements that probe web servers for their characteristics in the context of the environment on which the communications take place. We note that in some cases, information gathered via active measurements can be obtained as well via passive means. However, many of the TBIT tests are not amenable to straightforward post-facto analysis of packet traces. For instance, tests that involve actively attempting alternative schemes in connection initiation cannot be performed by passive trace analysis alone. Consider a test for middleboxes that block TCP SYN segments when the SYNs carry advertisements for ECN. Packet traces can indicate whether connections attempting to use ECN succeed or fail. However, determining that the failure of a connection attempting to negotiate ECN is due to a middlebox blocking ECN-capable SYNs takes active insertion of SYNs with and without ECN advertisements.

The measurements gathered for this work are the result of specific tests implemented in the TBIT framework. Each test is designed to examine a specific aspect of the behavior of the remote web servers, or of the path to and from the web server. Most of these tests examine the behavior of TCP implementations on the web servers. However, the tests are not restricted to TCP (e.g., the Path MTU Discovery [22] tests). TBIT establishes a TCP connection with the remote host at the user level. TBIT composes TCP segments (or segments from another protocol), and uses raw IP sockets to send them to the remote host. TBIT also sets up a host firewall to prevent incoming packets from reaching the kernel of the local machine; a BSD packet filter is used to deliver incoming packets to the TBIT process. TBIT's user-level connection is used to control the sending of carefully constructed packets (control, data, acknowledgment, etc.) as desired from the local host. Note that all the TBIT tests are susceptible to network conditions to some degree. For instance, if an ACK sent by TBIT is lost in transit to the web server the result of the test could be inconclusive or even wrongly reported. We have taken test-specific measures to make each of our tests as robust as possible. In addition, our large set of web servers (described below) helps to minimize any biases that bogus tests introduce into our results.

Server name	Location	Cache size
pb.us.ircache.net	Pittsburgh, PA	12867
uc.us.ircache.net	Urbana-Champain, IL	18711
bo.us.ircache.net	Boulder, CO	42120
sv.us.ircache.net	Silicon Valley, CA	28800
sd.us.ircache.net	San Diego, CA	19429
pa.us.ircache.net	Palo Alto, CA	5511
sj.us.ircache.net	MAE-West, San Jose, CA	14447
rtp.us.ircache.net	Research Triangle, NC	33009
ny.us.ircache.net	New York, NY	22846

Table 1: IRCache servers and locations

The list of target web servers used in our study was gathered from IRcaches, the NLANR Web Caching project [1]. We used web cache logs gathered from nine different locations around the United States. Table 1 shows the cache logs used from February 2004, along with the log sizes, expressed as the number of unique IP server addresses from each cache. Since the caches are located within the continental US, most of the cached URLs correspond to domain names within the US. However, the cache logs also contain a sizable set of web servers located in the other continents. Of the 84,394 unique IP addresses[1] found in the cache logs: 82.6% are from North America, 10.2% are from Europe, 4.9% are from Asia, 1.1% are from Oceania, 1.0% are from South America and 0.2% are from Africa.

All the TBIT tests outlined in this paper were conducted between February and May 2004. The TBIT client was always run from a machine on the local network at our research laboratory. There is no local firewall between the machine running TBIT and our Internet connection.

4. MIDDLEBOXES AND TRANSPORT PROTOCOLS

The increased prevalence of middleboxes puts into question the general applicability of the end-to-end principle. Middleboxes introduce dependencies and hidden points of failure, and can affect the performance of transport protocols and applications in the Internet in unexpected ways. Middleboxes that divert an IP packet from its intended destination, or modify its contents, are generally considered fundamentally different from those that correctly terminate a transport connection and carry out their manipulations at the application layer. Such diversions or modifications violate the basic architectural assumption that packets flow from source to destination essentially unchanged (except for TTL and QoS-related fields). The effects of such changes on transport and application protocols are unpredictable in the general case. In this section we explore the ways that middleboxes might interfere in unexpected ways with transport protocol performance.

4.1 ECN-capable Connections

Explicit Congestion Notification (ECN) [27] is a mechanism that allows routers to mark packets to indicate congestion, instead of dropping them. After the initial deployment of ECN-capable TCP implementations, there were reports of middleboxes (in particular, firewalls and load-balancers) that blocked TCP SYN packets attempting to negotiate ECN-capability, either by dropping the TCP

[1]We note that the list of servers could be biased by a single machine having multiple unique IP addresses – which would tend to skew the results. However, due to the size of the server list, we believe that such artifacts, while surely present, do not highly skew the overall results.

Year:	2000		2004	
ECN Status	**Number**	**%**	**Number**	**%**
Number of Servers	24030	100%	84394	100%
I. Classified Servers	21879	91%	80498	95.4%
I.A. Not ECN-capable	21602	90%	78733	93%
I.B. ECN-Capable	277	1.1%	1765	2.1%
I.B.1. no ECN-Echo	255	1.1%	1302	1.5%
I.B.2. ECN-Echo	22	0.1%	463	0.5%
I.C. Bad SYN/ACK	0		183	0.2%
II. Errors	2151	9%	3896	4.6%
II.A. No Connection	2151	9%	3194	3.8%
II.A.1. only with ECN	2151	9%	814	1%
II.A.2. without ECN	0		2380	2.8%
II.B. HTTP Error	–		336	0.4%
II.C. No Data Received	–		54	0%
II.D. Others	–		312	0.4%

Table 2: ECN Test Results

SYN packet, or by responding with a TCP Reset [10]. [23] includes test results showing the fraction of web servers that were ECN-capable and the fraction of paths to web servers that included middleboxes blocking TCP SYN segments attempting to negotiate ECN-capability. The TBIT test for ECN is described in [23].

Table 2 shows the results of the ECN test for 84,394 web servers. Only a small fraction of servers are ECN-Capable – this percentage has increased from 1.1% of the web servers tested in 2000 to 2.1% in 2004. After a web server has successfully negotiated ECN we send a data segment marked "Congestion Experienced (CE)" and record whether the mark is reflected back to the TBIT client via the ECN-Echo in the ACK packet. The results are given on lines I.B.1 and I.B.2 of the table. In roughly three-quarters of cases when ECN is negotiated, a congestion indication is not returned to the client. This could be caused by a bug in the web server's TCP implementation or a middlebox that is clearing the congestion mark as the segment traverses the network. Finally, we also observe a small number of web servers send a malformed SYN/ACK packet, with both the ECN-Echo and Congestion Window Reduced (CWR) bits set in the SYN/ACK packet (line I.C of the table).

For 3194 of the web servers, no TCP connection was established. For our TBIT test, if the initial SYN packet is dropped, TBIT resends the same SYN packet – TBIT does not follow the advice in RFC 3168 of sending a new SYN packet that does not attempt to negotiate ECN. Similarly, if TBIT receives a TCP Reset in response to a SYN packet, TBIT drops the connection, instead of sending a subsequent SYN packet that does not attempt to negotiate ECN-capability.

In order to assess how many of these connection failures are caused by the attempt of ECN negotiation, we run two back-to-back TBIT tests to each server. The first test does not attempt to negotiate ECN. After a two-second idle period, another connection is attempted using ECN. We observe that 814 connections (1% of the web servers, or 25% of the connection failures) are apparently refused because of trying to negotiate ECN, since the connection was established successfully when no ECN negotiation was attempted. Table 2 indicates that the fraction of web servers with ECN-blocking middleboxes on their path has decreased substantially since September 2000 – from 9% in 2000 to 1% in 2004.

We further explored the behavior of ECN-capable servers by recording the ECT codepoints in the data packets received by TBIT. Table 3 shows the number of servers from which the different codepoints were observed. TBIT received data packets with the ECT 00 codepoint from about 42% of the ECN-capable servers. The ECN specification defines two ECT code points that may be used by a sender to indicate its ECN capabilities in IP packets. The specification further indicates that protocols that require only one such a codepoint *should* use $ECT(1) = 10$. We observe that ECN-capable servers do use ECT(1) and found no server made use of the $ECT(0) = 01$ codepoint. We further observe that no router between our TBIT client and the ECN-capable servers reported Congestion Experienced (CE) in any segment. Finally, TBIT received both data segments with $ECT = 00$ and $ECT = 10$ in the same connection from about 10% of the ECN-capable servers. This behavior may indicate that the ECT code point is being erased by a network element (e.g. router or middlebox) along the path between the ECN-capable server and the client.

ECN fields in data packets	Number	% of total
ECN-capable servers	1765	100%
Received packets w/ ECT 00 (Not-ECT)	758	42%
Received packets w/ ECT 01 (ECT(1))	0	0%
Received packets w/ ECT 10 (ECT(0))	1167	66%
Received packets w/ ECT 11 (CE)	0	0%
Received packets w/ ECT 00 and ECT 10	174	10%

Table 3: Codepoints in data packets from ECN-Capable Servers

4.2 Path MTU Discovery

TCP performance is generally proportional to the segment size employed [16]. In addition, [16] argues that packet fragmentation can cause poor performance. As a compromise, TCP can use Path MTU Discovery (PMTUD) [22, 20] to determine the largest segment that can be transmitted across a given network path without being fragmented. Initially, the data sender transmits a segment with the IP "Don't Fragment" (DF) bit set and whose size is based on the MTU of the local network. Routers along the path that cannot forward the segment without first fragmenting it (which is not allowed because DF is set) will return an ICMP message to the sender noting that the segment cannot be forwarded because it is too large. The sender then reduces its segment size and retransmits. Problems with PMTUD are documented in [17], which notes that many routers fail to send ICMP messages and many firewalls and other middleboxes are often configured to suppress all ICMP messages, resulting in PMTUD failure. If the data sender continues to retransmit large packets with the DF bit set, and fails to receive the ICMP messages indicating that the large packets are being dropped along the path, the packets are said to be disappearing into a PMTUD *black hole*. We implemented a PMTUD test in TBIT to assess the prevalence of web servers using PMTUD, and the success or failure of PMTUD for these web servers. The test is as follows:

1. TBIT is configured with a *virtual link MTU*, MTU_v. In our tests, we set MTU_v to 256 bytes.

2. TBIT opens a connection to the web server using a SYN segment that contains an MSS Option of 1460 bytes (which is based on the actual MTU of the network to which the TBIT client is attached).

3. The TCP implementation at the server accepts the connection and sends MSS-sized segments, resulting in transmitted packets of MSS + 40 bytes. If the data packets from the server do not have the DF bit set, then TBIT classifies the

PMTUD Status	Number	% of total
Total Number of Servers	81776	100%
I. Classified Servers	71737	88%
I.A. PMTUD not-enabled	24196	30%
I.B. Proper PMTUD	33384	41%
I.C. PMTUD Failed	14157	17%
II. Errors	9956	12%
II.A. Early Reset	545	0.6%
II.B. No Connection	2101	2.5%
II.C. HTTP Errors	2843	3.4%
II.D. Others	4467	5.5%

Table 4: PMTUD Test Results

server as not attempting to use PMTUD. If TBIT receives a packet with the DF bit set that is larger than MTU_v TBIT rejects the packet, and generates an ICMP message to be sent back to the server.

4. If the server is capable of receiving and processing such ICMP packets, it will reduce the MSS to the value specified in the MTU field of the ICMP packet, minus 40 bytes for packet headers, and resume the TCP connection. In this case, TBIT accepts the proper-sized packets and the communication completes.

5. If the server is not capable of receiving and processing ICMP packets it will retransmit the lost data using the same packet size. Since TBIT rejects packets that are larger than MTU_v the communication will eventually time out and terminate and TBIT classifies the server/path as failing to properly employ PMTUD.

Table 4 shows that PMTUD is used successfully for slightly less than half of the servers on our list. For 31% of the servers on our list, the server did not attempt Path MTU Discovery. For 18% of the servers on our list, Path MTU Discovery failed, presumably because of middleboxes that block ICMP packets on the path to the web server.

Alternate methods for determining the path MTU are being considered in the Path MTU Discovery Working Group in the IETF, based on the sender starting with small packets and progressively increasing the segment size. If the sender does not receive an ACK packet for the larger packet, it changes back to smaller packets.

In a similar strategy, called *black-hole detection*, if a packet with the DF bit set is retransmitted a number of times without being acknowledged, then the MSS will be set to 536 bytes [2]. We performed a variant of the PMTUD test in which TBIT does not send the ICMP packets, to see if any server reduces the size of the packets sent simply because it didn't receive an ACK for the larger packet. We didn't find any servers performing black-hole detection.

Since a non-trivial number of network elements discard well-known ICMP packets the results of our tests do not offer hope for protocol designers proposing to use new ICMP messages to signal various network path properties to end systems (e.g., for explicit corruption notification, handoff or outage notification, etc.).

4.3 IP Options

IP packets may contain options to encode additional information at the end of IP headers. A number of concerns have been raised regarding the use of IP options. One concern is that the use of IP options may significantly increase the overhead in routers, because in some cases packets with IP options are processed on the *slow path* of the forwarding engine. A second concern is that receiving IP packets with malformed IP options may trigger alignment problems on many architectures and OS versions. Solutions to this problem range from patching the OS, to blocking access to packets using unknown IP options or using IP options in general. A third concern is that of possible denial of service attacks that may be caused by packets with invalid IP options going to network routers. These concerns, together with the fact that the generation and processing of IP options is nonmandatory at both the routers and the end hosts, have led routers, hosts, and middleboxes to simply drop packets with unknown IP options, or even to drop packets with standard and properly formed options. This is of concern to designers of transport protocols because of proposals for new transport mechanisms that would involve using new IP options in transport protocols (e.g., [15, 9]).

TBIT's IP options test considers TCP connections with three types of IP options in the TCP SYN packet, the *IP Record Route Option*, the *IP Timestamp Option*, and a new option called *IP Option X*, which is an undefined option and represents any new IP option that might be standardized in the future. We experimented with two variants of Option X, both of size 4. The first variant uses a copy bit of zero, class bits set to zero and 25 as the option number. The second variant of IP Option X sets the class bits to a reserved value, and uses an option number of 31. The results for experiments with both Option X variants are similar.

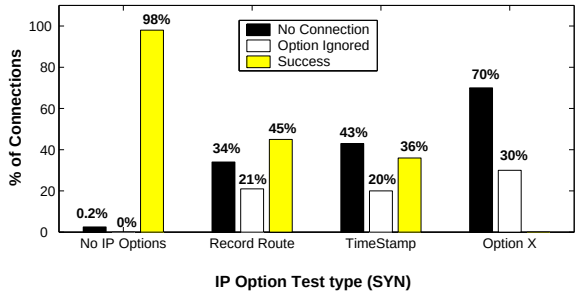

Figure 1: Handling IP Options in TCP SYN packets.

Figure 1 shows the TCP connection behavior with different IP options in the associated SYN packets. For each attempted connection there are three possible outcomes: no connection established, connection established with the IP option ignored, or IP option accepted. As Figure 1 shows, in many cases no connection was established when the Record Route Option or the Timestamp Option was included in the SYN packet. When IP Option X is included in the SYN segment, the connection was not established to over 70% of the web servers tested. This does not bode well for the deployment of new IP options in the Internet.

Most IP options are usually expressed in the first packet (e.g., the TCP SYN packet) in the communication between end hosts. We performed an additional test to assess the behavior when IP option X is placed in data packets in the middle of an established connection. For each established connection TBIT offers two classifications: "success" or "broken connection". The former indicates that the server successfully delivered its data regardless of the IP option insertion. The latter classification indicates that the insertion of the IP option forced the connection to be idle for at least 12 seconds (which we then define as "broken"). We performed two sets of tests, with and without insertion of option X. The connection failure rate across both sets of tests is roughly 3%. The tests without IP options show nearly 6% of the connections are "broken"

for some reason. Meanwhile, when inserting IP option X into the middle of the transfer, 44% of the connections are broken, indicating a significant issue when attempting to utilize IP options in mid-connection.

4.4 TCP Options

Next we turn our attention to potential problems when TCP options are employed. TCP options are more routinely used than IP options. For instance, TCP uses the timestamp option [14] to (among other things) take round-trip time measurements more frequently than once per round-trip time, for the Protection Against Wrapped Sequences [14] algorithm and for detecting spurious timeouts [18].

However, middleboxes along a path can interfere with the use of TCP options, in an attempt to thwart attackers trying to fingerprint hosts. Network mapping tools such as NMAP (Network Mapper) use information from TCP options to gather information about hosts; this is called *fingerprinting*. Countermeasures to fingerprinting, sometimes called *fingerprint scrubbers* [29], attempt to block fingerprinting by inspecting and minimally manipulating the traffic stream. One of the strategies used by fingerprint scrubbers is to reorder TCP options in the TCP header; any unknown options may be included after all other options. The TCP options test checks to see if sites reject connections negotiating specific or unknown TCP options, or drop packets encountered in the middle of the stream that contain those options.

The TCP options test first assesses the behavior of the web server when the TCP Timestamp option is included in the SYN packet. To test for performance with unknown TCP options, we also initiate connections using an unallocated option number, *TCP Option Y*, in the SYN packet.

Our tests indicate a connection failure rate of about 0.2% in all scenarios. Option Y is ignored in the remainder of the connections. The timestamp option is ignored by roughly 15% of the servers (but the connection is otherwise fine). The reason the servers ignore the timestamp option is not visible to TBIT, but could be either a middlebox stripping or mangling the option or the web server not supporting timestamps. Next we assess the use of options in the middle of a TCP connection, by establishing a connection without TCP options and then using the Timestamp option or Option Y on a data packet in the middle of the connection. The connection failure rate for both options is roughly 3% – indicating that sending unknown options midstream is not problematic for most web servers.

5. ADDITIONAL RESULTS

In addition to the measurements presented in this paper we executed additional tests to detect and quantify the presence of various algorithms and parameters in the web server's TCP implementations [21]. In this section we summarize several of the results we have obtained in the hopes of providing researchers with guidance on constructing models for their simulation and emulation studies. The tests that produce these results are similar to the tests outlined in [23].

- When not using selective acknowledgments (SACK) [19] (e.g., because of non-SACK-capable receivers), roughly 75% of the web servers we could classify used NewReno loss recovery [13]. This suggests that studies involving only the Reno TCP variant should be discouraged.
- Nearly 70% of web servers advertise themselves as SACK capable. Of the servers that advertise SACK support, over 95% make some use of the SACK information sent by the web client.
- Of the web servers that advertise SACK support, more than 95% correctly generate SACK blocks when data sent by the client is missing.
- RFC 3390 [5] allows a TCP to use an initial congestion window of 1–4 segments, depending on their size. We found that 42% of the web servers in our dataset used an initial congestion window of 1 segment, while 54% used an initial window of 2 segments. Less than 3% of the web servers used 3 or 4 segment initial congestion windows. We noted initial congestion window values as large as 129 segments (in small proportions of the servers).

In addition to the above results we have additional results regarding TCP's use of Limited Transmit, Appropriate Byte Counting, Congestion Window Validation, Window Scale Option, Minimum RTO, Minimum MSS, and the Deployment of D-SACK (Duplicate SACK). Also, we have measurement strategies for detecting middleboxes that perform TTL-rewriting, and for detecting the effects of reordering on transport protocols. Finally, another component of our work uses packet traces from near a set of web servers to assess the client-side deployment of various end-host algorithms and protocol mechanisms.

6. CONCLUSIONS AND FUTURE WORK

The contribution of the work presented in this paper is to illustrate the ways that the performance of protocol mechanisms in the Internet differ from theory. The insights gathered from our measurements involving the interactions between TCP and middleboxes along the network path are summarized in Table 5.

Additionally, there are a wealth of important TCP behaviors that we have not examined in our tests, and new TCP mechanisms are continually being proposed, standardized and deployed (e.g., High-Speed TCP [12]). Assessing their deployment, characteristics and behaviors in the context of the evolving Internet architecture are useful future work.

Another class of extensions to this work is exploring the behavior of TCP in alternate applications (e.g., peer-to-peer systems, email, web caching, etc.).

An additional interesting area for future investigation is using TBIT-like tools for *performance* evaluation. For instance, a performance comparison of servers using various initial congestion window values might be useful or servers with and without SACK-based loss recovery. Developing techniques for conducting this kind of performance comparison in a solid and meaningful way (and detecting when such a comparison is not meaningful) is a rich area for future investigation. Furthermore, performing tests from multiple vantage points would be an interesting extension for detecting differences in behavior among multiple paths which may point to middleboxes in some paths.

As new transport protocols such as SCTP and DCCP begin to be deployed, another area for future work will be to construct tools to monitor the behavior, deployment and characteristics of these protocols in the Internet.

While we examined some ways that middleboxes interfere with TCP communications, a key open question is that of assessing ways that middleboxes affect the *performance* of transport protocols or of applications. One middlebox that clearly affects TCP performance is that of Performance Enhancing Proxies (PEPs) [7] that break single TCP connections into two connections potentially changing end-to-end behavior. While [4] presents some results in this general area, additional active tests may be useful to investigate this further.

Behavior	Section	Possible Interactions with Routers or Middleboxes
ECN	4.1	Advertising ECN prevents connection setup for a small (and diminishing) set of hosts.
PMTUD	4.2	Less than half of the web servers successfully complete Path MTU Discovery. PMTUD is attempted but fails for one-sixth of the web servers.
IP Options	4.3	For roughly one-third of the web servers, no connection is established when the client includes an IP Record Route or Timestamp option in the TCP SYN packet. For most servers, no connection is established when the client includes an unknown IP Option.
TCP Options	4.4	The use of TCP options does not interfere with connection establishment. Few problems were detected with known and unknown TCP options included in data packets in mid-stream.

Table 5: Information on interactions between transport protocols and routers or middleboxes.

Finally, a completely different kind of test that could benefit from the active probing approach outlined in this paper would be one to detect the presence or absence of Active Queue Management mechanisms at the congested link along a path. To some extent, this can be done with passive tests, by looking at the pattern of round-trip times before and after a packet drop. However, active tests may be much more powerful, by allowing the researcher to send short runs of back-to-back packets, as well as potentially problematic, in attempting to induce transient congestion in the network.

Acknowledgments

Orion Hodson assisted with our TBIT measurements. Sourabh Ladha and the anonymous reviewers gave us detailed and useful feedback.

This material is based in part upon work supported by the National Science Foundation under Grant Nos. 0205519 and 0230921. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

[1] NLANR Web Caching project. http://www.ircache.net/.

[2] PMTU Black Hole Detection Algorithm Change for Windows NT 3.51. Microsoft Knowledge Base Artible - 136970.

[3] Mark Allman. A Web Server's View of the Transport Layer. *Computer Communications Review*, 30(5):10–20, October 2000.

[4] Mark Allman. On the Performance of Middleboxes. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 307–312, October 2003.

[5] Mark Allman, Sally Floyd, and Craig Partridge. Increasing TCP's Initial Window, October 2002. RFC 3390.

[6] Mark Allman and Vern Paxson. On Estimating End-to-End Network Path Properties. pages 229–240, 1999.

[7] John Border, Markku Kojo, Jim Griner, Gabriel Montenegro, and Zach Shelby. Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations, June 2001. RFC 3135.

[8] Douglas E. Comer and John C. Lin. Probing TCP Implementations. In *USENIX Summer 1994 Conference*, 1994.

[9] Wesley Eddy, Shawn Ostermann, and Mark Allman. New Techniques for Making Transport Protocols Robust to Corruption-Based Loss. January 2004. Under submission.

[10] S. Floyd. Inappropriate TCP Resets Considered Harmful, 2002. RFC 3360.

[11] Sally Floyd. Tools for Bandwidth Estimation. Web page, URL 'http://www.icir.org/models/tools.html'.

[12] Sally Floyd. HighSpeed TCP for Large Congestion Windows, December 2003. RFC 3649.

[13] Sally Floyd, Tom Henderson, and Andrei Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm, April 2004. RFC 3782.

[14] V. Jacobson, R. Barden, and D. Borman. TCP Extensions for High Performance, May 1992. RFC 1323.

[15] Amit Jain and Sally Floyd. Quick-Start for TCP and IP, 2002. Internet-Draft draft-amit-quick-start-02.txt, expired, URL: http://www.icir.org/floyd/papers/draft-amit-quick-start-02.txt.

[16] Christopher Kent and Jeffrey Mogul. Fragmentation Considered Harmful. In *ACM SIGCOMM*, October 1987.

[17] Kevin Lahey. TCP Problems with Path MTU Discovery, September 2000. RFC 2923.

[18] R. Ludwig and M. Meyer. The Eifel Detection Algorithm for TCP, 2003. RFC 3522.

[19] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. TCP Selective Acknowledgement Options, October 1996. RFC 2018.

[20] Jack McCann, Steve Deering, and Jeffrey C. Mogul. Path MTU Discovery for IP version 6, August 1996. RFC 1981.

[21] Alberto Medina, Mark Allman, and Sally Floyd. Measuring the Evolution of Transport Protocols in the Internet, 2004. URL http://www.icir.org/tbit/.

[22] Jeffrey C. Mogul and Steve Deering. Path MTU Discovery, November 1990. RFC 1191.

[23] Jitendra Padhye and Sally Floyd. Identifying the TCP Behavior of Web Servers. In *ACM SIGCOMM*, August 2001.

[24] Vern Paxson. Automated Packet Trace Analysis of TCP Implementations. In *ACM SIGCOMM*, September 1997.

[25] Vern Paxson. End-to-End Internet Packet Dynamics. In *ACM SIGCOMM*, September 1997.

[26] Jon Postel. Transmission Control Protocol, September 1981. RFC 793.

[27] K.K. Ramakrishnan, Sally Floyd, and David Black. The Addition of Explicit Congestion Notification (ECN) to IP, September 2001. RFC 3168.

[28] J.H. Saltzer, D.P. Reed, and David Clark. End-to-End Arguments in System Design. In *Proceedings of the Second International Conference on Distributed Computing Systems*, pages 509–512, August 1981.

[29] Matthew Smart, G. Robert Malan, and Farnam Jahanian. Defeating TCP/IP Stack Fingerprinting. In *9th USENIX Security Symposium*, pages 229–240, 2000.