UNIVERSITY OF AMSTERDAM

MSC. SYSTEMS & NETWORK ENGINEERING

Discovering Path MTU black holes on the Internet using RIPE Atlas

Commissioned by:

NLnet Labs

Authors:

Maikel DE BOER
Maikel.deBoer@OS3.nl

Jeffrey BOSMA
Jeffrey.Bosma@OS3.nl

Supervisors:

Benno OVEREINDER
Benno@NLnetLabs.nl

Willem TOOROP
Willem@NLnetLabs.nl

July 23, 2012

This page is intentionally left blank.

Abstract

The Internet as we know it today is a complex system that works because of the many different protocols standardized by the Internet Engineering Task Force (IETF) defined in Request for Comments (RFC). If people intentionally or unintentionally do not comply with these standards problems are inevitable.

Due to some network administrators that have configured their firewalls to filter Internet Control Message Protocol (ICMP) Packet Too Big (PTB) packets and Internet Protocol (IP) fragments and some Customer-premises equipment (CPE) devices which does this by default, several of the protocols on the Internet do not work as they are supposed to do. By filtering ICMP PTB packets and IP fragments, Path MTU (PMTU) black holes can occur. The effect of these particular black holes in a networked path is that packets, that are larger than the smallest link can assimilate, are forever lost because of the Maximum Transmission Unit (MTU) of this link. Such an event results in a situation where a typical end user thinks is caused by outage of their application, the remote service, or the connection in between.

To determine the scale of the problem and where it occurs we have build an experimental setup and in our experiments use RIPE Atlas as a worldwide measurement. We have conducted several different experiments for nine periods of four hours using an average of between 1134 and 1365 Internet Protocol version 4 (IPv4) vantage points and an average of between 397 and 502 Internet Protocol version 6 (IPv6) vantage points.

We observed that for IPv4 between 4% and 6% of the paths between the vantage points and our experimental setup filter ICMP PTB packets. For IPv6 this was between 0.77% and 1.07%. Furthermore, we found that when IPv4 Domain Name System (DNS) servers do not act on the receipt of ICMP PTB packets, between 11% and 14% of the answers from these DNS servers are lost. For IPv6 DNS servers this was between 40% and 42%. Lastly, we found that for IPv4 approximately 6% of the paths between the vantage points and our experimental setup filter IP fragments. For IPv6 this was approximately 10%. Both ICMP PTB packet and IP fragment filtering seems to be happening close to the end users and not in or near the core of the Internet.

The amount of ICMP PTB packet filtering is larger in IPv4 than in IPv6, although it is not likely many users will notice any problems because of this. The results for IP fragment filtering in IPv6 are more troublesome since it is likely that protocols like Domain Name System Security Extensions (DNSSEC) are not going to work as smoothly on the Internet as it exists today. Luckily it looks like the problems do not occur in or near the core of the Internet. This means that there is a high probability that everyone would be able to fix their own networks if these are not configured correctly.

Acknowledgements

We, Maikel de Boer and Jeffrey Bosma, enjoyed working together with several people that helped us to achieve the results that are described in this thesis.

We are grateful to Benno Overeinder and Willem Toorop from NLnet Labs for their their insight, suggestions, feedback and (technical) support. We would also like to thank Olaf Kolkman, the director of NLnet Labs, for providing us with the opportunity to work on this project.

Furthermore, we thank Vesna Manojlovic, Philip Homburg, Andreas Strikos and Emile Aben from RIPE NCC for their helpful and much appreciated work. Without them the project would not have been possible.

Contents

List of Figures

Listings

List of Tables

1 Introduction

With the ever increasing number of nodes on the Internet, the Internet Protocol version 4 (IPv4) is pushed more and more towards its addressing capability limits. This limit restricts the number of on-line users that can be connected without the use of translation mechanisms such as Network Address Translation (NAT). The use of such translation mechanisms in, e.g. the core of the Internet, adversely affects the growth of the Internet as a whole. The Internet Protocol version 6 (IPv6) is intended to be the next generation Internet protocol, succeeding the more than 30 years old IPv4 because of its deficiencies. The main limitation is the number of addressable nodes. With IPv4, a 32-bit addressing space is available which allows for the addressing of 4.294.967.296 individual nodes. Certain parts of this address space cannot be used because of predefined purposes ranging from local private networks to complete IP-blocks reserved for experimental or future usage. In the IPv6 address space several parts are also marked as unusable. However, IPv6 defines a 128-bit address space and therefore it is possible to address 2^{128} nodes, leaving an immense amount of usable addresses.

Now that the pool of unallocated IPv4 addresses is exhausting, or is even already entirely depleted for various regions, an increasing number of organizations are pushed towards the use of IPv6. Unfortunately, the transition from IPv4 to IPv6 is not at all considered a smooth deployment and has given rise to anomalies in IPv6-enabled services in the Internet. Some of these anomalies are unique to IPv6 regardless of its implementation, however, others occur due to a specific implementation of IPv6 and may as well occur with IPv4. The focus of this research is on Path MTU (PMTU) black holes, a type of anomalies that occur in both the IPv4 and IPv6-driven Internet.

1.1 Path MTU black holes

The Internet consists of millions of networks that are interconnected with links. The effect of a black hole in the Internet, on packets that flow along a networked path of links between a sender and receiver, is that these packets are forever lost. In the case of PMTU black holes only packets that are larger than the PMTU are lost. The PMTU is determined by the link with the smallest Maximum Transmission Unit (MTU) on the path. This particular link forms the bottleneck and will limit the maximum packet size that the entire path between a sender and receiver is able to assimilate.

There are two possible causes for the occurrence of PMTU black holes in the Internet. The first cause is the filtering of important signalling packets by intermediate nodes on the path. In the event that these nodes run a firewall, they could potentially be configured to disallow all or certain types of Internet Control Message Protocol (ICMP) packets to pass through. The effect of this filtering is the inoperability of Path MTU Discovery (PMTUD) in cases where it is required for a successful flow of packets along a path, i.e. if packets are larger than the link with the smallest MTU can assimilate. In such cases failure of PMTUD will lead to PMTU black holes.

Assuming PMTUD is done successfully and payload data is split and distributed over multiple smaller packets according to the discovered PMTU, successful delivery is not guaranteed even if the underlying network infrastructure would provide completely reliable data transfers. With IPv6 only a sender and receiver, that both act as absolute endpoints in a path, are allowed to split packets into what is known as fragments. Intermediate nodes are never allowed to do so. The same is true for IPv4 packets if and only if this is explicitly indicated in the packet's header. Due to these standardized restrictions, as well as the information available in the header of a fragment, intermediate nodes running firewalls will have to keep track of every fragment belonging to a split packet up until the final fragment. Only then the decision can be made to either let all fragments pass through or to filter them. Since keeping track of all a packet's fragments requires the use of system resources, some network administrators decide to simply filter all fragments to avoid potential Denial of Service (DoS) attacks that try to exploit the limited availability of these resources. The filtering of fragments is the second possible cause for PMTU black holes.

1.2 Research objective

PMTU black holes have been the subject of research in the work of various other researchers, as described in Chapter 2. In our research we will extent the work done in previous studies and explore new areas in the context of PMTU black holes on the Internet. The added value of our research lies in the ability to perform experiments on a worldwide scale by using a much larger number of vantage points. Because of the large amount of vantage points, as well as the ability to retrieve important feedback from each vantage point, we can more accurately pinpoint where on the Internet PMTU black holes occur and determine the type of filtering. Moreover, we can also measure to what extent this affects the functioning of the IPv4 and IPv6-driven Internet, and are able to make a comparison between both.

Ultimately, the goal of this research is to create awareness of PMTU black holes and the connectivity problems that they cause. This awareness in turn could prove to be useful in the prevention of these black holes which will, in the long run, result in a more functional and higher quality Internet.

1.3 Research questions

PMTU black holes remain a relevant problem on the Internet for both IPv4 and IPv6, even though being the subject of various previous studies and well described in literature. The objective for our research, as described in Section 1.2, has resulted in the following main research question:

- *Where on the Internet do Path MTU black holes occur?*

We also define the following sub-research question as a complement to the main question above:

- *Do Path MTU black holes occur more often on the IPv6-Internet compared to IPv4?*

1.4 Thesis outline

The structure of the remainder of this thesis is as follows. Chapter 2 gives an overview of the research works related to PMTU black holes. Chapter 3 introduces the concepts and background of the research. In Chapter 4 the experiments for the research are introduced. Chapter 5 gives an overview of the results for the performed experiments. In Chapter 6 the results of our experiments are discussed. Chapter 7 presents the conclusion. In Chapter 8 a discussion on future work is presented. Finally, Chapter 9 gives several recommendations for the mitigation of PMTU black holes.

2 Related work

In the year 2009, Geoff Huston, Chief Scientist at the Asia-Pacific Network Information Centre (APNIC), wrote on his blog about a problem he experienced while trying to retrieve a document from the website `www.rfc-editor.org` [1]. After extensive analysis of the problem he found that relatively small packets, like those used in the three-way handshake of the Transmission Control Protocol (TCP), could successfully travel between the web server and his laptop but that this was not the case for larger packets. He discovered that this behavior was caused by an improper functioning of PMTUD. He states that some firewalls treat ICMP packets as potentially hostile and prevent their traversal. The overzealous network administrators that have configured these kinds of access policies therefore unintentionally cripple PMTUD. The technicalities of PMTUD are well-defined in Request for Comments (RFC) 1981 [2] and will also conceptually be discussed in Chapter 3.

Besides Geoff Huston, PMTU black holes have been the subject of research by other Internet engineers and researchers. Particularly the research done by Matthew Luckie and Ben Stasiewicz [3] shares some commonalities with the research presented in this thesis. In their research they measure the behavior of PMTUD for 50,000 popular websites and explore the methods that operators of these websites use to reduce their dependence on PMTUD. For their measurements they used a total of five vantage points in geographically and topologically diverse locations. In our research we will be using many more vantage points, as described in Section 4.1, and in addition to measurements of certain ICMP packet filtering we will also measure the filtering of fragments on the Internet.

Matthew Luckie also participated in another study were he and other researchers inferred and debugged PMTUD failures [4]. They provided a review of modern PMTUD failure modes and present a tool designed to help network operators and users infer the location of a failure.

The problems with PMTUD are also described in RFC 2923 [5]. Solutions for failing PMTUD are described in RFC 4821 through the use of Packetization Layer PMTUD [6]. Recommendations for the filtering of ICMP packets are described in [7] where they present guidelines to prevent the rise of PMTU black holes due to this type of filtering.

Lastly, research was done by Lorenzo Colitti et al. regarding a number of techniques to detect and collect information about IPv6-in-IPv4 tunnels [8]. These tunnels play an important role in the migration to IPv6 for users that rely on IPv6-in-IPv4 tunnels when native IPv6 connectivity is not available. Detection of these tunnels is crucial because of their effect on the PMTU.

3 Background and concepts

3.1 Terminology

An understanding of the way the terminology is used in this and successive chapters is especially useful for those without prior insight. We will therefore provide several terminological definitions in order to elucidate the use of these terms.

Regarding the use of MTU, whenever we refer to a Maximum Transmission Unit, we mean the MTU of an interface that is connected to a link. The MTU of this interface will determine the MTU of the entire link, regardless of the MTU configured at the interface on the other side. From now, we will refer to this as the link MTU. When multiple of these links are connected together the smallest MTU will determine the MTU of the entire path. We will refer to the MTU of an entire path as the PMTU.

For the use of Internet Protocol (IP), regardless of whether it is used in the context of IPv4 or IPv6, we will always refer to it as IP. We will only specifically use IPv4 or IPv6 if the context does not provide sufficient means to be able to distinguish between IPv4 and IPv6, and if this differentiation is significantly important in the context.

Similarly as with IP, the use of Internet Control Message Protocol version 4 (ICMPv4), specific for IPv4, and Internet Control Message Protocol version 6 (ICMPv6), specific for IPv6, is prevented unless these terms are contextually important. Otherwise, we will simply refer to ICMP.

Lastly, the use of the words fragmentation and segmentation both indicate the splitting of payload data contained in a packet into smaller pieces such that it is distributed over multiple packets. However, segmentation happens at the transport layer, also known as the fourth layer, of the Open Systems Interconnection (OSI) model. The packets carrying the segmented data will have both a network layer and a transport layer header. For TCP this type of packets are called TCP segments. Segmentation is done by a sender and receiver that participate in the TCP session of the corresponding TCP segments. Fragmentation differs from segmentation in that it is done at a lower layer of the OSI model, namely the third layer that is also known as the network layer. As a result of this all packets carrying the fragmented data, except for the first packet, will not have a transport layer header. This is the case for fragmented User Datagram Protocol (UDP) packets. These packets without a transport layer header, better known as IP fragments, will therefore lack information such as port numbers, in contrary to TCP segments. Furthermore for IPv4, fragmentation can in some cases also be done by intermediate nodes.

3.2 MTU and MSS

In addition to the MTU, the Maximum Segment Size (MSS) of TCP segments also fulfils an important role in our research. Although they relate to each other, they do not have the same meaning.

The MTU is a limit on the maximum amount of data that a Network Interface Card (NIC) can transmit and receive in a single packet. The MTU is often expressed in bytes. The use of IPv4 defines 68 bytes as an absolute minimum MTU, however, any Internet destination must be able to receive a packet of 576 bytes either in one piece or in IP fragments for later reassembly [9]. The use of IPv6 defines a much larger MTU of 1280 bytes as the bare minimum. While for both IPv4 and IPv6 the commonly used MTU in the Internet is 1500 bytes, a variety of other MTUs are being used as well. One of the reasons for the MTU not being identical for all nodes in the Internet is the use of tunnels to encapsulate network traffic. An example are the IPv6-in-IPv4 tunnels that can be used as a transition mechanism for users who are not offered native IPv6 Internet access by their Internet Service Provider (ISP). Tunnelling the other way around functionality wise is also possible, e.g. with IPv4-in-IPv6 tunnels so that native IPv6-only users are still capable of reaching IPv4 nodes on the Internet [8]. These encapsulation mechanisms all share a common property, that is, to encapsulate traffic, a tunnelling protocol adds additional information to the original packet in order to properly interpret and process them at the tunnel endpoint. Naturally, this additional information decreases the available space for payload data.

An MSS is the maximum size for a TCP segment that a node is willing to receive. The MSS is always advertised at the beginning of a TCP session establishment, i.e. in the first two packets of the TCP three-way handshake. The MSS is used to restrict the amount of payload data that is transmitted in packets. When two communicating nodes have advertised their preferred MSS, the lowest preferred MSS is used by both nodes for the entire duration of the TCP session. In IPv4, if a node has an MTU of 1500 bytes, the advertised MSS will be 1460 bytes. The 40 bytes that are missing are due to the space that the headers of IPv4 and TCP take up. In the case of IPv6 an MSS of 1440 bytes is advertised due to the larger size of the IPv6 header.

3.3 PMTUD

As described in Section 3.2, each node on the Internet can potentially configure any MTU on their NIC as long as it conforms to the standardized IP minimum. Consider two nodes on the Internet that want to share data belonging to a set of images. The amount of data that they can pass at once in a single packet not only depends on their own MTU, but also on that of every intermediate node, such as a router, in the path. In order to have a successful flow of packets between the two nodes it is crucial that they do not put more data in a packet than the routers in the path can convey. For an efficiently as possible packet flow this would require both nodes to put as much data as possible in a single packet that the path would still be able to convey. However, because the PMTU, the smallest MTU on the path, is not know beforehand, a detection mechanism is required.

Otherwise, for a successful but less efficient packet flow, the two nodes could simply transmit packets of 576 bytes or 1280 bytes for IPv4 or IPv6, respectively. However, this would require more packets to convey all the data and therefore increase the network protocol overhead. Minimizing the inefficiencies of additional protocol overhead is not the only purpose of having a large as possible packet size. The use of small packets could potentially adversely impact the performance of network components and therefore also downgrade the throughput performance. The underlying reason is simple, namely that the component's processor is more often interrupted relative to what would have been the case if larger packets were used.

PMTUD is a PMTU detection mechanism to discover the PMTU of a path between two nodes, resulting in an efficiently as possible packet flow [10]. For its operation it utilizes a certain ICMP packet type known as an ICMP Packet Too Big (PTB). These packets are used to signal the sending node when it is transmitting a packet that is too large for a router to pass on. Figure 1 illustrates an example of how PMTUD operates during such an event. Note that this figure depicts a slightly simplified example of the actual behavior of TCP that may be different depending on other environmental variables. Furthermore, we assume that in the case of IPv4 all packet headers indicate that fragmentation by the routers in the path is prohibited.

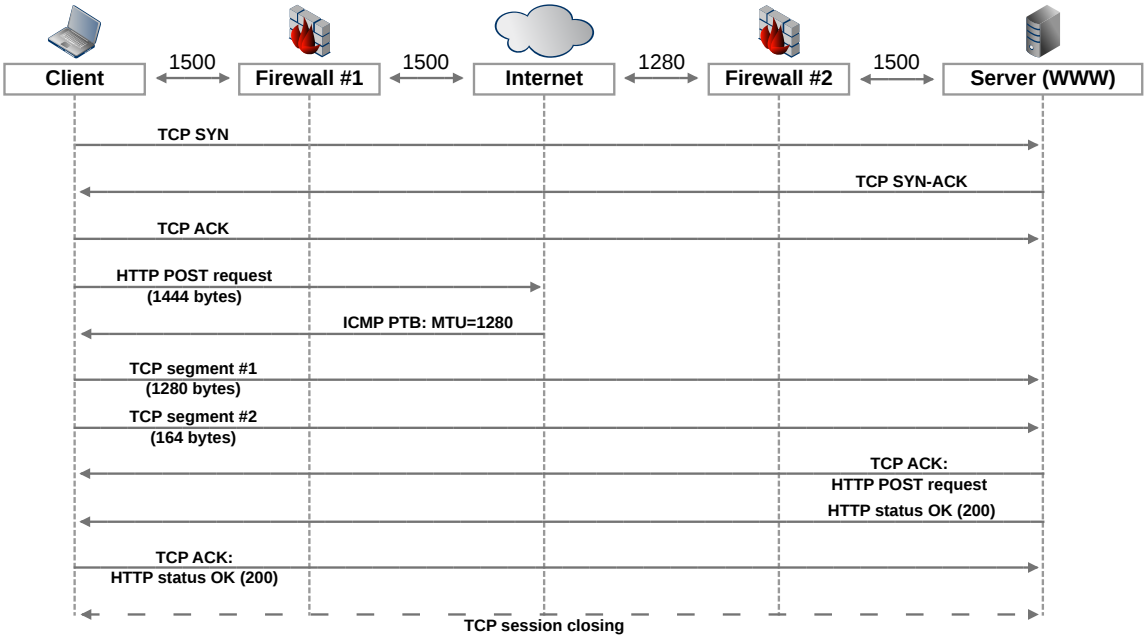

Figure 1: PMTUD: successful HTTP POST - MTU: 1280

The client, on the far left, wants to send a filled in web form with the Hypertext Transfer Protocol (HTTP) to the web server, on the far right. Both are connected to the Internet through a firewall that forwards traffic to and from the Internet. Note that the PMTU between the client and the web server is 1280 bytes because of the link MTU between the Internet and the second firewall.

The first three packets that go back and forth are part of the TCP three-way handshake that is done to establish a TCP session. The initiation of a TCP session is done with packets that are small and will always be able to be forwarded by all routers in the path. After the session has been established, the client will transmit a packet of 1444 bytes, containing the web form data, in a HTTP POST request. Obviously, this packet will never make it to the web server since it is too large for the path to convey. In response to this event the router, connected to the link that has a link MTU of 1280 bytes, will send an ICMP PTB packet back to the client. The ICMP PTB packet indicates that there was a problem with the HTTP POST request packet that the client transmitted, and that it should lower the packet size to 1280 bytes to be able to be passed on. When the client receives such an ICMP PTB packet, it will adjust the MSS and retransmit the web form data after segmenting the payload data into two separate packets: one packet of 1280 bytes and another packet of 164 bytes. After the retransmission by the client, the packets will be passed on by all routers and successfully arrive at the web server if no other bottleneck is present in the path, which is not the case in this example. Once the web server has received these packets, it will transmit a TCP Acknowledgement (ACK) packet and a HTTP status 200 packet to indicate the client's web form data was successfully received. The client will acknowledge the latter packet and the TCP session is closed (actual TCP session closing packets are not shown).

Although the operation of PMTUD is the same for both IPv4 and IPv6, there are some subtle differences that are due to the use of ICMPv4 or ICMPv6, respectively. These differences are outlined in Section 3.3.1 and 3.3.2.

3.3.1 IPv4

In IPv4 a sending node can choose whether it sets the special purpose Don't Fragment (DF) flag in the packet's IP header to prohibit any fragmentation done by routers. Otherwise, if this flag is not set, routers may decide to fragment the packets themselves if required because of a bottleneck in the path. In cases like this there is no need to perform PMTUD if all routers support fragmentation and are willing to do so.

PMTUD in IPv4 operates on ICMPv4 for PTB signalling. The ICMPv4 packet type that is used for PTB signalling is type 3 (*Destination Unreachable*) along with code 4 (*Fragmentation required and DF flag set*).

3.3.2 IPv6

In IPv6 a sending node does not have the ability to choose whether the packets are allowed to be fragmented by routers. Instead of an explicit DF flag in the packet's IP header, fragmentation of IPv6 packets by routers is prohibited and may only be done by the two nodes that act as endpoints in a path.

PMTUD in IPv6 operates on ICMPv6 for PTB signalling. The ICMPv6 packet type that is used for PTB signalling is type 2 (*Packet Too Big*) along with code 0.

3.4 PMTU black holes

There are two problems regarding PMTU that are caused by the variety of different MTUs being used in the Internet. The first is due to the fact that PMTUD relies on ICMP packets. Network administrators are sometimes afraid of potential abuse of ICMP packets or fragments, e.g. for DoS attacks. For this reason some firewalls are configured with access policies for filtering certain or all types of ICMP packets, and possibly also fragments. Both types of filtering cause PMTU black holes, the former is discussed in Section 3.4.1 and the latter in Section 4.2.2.

3.4.1 ICMP PTB filtering

In Section 3.3 we have shown how PMTUD operates in an environment without filtering. Figure 2 illustrates how PMTUD fails to operate successfully due to the filtering of ICMP PTB packets. Moreover, it illustrates how the occurrence of this failure creates a PMTU black hole. The layout and use-case of this figure are identical to that of Figure 1.

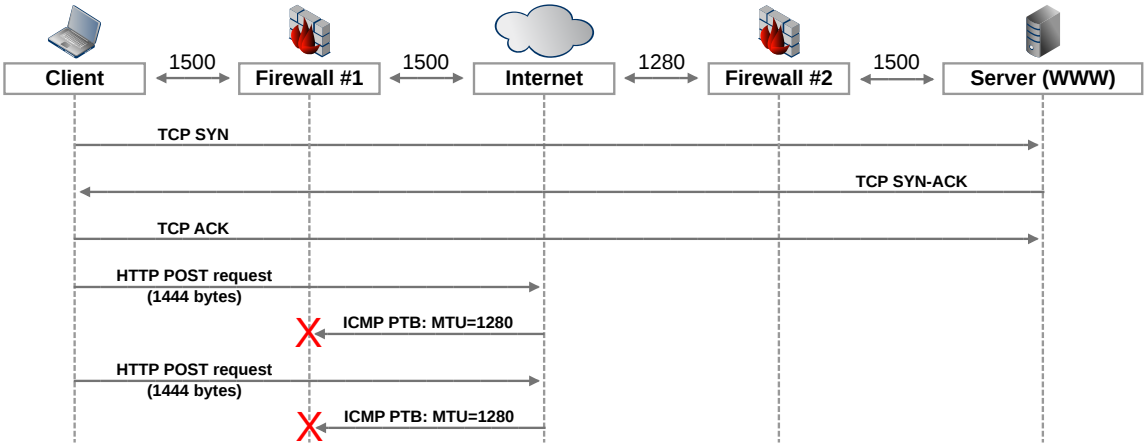

Figure 2: PMTU black hole: ICMP PTB filtering

Again, the first three packets that go back and forth are small packets that are used to establish a TCP session. After the session has been established, the client will transmit a packet of 1444 bytes, containing the web form data, in a HTTP POST request. This HTTP POST request packet will trigger an ICMP PTB packet to be send back to the client. However, this packet will never reach the client since the first firewall is configured to filter ICMP PTB packets. What happens is that after a while a timeout occurs at the client because of the missing TCP ACK packet that the web server should normally have send in response to the HTTP POST request packet. The client will respond to this event by retransmitting the HTTP POST request packet. Unfortunately, the situation remains unchanged. Despite the recovery attempt performed by the client in response to the absence of a TCP ACK for these black holed packets, large packets will never be acknowledged by the web server unless they are each split into multiple smaller TCP segments that do fit.

3.4.2 IP fragment filtering

Figure 3 illustrates an example of a path that does not filter IP fragments. The client, on the far left, wants to resolve a certain Fully Qualified Domain Name (FQDN) into the corresponding IP addresses. We assume that the Domain Name System Security Extensions (DNSSEC) server, on the far right, knows the IP addresses associated with the FQDN. Both are connected to the Internet through a firewall that forwards traffic to and from the Internet. Note that the PMTU between the client and the DNSSEC server is 1280 bytes because of the MTU configured on the NIC of the DNSSEC server.

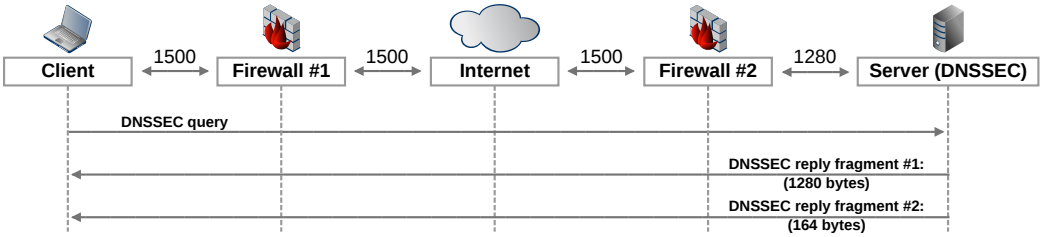

Figure 3: PMTU: successful DNS query - MTU: 1280

The first packet is initiated by the client and contains a DNSSEC query with the FQDN to be resolved. Once the DNSSEC server has received this packet it will fragment the response data, i.e. the IP address records and signatures, into two fragments and transmit these to the client. Since the DNSSEC server has the lowest MTU in the path, the fragments will be delivered successfully. Now that the client has received both fragments it will be able to reassemble the response data.

Figure 4 illustrates how the DNSSEC query fails due to IP fragment filtering. Moreover, it illustrates how the occurrence of this failure creates a PMTU black hole. The layout and use-case of this figure are identical to that of Figure 3.

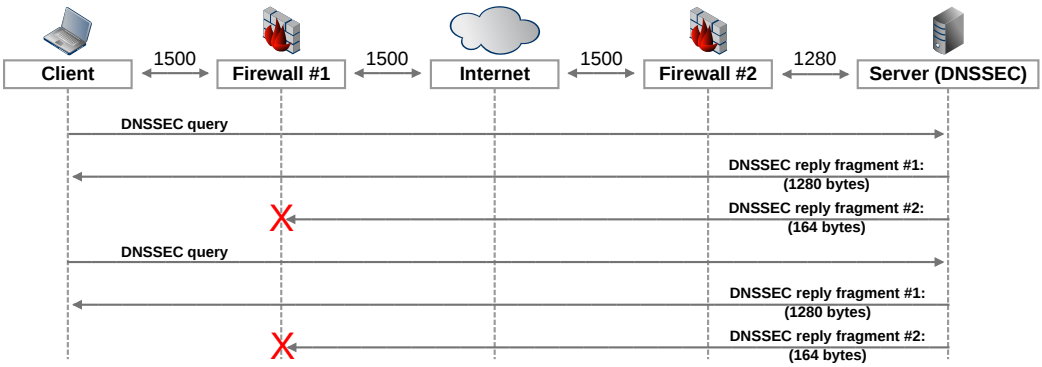

Figure 4: PMTU black hole: IP fragment filtering

Again, the client initiates the first packet containing a DNSSEC query. The DNSSEC server will fragment the response data into two fragments. The first fragment contains a transport layer header since it is crucial for the reassembly of fragments. In the case of DNSSEC the transport layer header is UDP. The second fragment does not contain a UDP header. Because the first firewall is configured to filter fragments, the last fragment, containing no UDP header, never reaches the client. However, the first fragment that has a UDP header will not be filtered. Because the client did only receive one fragment it is unable to reassemble the response data. Unfortunately, the situation remains unchanged even after repeated DNSSEC queries.

4 Experiments

4.1 RIPE Atlas

For the experiments in our research we will be making use of RIPE Atlas. This is an Internet measurement network developed and deployed by RIPE NCC in order measure the Internet infrastructure in real time. To do so, a large number of probes have been distributed worldwide. A probe is a small USB-powered embedded device that is controlled by the systems of RIPE Atlas to run measurements. The results of all measurements are reported to the data collection components of Atlas. Each probe acts as a vantage point for the performed measurements.

Figure 5 illustrates a graph with the number of active probes worldwide. This graph shows that there currently are a total of 2000 probes of which 1650 are up and running, meaning that these probes are connected to the Internet and the RIPE Atlas infrastructure. The remaining 350 probes are marked as being down. Of the 2000 probes in total, around 800 probes are IPv6-enabled aside from IPv4.

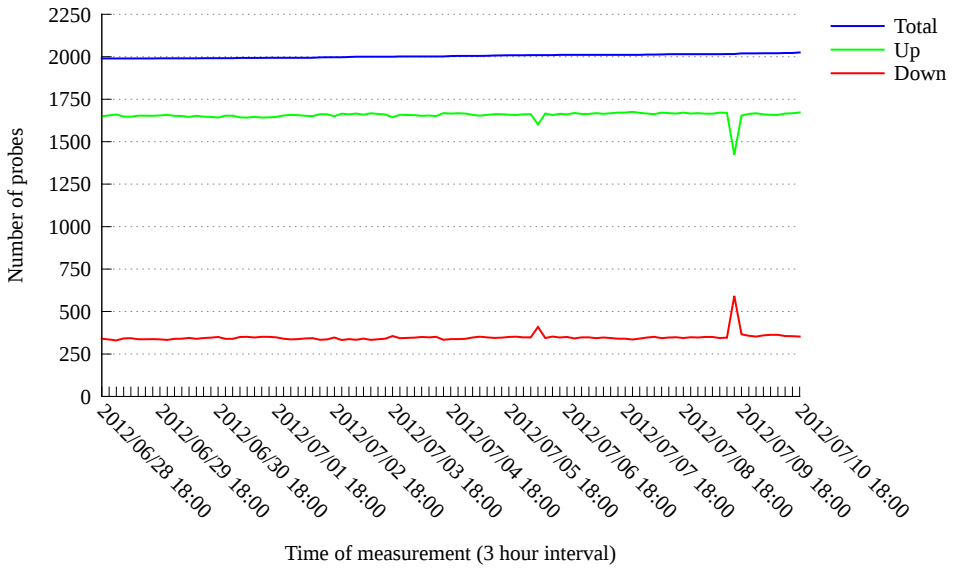

Figure 5: RIPE Atlas number of active probes

The majority of the Atlas probes are located in the RIPE NCC service region, however, there are also probes in other regions around the globe. Figure 6 illustrates the approximate location extent of the RIPE Atlas network and shows that most of the probes are located in Europe. Furthermore, Figures 25 and 26 in Appendix A illustrate the number of probes per country for those with at least 5 probes.

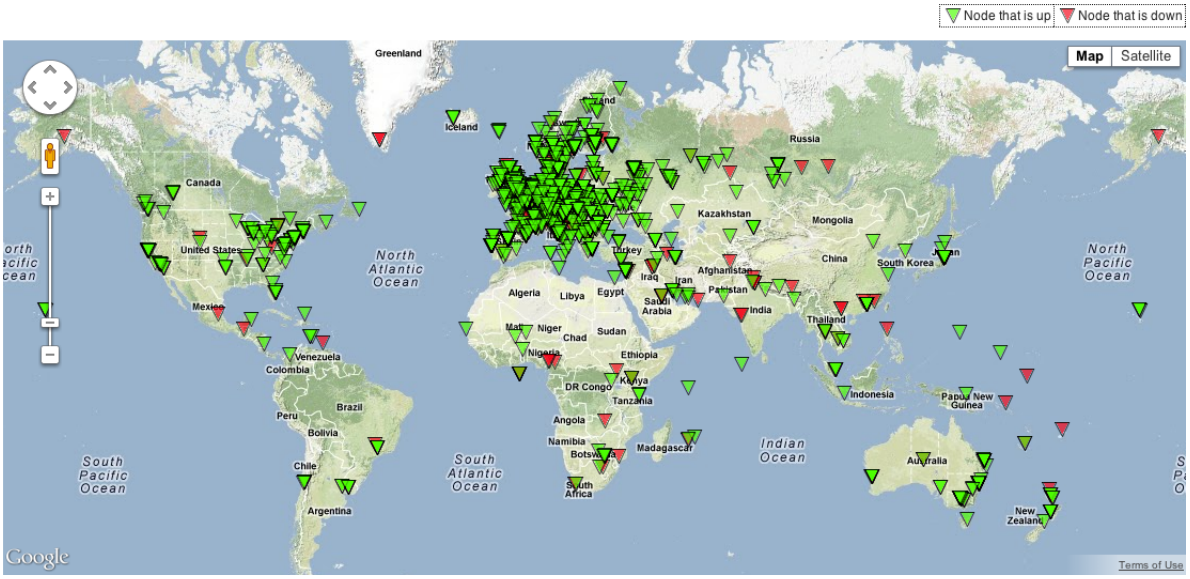

Figure 6: RIPE Atlas active probes map

By default a registered RIPE Atlas user has the ability to perform User Defined Measurements (UDMs) that consist of *ping* and *traceroute* measurements on the Internet for both IPv4 and IPv6. The execution of a UDM requires credits that can be earned by owning a probe and keeping it up and running. With these credits users can, for example, run measurements to verify if a specific network node or path is functioning properly when accessed from geographically and topologically diverse locations. All probes in the RIPE Atlas network also participate in measurements that are predefined by RIPE NCC such as the Round Trip Time (RTT) for various Domain Name System (DNS) root servers.

RIPE Atlas plays a critical role in our research. To be able to perform our experiments, the availability of additional measurements that normally are not available to a user is a must. Close collaboration between NLnet Labs and RIPE NCC has resulted in the ability to let the probes perform HTTP GET and HTTP POST request measurements, and DNS query measurements in combination with our experimental setup. A probe does not have the ability to capture and store the traffic of such measurements, however, each probe stores a summary of its measurement results on the systems of RIPE Atlas. Although the results provide a summarized overview of events, their granularity is high enough for the experiments we perform. Because of a probe's current inability to capture and store traffic we decided not to build a full mesh network of probes for triangulation purposes. We could therefore only measure the paths between each probe and our experimental setup.

With our experimental setup we will discover where on the IPv4 and IPv6-driven Internet PMTU black holes occur. More precisely, with this setup we measure which underlying problem is the cause for these black holes and how often these problems occur. In addition to these measurements we will also inventory the PMTU for each probe.

4.2 Experimental setup

The three different types of experiments that will be conducted are ICMP PTB packet filtering, IP fragment filtering, and PMTU determination. The experimental setup consists of three different servers located in the lab at the University of Amsterdam, The Netherlands. These servers are directly connected to the Internet via SURFnet, have native IPv4 and IPv6 connectivity, and are configured to allow any type of traffic from and to the Internet.

They run unmodified versions of the Linux distribution Ubuntu, except for a subtitle change in the way the operating system handles network traffic. By default, generic and TCP segmentation operations are offloaded to the NIC. Effectively this means that the network driver can simply pass on large bulks of data to the NIC since it is responsible for the segmentation. The problem is that programs that capture network traffic, e.g. *tcpdump*, use the network driver for this purpose and thus only see the large bulks of data and not the individual segments. Since it is important for our research to have the ability to see the actual network traffic in its original packetized state, we have completely disabled any form of network traffic offloading.

4.2.1 ICMP PTB filtering

In this experiment we determine if ICMP PTB packets are filtered in the path between a probe and our experimental setup. Figure 7 illustrates the experimental setup that was created for this experiment. The figure shows that we have two servers connected to the Internet: Chummi and Belgrade (which is the actual name of the server, not its location). To provide Belgrade with Internet access, Chummi is configured to route traffic between Belgrade and the Internet. The link MTU of the link between Chummi and Belgrade is controlled by altering the MTU on Chummi. This way we do not introduce any potentially beneficial effects on, e.g. the MSS advertized by TCP, that could lead to the absence of PMTUD. A web server (Apache 2.2) is running on Belgrade and is configured to allow HTTP POST requests from the Internet. Chummi runs *tcpdump* to capture all the traffic that travels to and from itself, e.g. ICMP PTB packets, but also captures all traffic that travels to and from Belgrade. The recorded packet trace is stored for later analysis.

During the experiment we let every probe do a HTTP POST request of 65528 bytes to the web server running at Belgrade. In the IPv4 HTTP POST request packets, the DF flag is set in the IPv4 header to prohibit any fragmentation by routers. The large packets containing the payload data travel from a probe to Chummi and reach their final destination at Belgrade, as illustrated in Figure 7. However, when these large packets travel to our experimental setup they hit the bottleneck between Chummi and Belgrade. In response to this event Chummi will generate an ICMP PTB packet and send this to the probe in order to request smaller packets. If the probe receives the ICMP PTB packet, it will retransmit the packets with a reduced size so that the HTTP POST request is successfully received. Note that this situation only occurs if the probe or a router is not configured with an MTU of 1280 bytes.

If the packet trace recorded at Chummi contains a successful TCP three-way handshake and a header belonging to the HTTP POST request, but no actual data, we observe that a HTTP response with status code of 400 is send back from Belgrade. This particular status code is used to indicate a bad request has occurred, which in this case happens when no data is received within the timeout interval of approximately 10 seconds after the HTTP POST request was issued. We therefore interpret such an event as a path that filters ICMP PTB packets. This process is repeated for all probes that participate in the experiment.

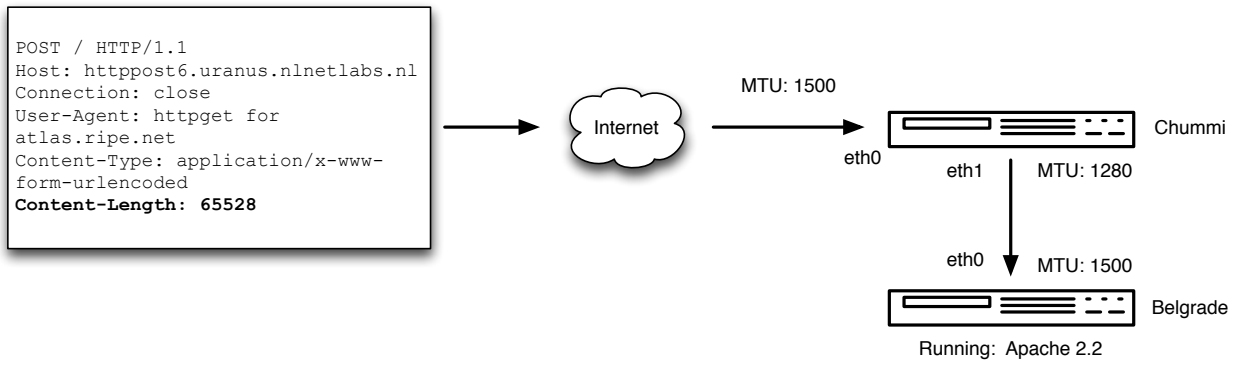

Figure 7: ICMP PTB filtering setup

4.2.2 IP fragment filtering

In this experiment we determine if IP fragments are filtered in the path between a probe and our experimental setup. Figure 8 illustrates the experimental setup that was created for this experiment. The figure shows that we use Chummi and Berlin (again, not the server's location) in this experiment. For the IPv4 experiment we set the MTU on Berlin to the IPv4 minimum MTU of 576 bytes. For the IPv6 experiment we set the MTU on Chummi to the IPv6 minimum of 1280 bytes. During the experiment both Chummi and Berlin will be running *ldns-testns* in additional to *tcpdump*. The former is a lightweight DNS server that responds to requests with specially crafted replies, i.e. a large amount of data that must be fragmented in order fit onto the path.

During the experiment we let every probe send a DNS query to Chummi and Berlin for the *version.bind* record. After receiving the DNS query at both servers, fragments containing the requested DNS data of 1590 bytes in total are transmitted to the probe. An truncated overview of this response is illustrated in Figure 8. By setting the MTU on both servers to IP minimum MTU we assure that the DNS reply is send with the smallest possible fragments that all routers in the path should be able to forward. This would mean that no ICMP PTB packets are generated because PMTUD should not be needed. Nevertheless, in the IPv4 fragments the DF flag is set in the IPv4 header to prohibit any fragmentation by routers.

If the packet trace recorded at Chummi and Berlin contains a DNS query and the corresponding fragments that are send back, we verify the probe's results stored at RIPE Atlas to see whether or not the transmitted fragments were successfully received. If these results does not confirm the receipt of fragments for the probe, we conclude that the fragments were filtered somewhere in the path between the probe and our experimental setup. This process is repeated for all probes that participate in the experiment.

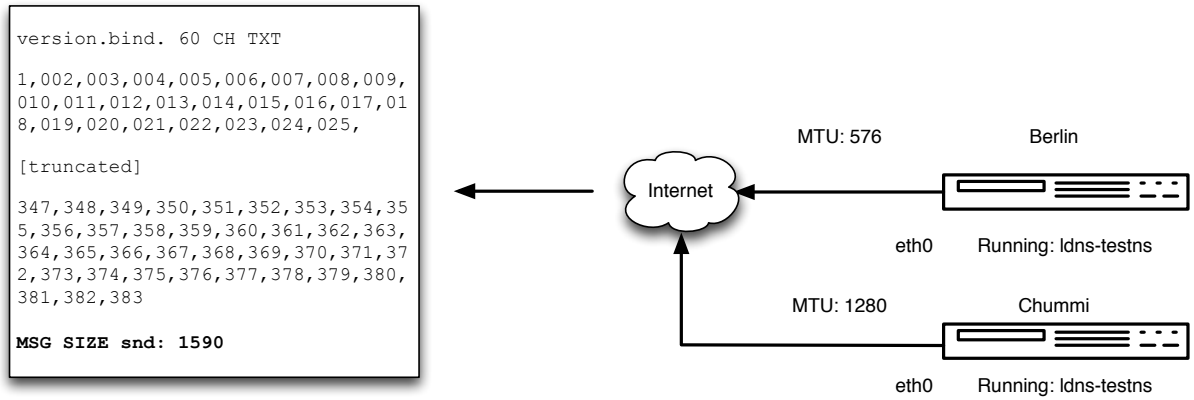

Figure 8: IP fragment filtering setup

4.2.3 PMTU determination

In this experiment we determine the PMTU for the path between a probe and our experimental setup. Figure 9 illustrates the experimental setup that was created for this experiment. The figure shows that we only use Berlin in this experiment. The link MTU between Berlin and the Internet is configured with the default of 1500 bytes. This is done to advertize a large MSS of 1460 bytes or 1440 bytes for IPv4 and IPv6, respectively. A web server (Apache 2.2) is run on Berlin and is configured to allow HTTP GET requests from the Internet. It also runs *tcpdump* to capture traffic for later analysis.

During the experiment we let every probe do a HTTP GET request to the web server running at Berlin. In response to such a request, it will serve a 5000 bytes file. The large packets that are send back containing this response travel from Berlin to a probe, as illustrated in Figure 9. In the IPv4 HTTP GET response packets the DF flag is set in the IPv4 header to prohibit any fragmentation by routers.

If these large packets hit an MTU bottleneck somewhere on the way to a probe, an ICMP PTB packet is send to Berlin. In response to this event Berlin will retransmit the file using smaller packets.

This process continues until the probe has successfully received the entire file. With the information of the TCP MSS, as well as that in the ICMP PTB packet if it was received, we can determine the PMTU for the path between the probe and our experimental setup.

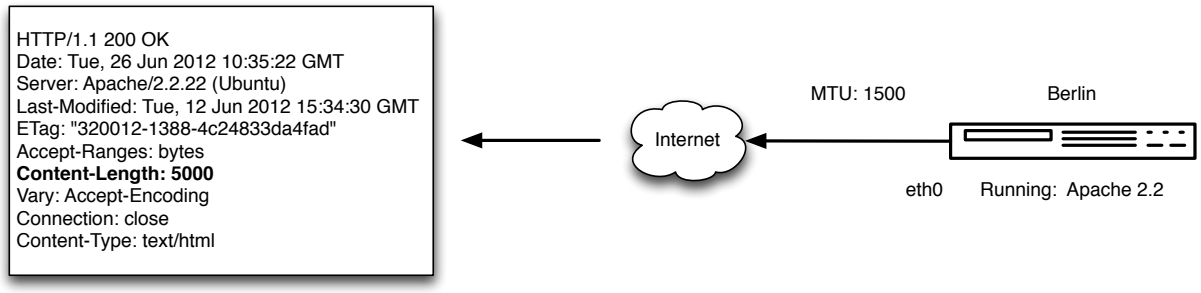

Figure 9: PMTU determination setup

4.3 Measurement periods

All of the experiments described in Section 4.2 are performed during different measurement periods to ensure they do not interfere with each other and adversely affect the results. For each experiment the duration of a single measurement period during which an experiment is performed is four hours. Each probe participating in an experiment will conduct the measurements with an one hour interval. Effectively this means that during each experiment period, a probe will conduct four measurements in total.

For the ICMP PTB packet filtering and PMTU determination experiments, a total of six measurement periods were performed. For the IP fragment filtering experiment a total of three measurement periods were performed. Table 1 illustrates the time of each measurement period as well as the configured MTU in the experimental setup if it differs from the default 1500 bytes.

For the ICMP PTB packet filtering experiments we ran three periods with the MTU set to 1500 bytes to create a baseline of what happens in the reality when there are no artificial bottlenecks added. We suspect that less ICMP PTB packets will be generated because the complete PMTU between the probes and our experimental setup is 1500 bytes. To confirm these thoughts we ran three measurement periods with the MTU set to 1280 bytes to see how much would go wrong if an artificial bottleneck is present in the path. Assuming a probe is not configured with an MTU of 1280 bytes, ICMP PTB packets are generated and the probe should act accordingly.

Date and time	**Interface MTU**
2012/06/26, 12:00 - 16:00	Chummi eth1 MTU: 1280
2012/06/26, 16:00 - 20:00	Chummi eth1 MTU: 1500
2012/06/27, 20:00 - 00:00	Chummi eth1 MTU: 1500
2012/06/27, 12:00 - 16:00	Chummi eth1 MTU: 1500
2012/06/27, 16:00 - 20:00	Chummi eth1 MTU: 1280
2012/06/28, 20:00 - 00:00	Chummi eth1 MTU: 1280
2012/07/09, 16:00 - 20:00	Berlin eth0 MTU: 576, Chummi eth0 MTU: 1280
2012/07/10, 20:00 - 0:00	Berlin eth0 MTU: 576, Chummi eth0 MTU: 1280
2012/07/10, 0:00 - 4:00	Berlin eth0 MTU: 576, Chummi eth0 MTU: 1280

Table 1: Experiment measurement periods

5 Results

In this chapter we will discuss the results we gathered without further reasoning or analysis. The results may look counterintuitive but are explained in great detail in Chapter 6 and 7.

5.1 Number of probes

The number of probes differ per experiment as well as experiment period since we had no control over which probes were actually participating. In Table 2 we observe the average number of probes that participated each of the different experiments. For the IPv4-based experiments, the average number of probes that participated was between 1143 and 1365, and between 397 and 502 for IPv6-based experiments.

Experiment	IPv4	IPv6
HTTP POST request	1139	400
DNS query	1365	502
HTTP GET request	1148	397

Table 2: Average number of probes per experiment

5.2 ICMP PTB filtering

Figure 10 illustrates the results of the ICMPv4 PTB packet filtering experiment when the MTU was set to 1500 bytes. During this experiment it could be the case that less ICMP PTB packets were generated opposed to what was previously described in Section 4.3. An average of 1144 probes probes participated during three measurement periods. For all three periods we observe that no paths between the probes and our experimental setup filter ICMPv4 PTB packets.

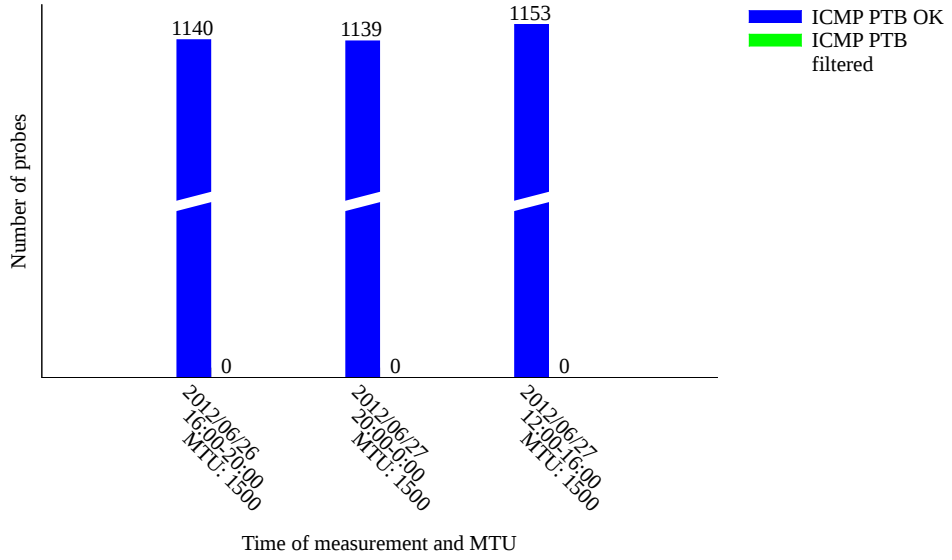

Figure 10: ICMPv4 PTB filtering - MTU: 1500

Figure 11 illustrates the results of the ICMPv6 PTB packet filtering experiment when the MTU was set to 1500 bytes. During this experiment it could be the case that less ICMP PTB packet were generated opposed to what was previously described in Section 4.3. An average of 396 probes participated during three measurement periods. We observe that between one or two paths between the probes and our experimental setup filter ICMPv6 PTB packets.

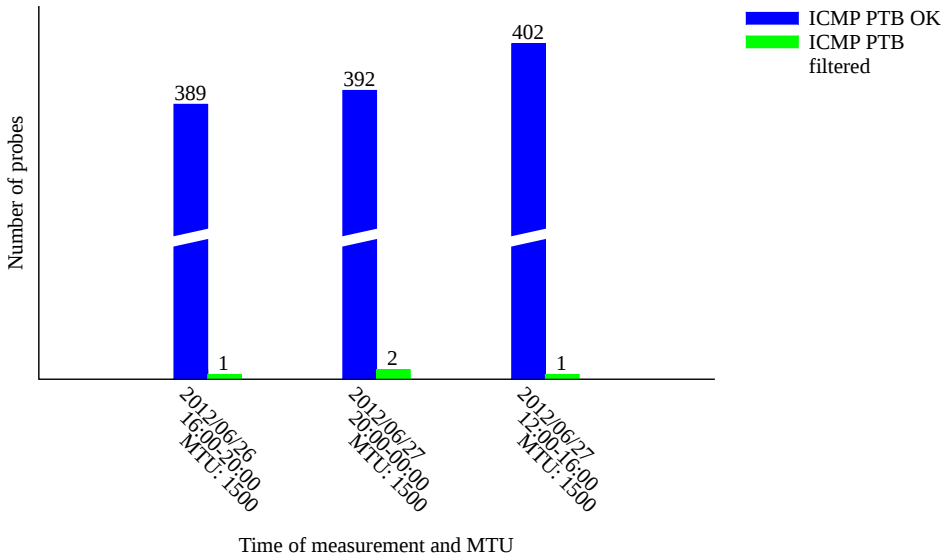

Figure 11: ICMPv6 PTB filtering - MTU: 1500

In Figure 12 a comparison in percentages between the ICMPv4 and ICMPv6 PTB packet filtering experiments are made when for both protocols the MTU was set to 1500 bytes. Again, during these experiments it could have been the case that less ICMP PTB packets were generated opposed to what was previously described in Section 4.3. For IPv4 0% of the paths between the probes and our experimental setup filter ICMP PTB packets. For IPv6 this is between 0.25% and 0.5%.

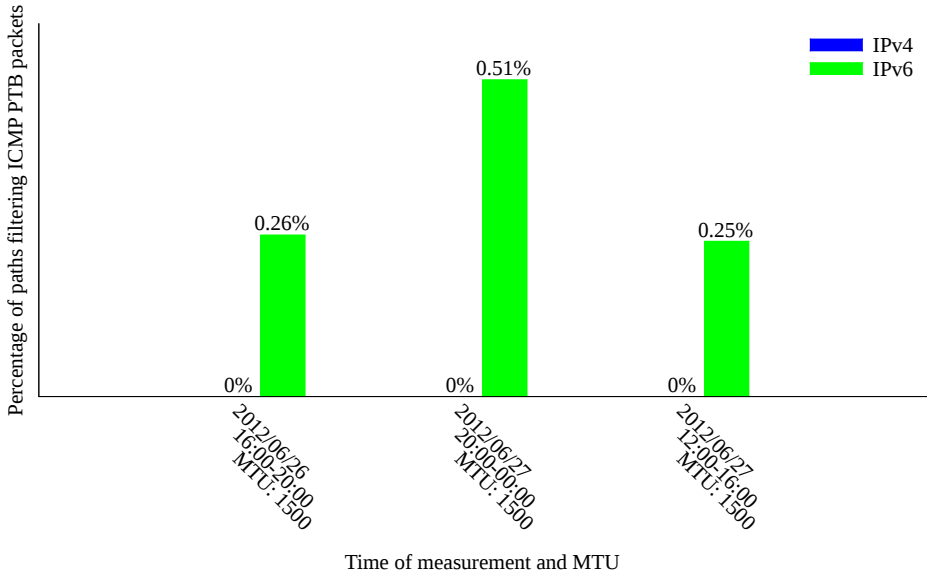

Figure 12: ICMP PTB filtering percentages - MTU: 1500

Figure 13 illustrates the results of the ICMPv4 PTB packet filtering experiment when the MTU was set to 1280 bytes. An average of 1115 probes probes participated during three measurement periods. We observe that between 43 and 69 paths between the probes and our experimental setup filter ICMPv4 PTB packets.

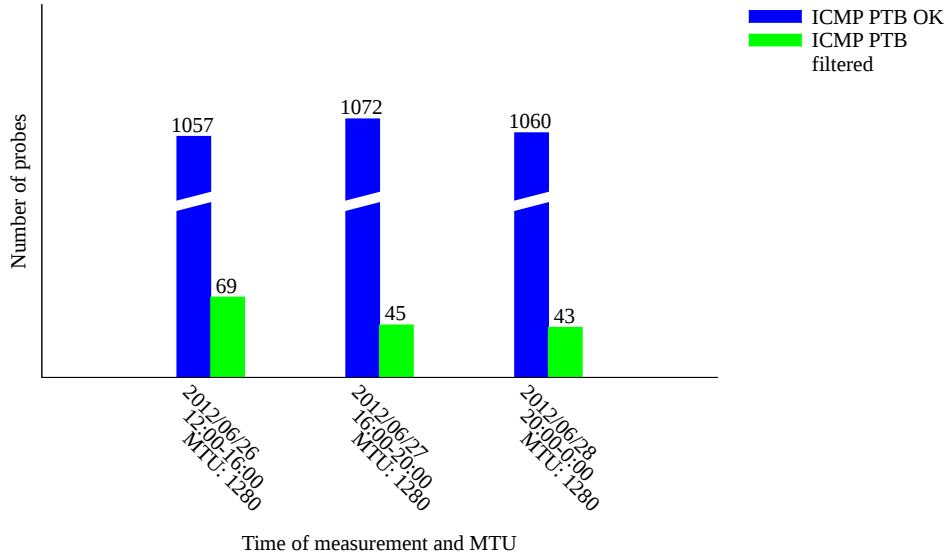

Figure 13: ICMPv4 PTB filtering - MTU: 1280

Figure 14 illustrates the results of the ICMPv6 PTB packet filtering experiment when the MTU is set to 1280 bytes. An average of 383 probes probes participated during three measurement periods. We observe that between 3 and 4 paths between the probes and our experimental setup filter ICMPv6 PTB packets.

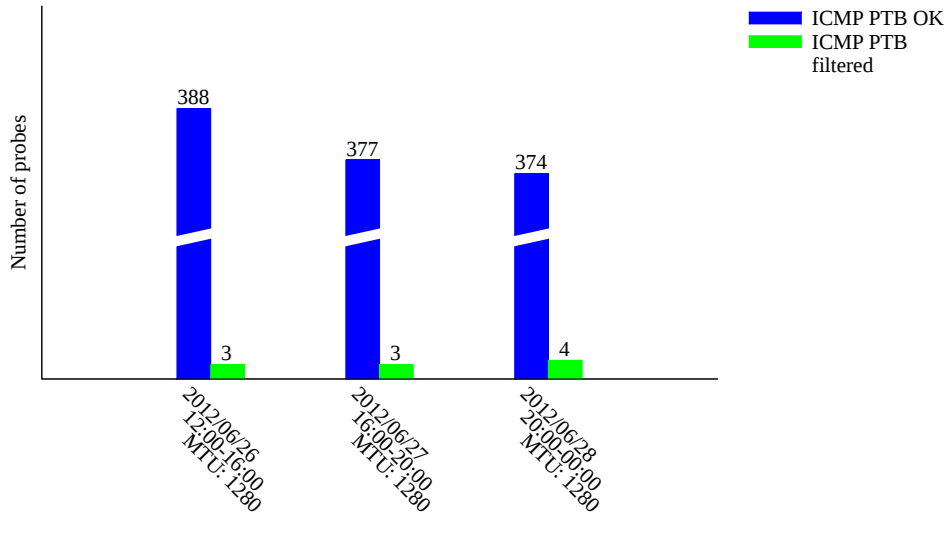

Figure 14: ICMPv6 PTB filtering - MTU: 1280

In Figure 15 a comparison in percentages between the ICMPv4 and ICMPv6 PTB filtering experiments are made when for both protocols the MTU is set to 1280 bytes. For IPv4 between approximately 6% and 4% of the paths between the probes and our experimental setup filter ICMP PTB packets. For IPv6 this is between 0.77% and 1.07%.

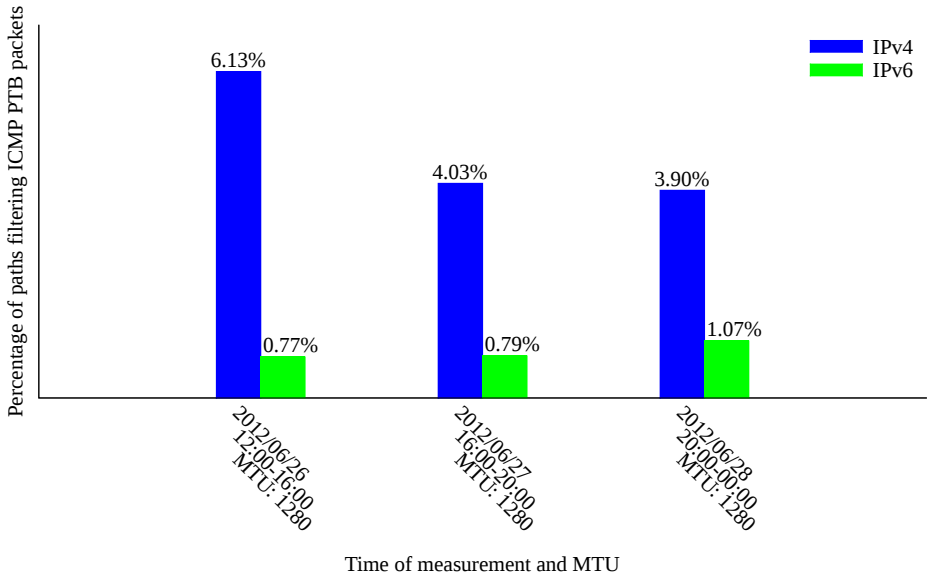

Figure 15: ICMP PTB filtering percentages - MTU: 1280

5.3 IP fragment filtering

In Figure 16 we observe the results of the IPv4 fragment filtering experiment when the MTU was set to 1500 bytes. The experiment was conducted for six measurement periods with the average number of participating probes being 1181. We observe that between 1013 and 1060 paths between the probes and our experimental setup do not filter fragments and between 129 and 164 paths do filter fragments.

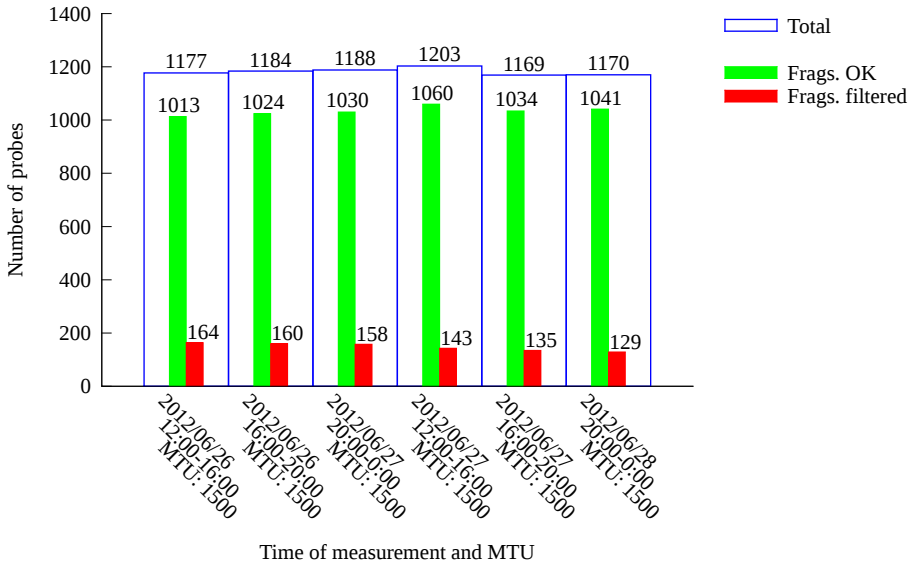

Figure 16: IPv4 fragment filtering - MTU: 1500

In Figure 17 we observe the results of the IPv6 fragment filtering experiment when the MTU was set to 1500 bytes. The experiment was conducted for six measurement periods with the average number of participating probes being 423. We observe that between 241 and 254 paths between the probes and our experimental setup do not filter fragments and between 173 and 181 paths do filter fragments.

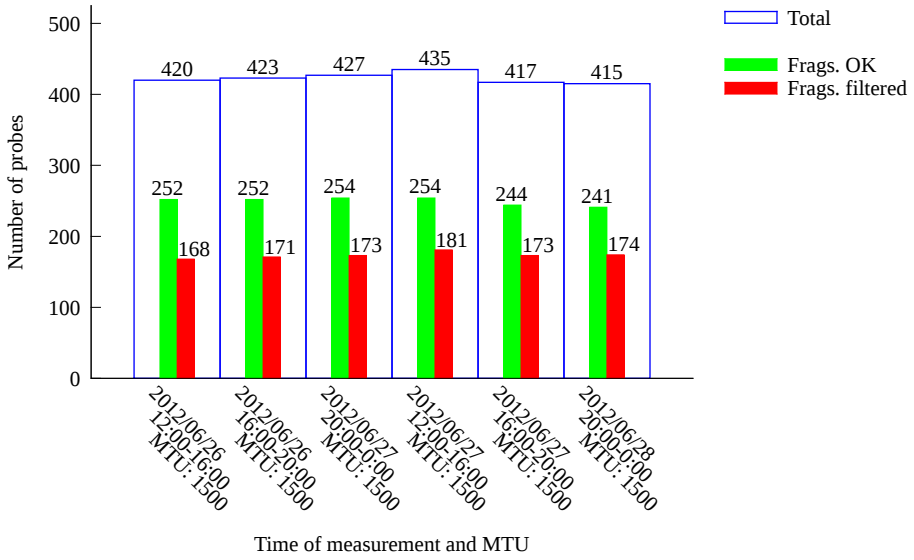

Figure 17: IPv6 fragment filtering - MTU: 1500

Figure 18 illustrates a comparison in percentages between the IPv4 and IPv6 fragment filtering experiments when for both protocols the MTU was set to 1500 bytes. We observe that in IPv4 less paths between the probes and our experimental setup filter fragments compared to IPv6. For IPv4 between approximately 11% and 14% of the paths filter fragments. For IPv6 this is between 40% and nearly 42%.

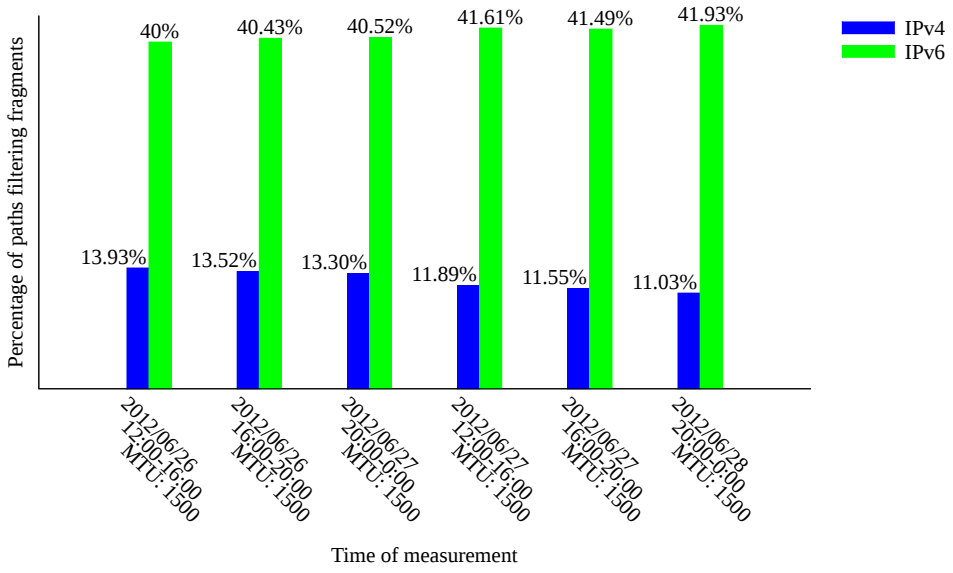

Figure 18: IP fragment filtering percentages - MTU: 1500

In Figure 19 we observe the results of the IPv4 fragment filtering experiment when the MTU was set to the protocol minimum MTU of 576 bytes. The experiment was conducted for three measurement periods with the average number of participating probes being 1365. We observe that between 1268 and 1291 paths in between the probes and our experimental setup do not filter fragments and 84 paths do filter fragments.

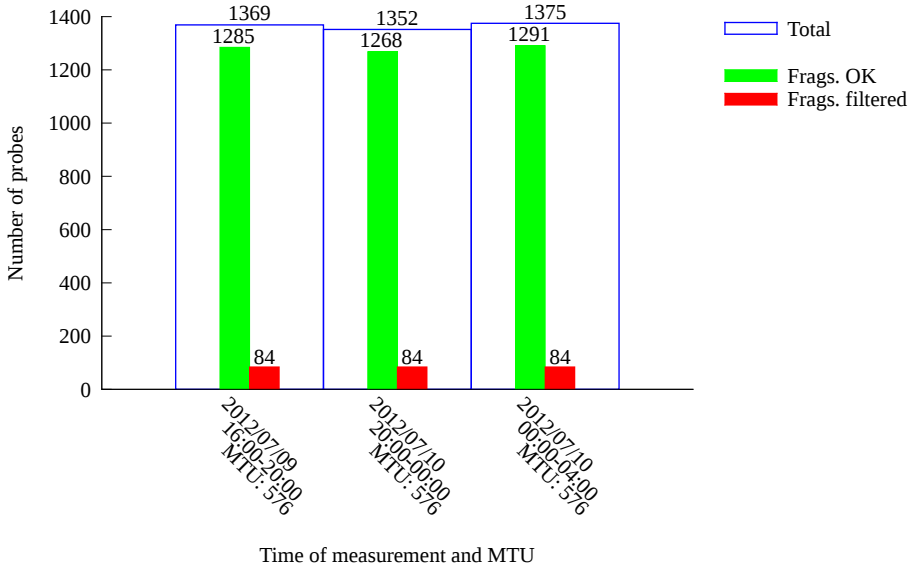

Figure 19: IPv4 fragment filtering - MTU: 576

In Figure 20 we observe the results for the IPv6 fragment filtering experiment where the MTU is set to the protocol minimum MTU of 1280 bytes. The experiment was conducted for three periods with the average number of participating probes being 502. We observe that between 448 and 456 paths between the probes and our experimental setup do not filter fragments and between 49 and 51 paths do filter fragments.

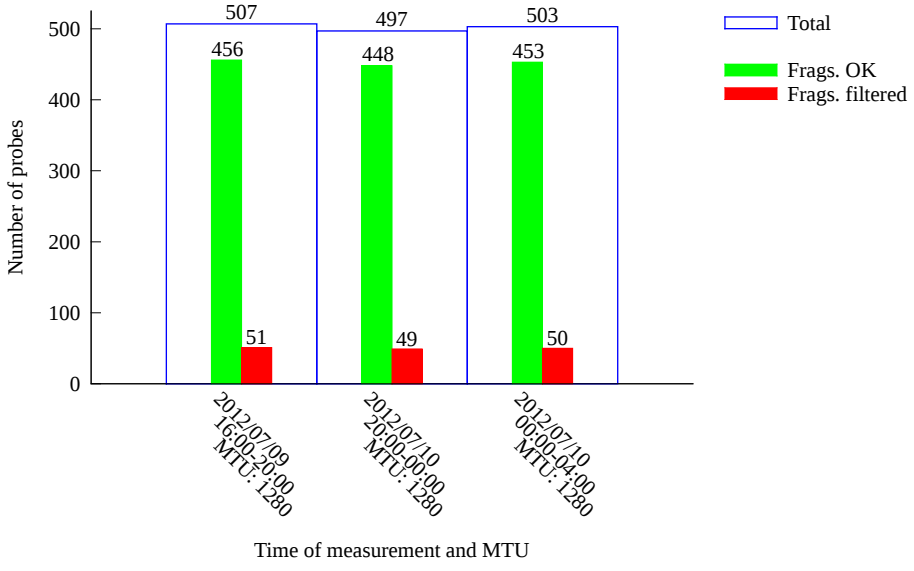

Figure 20: IPv6 fragment filtering - MTU: 1280

Figure 21 illustrates a comparison in percentages between the IPv4 and IPv6 fragment filtering experiments when the MTU for both protocols was set to the protocol minimum MTU. For IPv4 the minimum MTU is 576 bytes and for IPv6 this is 1280 bytes. We observe that in IPv4 less paths between the probes and our experimental setup filter fragments compared to IPv6. For IPv4 approximately 6% of the paths filter fragments. For IPv6 this is approximately 10%.

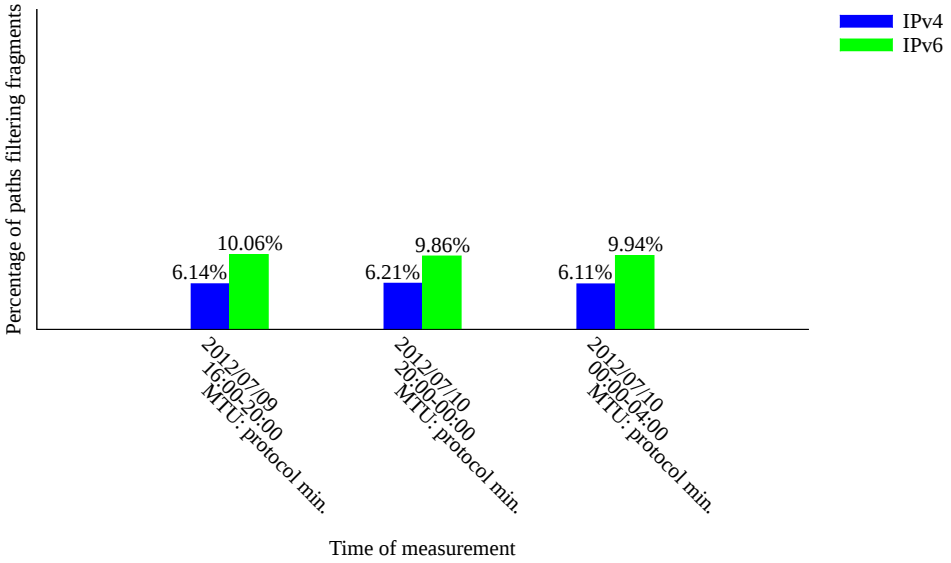

Figure 21: IP fragment filtering percentages - MTU: IP minimum

5.4 Location determination

To be able to pinpoint where on the Internet ICMP PTB packets and IP fragments get filtered, we developed an algorithm we called "hop counting". During the experiments we try to find where ICMP PTB packets and fragments get filtered. The experiments resulted in two lists with IP addresses of probes depending on whether the corresponding paths do or do not filter ICMP PTB packets and fragments. For every IP address in these lists we run a *traceroute* and store the output in a file. Once we have collected all the routes, we will analyze them automatically using a script.

In the script we define a dictionary data structure with the IP address as the key and a list with two fields as the value. We first analyze all the routes for which the paths do not filter ICMP PTB packets and fragments. We iterate through the file hop by hop. If the hop is in the *traceroute* results but does not exists as a key in the dictionary, we add it with the IP address as key and with a list of {1, 0} as the value. If the hop does already exists as a key in the dictionary we increase the left side of the list value by one, e.g. {2, 0}.

Now we do the same for all the routes for which the paths to the probes filter ICMP PTB packets and fragments. Again, if the IP address of a specific hop in the *traceroute* does not exist in the dictionary we add it but this time with a list of {0, 1} as value. If the key exists and the value is for example {4, 0}, we increase the by one so that is will become {4, 1}.

An example of how the hop counting algorithm works is illustrated in Figure 22. In the example two lists of IP addresses exists. The first with IP addresses for which the paths do not filter ICMP PTB packets consist of two IP addresses, and the second for which the paths do filter ICMP PTB packets consist of four IP addresses.

In the figure we see that every *traceroute* first goes through router 1. At router 2 we observe that one *traceroute* for a working path went through and three for bad paths. This indicates that the router is probably not responsible for filtering ICMP PTB packets and fragments since it is also part of a working path.[1]

When we look at router 3 we observe that every *traceroute* to the three probes behind this router are to probes that are on the list of paths that filter ICMP PTB packets and fragments. If this is the case we assume this router is to blame for the filtering of ICMP PTB packets and fragments.

If a relatively popular router (a router that is part of many routes) seems to be guilty of filtering ICMP PTB packets and fragments, we assume this is somewhere in the core in of the Internet. If a relatively small number of routes come across a specific IP address we assume a router more on the edges of the Internet is responsible for the filtering of ICMP PTB packets and fragments. Examples of such routes are those located at home or in companies.

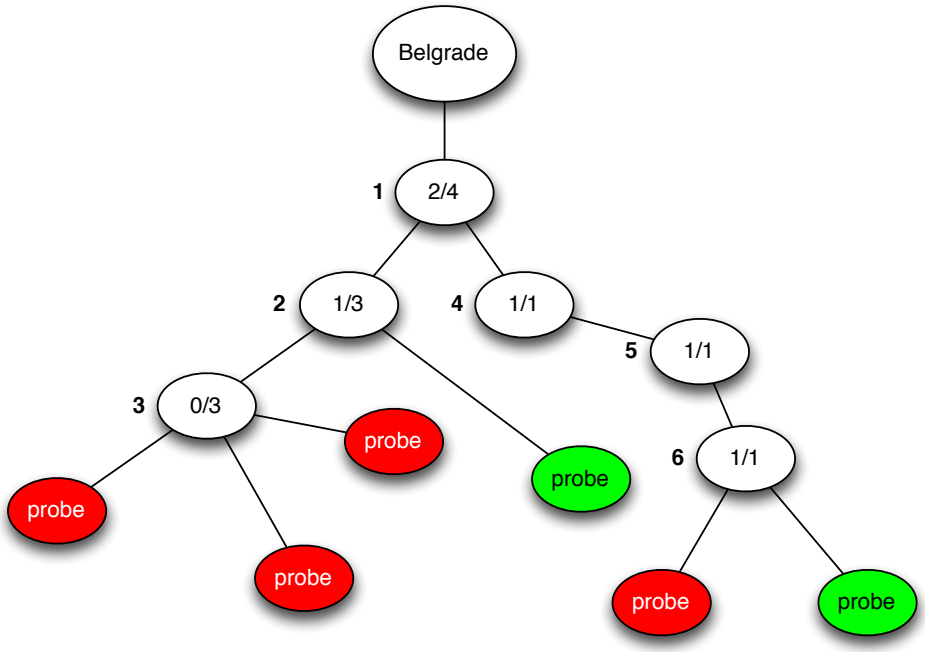

Figure 22: Hop counting algorithm

5.4.1 ICMP PTB filtering

In Listing 1 we observe the results of the hop counting algorithm for the ICMPv4 PTB filtering experiment on 26-06-2012 from 16:00 till 20:00. The second line in Listing 1 shows the SURFnet router (*145.145.19.190*) to which the network in the lab is connected. All routes go through this router. In Listing 1 many of the routes are omitted for the sake of readability. The first router in the complete list that has an error percentage of 100% is *203.50.6.78*.

```
1 Bad     Total  Error percentage     IP address
2 69      1126   6.1%                 145.145.19.190
3 53       810   6.5%                 145.145.80.65
4 16       311   5.1%                 145.145.80.73
5 13       214   6.1%                 77.67.72.109
6 7        199   3.5%                 109.105.98.33
7 2         60   3.3%                 62.40.124.157
8 ...
9 2          2   100.0%               203.50.6.78
```

[1] Of course there can be some kind of access policy that does not allow specific hosts through but these policies are more often applied at the business or customer premises and not in the core of the Internet.

```
2          2   100.0%              203.50.6.89
2          2   100.0%              61.10.0.118
2          2   100.0%              80.231.159.10
2          2   100.0%              84.116.238.49
```

Listing 1: Location of ICMPv4 PTB packets filtering

In Listing 2 we observe the results of the hop counting algorithm for the ICMPv6 PTB filtering experiment on 26-06-2012 from 16:00 till 20:00. Again, the second line in Listing 2 displays the SURFnet router (*2001:610:158:1916:145:100:99:17*) to which the network in the lab is connected. All routes go through this router. In Listing 2 again many of the routes are omitted for the sake of readability. When we go through the entire list there are no routers with an error percentage of 100%.

```
Bad     Total  Error percentage     IP address
3        391   0.8%                 2001:610:158:1916:145:100:99:17
2        292   0.7%                 2001:610:e08:64::65
2        131   1.5%                 2001:7f8:1::a500:6939:1
1          9   11.1%                2001:470:0:217::2
1          6   16.7%                2001:470:0:67::2
1         46   2.2%                 2001:470:0:3f::1
...
No routers with 100% failure rate.
```

Listing 2: Location of ICMPv6 PTB packets filtering

5.4.2 IP fragment filtering

In Listing 3 we observe the results of the hop counting algorithm for the IPv4 fragment filtering experiment on 09-07-2012 from 16:00 till 20:00. The second line in Listing 3 displays the SURFnet router to which the network in the lab is connected. All routes go through this router. In Listing 3 again many of the routes are omitted for the sake of readability. The first router in the complete list with a error percentage of 100% is *212.188.22.158*.

```
Bad     Total  Error percentage     Ip
84      1369   6.1%                 145.145.19.190
56       983   5.7%                 145.145.80.65
28       381   7.3%                 145.145.80.73
14       256   5.5%                 109.105.98.33
21       247   8.5%                 77.67.72.109
9         62   14.5%                62.40.124.157
...
3          3   100.0%               212.188.22.158
2          2   100.0%               146.97.33.137
2          2   100.0%               158.64.16.189
2          2   100.0%               174.35.131.38
2          2   100.0%               188.230.128.10
```

Listing 3: Location of IPv4 fragment filtering

In Listing 4 we observe the results of the hop counting algorithm for the IPv6 fragment filtering experiment on 09-07-2012 from 16:00 till 20:00. The first router in the complete list with a error percentage of 100% is *2001:16d8:aaaa:5::2*.

```
Bad     Total  Error percentage     Ip
51      507    10.1%                2001:610:158:1916:145:100:99:17
39      378    10.3%                2001:610:e08:64::65
20      168    11.9%                2001:7f8:1::a500:6939:1
6       107    5.6%                 2001:610:e08:72::73
7        66    10.6%                2001:470:0:3f::1
2        57    3.5%                 2001:948:2:6::1
...
3          3   100.0%               2001:16d8:aaaa:5::2
2          2   100.0%               2a01:270:0:ffff:720:103:0:2
2          2   100.0%               2a01:270:c:c01:c04::1
2          2   100.0%               2a01:270:dd00:300::1
2          2   100.0%               2a02:2928::3
```

Listing 4: Location of IPv6 fragment filtering

5.5 PMTU determination

Figure 23 illustrates the results of the PMTU determination experiment for IPv4. We observe the averages of the time a specific MTU was found during the six experiments periods. A complete set of data per experiment can be found in Appendix B. In Figure 23 we observe 34 different MTU sizes. We also observe that MTUs of 1420, 1460, 1492 and 1500 bytes occur more often than other sizes.

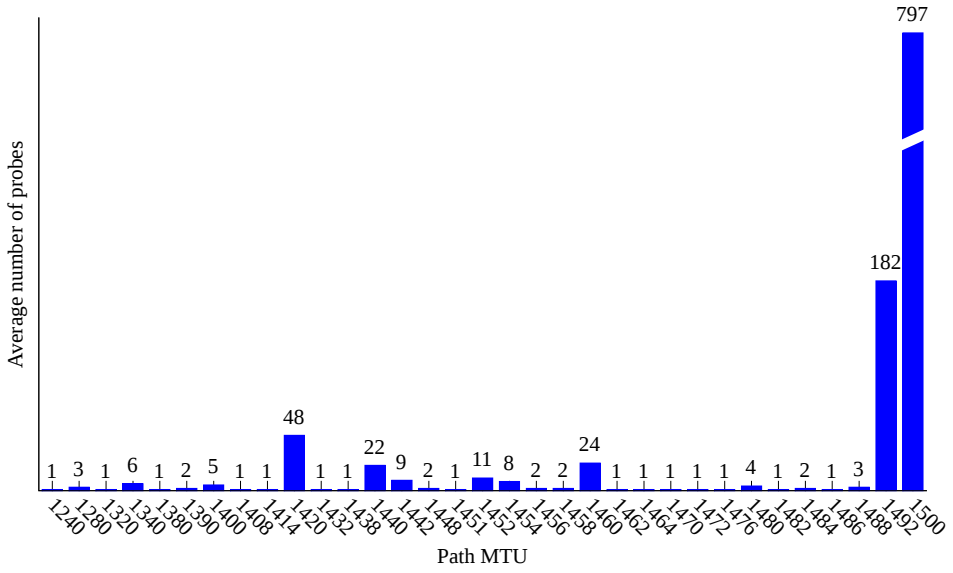

Figure 23: IPv4 common Path MTUs

In Figure 24 we observe the results of the PMTU determination experiment for IPv6. Again, we observe the averages of the time a specific MTU was found during six periods of experiments. A complete set of data per experiment is also listed in Appendix B. In Figure 24 we observe 20 different MTU sizes. We also observe that MTUs of 1280, 1440, 1472, 1480 and 1492 bytes occur more often than other sizes.

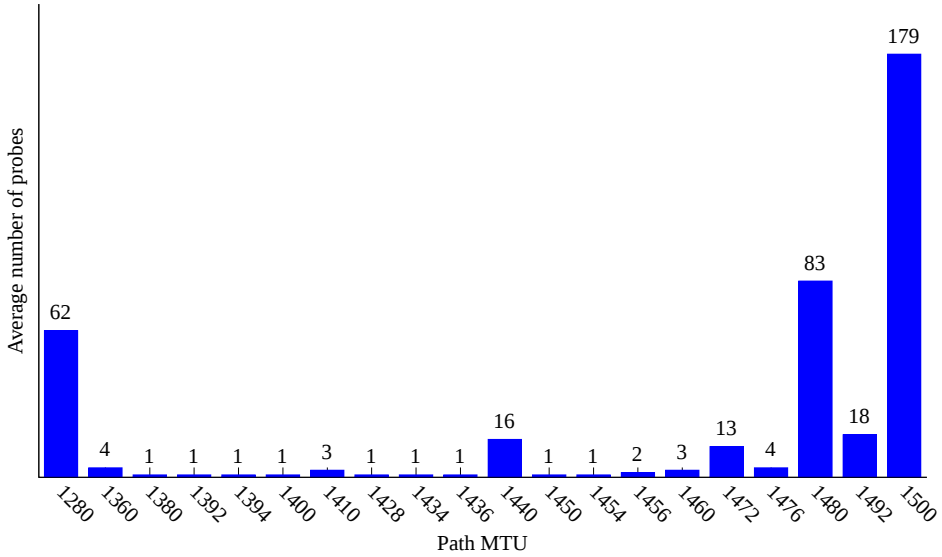

Figure 24: IPv6 common Path MTUs

6 Discussion

In Table 2 we observe that the number of IPv6-enabled probes is approximately a factor 3 less compared to the number of IPv4-enabled probes. It is good to see that the RIPE Atlas infrastructure has probes that are actually IPv6 enabled but IPv4 still has the upper hand. Since the IPv4 addresses are starting to run out[2] we believe it would not take long before more probes will become dual-stack or even IPv6 only.

When we where running the ICMP PTB packet filtering experiments we first did this with the link MTU between Belgrade and Chummi set to 1500 bytes. When we look at Figure 12 we observe that the problem is almost negligible: 0% for IPv4 and up to 0.5% for IPv6. The reason that we do not observe more paths that filtering ICMP PTB packets is probably because there is no MTU bottleneck between our experimental setup and the probes. This means that during communication, no ICMP PTB packets are send to which the probes should react so we can not really determine if something would break if a bottleneck would have existed.

To find out how probes react to ICMP PTB packets we have created an artificial bottleneck in our experimental setup. This was done between Belgrade and Chummi by configuring the MTU of the corresponding interface on Chummi to 1280 bytes. When looking at Figure 15 we observe that between 4% and 6% of the IPv4 paths filter the ICMPv4 PTB packets, which is more than for IPv6 where this is between three quarter of a percent till up to 1%. We believe the problem is worse in IPv4 because of network administrators that do not fully understand how IPv4 works and just filter every type of ICMP. There are two reason why not many people notice the problems in IPv4. The first is because most of the paths in the IPv4-driven Internet have a PMTU of 1500 bytes, as illustrated in Figure 23. The second reason is because of IPv4 routers that are allowed to fragment packets if the DF flag is not set in a packet's header, meaning that the core of the Internet solves the problem of packets that are too large. In our experiments the DF flag was set at all times.

For IPv6 the effects of the bottleneck are less than for IPv4, as also illustrated in Figure 15. In IPv6 the DF flag does not exist since routers are never allowed to fragment packets. Despite the fact that IPv6 packets are not allowed to be fragmented by routers, less paths between the probes and our experimental setup filter ICMPv6 PTB packets. We believe this is because administrators implementing IPv6 have more in-depth knowledge about the protocol. In IPv4 for most of the ICMPv4 packets it is not that big of a deal when they get filtered (except for those with type 4 and code 3), while in IPv6 there are more features that rely on ICMPv6 packets for their functioning. Administrators implementing IPv6 probably did there homework and know that filtering all ICMPv6 packets will break functionality such as router advertisements, neighbor solicitations and PMTUD.

When we look at the percentage results for the IP fragment filtering experiment where the MTU was set to 1500 bytes, as seen in Figure 18, we observe that the IPv6 paths perform way worse compared to the IPv4 paths. When the DNS server replies with a fragmented answer that does not fit through a router somewhere in the path, the node before this bottleneck generates an ICMP PTB packet and sends this back to the server. In the ICMP PTB packet enough information should be available for the DNS server to do a retransmission, but this is not the case with *ldns-testns* which just ignores the ICMP PTB packets resulting in a high error rate. IPv4 benefits from the DF flag not being set in the answer, resulting in a less higher error rate because the router in the path can solve the problem by fragmenting and reassembling the packet along the way.

If PMTUD is disabled one should set the interface MTU to the protocol minimum MTU, and by doing so, all routers in the path that honor the IPv4 and IPv6 specifications should be able to forward the fragments. The results in percentages when the MTU for our DNS server was set to the minimum MTU are illustrated in Figure 21. Figure 21 gives an accurate estimation of how many paths actually filter fragments when PMTU is not an issue. We observe that the paths to IPv6-enabled probes filter relatively more fragments compared to the IPv4 paths. This is a negative result meaning that almost 10% of the networks where IPv6-enabled probes are located, and approximately 6% of the networks where the IPv4-enabled probes are located, will get trouble with e.g. DNSSEC when retrieving large zone data. A difference between the IPv6-enabled paths and the IPv4 counterparts with respect to fragment filtering is present but relatively small. We can therefore not explain this subtitle difference.

[2] http://www.potaroo.net/tools/ipv4/index.html

To determine where along the path ICMP PTB packets and IP fragments are filtered we run the hop counting algorithm. From the data showed in Listing 1, 2, 3, and 4 we can conclude that both ICMP PTB packet and fragment filtering happens at the edges of the Internet at for example customer or company premises. When looking at the complete output of the hop counting algorithm for all experiments there are no routers that appear relatively many times in the *traceroute* output and have an error percentage of 100%. If this would have been the case, i.e. when a router would appear in for example 30 different paths between the probes and our experimental setup, this would indicate the the router is probably located closer to the core of the Internet.

However, our method is not 100% precise because we can mostly only determine if ICMP PTB packets and fragments are filtered in the European part the internet. The simple reason is that there are more probes in Europe compared to the rest of the world, resulting in a better overview of what routes are commonly used in the European part of the Internet. If a ISP router in South Korea (where only 1 probe is located) would filter ICMP PTB packets and fragments, this would show up as if 1 out of 1 paths filter these packets. We would then suspect this is a edge router, but this is not necessarily the case.

In Figure 23 we observe that the MTUs of 1420, 1460, 1492 and 1500 bytes are the most often occurring for IPv4. We can only speculate as to why a specific value is occurring more often than another. The standard MTU on Ethernet-based networks is 1500 bytes, and this explains why the MTU of 1500 bytes is occurring so often. An MTU of 1492 bytes is used by default on Ethernet LANs with Logical link control (LLC) and Subnetwork Access Protocol (SNAP) or RFC 2004 [11] compliant minimal encapsulation within IP tunnels. We suspect that the minimal encapsulation within IP tunnels are more likely the cause of this MTU occurring so often. A TCP header is 20 bytes and an IPv4 header is 20 bytes, when added to 1460 bytes adds up nicely to 1500 bytes. We suspect an MTU of 1460 bytes is due to IPv4-in-IPv4 tunnelling. The value of 1420 bytes is a bit more difficult to explain. A TCP header again is 20 bytes, but an IPv6 header is 40 bytes. Adding these numbers with an additional 1420 bytes makes 1480 bytes. So it could be because of IPv4-in-IPv6 tunnels but this is not a nice fit. It could always be the case that administrators try to keep the MTU low so that packets can be encapsulated somewhere in the path without the packet size becoming larger than 1500 bytes. Remarkable is the fact that the IPv4 minimum MTU of 576 bytes is not used at all.

In Figure 24 we observe that the MTUs of 1280, 1440, 1472, 1480, 1492 and 1500 bytes are the most often occurring for IPv6. Again, 1500 bytes is the default MTU on Ethernet-based networks and 1492 is probably because of Ethernet LANs with LLC and SNAP or RFC 2004 [11] compliant minimal encapsulation within IP tunnels. The MTU of 1472 bytes is probably because of IPv4-in-IPv6 tunnels where IPv4 options are also used, this adds another 8 bytes to the header. The 1480 bytes MTU is probably because of IPv4-in-IPv6 tunnels where IPv4 options are not used. The value of 1440 bytes is a logical decision if one thinks the IPv6 packets will be encapsulated in IPv6 packets. The value of 1280 bytes is the IPv6 minimum MTU which every IPv6-enabled node should be able to forward. Setting the MTU to 1280 bytes makes the use of PMTUD unnecessary and would be a save decision.

7 Conclusion

In this thesis we have explained what PMTU black holes are and what the implications of ICMP PTB packet filtering and IP fragment filtering have on connectivity. We explained what RIPE Atlas is, what probes are and how we are using them as vantage points to find out where ICMP PTB packets and IP fragments are filtered on the Internet. We talked about all the results we have gathered during our experiments and have discussed the results in Chapter 6.

The goal of this research was to find out where on the Internet PMTU black holes occur and if this happens more often for IPv4 than for IPv6 or vice versa. By knowing how often ICMP PTB packets and fragments are filtered, we can measure the quality of the internet and hopefully create awareness amongst network administrators, vendors and end users about the implications of filtering such packets.

We found out that ICMP PTB packets are more often filtered in IPv4-enabled paths between the probes and our experimental setup opposed to the IPv6-enabled paths. Although IPv4 seems to have more problems, probably not a lot of people notice this because the complete PMTU in IPv4 is often 1500 bytes. Moreover, the problems sometimes are transparent because the DF flag is not set in an IP packet's header. We also found that fragment filtering is a bigger problem than ICMP PTB packet filtering. In 6% of the IPv4 paths and approximately 10% of the IPv6 paths between the probes and our experimental setup fragments are filtered. These numbers get even higher when IPv6-enabled DNS servers do not act upon the receipt of ICMP PTB packets. If such a DNS server does not perform PMTUD, the replies for approximately 40% of the requests will not make it to the querying probes. By using the "hop count" algorithm we developed, we found that ICMP PTB packets and fragments are probably not filtered in the core of the Internet although we could only say this with some certainty for Europe and not for the rest of the world.

Filtering ICMP PTB packets and fragments on a network could potentially give the illusion to the end user that their connection is not working properly. Fragment filtering for example could cause a lot of problems when people are starting to use DNSSEC where fragmented replies are more often the rule than exception. Because all problems seem to occur on the edges of the Internet, i.e. at home or company premises, network administrators can solve most of the problems themselves.

In previous research the search for paths that filter ICMP PTB packets was already done, however, in our research we have verified if the numbers of these previous studies are still valid and if the situation did improve since then. Finding out where fragments are filtered was previously not possible. Because of RIPE Atlas we where able to use a large number of vantage points worldwide that we could use for our experiments. Moreover, we also had the ability to retrieve important feedback from each vantage point in order to substantiate our experiments.

8 Future work

Our research shows that problems with black holes still occur on the internet. The amount of problems related to black holes could partially give an estimation on how healthy the Internet is. By further automating the scripts we used to get to our results, a daily or even hourly measurement could be done automatically. The results for these automated measurements could then be illustrated on a webpage. It would be interesting to create several experimental setups on geographically different locations to determine if the ICMP PTB packet and fragment filtering on continents other than Europe also occurs at customer or company premises.

During our research we where unable to see if a dual-stack probe giving problems in one of the two IPs would also give problems in the other IP. It would be interesting to look into this. If this is not the case we could conclude that the IPv4 firewall configurations differ from those in IPv6 and maybe have security issues.

Our *lnds-testns* DNS server did not take any action upon receipt of ICMP PTB packets and therefore resulted in the client's query not being answered. It would be interesting to see if other DNS servers also have this problem. If this is the case, our initial thoughts were wrong and an even bigger amount of clients using these DNS servers over IPv6 would get in trouble. If DNS servers do not respond to the ICMP PTB packets other actions should be taken for a properly working IPv6 Internet.

RIPE Atlas is a great tool for many other kinds of network-related research. For example, it would be interesting to research if the IPv4 path to a location is the same as the IPv6 path to that location. In the networking world sending data over a specific path costs money one way or the other. If providers still think IPv6 is not a production technology they would probably route the IPv6 traffic over different (cheaper) paths than the IPv4 traffic.

9 Recommendations

As described in Chapter 7, the root cause of PMTU black holes are overzealous network administrators that configure their firewalls to filter ICMP PTB packets and fragments or Customer-premises equipment (CPE) which does this by default. The best way to solve this problem is through education.

For IPv6, RFC 4890 [7] proofs to be a valuable source of information when designing firewall access policies for filtering ICMP packets. This RFC is a 20 page document describing the implications and pros and cons of filtering a particular type of ICMP packet. For IPv4, there is no RFC with recommendations regarding the filtering of ICMP packets except for a draft that might prove to be fruitful [12]. Otherwise, we would like to recommend to not filter ICMP packets that are type 3 and code 4. These ICMP packets are the type that is crucial for successful PMTUD.

Because many home users just plug their routers in and do not change the default configuration settings, we would like to see that manufactures of consumer networking products by default do not filter ICMP PTB packets or IP fragments. If home users want to tinker with their firewall at home, the manufactures should supply simple documentation describing some of the problems which may arise.

Unfortunately, we do not live in a perfect world and it would be next to impossible to educate every home user and network administrator about how to configure their firewall correctly. In RFC 4821 [6], a technique named packetization layer PMTUD is described as an alternative for the regular PMTUD. This technique does not rely on ICMP packets but on TCP or some other packetization layer to determine the PMTU. It does so by probing the path with progressively larger packets. Packetization layer PMTUD should be enabled at the endpoint where large packets originate from. For example, when hosting a website, small requests come in and the web server then probes the path using packetization layer PMTUD to determine the PMTU. The major advantage of this technique is that works even if somewhere in the path ICMP PTB packets are filtered.

An implementation of packetization layer PMTUD is available by default in the Linux kernel starting from version 2.6.17.[3] However, it is turned off by default so must be enabled manually. This is done by changing the value in */proc/sys/net/ipv4/tcp_mtu_probing*. A value of 0 means off, a value of 1 indicates that packetization layer PMTUD should only be used after a black hole is detected, and when set to 2 packetization layer PMTUD is always performed.

In Microsoft Windows a comparable feature called Path MTU Black Hole (PMTUBH) detection is available.[4] It is also disabled by default and can be enabled by adding *EnablePMTUBHDetect* set to 1 at the following location in the registry: *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters*.

There are two other ways to circumvent the problems caused by ICMP PTB packet filtering, however, we do not consider these best practices. First, reducing the MTU to the minimum MTU of the used IP, which is 576 bytes for IPv4 and 1280 bytes for IPv6. Doing so will affect the TCP MSS that is advertised to the other side during the TCP three-way handshake. The other side will adjust its TCP MSS to the lowest advertised value, and will therefore never send packets larger than this value for the duration of the TCP session. Because the minimum MTU is advertised, every intermediate node should be able to route the packets and as a result, PMTU black holes will not occur.

The second way to circumvent the problems with ICMP PTB packet filtering is to configure a firewall to perform an operation that is known as TCP MSS clamping. Basically, MSS clamping has the same effect as reducing the MTU. Only this time the firewall will rewrite every first packet of every TCP three-way handshake that is processed by the firewall to the MSS you configure it for.

[3] http://staff.psc.edu/jheffner/projects/mtup/
[4] http://technet.microsoft.com/en-us/library/cc958871.aspx

Obviously, lowering the MTU of an interface will reduce the efficiency of the IP. By adding smaller payloads to packets the protocol becomes less efficient and, since routers need to inspect every packet's header, it is better to have larger payloads so less headers needs to be inspected. Another problem is that TCP MSS clamping will not solve the problem for UDP-based traffic. For UDP no real solutions exists except for convincing network administrator not to filter IP fragments.

Lastly, Gijs van den Broek et al. [13] proposes two solutions for DNS servers that do not act upon the receipt of ICMP PTB packets. There first solution is to minimize the MTU on the serving interface of DNS servers as already has been described above. The second solution is a more difficult one that should be implemented in software. The DNS software should alter the query result in smaller responses and avoid fragmentation of the DNS reply. Both solutions would leave firewalls and DNS resolvers unchanged.

Bibliography

[1] G. Huston, "A Tale of Two Protocols: IPv4, IPv6, MTUs and Fragmentation," *The ISP Column*, 2009.

[2] J. McCann, S. Deering, and J. Mogul, "Path MTU Discovery for IP version 6," *Internet RFC 1981*, 1996.

[3] M. Luckie and B. Stasiewicz, "Measuring Path MTU Discovery Behaviour," 2010.

[4] M. Luckie, K. Cho, and B. Owens, "Inferring and Debugging Path MTU Discovery Failures," 2005.

[5] K. Lahey, "TCP Problems with Path MTU Discovery," *Internet RFC 2923*, 2000.

[6] M. Mathis and J. Heffner, "Packetization Layer Path MTU Discovery," *Internet RFC 4821*, 2007.

[7] E. Davies and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls," *Internet RFC 4890*, 2007.

[8] L. Colitti, G. D. Battista, and M. Patrignani, "IPv6-in-IPv4 Tunnel Discovery: Methods and Experimental Results," *eTransactions on Network and Service Management*, April 2004.

[9] J. Postel, "Internet Protocol," *Internet RFC 791*, 1981.

[10] J. Mogul and S. Deering, "Path MTU Discovery," *Internet RFC 1191*, 1990.

[11] C. Perkins, "Minimal encapsulation within IP," *Internet RFC 2004*, 2004.

[12] F. Gont, G. Gont, and C. Pignataro, "Recommendations for filtering ICMP messages," *draft-ietf-opsec-icmp-filtering-03*, 2012.

[13] Gijs van den Broek, Roland van Rijswijk, Aiko Pras, and Anna Sperotto, "DNSSEC Meets the Real-World: Improving Response Deliverability," *MSc.Thesis, University of Twente, The Netherlands. (Private communication.)*, 2012.

Glossary

ACK	Acknowledgement
APNIC	Asia-Pacific Network Information Centre
CPE	Customer-premises equipment
DF	Don't Fragment
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
DoS	Denial of Service
FQDN	Fully Qualified Domain Name
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ICMPv4	Internet Control Message Protocol version 4
ICMPv6	Internet Control Message Protocol version 6
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISP	Internet Service Provider
LLC	Logical link control
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NIC	Network Interface Card
OSI	Open Systems Interconnection
PMTU	Path MTU
PMTUBH	Path MTU Black Hole
PMTUD	Path MTU Discovery
PTB	Packet Too Big
RFC	Request for Comments
RTT	Round Trip Time
SNAP	Subnetwork Access Protocol
TCP	Transmission Control Protocol
UDM	User Defined Measurement
UDP	User Datagram Protocol

A RIPE Atlas probes per country

The graph in Figure 25 illustrates the countries that have at least 25 probes, and the graph in Figure 26 those that have less than 25 but at least 5. Countries with less then 5 probes are omitted.

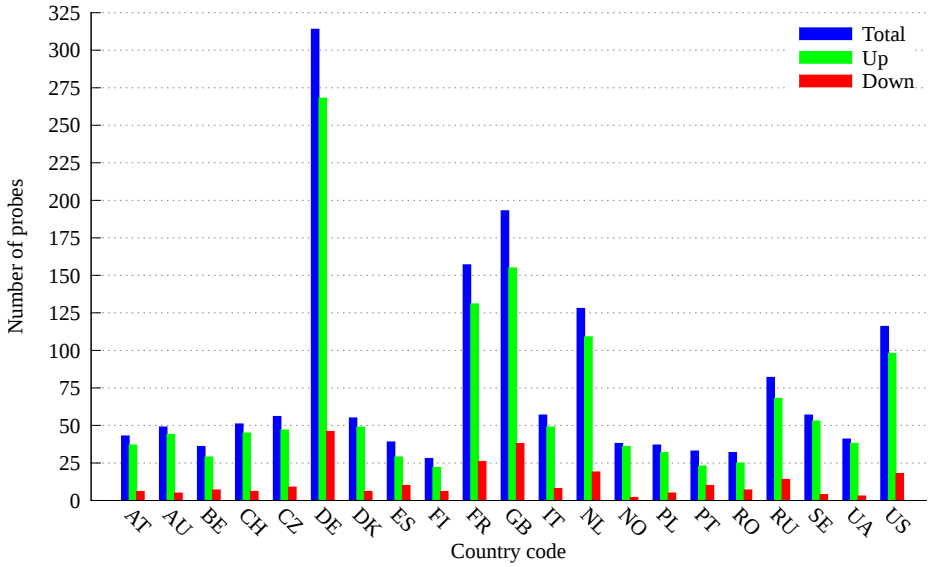

Figure 25: RIPE Atlas probes per country (high)

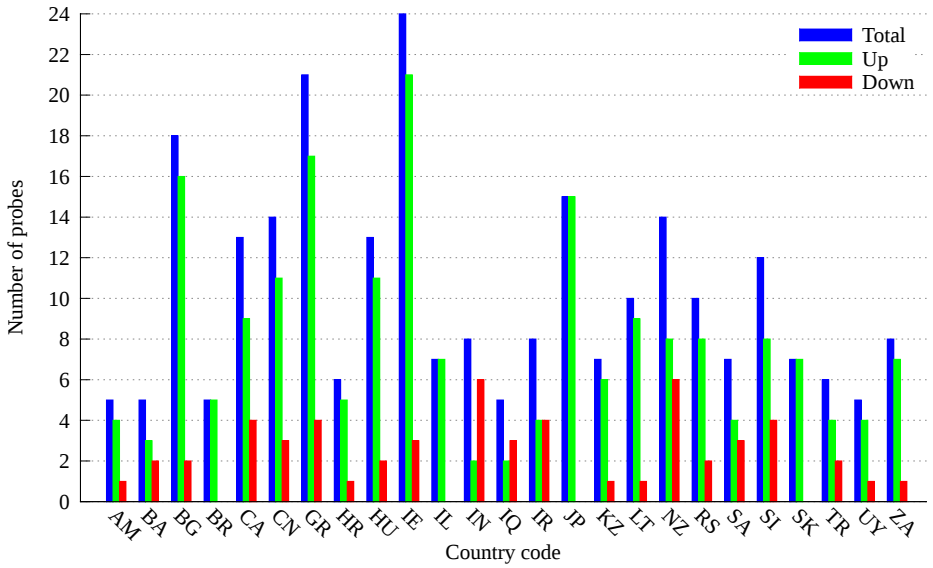

Figure 26: RIPE Atlas probes per country (low)

B PMTU determination

	Period 1		Period 2		Period 3		Period 4		Period 5		Period 6	
Path MTU	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4	IPv6
1240	1		1		1							
1280	3	64	3	64	3	62	3	64	3	59	3	58
1320	1		1		1		1		1		1	
1340	5		6		6		7		7		6	
1360		3		4		4		4		4		4
1380	1	1	1	1	1	1	1	1	1	1	1	1
1390	2		2		2		2		2		2	
1392		1		1		1		1		1		1
1394		1		1		1		1		1		1
1400	5	1	5	1	5	1	5	1	4	1	3	1
1408	1		1		1		1		1		1	
1410		3		3		3		3		3		3
1414	1		1		1		1		1		1	
1420	47		47		48		50		49		48	
1428		1		1		1		1		1		1
1432	1		1		1		1		1		1	
1434		1		1		1		1		1		1
1436		1		1		1		1		1		
1438	1		1		1		1		1		1	
1440	20	16	21	16	24	15	22	16	22	16	21	16
1442	8		8		9		9		9		9	
1448	2		2		2		2		2		2	
1450	1	1	1	1		1		1		1		1
1451	1		1		1		1		1		1	
1452	12		12		12		10		9		9	
1454	7	1	8	1	7	1	8	1	10	1	8	1
1456	2	2	2	2	2	2	2	2	2	2	2	2
1458	2		2		2		2		2		2	
1460	20	2	23	3	22	3	27	3	25	3	25	3
1462	1		1		1		1		1		1	
1464	1		1		1		1		1		1	
1470	1		1		1		1		1		1	
1472	1	13	1	13	1	13	1	12	1	12	1	12
1476	1	4	1	4	1	4	1	4	1	4	1	4
1480	4	79	4	82	4	83	4	86	4	85	3	81
1482	1		1		1		1		1		1	
1484	2		2		2		2		2		2	
1486							1		1		1	
1488	3		4		3		3		3		3	
1492	189	19	186	19	182	19	181	20	175	17	178	16
1496	1		1									
1500	800	181	803	181	806	181	812	183	781	173	777	176
Total number of probes	1149	395	1156	400	1155	398	1165	406	1125	387	1117	383

Figure 27: IP Path MTUs (complete)