Probing the viability of TCP extensions

Adam Langley
Google Inc
agl@google.com

ABSTRACT

TCP was designed with extendibility in mind, chiefly reflected in the options mechanism. However, there have been repeated observations of misbehaving middleware that have hampered the deployment of beneficial TCP extensions.

This paper reports on an experiment to determine how prevalent three types of behaviour are. We find that 9.08% of hosts don't respond to SYN frames with payloads, 0.17% of hosts don't respond to SYN frames with non-standard options and 0.56% of hosts don't respond to SYN frames which attempt to negotiate ECN.

1. INTRODUCTION

When developing TCP extensions, authors have to be constantly aware that the public Internet is a hostile place for the non-conformist frame.

Explicit congestion notification[11] allows for routers using active queue management to mark frames that would otherwise have been dropped. These marked frames affect congestion control exactly as if a frame had been dropped, but do not require retransmissions etc.

ECN has been shown in simulations to be of significant benefit[6] and development continues with the recent work on adding ECN to SYN/ACK frames[7].

However, ECN is rarely enabled for fear of running afoul of malconforming middleware that drops such frames or even crashes at the sight of them. We seek to establish how many public webservers have ECN enabled and how many become unreachable when ECN support is advertised in a SYN frame.

New TCP extensions often involve new option kinds. However, designers have to be aware that malconforming middleware may also erroneously drop TCP frames with unknown options. We aim to evaluate how common this behaviour is.

Table 1: DNS error rates

Domain does not exist	98625
Timed out (10 seconds)	42101
Server failure	19840
Unknown error	2227
Response truncated	1

Additionally, the limited TCP option space is an occasional source of friction, especially in SYN frames where nothing about the destination host can be assumed. Half of the option space (20 bytes of a possible 40) in SYN frames is generally spoken for now, leaving only 20 bytes remaining.

Designs which require more option space in SYN frames are faced with a dilemma: The payload of a SYN frame is almost universally ignored by TCP stacks and would present a rich source of space. However, TCP clearly states that a payload in a SYN frame is perfectly acceptable and that stacks may enqueue such payloads for delivery to the application layer. We aim to determine what fraction of hosts actually ignore payloads in SYN frames.

2. EXPERIMENTAL SETUP

Approximately 10 million hostnames were extracted from a Google web index from August 2008. These hostnames have been referenced in a webpage that the Google crawler has seen but have not been checked to exist. The extraction takes the form of a MapReduce[5] which extracts hostnames from URLs in the map phase and then reduces by removing duplicates.

Since the number of duplicates can't be exactly predicted, the 10 million goal was only an estimate and the actual number extracted was 10,240,701.

Each hostname was then resolved to a single IPv4 address (the resulting DNS errors are broken down in table 1). After eliminating many duplicate IP addresses, 1,445,303 remained.

Each of these were probed using the state machine in figure 1. Upon entry to one of the lettered states, a frame is transmitted. The type of frame depends on the letter of the state.

'`S`' states transmit a simple SYN frame with no payload nor options. '`E`' states transmit a similar SYN, but with the

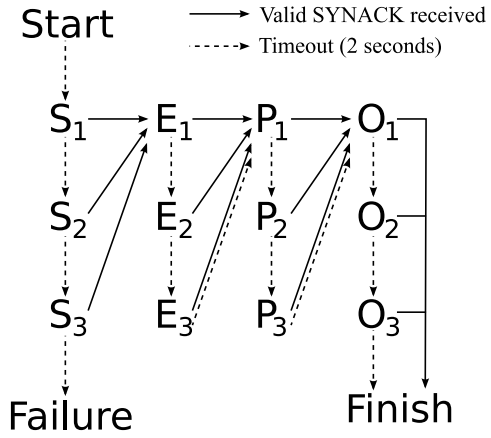

Figure 1: Prober state machine

CWR and ECN flags set as per [11]. 'P' states transmit a SYN with a 16 byte payload. 'O' states transmit a SYN with a 2 byte option with undefined kind number 42. Each type of SYN uses a distinct source port number to distinguish SYN/ACK frames from different experiments.

Each type of SYN is transmitted at most three times with a two second timeout between them. A SYN/ACK with the correct source address and port numbers is sufficient to move onto the next type of SYN frame. A passive open port is required on the target host and so all SYNs were transmitted to port 80.

Failure to get a response to the plain SYN probing results in marking the host as down and no other SYNs are sent.

For each each type of SYN frame transmitted, the number of SYNs transmitted before getting a reply is recorded. For the SYN with ECN flags, the presence of ECN flags in the SYN/ACK is recorded. For the SYN with a payload, the ACK number is examined to determine if the peer enqueued the payload (by ACKing at least a byte of it), if it ignored the payload (by ACKing only the SYN) or returned an invalid ACK number.

After probing, 1,357,338 (93.91%) of hosts were found to respond to a simple SYN to port 80 with a SYN/ACK and thus were considered 'up'.

3. UNKNOWN OPTIONS

Of 28, non experimental TCP option kinds assigned by IANA at the time of writing[3], only a handful are in common usage (EOL, NOP and maximum segment size[10], window scaling[4], SACK permitted and SACK[8] and timestamps[4]). New developments in TCP often require the use of new option kinds. Middleware which drops TCP frames containing unknown options thus hamper the development of TCP.

Of the hosts tested, 2335 (0.17%) responded to the initial SYN, but failed to respond to SYNs with the additional option. Although not zero, this is a pleasingly low number.

4. SYN PAYLOADS

Table 2: Prober results

Initial set of non-duplicate IPs	1,445,303
Host responds to a simple SYN	1,357,338 (93.91%)
No response to non-standard options	2,335 (0.17%)
Responds to SYNs with payloads	1,234,146 (90.92%)
Ignores payload	1,234,146 (100%)
Acks payload	0 (0%)
Invalid ACK number	0 (0%)
No response to SYNs with payloads	123,192 (9.08%)
Responds to SYNs with ECN bits	1,349,711 (99.44%)
ECN successfully negotiated	14,407 (1.07%)
ECN not supported	14,407 (98.93%)
No response to SYNs with ECN bits	7,627 (0.56%)

TCP allows for payloads to be carried in SYN frames. However, due to the denial of service issues with enqueuing data from unconfirmed connections, most stacks ignore such payloads and, validly, ACK only the SYN in a SYN frame, causing the host to retransmit the application layer data in a future packet.

When transmitting a SYN frame nothing can be assumed about the destination peer except that it supports TCP. The option space in a SYN is limited to 40 bytes, of which 20 bytes are taken by commonly used options.

It's sometimes difficult or impossible to fit a desired option in the remaining space, leading to workarounds like long options[2]. If SYN payloads were universally ignored, this could allow, after careful consideration, more aggressive extensions to TCP including a backwards compatible method of upgrading to a new transport layer entirely.

Of the 1.3 million hosts queried, every one of those which replied to a SYN with a payload ignored the payload and ACKed only the SYN.

However, 123,193 (9.08%) ignored the SYN entirely.

5. EXPLICIT CONGESTION NOTIFICATION

ECN enjoys widespread support (in Linux, OS X and Windows Vista at least). However, it is usually disabled by default because of compatibility issues.

Although ECN principally affects the Type of Service field in the IP header, its use is negotiated using a pair of formerly reserved bits in the TCP header.

Floyd et al report in[9] that 9% of hosts ignored SYNs that attempted to negotiate ECN in 2000 and that number had ignored to 1% by 2004. In the same paper, 1.1% of hosts were found to support ECN in 2000 and 2.1% by 2004.

We found that 14,407 (1.06%) of hosts supported ECN. However, 7,627 hosts (0.56%) ignored the SYN frames. This suggests that the rate of improvement in passing ECN is falling and that the support is stagnant or falling.

However, the failing hosts are not distributed randomly. The 7,627 hosts span only 4,613 /24 subnets. The top twenty subnets account for 778 failing hosts (10%). WHOIS[1] in-

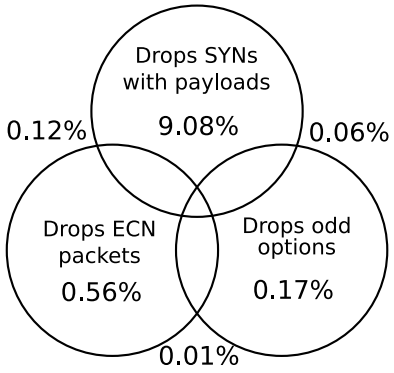

Figure 2: Overlap between failure modes

formation for 18 of those 20 subnets suggests that they are located in China.

6. CONCLUSIONS

Too much middleware fails to adhere to published specifications. This imposes a cost on the rest of the world which must delay or abandon positive extensions to TCP. This is not a new problem, although no good solutions have arisen in the years that it has been recognised.

We modified a version of `mtr`, a common traceroute tool, to transmit TCP SYNs with ECN bits set in an effort to find the middleware which was dropping these packets. Sadly, it appears that middleware which drops such SYN frames also tends to drop ICMP messages.

In the few cases where we did identify a good candidate, no information was forthcoming. No common TCP ports were open and OS fingerprinting was inconclusive.

The three types of failure which we investigated are certainly not independent, as figure 2 shows. A host failing on one count is much more likely to fail on one or both of the others. 0.01% of hosts failed on all three counts.

We can conclusively say that assuming that payloads in SYN frames will be ignored appears to be safe. However, the very high number of destinations which discarded such packets completely hampers any developments which might make use of this.

It's unclear what to do about ECN. Without pressure it appears that the rate of fixing malconforming middleware is slowing, although it does appear that fixing a few key locations would drastically improve the situation. The bias towards Chinese hosts is puzzling but it may be caused by national infrastructure filtering such packets.

7. REFERENCES

[1] L. Daigle. WHOIS Protocol Specification. RFC 3912 (Draft Standard), Sept. 2004.

[2] W. Eddy and A. Langley. Extending the space available for tcp options, 2008.

[3] IANA. Transmission control protocol (tcp) option numbers, 2008.

[4] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[5] S. G. Jeffrey Dean. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, December 2004.

[6] A. Kuzmanovic. The power of explicit congestion notification. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 61–72, New York, NY, USA, 2005. ACM.

[7] A. Kuzmanovic, A. Mondal, S. Floyd, and K. Ramakrishnan. Adding explicit congestion notification (ecn) capability to tcp's syn/ack packets, 2008.

[8] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard), Oct. 1996.

[9] A. Medina, M. Allman, and S. Floyd. Measuring the evolution of transport protocols in the internet. *SIGCOMM Comput. Commun. Rev.*, 35(2):37–52, 2005.

[10] J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFC 3168.

[11] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001.