Measuring Path MTU Discovery Behaviour

Matthew Luckie
Department of Computer Science
University of Waikato
Hamilton, New Zealand
mjl@wand.net.nz

Ben Stasiewicz
Department of Computer Science
University of Waikato
Hamilton, New Zealand
ben@wand.net.nz

ABSTRACT

Path MTU Discovery (PMTUD) is widely believed to be unreliable because of firewalls that discard ICMP "Packet Too Big" messages. This paper measures PMTUD behaviour for 50,000 popular websites and finds the failure rate in IPv4 is much less than previous studies. We measure the overall failure rate between 5% and 18%, depending on the MTU of the constraining link. We explore methods webserver operators are using to reduce their dependence on PMTUD, and find 11% limit themselves to sending packets no larger than 1380 bytes. We identify a number of common behaviours that seem to be software bugs rather than filtering by firewalls. If these are corrected PMTUD failures could be reduced by 63%. We further find the IPv6 failure rate is less than the IPv4 rate even with more scope for failure in IPv6.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Measurement

Keywords

TCP, Path MTU Discovery, IPv6, IPv4, middleboxes

1. INTRODUCTION

Path MTU Discovery (PMTUD) is widely believed to be unreliable because of firewalls that discard ICMP "Packet Too Big" (PTB) messages. Previous work by Medina, *et al.* in 2004 [1] reported 17% of the IPv4 webservers they tested failed PMTUD by not reducing their packet size when requested. They reason this is likely due to firewalls discarding PTB messages. IPv4 hosts may choose whether or not they use PMTUD, but IPv6 hosts do not have a choice and are required to act on PTB messages. This paper investigates PMTUD behaviour for IPv4 and IPv6 webservers.

IMC'10, November 1–3, 2010, Melbourne, Australia.

The PMTUD mechanism is similar for both IPv4 [2] and IPv6 [3]. When a router receives a packet that is too big for it to forward, it discards the packet and sends a PTB message telling the host the maximum transmission unit (MTU) it can forward. The host should then reduce its packet size to the size advised. Using the largest packet size that can be forwarded intact through a network is optimal, as "much of the cost of packetized communication is per-packet rather than per-byte" [4].

Unfortunately, there are many causes of PMTUD failure [5]. When PMTUD does not work, it causes Internet communications to fail in a way that is difficult to diagnose [6]. The most well-known cause is the firewall that discards all ICMP messages, including the PTB. Similarly, some routers are configured to not send ICMP destination unreachable messages; in IPv4, this also suppresses the PTB message. IPv6 separates PTB messages from the destination unreachable message so disabling destination unreachable messages should not also cause PTB messages to be suppressed. Finally, a router will not send a PTB message if it believes the network is capable of forwarding a larger packet than it actually can; this failure mode is known as an MTU mismatch.

Based on experiences with IPv4 and the IPv6 requirement for PTB messages to be delivered and acted on, there is concern PMTUD will fail more often when IPv6 is used, hampering end-user experience and thus IPv6 adoption. Some have recommended operators configure the MTU of interfaces on end-hosts to 1280 bytes [7, 8, 9]. The rationale behind this advice is the IPv6 minimum MTU is 1280 bytes so a host will never have to send a smaller packet or rely on a server accepting PTB messages. If this advice is taken, the host no longer has to use PMTUD as it will never send a packet larger than 1280 bytes, and it will also advertise a TCP Maximum Segment Size (MSS) that should cause the remote host to never send a packet larger than 1280 bytes over the TCP connection. This advice has the downside of not using the capabilities of paths which are able to forward larger packets. In addition, RFC 2460 requires a host to act on a PTB message with an MTU smaller than 1280 bytes by sending all packets with an IPv6 fragmentation header regardless of size [10].

This paper investigates PMTUD behaviour for IPv4 and IPv6 webservers. The method we use is similar to that used by Medina, *et al.* [1]. Our contributions are: (1) measurements of IPv4 webserver response to PTB messages containing different MTU values, (2) longitudinal measurement of IPv4 PMTUD behaviour from a packet trace, (3) a method

to measure IPv6 webserver PMTUD behaviour and comparison with IPv4 behaviour for those webservers, and (4) identification of apparent software flaws responsible for more than half of the failures. The rest of this paper begins with a survey of related work. Section 3 describes the methodology, the hosts surveyed, and the vantage points we use. Section 4 presents the data collected for a set of 50,000 popular websites reachable with IPv4, analysis of the effect of popularity on ICMP filtering behaviour, and comparison with the results of Medina, *et al.* Section 5 presents the data collected for all webservers with globally routable IPv6 addresses in the Alexa top 1 million list [11]. Finally, section 6 concludes.

2. RELATED WORK

Medina, *et al.* [1] characterised the TCP behaviour of webservers in 2004 using TBIT [12]. Overall, 17% of 81,776 targets they tested and 35% of the top 500 webservers as reported by Alexa at the time did not send smaller packets when requested to reduce their packet size to 256 bytes. They hypothesise this behaviour is caused by firewalls that discard PTB messages. We derive our methodology from theirs; we measure the response to PTB messages containing different MTU values, separately classify hosts that respond to a PTB message by clearing the DF bit, add reassembly support for fragmented packets, and add IPv6 support.

Luckie, *et al.* [13] characterised PMTUD failures for hosts that are able to send 9000 byte packets towards hosts that can only receive 1500 byte packets. In this situation, routers in the path are required to send PTB messages for PMTUD to work. In total, 30% of the paths tested to 136 hosts failed PMTUD. Many are caused by operators configuring their routers to not send ICMP destination unreachable messages, but some are also caused by MTU mismatches where a router believed it could forward larger packets than the network could carry. The tests in this work measure the ability of hosts to receive and act on PTB messages, rather than the ability of routers to send PTB messages.

The IETF PMTUD working group released an alternative PMTUD method known as Packetization Layer Path MTU Discovery (PLPMTUD) [14]. PLPMTUD does not rely on the delivery of PTB messages, but requires the use of a reliable transport protocol. For TCP, the PLPMTUD technique causes a large segment designed to test the network's ability to carry that packet-size to be inserted before a sequence of smaller segments that are known to be able to be forwarded. If the large segment is discarded because it is too big, the smaller segments that follow will cause duplicate acknowledgements to be returned by the remote TCP; the sender will then break the data into smaller segments and retransmit, avoiding a delay waiting for the TCP retransmission timer to be activated, and in future try a smaller maximum segment size. Implementation of PLPMTUD is complicated as it is difficult to determine if a large segment is lost due to congestion or MTU issues. The only deployed implementation is found in the Linux kernel from version 2.6.17, and it is off by default [15].

Microsoft Windows implements PMTUD blackhole detection for TCP, which is disabled by default [16]. If a system does not receive an acknowledgement after two retransmissions, it will either clear the DF bit, or lower its MSS value for the connection to 536 bytes, depending on the version of Windows. If acknowledgements are then forthcoming, it will send 536 byte packets for the lifetime of the connection.

3. METHOD

3.1 Measuring PMTUD Behaviour

We use an approach derived from the TCP Behaviour Inference Tool (TBIT) work in [1] to measure PMTUD behaviour. Briefly, PMTUD behaviour is inferred by the behaviour of a remote TCP after a PTB is sent asking it to send smaller packets. If the host reduces the packet size as requested, we infer the PTB message was received and acted on. If the host does not reduce the segment size and retransmits the data, another PTB is sent requesting it do so. We infer failure if the host retransmits the data in question three times and does not reduce the segment size or clear the DF bit.

3.1.1 An IPv4 PMTUD test

We test the PMTUD behaviour of webservers with four different MTU values; 1480, 1280, 576, and 256 bytes. A 1480 byte MTU corresponds to an IP tunnel over Ethernet, 1280 is a popular choice for tunnel software, 576 is the minimum size packet a host is required to reassemble, and 256 bytes is the MTU value tested in [1]. We send PTB messages if the DF bit is set in a packet larger than the MTU we are testing. We advertise an MSS of 1460 bytes to each webserver for all IPv4 tests.

If we do not receive any packet larger than the MTU value we are testing, we classify the test as such. If we receive data packets larger than the MTU value we are testing but the DF bit is not set, we classify these hosts as not attempting PMTUD. A host that supports PMTUD might react to a PTB message by clearing the DF bit in subsequent packets; we classify these hosts as clearing the DF bit. This behaviour is a defence against an adversary sending a PTB message with a small MTU which could then lead to the host sending many small data packets [17].

3.1.2 An IPv6 PMTUD test

An IPv6 host must accept and process PTB messages, but does not have to reduce its packet size below 1280 bytes [10]. Tunnelled connectivity is common in IPv6, so our ability to directly test PMTUD is limited if a tunnel with a 1280 byte MTU is in the path. Similarly, if the webserver sends segments that are too big for the network to carry to the vantage point and the webserver does not receive a PTB message, the vantage point will not see any data packets that can be used to test for PMTUD behaviour. Therefore we begin by advertising an MSS of 1440 bytes to the webserver, sending a PTB with an MTU of 1480 bytes if a data packet larger than this is received. Then, we test a connection with an MSS of 1380 bytes and an MTU of 1280 bytes. Finally, we test a connection with an MSS of 1220 bytes and an MTU of 576 bytes. For the last test, we expect subsequent packets to arrive with an IPv6 fragmentation header, even if the packet is smaller than 1280 bytes. The last test also allows us to infer PMTUD failure for earlier tests where no data packets are received as an MSS of 1220 bytes should be small enough in practice that packets could be carried.

3.2 Vantage Points

For our IPv4 tests we use two vantage points: a host located in San Diego, USA, and another in Hamilton, New Zealand. We use two hosts to reduce the chance that some aspect of the network they are hosted on or their service

	Hamilton, NZ, 2010				San Diego, USA, 2010				2004 [1]
MTU tested	1480	1280	576	256	1480	1280	576	256	256
No TCP connection	1.7%	2.4%	1.6%	1.5%	0.7%	1.5%	0.7%	0.6%	2.5%
Early TCP reset	0.4%	0.3%	0.3%	0.3%	0.4%	0.2%	0.2%	0.2%	0.6%
No data packets	0.2%	0.2%	0.2%	0.3%	0.2%	0.2%	0.2%	0.2%	–
Data packets too small	18.0%	6.1%	3.9%	0.3%	17.2%	5.2%	3.2%	0.1%	–
DF not set on data packets	1.5%	2.0%	2.2%	3.3%	1.5%	2.0%	2.2%	3.3%	29.6%
Cleared DF after PTB	1.6%	2.7%	4.5%	70.8%	1.6%	2.7%	4.6%	72.1%	–
PMTUD success	71.1%	77.4%	78.0%	5.2%	72.6%	79.1%	79.5%	5.3%	40.8%
PMTUD failure	5.7%	8.9%	9.4%	18.3%	6.0%	9.1%	9.5%	18.3%	17.3%

Table 1: Results of our experiments and the earlier work of Medina, *et al.* [1]. The portion of webservers that failed at PMTUD increases from 6% for an MTU of 1480 to 18% for an MTU of 256. Few webservers will reduce their packet size to 256 bytes; instead, most reduce their packet size to 552 bytes, and most send these packets with the DF bit cleared.

providers influencing the result; for example, by setting or clearing the DF bit on received packets, or by re-writing the MSS option contained in our SYN packets. For our IPv6 tests we use five vantage points in geographically and topologically diverse locations. We use the same San Diego vantage point, as well as hosts in Amsterdam, Tokyo, New York, and Christchurch, New Zealand. More vantage points are required for the IPv6 tests owing to the belief that tunnelled paths are widespread; because PMTUD relies on a tunnel sending a PTB message if it reduces the Path MTU, they provide an additional failure mode.

3.3 Targets

We derive our targets from the Alexa Top 1,000,000 websites list [11]. For our IPv4 tests, each vantage point independently derives its list of IP addresses to probe from the first 50,000 entries. We do this so the appropriate addresses for each vantage point will be tested if the location of the vantage point influences the addresses supplied. We test all IPv4 addresses the domain resolves to. We use wget [18] to determine a suitable URL to use in our tests. If the default page for the domain is at least 1600 bytes in size, we ask for it when we test PMTUD behaviour. Otherwise, we use wget to obtain all objects required to display the default page that are on that webserver, and select the first object that is at least 1600 bytes in size, or the largest object available. The San Diego vantage point tests the behaviour of 45,752 addresses, the Hamilton vantage point tests 45,990 addresses, and 54,008 addresses in total are tested with an intersection of 70% across 5,492 ASes.

For our IPv6 tests, we resolve the domains for the million websites in the list for IPv6 addresses. For those with IPv6 addresses, we resolve the domain for IPv4 addresses as well to enable PMTUD behaviour comparison for these dual-stacked webservers. The same 1,067 IPv6 addresses are tested by all five vantage points.

3.4 Implementation

We implemented a TBIT-style PMTUD test in scamper, a parallelised packet-prober [19]. For each measurement made our tool records, in a single data unit, meta-data about the test such as the URL, the server MSS seen and the MTU value used, as well as all packets sent and received for that test. It is simple to ensure inferences are valid given the packets recorded. We also implemented a driver to coordinate scamper's measurements. For each IP address, the driver begins by sending up to four ping packets to see if the network allows any ICMP through. Then, it does the sequence of PMTUD tests towards the webserver one at a time, testing the MTU values from highest to lowest. The driver waits at least one minute between PMTUD tests to each unique IP address to avoid being a nuisance.

Server MSS	Portion	Fail rate (1280)
1460	86.5%	6.7%
1380	10.8%	27.1%
1400	0.4%	6.8%
1414	0.3%	6.9%
536	0.2%	–
other	1.8%	–

Table 2: Top five server MSS values advertised and their corresponding failure rate. 10.8% of the population advertise an MSS of 1380, and these are four times more likely to fail at PMTUD.

4. IPV4 PMTUD BEHAVIOUR

Table 1 shows the overall results of our IPv4 tests, with the final column containing corresponding classifications from [1]. We see a similar failure rate in 2010 for the tests with a 256 byte MTU as was seen in the 2004 study. However, the failure rate is much lower for the other MTU values tested. Compared with the 2004 study we measure a success rate of 78–80% which is nearly double the 2004 result, and the number of servers we measure that do not set the DF bit on data packets is nearly an order of magnitude less than the 2004 result. We believe the results are different in the 2004 study because of a bug in TBIT that classifies hosts that clear the DF bit as not attempting PMTUD, and because of the choice of 256 as the MTU value to use in PTB messages. Figure 1 shows the fraction of webservers that always set the DF bit in a long-term packet header trace collected at the border of the University of Waikato network [20]. In 2004, 94% of webservers always set the DF bit, and is independent of the website population changing when students are on campus. This is consistent with the behaviour of most operating systems to enable PMTUD by default.

The proportions of classifications are similar for the two vantage points, so the following analyses will use the data obtained from Hamilton. The proportion of webservers that

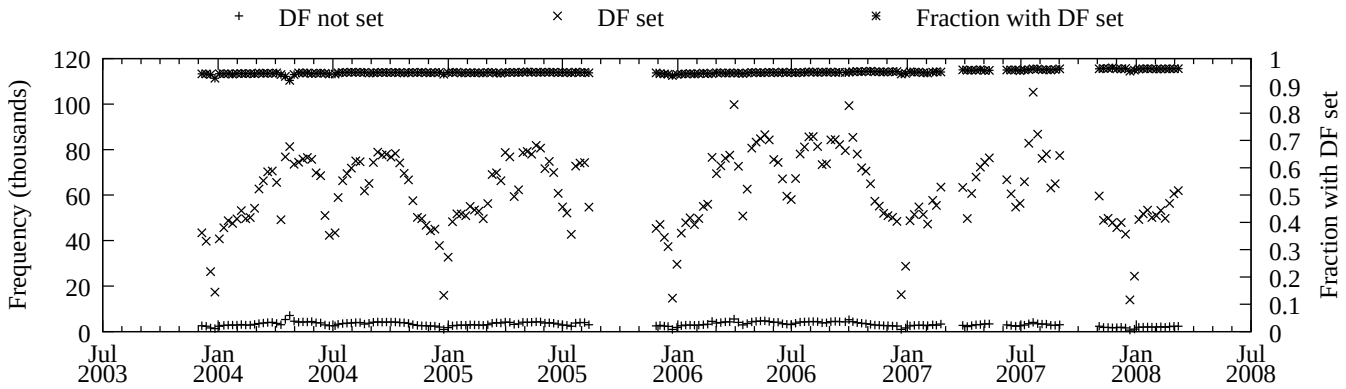

Figure 1: IPv4 webservers that send data packets with the DF bit set over time as measured by packet traces at the edge of the University of Waikato network. Each point represents one week. The bottom series is the number of webservers that never set the DF bit on any data packet. The middle series is the number of webservers that set the DF bit on all data packets. The top series is the fraction of these webservers that always set the DF bit; the fraction grows from 94% in 2004 to 96% by 2008.

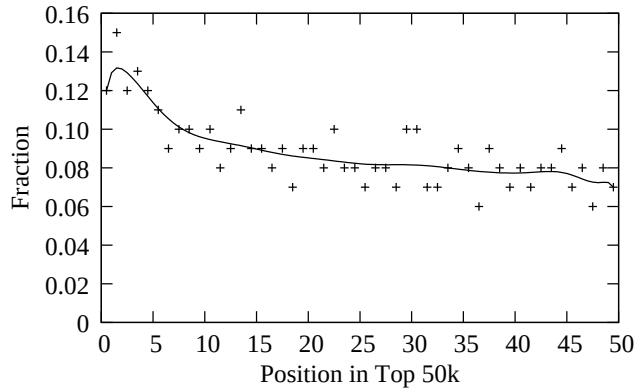

Figure 2: Fraction of IPv4 webservers that fail at PMTUD with a 1280 byte MTU in buckets of 1000, by popularity. The line is a Bézier curve showing the trend that the more popular a webserver is the more likely it is to fail at PMTUD.

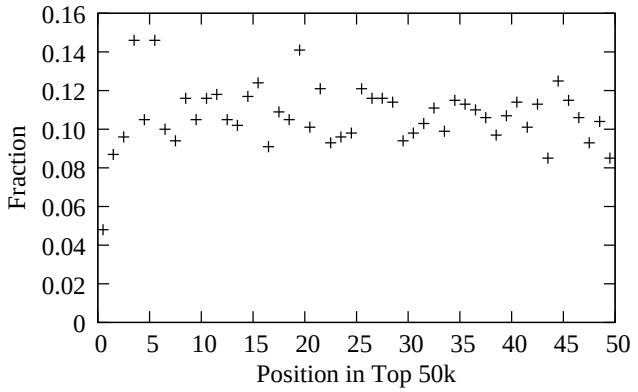

Figure 3: Fraction of IPv4 webservers that advertise an MSS of 1380, ordered by popularity in buckets of 1000. There is no correlation evident between the popularity of a webserver and the advertisement of 1380.

do not send packets large enough to allow for a test using an MTU of 1480 is 18%, compared with 6% for an MTU of 1280. Table 2 shows why: 10.8% of servers in our population advertise an MSS of 1380 bytes and send packets no larger than 1420 bytes. This MSS is associated with middleboxes that clamp the MSS of TCP connections they pass to avoid PMTUD problems when the client is behind a tunnel. Table 2 also shows the PMTUD fail rate associated with these servers, which is 4 times higher than the fail rate of the other MSS values listed. It seems worthwhile to identify the manufacturers of the middleboxes involved to determine if there is a bug involved, as if these webservers had a failure rate of 6.7% like most of the rest of the population the overall failure rate would drop 25%.

Figure 2 shows how the popularity of a webserver corresponds with the probability it will fail PMTUD with an MTU of 1280 bytes. As with Medina [1], we find the most popular webservers are much more likely to filter PTB messages than webservers that are less popular, but the failure rate for the most popular 500 we measure is half the rate reported in 2004. Because servers that advertise an MSS of 1380 are more likely to fail at PMTUD, we wondered if this was more likely to be a feature of the popular webservers. Figure 3 shows there is no correlation evident; in fact the most popular 1000 webservers have the lowest fraction of 1380 MSS.

As noted in section 3.4, before we commenced each PMTUD test, we tested the webserver's responsiveness to ping. Overall, 85% of the webservers that succeeded at PMTUD are responsive, but only 33% of those which failed PMTUD are, suggesting the presence of firewalls configured to deny by default. Of the systems that did not set the DF bit on any data packet, 50% are responsive, indicating a deliberate choice to disable PMTUD because of the presence of a firewall. Figure 4 shows how the popularity of a webserver influences the probability it will be responsive to ping. More popular webservers are more likely to be unresponsive.

Figure 5 shows the size of a segment observed after a PTB message is sent regardless of the PMTUD classification made. For the 576, 1280, and 1480 MTU tests, at least

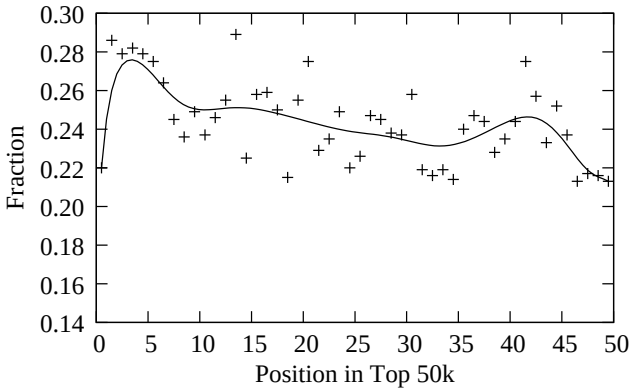

Figure 4: Fraction of IPv4 webservers that are not responsive to ping, ordered by popularity in buckets of 1000. The line is a Bézier curve showing the trend that the more popular webservers are more likely to be unresponsive to ping.

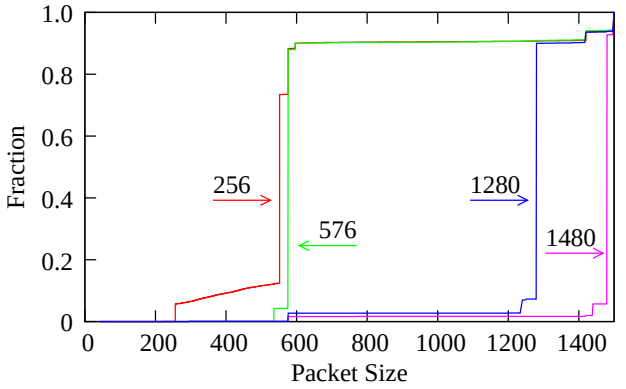

Figure 5: The subsequent size of a segment after a PTB message is sent, for each MTU tested. Most webservers behave as expected and reduce their packet size to the size advised, although most webservers will not reduce their packet size below 552 bytes if asked.

70% of the population behave as expected and reduce their packet size to the MTU advised. 5% reduce their packet size 40 bytes further, perhaps to ensure they send packets that will easily clear the link concerned. Few servers will reduce their packet size below 552 bytes in response to a PTB with MTU 256. Instead, most clear the DF bit to allow packets to be fragmented. The TBIT tool used in [1] classifies these systems as not attempting PMTUD, though it is clear from tests with other MTUs that most will attempt and successfully complete PMTUD. Systems that reduce their packet size to 552 bytes but do not clear the DF bit are failing PMTUD due to a bug in the operating system of the webserver; FreeBSD is one operating system with this bug [21]. Correcting these bugs will cut the failure rate for MTU 256 in half. The choice of 256 as an MTU value is not optimal for measuring the impact of middleboxes filtering PTB messages, because (1) most operating systems disable PMTUD when a PTB with this MTU is received, and (2) some reduce their packet size to a threshold but do not clear the DF bit.

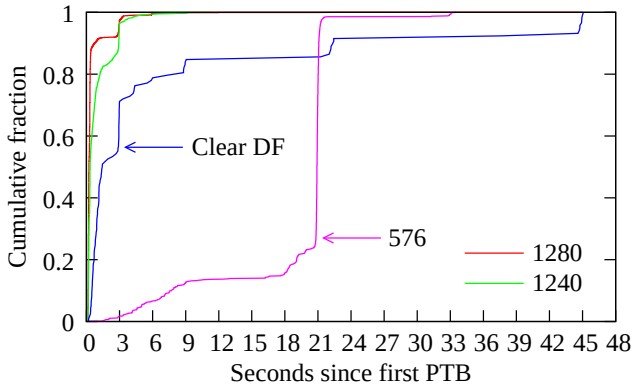

Figure 6: The delay between the first PTB sent and the action by the host. The delays measured in reducing to 576 bytes suggest the reductions are due to the hosts filtering PTB messages but having PMTUD blackhole detection enabled.

For the 1280 MTU test, 2.7% reduce their segment size to 576 bytes, and 91% of these reductions also clear the DF bit. This behaviour indicates the use of blackhole detection by the webserver, such as that done by Windows [16], in the presence of a firewall that discards PTB messages. Figure 6 shows the delay between the first PTB sent and the webserver transmitting packets that will pass a link with a 1280 byte MTU. 91% of the webservers that reduce their packet size to 1280 do so within one second. In addition, 94% of all webservers that appear to take action based on receiving a PTB message also do so within three seconds. However, 77% of the webservers that reduce their packet size to 576 do so at least 20 seconds after the first PTB is sent, and are likely to be Windows hosts that have inferred a PMTUD blackhole; 74% of these webservers are unresponsive to ping. The Clear DF line represents 118 webservers that cleared the DF bit but did not reduce their packet size so it would clear a link with a 1280 byte MTU. 71% of these webservers respond within three seconds, though the tail is long, with 8% waiting 43 seconds to take their action.

We searched the data for evidence of other webservers using techniques to avoid PMTUD failures if they do not receive a PTB message when they should. Linux implements two techniques, neither of which are enabled by default. The first technique begins by sending large packets, and if the first large packet is lost twice and no PTB message is received, then after nine seconds the host will revert to sending 552 byte packets with the DF bit cleared. The second technique is PLPMTUD [14]; Linux begins by sending 552 byte packets by default, and after the window size has grown to 11 segments it will try sending a 1064 byte segment with the DF bit set. MacOS since 10.6 also implements blackhole detection; it begins by sending a small segment followed by a flight of several large packets. If the large packets do not cause acknowledgements, and the retransmission of the first large packet is not acknowledged either, it will lower its packet size to 1240 bytes and clear the DF bit. We observed no evidence of any of these in our data, though the tests are unlikely to find evidence of the first or third techniques unless a middlebox discards our PTB messages before they reach the destination.

	Tokyo			New York		San Diego		Amsterdam		Christchurch	
MTU tested	1480	1280	576	1480	1280	1480	1280	1480	1280	1480	1280
Early TCP reset	0.1%	0.1%	nil	0.1%	0.1%	0.2%	0.1%	0.2%	0.1%	0.2%	0.1%
No data packets	0.3%	0.1%	0.1%	0.1%	nil	0.3%	nil	0.6%	0.4%	0.6%	0.2%
Too small	35.9%	16.8%	–	32.9%	17.0%	32.7%	17.6%	33.4%	17.2%	34.2%	17.3%
PMTUD success	62.7%	80.5%	44.1%	66.0%	80.7%	65.9%	80.1%	64.9%	80.0%	64.1%	79.9%
PMTUD failure	1.0%	2.4%	55.7%	0.9%	2.2%	0.9%	2.2%	1.0%	2.2%	0.9%	2.4%

Table 3: Results of our IPv6 experiments. The results across vantage points are similar. At least 80% of the webservers we measure will reduce their packet size when asked, but less than half will include a fragmentation header as RFC 2460 requires when the MTU supplied is 576 bytes.

Server MSS	Portion	Fail rate (1280)
1440	83.6%	1.6%
1220	5.3%	–
1420	5.0%	2.2%
1380	3.8%	23.5%
other	1.8%	–

Table 4: Top four MSS values advertised for IPv6 and their corresponding failure rate. As with IPv4, IPv6 systems that advertise an MSS of 1380 are much more likely to fail PMTUD.

5. IPV6 PMTUD BEHAVIOUR

As noted in section 3.3, there are only 1,067 unique IPv6 addresses that are globally routable in the Alexa list of the top million websites [11]. We could not establish a connection to 15%, which we exclude from the following results so the summary statistics for PMTUD behaviour are not underestimated. However, the results are encouraging for the 910 websites for which we could test PMTUD behaviour.

Table 3 shows the results of our IPv6 experiments. The proportion of servers that fail at PMTUD for 1280 bytes is 2.4%. The corresponding failure rate for the IPv4 servers for this population is 4.4%. This may be due to fewer webservers having an IPv6 firewall enabled where they currently have an IPv4 firewall. As with the IPv4 tests, we assessed the responsiveness of each IPv6 webserver to ping. For the systems where PMTUD is successful 96% respond to ping, but for the failures 50% do so. Table 4 shows the top four server MSS values seen, and their corresponding failure rates. As with IPv4, systems that advertise an MSS of 1380 are much more likely to fail PMTUD.

Less than half of the webservers successfully complete PMTUD when sent a PTB message with an MTU of 576. In this scenario, we expect to receive all packets with an IPv6 fragmentation header even if the packets are not sent in fragments. This restricts our ability to directly test if a webserver receives and acts on a PTB message if it sends packets too small to test with an MTU of 1280 bytes. Table 3 contains results for Tokyo; we saw nearly identical results and behaviour with the other vantage points. We did not observe many TCP retransmissions in these tests; 72% of the failures saw just one retransmission, usually at least 45 seconds after the PTB was sent, before the TCP connection timed out. It is possible that another software bug is behind this result; some firewalls discard IPv6 packets with a fragmentation header if they are not part of a fragmented series [22], even if their only rule is to accept all IPv6 packets.

For 32% of 1480 MTU tests and 17% of 1280 MTU tests, we are not able to measure PMTUD behaviour because the packets we receive are too small. Because this figure is relatively large compared with the IPv4 population of section 4, we examined the packet traces for further evidence of PMTUD behaviour related to tunnelled IPv6 connectivity between our vantage points and the webservers. We filtered TCP connections where the maximum packet size received was less than the maximum packet size suggested by the server's MSS advertisement, and then looked for clusters of maximum packet size values. For all vantage points, we see clusters at 1480 and 1280, and for Tokyo we also see a cluster at 1476 corresponding to GRE [23]. However, the proportion of hosts that send packets smaller than their MSS is only 20%, and these clusters account for half of the maximum packet sizes we see. This indicates 10% of TCP sessions in our tests were carried over a tunnel, suggesting that tunnelled IPv6 connectivity is not very prevalent. This result is consistent with other measurements of the prevalence of IPv6 tunnelling [24], which have shown the percentage of tunnelled IPv6 reducing from 40% in 2004 to 10% in 2010.

6. CONCLUSION

Path MTU Discovery is a relatively simple algorithm to implement, but it depends on packet too big messages being generated by routers, forwarded by middleboxes, and correct action being taken by end-hosts. This paper shows the proportion of webservers that receive and correctly act on PTB messages is at least 80% for both IPv4 and IPv6. We identified two strategies to significantly cut the IPv4 failure rate. First, operating systems that refuse to lower their packet size below a threshold must be modified to not set the DF bit if the Path MTU is lower than the threshold. Second, the middleboxes that rewrite the TCP MSS to 1380 bytes should be debugged to ensure they correctly forward PTB messages, if it is possible to identify their manufacturers. A third strategy, which is arguably the most challenging, is to educate system administrators about the necessity of forwarding PTB messages.

Acknowledgements

This work was supported by New Zealand Foundation for Research Science and Technology (FRST) contract UOWX0705. The following provided data or access to hosts to conduct experiments from: Emile Aben (RIPE NCC), Daniel Andersen (CAIDA), Kenjiro Cho (IIJ), Bill Owens (NYSERNet) and Bill Walker (SNAP Internet).

7. REFERENCES

[1] Alberto Medina, Mark Allman, and Sally Floyd. Measuring interactions between transport protocols and middleboxes. In *IMC '04*, pages 336–341, Taormina, Sicily, Italy, October 2004.

[2] J. Mogul and S. Deering. Path MTU Discovery. RFC 1191, IETF, November 1990.

[3] J. McCann, S. Deering, and J. Mogul. Path MTU Discovery for IP version 6. RFC 1981, IETF, August 1996.

[4] C.A. Kent and J.C. Mogul. Fragmentation considered harmful. *ACM SIGCOMM Computer Communication Review*, 17(5):390–401, 1987.

[5] K. Lahey. TCP problems with path MTU discovery. RFC 2923, IETF, September 2000.

[6] Richard van den Berg and Phil Dibowitz. Over-zealous security administrators are breaking the Internet. In *Proceedings of LISA '02: Sixteenth Systems Administration Conference*, pages 213–218, Berkeley, CA, November 2002.

[7] Geoff Huston. A tale of two protocols: IPv4, IPv6, MTUs and fragmentation. `http://www.potaroo.net/ispcol/2009-01/mtu6.html`.

[8] Geoff Huston. Mutterings on MTUs. `http://www.potaroo.net/ispcol/2009-02/mtu.html`.

[9] Iljitsch van Beijnum. IPv6 and path MTU discovery black holes. `http://www.bgpexpert.com/article.php?article=117`.

[10] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. RFC 2460, IETF, December 1998.

[11] Alexa. Top 1,000,000 sites. `http://www.alexa.com/topsites`.

[12] TBIT: TCP behaviour inference tool. `http://www.icir.org/tbit/`.

[13] Matthew Luckie, Kenjiro Cho, and Bill Owens. Inferring and debugging path MTU discovery failures. In *IMC '05*, pages 193–198, San Francisco, CA, USA, October 2005.

[14] M. Mathis and J. Heffner. Packetization layer path MTU discovery. RFC 4821, IETF, March 2007.

[15] Matt Mathis. Raising the Internet MTU. `http://staff.psc.edu/mathis/MTU/`.

[16] Microsoft. Documentation for EnablePMTUDBHDetect registry key.

[17] F. Gont. ICMP attacks against TCP. RFC 5927, IETF, July 2010.

[18] Wget 1.12. `http://www.gnu.org/software/wget/`.

[19] Matthew Luckie. Scamper. `http://www.wand.net.nz/scamper/`.

[20] WITS: Waikato Internet traffic storage. `http://www.wand.net.nz/wits/waikato/`.

[21] Matthew Luckie. kern/146628: [patch] TCP does not clear DF when MTU is below a threshold. `http://www.freebsd.org/cgi/query-pr.cgi?pr=146628`.

[22] Matthew Luckie. kern/145733: [patch] ipfw flaws with IPv6 fragments. `http://www.freebsd.org/cgi/query-pr.cgi?pr=145733`.

[23] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic routing encapsulation (GRE). RFC 1701, IETF, October 1994.

[24] RIPE Labs. Untunneling IPv6. `http://labs.ripe.net/content/untunneling-ipv6`.