University of Amsterdam

System and Network Engineering

Research Project 1

# Making do with what we've got: Using PMTUD for a higher DNS responsiveness

*Authors:*
Hanieh Bagheri
hanieh.bagheri@os3.nl

Victor Boteanu
victor.boteanu@os3.nl

*Supervisors:*
Willem Toorop
willem@nlnetlabs.nl

Benno Overeinder
benno@nlnetlabs.nl

February 28, 2013

**Abstract**

The goal of this research project is to find out if ICMPv6 Packet Too Big (PTB) messages can be used to improve a name server's responsiveness.

Path MTU Discovery (PMTUD) is a mechanism used to signal the sender that its packet does not fit in the MTU of a link along its path. The sender receives a PTB message, which contains the MTU of the next link. In IPv6, the ICMPv6 PTB also contains as much of the original packet that would fit on the next link.

A DNS response includes the question that generated the response. However, DNS name servers are stateless and do not remember the queries that generated the response. Once a question has been answered, it is forgotten. A client would have to timeout and requery the server to get the answer.

The aim of this research is to make use of the information in ICMPv6 PTB messages to give a sense of state to name servers, and resend the response to the client. This would improve the responsiveness of the name server, as it does not have to wait for the client to time out and resend the query. Three solutions of using PTBs are presented in this report, and we covered the effects they would have if implemented. Packet captures provided by SURFnet and SIDN are analyzed to provide statistical data from real name servers. A test implementation of one of the solutions is performed using the RIPE Atlas network, and observations are made on the results.

The main conclusion of this paper is that PTB messages do contain enough information to be used by name servers to resend the response to the client.

# Contents

## List of Figures

## List of Tables

## 1 Introduction

The Domain Name System (DNS) is an essential component of the current Internet. It is responsible for associating information, such as IP addresses, with domain names. DNS normally uses User Datagram Protocol (UDP) as the transport layer protocol. UDP is an ”stateless” and ”unreliable” protocol, in that it does not implement acknowledgements or retransmission. By using UDP, DNS is relatively fast, and if a client does not get a response in due time, it can query the server again, or try a different server.

The classical DNS protocol limits the response size over UDP to 512 bytes. When a response does not fit in the available size, it will be truncated. In this case the TC flag in the response is set and the client should resend the query over TCP.

Original DNS Does not have any security mechanism to prevent forging the source or the contents of the response. The Domain Name System Security Extensions (DNSSEC) is introduced to provide authenticity and origin integrity. DNSSEC causes the zone information to be cryptographically signed. Those cryptographic signatures are conveyed in the responses, resulting into larger responses.

With the growing trend towards using DNSSEC (especially after the root zones being signed in 2010), we expect larger DNS responses to become more frequent. The Extension mechanisms for DNS (EDNS) introduced (among others) a mechanism for DNS clients and servers to agree on a larger response size over UDP. With EDNS, the name servers are allowed to exchange larger responses over UDP.

The immediate effect of having large DNS messages over UDP is the possibility of fragmentation. Every link in the Internet is capable of transmitting packets with a certain size, which is called Maximum Transmission Unit (MTU). Packets larger than this limit must be fragmented before being sent over that link. As shown in Figure 1, using UDP, when the response is smaller than the size defined in EDNS option, but it is not small enough to be sent as a single network packet, it has to be fragmented.

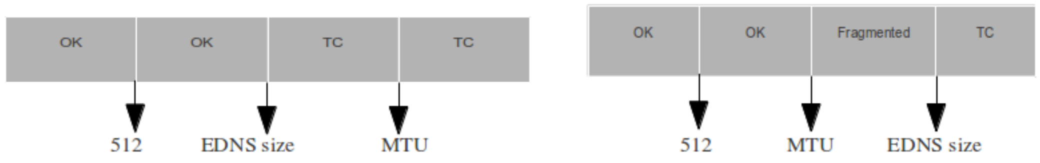


Figure 1: Effect of response size on its status over UDP

Although fragmentation addresses the issue with large DNS responses that are sent over UDP, the client-side firewalls might filter fragments [1]. In IPv6, routers are not allowed to fragment the packets. Fragmentation must be done by the end hosts. Those end hosts have to do Path MTU Discovery (PMTUD) to find out the smallest MTU in the path towards the destination. In the PMTUD process, the source sends a packet into the network. If the packet encounters a router with a smaller MTU, it is dropped and an ICMPv6 PTB is sent back to the source. There is a similar message in IPv4, called

"ICMP fragmentation needed". The main difference between the ICMPv6 PTB and its corresponding message in IP4 is that in the former, in addition to the passable MTU by the router, it contains the trimmed part of the original message. So looking at the PTB, one can see the part of the message that can traverse the path trough the router.

## 1.1 Research objective

In this research, we are interested in leveraging the extra payload in ICMPv6 to increase responsiveness of name servers in the case of having large responses. The advantage of ICMPv6 PTB, which makes it helpful for our problem is that it contains the first part of the payload of the dropped message, which can fit into the minimum MTU supported by all IPv6 routers (1280 bytes). Having the returned ICMPv6 PTB, the server would be able to find out the original query, in addition to the appropriate MTU size. In this research, we will explore several approaches to utilize this information.

## 1.2 Research question

This project is mainly aimed to answer the following question:

*Is it feasible to leverage the original packet payload in ICMPv6 to increase a name server responsiveness?*

The sub-questions that will be answered in this report will be:

*What strategies can be applied and what effects and risks would they have?*

*How would this be best implemented?*

*What is the impact of fragmentation and PMTUD on the performance/responsiveness of DNS servers?*

## 1.3 Related work

Previous research related to the topic of this project has been done by Maikel de Boer and Jeffrey Bosma, on the IPv6 Path MTU black holes, caused by ICMPv6 filters or fragment filters [1]. They made use of the RIPE Atlas network in order to find ICMPv6 filters or fragment filters, which cause IPv6 Path MTU black holes. Their research is relevant to our project, because it gives us information regarding IP fragment filtering, which is one of the issues that we have to take into consideration for this project.

In a recent research by Gijs van den Broek et. al. [4], real-world resolvers are monitored to analyze their behavior, when dealing with fragmented DNS responses. In this work, two server-side solutions are offered and tested to prevent the responses from being fragmented. The idea in the first solution is to set the maximum response size to 1232 bytes. The second solution is more complicated. It needs to implement a module for dynamically adjusting the EDNS size option in queries related to the resolvers behind fragment-filtering firewalls.

There is an Internet-draft written by Mark Andrews [5], mentioning that PMTUD does not work well with the stateless DNS, because there is no automatic mechanism for handling PTB messages in name servers. He recommends to enable IPV6_USE_MIN_MTU socket option in IPv6 to fragment packets at the minimum IPv6 MTU, instead of using PMTUD. In other words, the name servers are encouraged to fragment the responses at the smallest possible size, in order to avoid receiving PTB messages.

## 1.4 Outline

The core concepts of fragmentation and PMTUD are presented in Section 2. Packet Too Bid and Fragment Reassembly Time Exceeded messages are discussed in Subsection 2.3 and we describe their importance to this research. In section 3, we describe the resources that were used. The packet captures from SURFnet and SIDN are treated in separate subsections, as well as the initial RIPE Atlas experiment. The proposed solutions are presented in 4, followed by their implications and implementation options. A test implementation of solution 2 is described and analyzed in Section 5, followed by the conclusions of this paper in Section 6.

# 2 Background concepts

In this chapter, protocols and mechanisms relevant to our research are discussed more in-depth.

## 2.1 Fragmentation: IPv4 vs IPv6

In IPv4, fragmentation and reassembly of packets can be handled by the routers in the middle, completely transparent for the end-nodes. When a router receives a packet larger than the MTU of the next link, it can break it into multiple smaller "fragments" not larger than that MTU. Routers further down the routing-path of a packet can reassemble those fragments to reconstruct the original packets.

End-nodes may set the Don't Fragment (DF) bit in the IPv4-header to indicate that routers may not perform fragmentation. In this case, when a router receives a packet larger than next link's MTU, an ICMP "Fragmentation Needed" message containing the MTU is returned. The sender should resend smaller packets according to the new MTU. This sequence of events may repeat until the data reaches the destination. This process is called Path MTU Discovery (PMTUD) and tries to find the smallest MTU in the path. PMTUD is explained in more detail in subsection 2.2. With IPv6, fragmentation is only allowed end-to-end and nodes run a PMTUD algorithm to discover the proper packet size that can go through the path. When a router receives a packet larger than its MTU size, it drops the packet and sends an ICMPv6 Packet Too Big (PTB) message back to the sender.

Traditionally, name servers do not work well with PTB. Name servers do not keep state, and thus do not remember queries or responses. Upon reception of an IPv4 "Fragmentation Needed" messages, the name server cannot reproduce the answer and the requester will not get it. Current name servers handle IPv6 PTB similarly and simply ignore them.

Modern name servers avoid PTB to occur at all, by fragmenting to 1280 bytes. However, unlike the IPv4's PTB compeer [1], the IPv6 PTB contains a fair piece of the original payload.

One way to circumvent PMTUD with IPv6 is to make sure no packets larger than the minimum PMTU are sent [5]. With this method, all responses larger than 1232 bytes (1280 bytes - 8 bytes UDP header - 40 bytes IP header) will be fragmented.

Another way, is to make sure that the maximum (EDNS) response size is 1232 bytes [4]. This will prevent PMTUD and fragmentation, but will result in more truncated responses. More truncated responses will in turn lead to more TCP connections to the name server, which increases the load of the name server (because of the overhead of the state that needs to maintained for TCP).

[1]Destination unreachable, the datagram is too big. Packet fragmentation is required, but the "Don't fragment" (DF) flag is on

## 2.2 PMTUD

When an IPv6 host sends a large message to another host over UDP, it is fragmented into multiple IPv6 packets. The size of the fragments should be the largest value that can successfully go through the path between the two nodes. This size is equal to the minimum link MTU of all the links in a path and is called Path MTU (PMTU). The MTU is dynamically discovered using a Path MTU Discovery (PMTUD) mechanism.

In PMTUD, first the source sends the message using the MTU of the first link in the path. If any of the routers in the path has a smaller MTU, it will drop the packet and return an ICMPv6 Packet Too Big (PTB) message to the source. By receiving such a message, the source decreases its optimistic assumption of the PMTU. The same process might reoccur when the source sends the message again using the new PMTU. The value of the PMTU is cached at the source (by destination IP) and new traffic will use the learned PMTU.

Changes in the routing topology may cause the PMTU of a path change over time. Estimated PMTU in the source is reduced after receiving PTB messages. Each host increases its estimation of PMTU periodically to become aware of the likely increases in PMTU. In most cases, this increasing causes packet dropping and PTB generation. Therefore, it is recommended that the time interval between two increments to be a large enough. In other words, when the host learns the Path MTU, it must not perform PMTUD for discovering larger MTUs for a certain amount of time (5 minutes) and the recommended setting is twice of this value (10 minutes) [3].

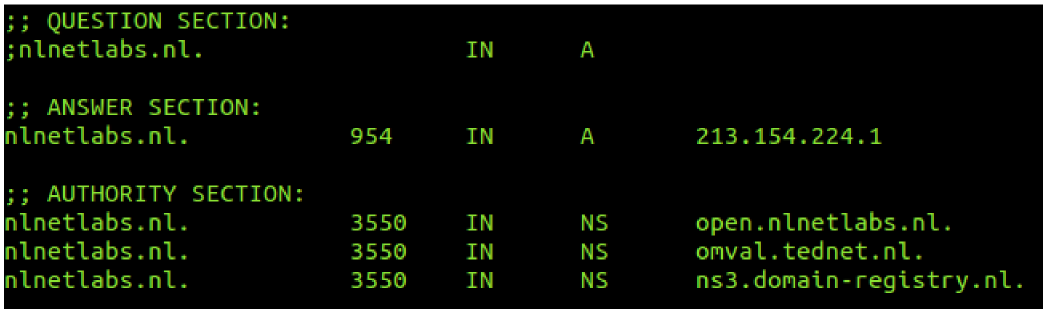
In the case of TCP communication, the maximum segment size (MSS) is advertised at the beginning of the TCP session. After the two nodes participating in a session have advertised their preferred MSS, the lowest one will be chosen for the session. The advertised MSS, in case of IPv6, will be 60 bytes smaller than the MTU, which is needed for headers. The MSS is adapted after the receipt of a PTB message, as the PTB adjusts the MTU on the source. This is possible over TCP, but not when using UDP, because with TCP the packets are remembered until they are acknowledged.

## 2.3 ICMPv6

This section will describe the ICMPv6 messages used in this project, why they are relevant, and how we make use of the information contained in them. The section is based on RFC 4443, in which the ICMPv6 protocol is specified.

### 2.3.1 Packet Too Big

A Packet Too Big (PTB) message will be originated by a router along a path when a packet will not be able to fit through the next link of that router along that path. It will compare the size of the packet to the MTU of its next link, and if the size of the packet will be larger, it will send back an ICMPv6 PTB message to the source that sent that packet. In our case the source will be the DNS server.



```
;; QUESTION SECTION:
;nlnetlabs.nl.                  IN      A

;; ANSWER SECTION:
nlnetlabs.nl.           954     IN      A       213.154.224.1

;; AUTHORITY SECTION:
nlnetlabs.nl.           3550    IN      NS      open.nlnetlabs.nl.
nlnetlabs.nl.           3550    IN      NS      omval.tednet.nl.
nlnetlabs.nl.           3550    IN      NS      ns3.domain-registry.nl.
```

Figure 2: DNS response

An example of a DNS response is shown in Figure 2, with the question section as part of the response.

The format of a PTB message [2] is illustrated in Figure 3:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             MTU                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    As much of invoking packet                 |
+               as possible without the ICMPv6 packet           +
|               exceeding the minimum IPv6 MTU [IPv6]           |
```

Figure 3: Packet Too Big message format

Operating systems use PTB messages to store the MTU of that path. Next time it will have to send a message on that same path, it will use the new MTU instead.

When a smaller MTU is present behind the signalling router, the process repeats itself until the minimum MTU up to the destination is learned. For example, if the initial MTU is 1500 bytes, but there is a link along the path to a destination with MTU of 1280 bytes, the router with that link will originate a PTB. This process is shown in Figure 4.

One of the focuses of this project is to make extra use of PTB messages. Up to now, these messages have been ignored by name servers, but we aim at extending their functionality.

### 2.3.2 Time Exceeded Message

Another commonly found ICMPv6 error message during this project was the Time Exceeded Message. There are actually two types of TEM messages: Hop limit exceeded in transit, and Fragment Reassembly time exceeded.

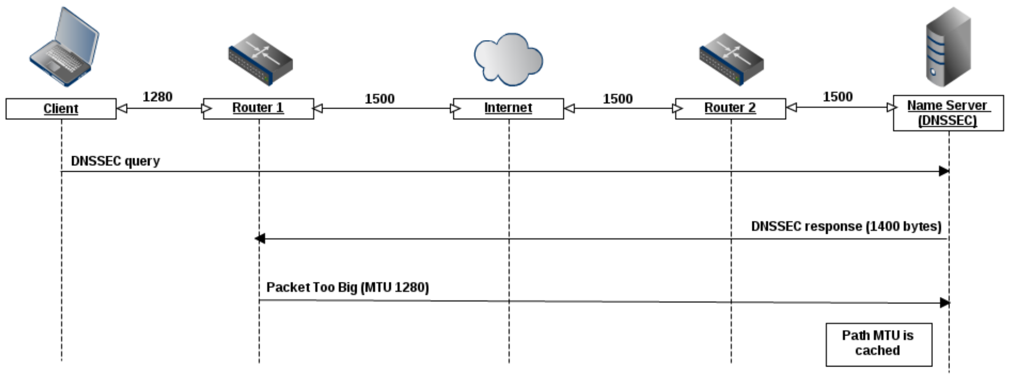


Figure 4: Path MTU Discovery

A Fragment Reassembly Time Exceeded (FRTE) message is originated from an end-host, if it did not succeed in reassembling all the fragments of a packet. Section 4.5 of RFC 2460 (specification of IPv6) states that if the packet cannot be reassembled in 60 seconds after the reception of the first fragment, a Fragment Reassembly Time Exceeded message will have to be sent back to the source.

All ICMPv6 messages with type <100 contain the same last field as the PTB, in which as much of the original message is stored. The format of Time Exceeded messages is almost identical to the format of PTB messages, and will thus not be repeated. The two differences are:

- The Code field is used to identify either Hop Limit Exceeded messages (code=0) or Fragment Reassembly time exceeded messages (code=1)
- The MTU field found in PTB messages is not used in this case, and set to 0.

FRTE messages are relevant to this project, because it is an indication that fragments are being filtered along the path. This packet will reach its destination, triggering the timeout counter, in which all fragments should be delivered. However, no other fragments will be received due to the fragment-filtering firewalls, and thus, the client will send back an ICMPv6 Fragment Reassembly Time Exceeded message.

Although these messages aid a lot in detecting the problem, we will not be using them in any of our solutions. The reason for this is that the requester will wait for 60 seconds before sending back an FRTE. However, they are of great use in troubleshooting fragmentation issues.

# 3 Real-world observations

During this research project, we were provided with packet traces captured on name servers from both SURFnet and SIDN (registry for .nl ccTLD). The captures from SURFnet span a one hour period, and were done on 8 name servers. SIDN captures were done over a period of two hours. The data we gathered from these traces is discussed in the following subsections.

The experimental setup we used consisted of a name server located on the NLNOG Ring, which was queried by probes in the RIPE Atlas network. The NLNOG Ring is a network of servers, on which every member of the Ring has access, while in turn giving access to other members to its server. Its creation was an initiative to streamline cooperation between network operators, which would commonly make ”shell access” deals with other network operators. Using the Ring can give a very useful outside point of view of one's network. The server was only running IPv6 on its eth0 interface, and NSD 3.2.14 was used for the name server. The name server was modified to send out packets larger than 1280 bytes, as opposed to draft-andrews-dnsext-udp-fragmentation-01 [5].

The Atlas experiment and the analysis of the packet traces from SURFnet and SIDN have been carried out to show the relevance of this research project. These will provide a baseline in order to determine the effect of our proposed solutions.

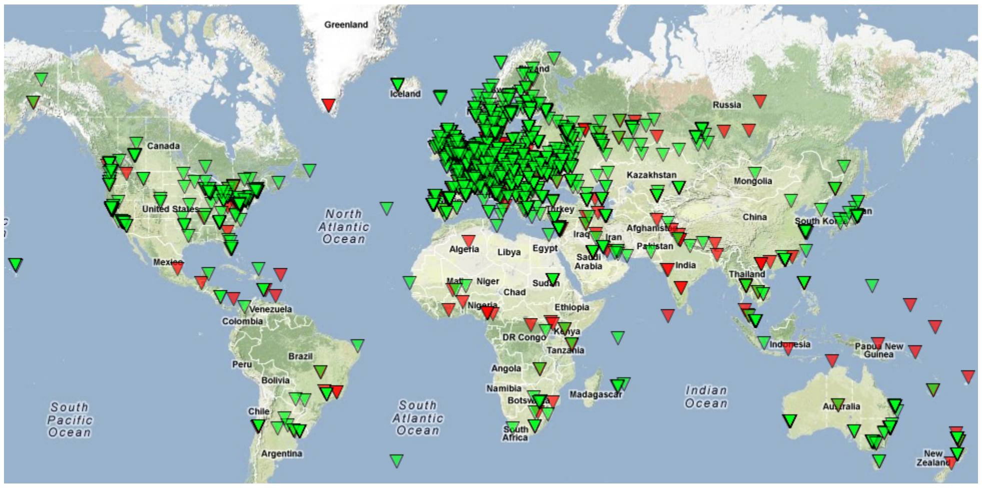


Figure 5: RIPE Atlas network of probes

RIPE Atlas is a real-time Internet measurement network. It is a project that was started by RIPE NCC and aims at deploying thousands of probes, which are used to measure the infrastructure of the Internet in real-time. Figure 5 shows the distribution of probes around the world. Probes are small USB devices that can be used to run various network measurements, defined in the interface provided by RIPE Atlas. Because our project focuses on IPv6 protocols, all the probes used in the experiment had to be IPv6-ready. The 2012 review states that there are over 800 IPv6-ready Atlas probes [6].

## 3.1 SURFnet packet captures

The SURFnet traces are the result of capturing the DNS traffic at 8 different name servers, during 1 hour, on January 15, 2013. All of the authoritative name servers are running BIND 9.8.2 and the resolvers run Unbound 1.4.19.

Some information obtained from these captures are demonstrated in Table 1. Although the ratio of IPv6 messages are way less than the IPv4 ones, it is expected that in near future, we see a significant growth in utilizing IPv6 addresses.

Table 1: SURFnet name servers IPv4/IPv6 ratio

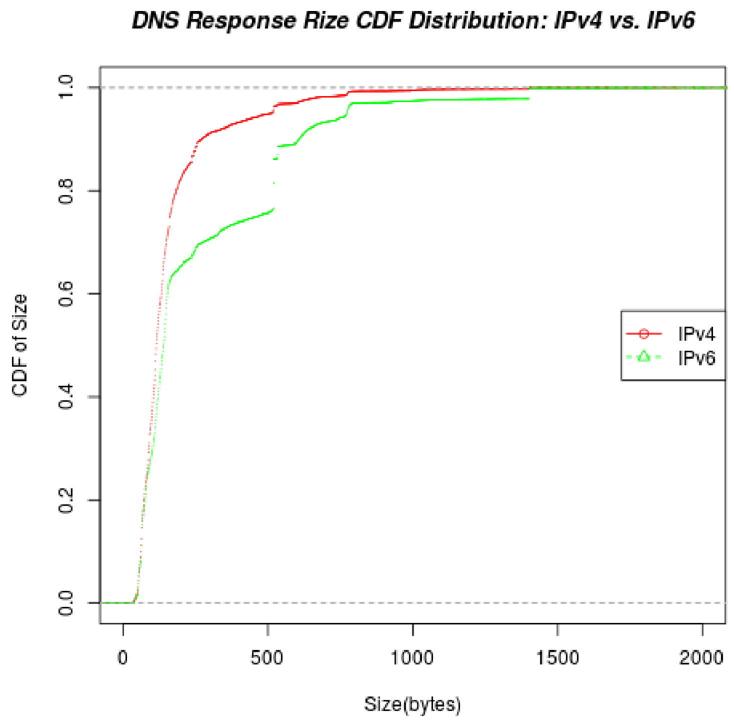
| Server | IPv4 | IPv6 | Number of Packets |
|---|---|---|---|
| ns0.amsterdam1.surf.net | 94.49% | 5.51% | 5915213 (29.57%) |
| ns0.nijmegen1.surf.net | 91.58% | 8.42% | 3999195 (19.99%) |
| ns0.tilburg1.surf.net | 84.11% | 15.89% | 5386692 (26.93%) |
| ns1.surfnet.nl | 87.18% | 12.82% | 2699609 (13.50%) |
| ns1.zurich.surf.net | 91.69% | 8.31% | 383691 (1.92%) |
| ns2.surfnet.nl | 93.16% | 6.84% | 1009883 (5.05%) |
| ns3.surfnet.nl | 92.95% | 7.055% | 510285 (2.55%) |
| ns6.amsterdam1.surf.net | 90.39% | 9.61% | 99444 (0.50%) |
|  | 89.95% | 10.05% | 20004012 (100%) |

Figure 3.1 compares the cumulative response size distribution in IPv4 versus IPv6. It can be inferred that dealing with large DNS responses is a more important issue in IPv6. Our hypothesis about the cause of this observation is that utilization of DNSSEC is more common in IPv6. One explanation is the people who support a pretty novel protocol, like IPv6, also care more about using security extensions like DNSSEC. We will come back to this issue again in this section.

Another observation in the SURFnet traces is related to the ratio of UDP and TCP packets. In case of IPv4, about 0.008% of responses are sent over TCP and in Ipv6 traffic, this ratio is even less, about 0.004%. According to the previous observation, in IPv6, DNS responses are generally larger than in the ones in IPv4 case, so one might expect to see more truncated responses in IPv6. However, it seems that the clients who are utilizing IPv6, configure the size option in EDNS to a large value.

*Note*: From what discussed above, it can be inferred that having large DNS responses is more common in IPv6 rather than IPv4 and using IPv6 is evergrowing. Therefore, investigating the consequences of having such messages in IPv6 is a crucial issue. Therefore, in this section, we focused on the IPv6, and by DNS response, UDP, TCP or IP packets and other similar messages, we are referring to the corresponding concepts in IPv6.

One of our concerns about the traces is finding the information about the large DNS responses and figuring out how the name servers deal with these messages. As shown in Table 2, most of DNS responses are smaller than 1232 bytes and they are small enough to fit in a single IP packet. A small number of the messages are larger than 1452 bytes and must be fragmented before being sent over UDP. For the messages in the middle range (between 1232 and 1452 bytes), depending on the policy, they may or may not be



fragmented. For example, given a DNS response with the size of 1300 bytes, if generally, the responses are sent by breaking them into 1280-byte IP fragments, the given response has to be sent as two fragments. However, if the server apply fragmentation by size of 1480 bytes, the message can go through the network without any need to fragmentation.

Table 2: SURFnet response sizes

| Sizes | Number of responses |
|---|---:|
| 1-1232 bytes | 97.77% |
| 1233-1452 bytes | 2.14% |
| 1453-65535 bytes | 0.07% |

As we discussed in Chapter 1, the main cause of appearing large DNS responses is signing the zone data using DNSSEC. The statistics obtained from the captures show that from the total IPv6 traffic 30.04% is related to DNSSEC traffic and almost all of it is transferred over UDP (TCP share is only 0.003%).

We also inspect the traces to see if there are any non-DNSSEC messages, which cause having large messages. The results of this inspection is demonstrated in Table 3. It is noticeable that DNSSEC is the most important reason for the DNS responses to become large. We noticed that some other big responses also exists in the captures which are non DNSSEC. However, there are a few of them and they are caused by very large TXT records, multiple TXT records, or having multiple PTR records in the RRset of the answer.

Table 3: SURFnet large response

| | 1233-1452 Bytes | 1453-65535 Bytes | |
|---|---|---|---|
| DNSSEC | 43165 | 1544 | 44709 (99.95%) |
| large TXT records | 1 | 6 | 7 (0.02%) |
| Multipe PTR records | 0 | 17 | 17 (0.04%) |
| | 43166 | 1567 | 44733 |

According to this table, our hypothesis about the first observation seems to be true. Nearly, all of the large responses in IPv6 are resulted from signing the zone data using DNSSEC. Having large responses in IPv6 is more common than IPv4, because DNSSEC is more popular among IPv6 users.

Another matter of interest for us is the number of fragmented responses. Table 4 shows the number of fragmented responses, grouped according to their sizes. As we except larger responses are more influenced by fragmentation. Contrary to our expectation, most of the messages with the size in range between 1232 and 1452 are responded without being fragmented. In this case, we expected to see many PTB messages.

Table 4: SURFnet fragmented responses

| | 1232-1452 Bytes | 1453-65535 Bytes | Total |
|---|---|---|---|
| Fragmented DNS responses | 43 (7.28%) | 548 (92.72%) | 591 |

Table 5 shows the payload length of each fragment in the fragmented messages. The two popular sizes for fragmentation by the servers is 1240 and 1456. It means that most of the messages are fragmented with MTU of 1280 or 1496. The name servers that use MTU of 1280 are the ones that follow the recommended approach in [5]. The ones that fragment at size of 1496 are using an MTU of 1500 bytes. This is because the "fragment offset" field in the fragment header is counted in multiples of 8 bytes.

Table 5: SURFnet fragment payload length

| Payload Length | Number of DNS responses |
|---|---|
| 1224 | 2 (0.34%) |
| 1240 | 231 (39.09%) |
| 1456 | 357 (60.41%) |
| 416 | 1 (0.17%) |

Table 6 shows the percentage of ICMPv6 messages with respect to the total number of IPv6 DNS responses. The ratio of PTB messages are not as big as we expected, so we cannot explain the observations related to Table 4.

Table 6: SURFnet ICMPv6

| Type of message | SURFnet |
|---|---:|
| Time Exceeded Fragment Reassembly | 26(0.001%) |
| Packet Too Big | 16(0.001%) |
| Administratively prohibited | 3624(0.180%) |

## 3.2 SIDN packet captures

Stichting Internet Domeinregistratie Nederland (SIDN) is the official registrar and manager for the country-code top-level domain (cc-TLD) of .nl. SIDN name servers are responsible for the .nl zone. The SIDN traces are obtained from capturing the IPv6 DNS traffic on one of the SIDN name servers for duration of 2 hours, on February 4, 2013.

As shown in Table 7, similar to the SURFnet traces, most of the response messages are still less than or equal to 1232 bytes. The difference is in responses larger than 1232 bytes. For the case of SURFnet, we observed that the number of responses in the range between and 1232 and 1452 is almost 30 times more than the the number of messages larger than 1452 bytes. We noticed that in the SIDN captures, the number of responses larger than 1452 is almost 165 times more than the number of messages in the range between 1232 and 1452. The reason is there are many queries asking about the NS records signed using DNSSEC. we will come back to this issue later, in this section.

Table 7: SIDN response sizes

| Sizes | Number of responses |
|---|---:|
| 1-1232 bytes | 99.67% |
| 1233-1452 bytes | 0.002% |
| 1453-65535 bytes | 0.33% |

We also noticed that there is no truncated response in the SIDN traces. It is a result of advertising large EDNS size options by the clients.

We also looked into the DNSSEC traffic. About 54.42% of the total DNS traffic was signed using DNSSEC. This ratio is more than the observed ratio for SURFnet. As we expect, using DNSSEC is more common in TLD name servers rather than the name servers in lower levels. Another observation is that all of the responses larger than 1232 bytes belong to the DNSSEC traffic. These queries are the ones asking

Table 8: SIDN DNSSEC response sizes

| | 1-1232 Bytes | 1233-1452 Bytes | 1453-65535 Bytes |
|---|---|---|---|
| DNSSEC | 666941 (99.39%) | 34 (0.06%) | 4047 (0.60%) |

Although all of the large messages are resulted by DNSSEC, most of the DNSSEC responses are smaller than 1232 bytes. Big responses from the SIDN traces are very similar and all of them are larger than 1452 bytes. However, we have seen that they would fit in smaller responses, if asked for with a smaller max-response-size EDNS0 option. The delivery of these responses is essential to the validation path in DNSSEC. Given the behaviour of TLD name servers and their role in DNSSEC, we conclude that a more customized solution needs to be found for such cases.

Looking at the fragmented responses, we noticed that all of the fragmemnted responses are larger than 1452 bytes. Table 9 shows that all of the fragmented responses are fargmented at one of these two sizes: 2218 or 1443 bytes.

Table 9: SIDN fragment payload length

| Size (Bytes) | Number of responses |
|---|---:|
| 2218 | 99.85% |
| 1443 | 0.15% |

Table 10 shows the number of ICMPv6 messages in the SIDN traces. Comparing to the SURFnet traces, We notice an increase in the number of Fragment Reassembly messages, and an even bigger increase in the number of Administratively Prohibited messages. The percentages are calculated with respect to the total number of DNS responses.

Table 10: SIDN ICMPv6 messages

| Type of message | SIDN |
|---|---:|
| Time Exceeded Fragment Reassembly | 333(0.026%) |
| Packet Too Big | 43(0.00%) |
| Administratively prohibited | 7991(0.65%) |

## 3.3 RIPE Atlas experiment

The MTU of the server was set at 1280 bytes to ensure that responses will be fragmented. In this way, we can deduce which probes filter fragments.

For this experiment, the RIPE Atlas network of probes was used to query a large TXT record on the name server running on the NLNOG Ring node. The reason for this is that a response to a query for this record will be larger than 1500 bytes.

A map of the probes used is shown in Figure 6. A total of 442 probes were available for this experiment, and the captures were taken over a three hour period.

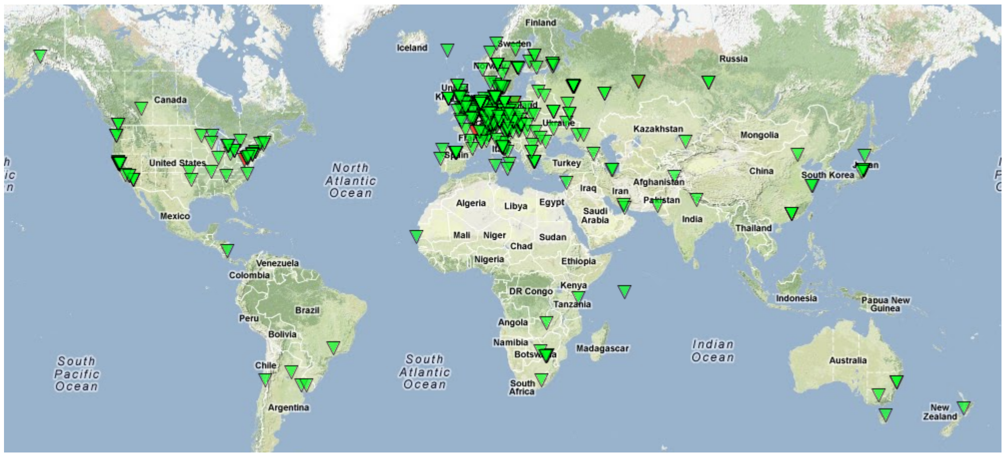


Figure 6: Probes used in experiment

Figure 7 shows the PMTUs on the paths to the probes which are less than 1500 bytes:

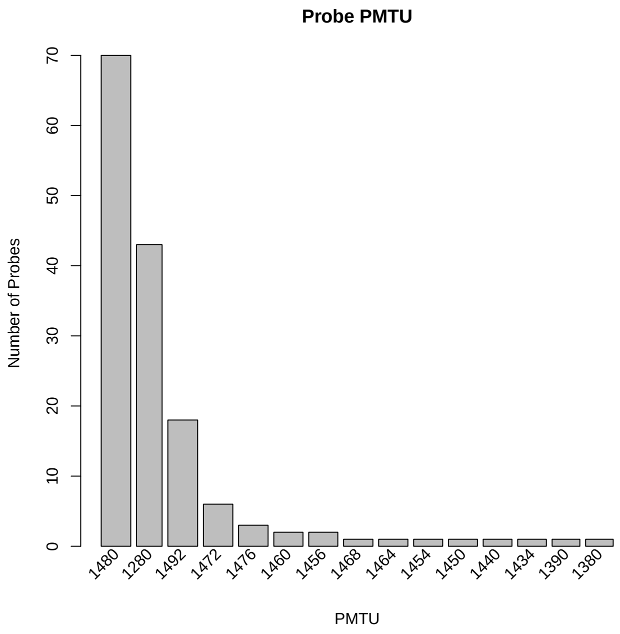


Figure 7: Atlas Probe PMTU

In this experiment 34 probes originated Fragment Reassembly Time Exceeded messages (7.6%), and there were 19 sources of Administratively Prohibited messages (4.2%), out of which 13 had also sent FRTE. Both message types indicate fragment filtering on those paths. Our theory is that fragment filtering hosts originate Administratively Prohibited messages because they do not accept fragments. Further research in the origin of these messages and their anticipation could provide an improvement in name server responsiveness.

Despite the fact that the MTU on the server was set to the minimum IPv6 MTU, one probe still sent back a Packet Too Big message. This probe's maximum MTU was of 1240 bytes and could thus only receive messages of at most 1232 bytes.

The data gathered in this experiment can serve as a baseline for future experiments.

# 4 Proposed solutions

Because the name server does not keep any state about previous queries, receiving PTB messages will not trigger resending a smaller packet into the network. However, these messages contain the original query and also the fitted part of response into the MTU. This option looks very beneficial to remind the name server about the previous queries, which were dropped due to the large size of packets. However, current name servers do not use this facility.

The following subsections describe our proposed solutions to make use of PTB messages to increase a name server's responsiveness.

## 4.1 Solution 1 - TC bit solution

The idea of this approach is this: after receiving a PTB message, a smaller DNS reply with the contents of the PTB payload can be sent to the client. In this case, the TC flag in the response is set to 1, indicating that the answer is truncated and the client should come back to the server with a TCP connection to get the complete answer.

## 4.2 Solution 2 - Resubmit query

This approach will use the payload contained in the ICMPv6 PTB message to resubmit the query (contained in the question section of the answer) to the name server. The aim of this solution is to give a sense of state to the system, but without the overhead that comes with stateful protocols.

## 4.3 Solution 3 - Resubmit query with EDNS0

This solution is similar to the one proposed in case 2, but aims at further improving deliverability of the name server at the cost of giving shorter answers. Similarly to resubmitting the query such as in the previous case, we would be processing the payload contained in the PTB message. The main difference is that we would modify the maximum-message-size EDNS0 option. The maximum-message-size option specifies the maximum size of a DNS response that the client will accept. By using the MTU from the PTB message, we can compute the length of a DNS response such that it will not get fragmented.

The length of the response can be calculated by subtracting the length of the IPv6 and UDP headers from the MTU received in the PTB message. For example, if we receive a PTB saying that the smallest MTU on a path is 1400 bytes, the maximum message size we would need to set would be:

1400 - 40 (IPv6 header) - 8 (UDP header) = 1352 bytes

This solution relies on the name server to properly adjust the answer to the maximum message size, and leave out the information that it deems less necessary to the client.

Although this would give the client a shorter answer (for example omitting the ADDITIONAL section), it would circumvent firewall fragment filtering. PMTUD must be disabled at the OS level, otherwise a second answer would be returned fragmented. The difference between this solution and the first solution in Gijs's research [4] is that instead of limiting the response to the minimum of 1232 bytes, we limit the response according to the MTU specified by the received PTB. In this we would have less truncated responses than if we would limit them at 1232.

## 4.4 Implications

All of the presented solutions aid in mitigating DNS ID hacking. In order to spoof the answer to a query, the attacker needs to send the response using the source IP of the name server and the ID of the query, otherwise the client will not accept the answer. The query ID is a 16 bit value, so the attacker can generate random numbers between 0 and 65535 to find a valid ID. In normal situations, the DNS response comes back from the server in a short time (a few milliseconds). In a situation like having big DNS responses that cannot go through the routers and cause ICMPv6 PTB, in current DNS systems, the client waits until the query expires (about 5 seconds). Because the original answer never reaches the client due its size, the spoofed answer is easily accepted and used by the client, which misdirects it to the desired address by the attacker.

According to RFC 1981 about PMTUD in IPv6, there may exist malicious parties who are willing to give the server wrong information about the path MTU. Since the path MTU cant be smaller than the IPv6 minimum link MTU, receiving PTB with smaller MTU than the real value is not a big issue in general. In case of receiving a larger MTU than the real path MTU, the server may decide to send to a larger packet. This mistake will be compensated by receiving another PTB. However, repeating this attack can cause lots of packet drops. As mentioned in this RFC, the server should not increase its estimate of the Path MTU by receiving a PTB message.

Solution 2 relies on the OS to adjust its MTU for the path, and decrease it to the required value. In case of larger DNS responses, this could cause additional fragmentation at the lower levels of the protocol stack. Fragmentation can cause problems due to the fact that firewalls are sometimes configured to filter fragments as a security measure.

Both case 2 and case 3 would still be susceptible to attacks in which the original packet is altered. However, these two solutions only handle the question section of the original answer, and disregard the rest of the answer. Depending on the size of the DNS response, reducing the maximum message size could cause a small rise in truncated responses. We do not consider this to be a problem, because it would not even be so prevalent. It would still be a better option rather than leaving the client without an answer.

There is the possibility of bypassing BCP38 [7] by constructing PTB messages, which would not require spoofing the source address. An attacker can spoof the IP address inside the ICMPv6 payload. In this way, the attacker does not need to spoof his own IP, thus bypassing BCP38. The name server is not able to distinguish forged payloads from its own generated responses. In this way, an attacker could flood a victim by forging many PTB messages. Receiving each PTB, the name server will send a response packet to the victim. Therefore, the victim receives lots of packets in a short period. So forging PTB messages can result a Denial of Service (DoS) attack in the clients.

## 4.5 Implementation

For implementing these methods, there are two options:

1. modifying the name server to receive the PTB messages from the OS
2. developing a server-side module in charge of handling the ICMPv6 PTB messages

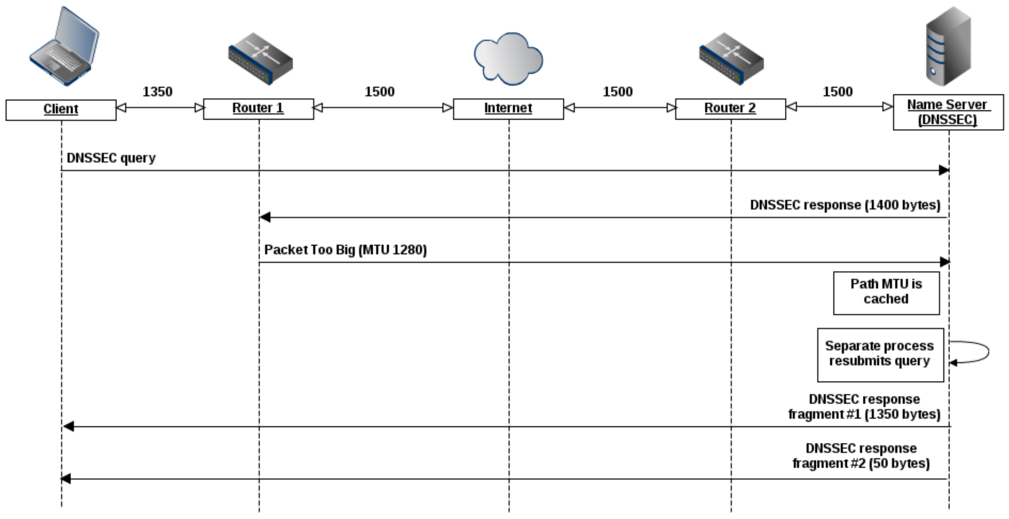
It seems the second option is better, because:

- it is a general-purpose solution, which can work well with different DNS software
- it does not affect the normal operation of the name server

We will make use of raw sockets for implementing solution 2, and present the results in the following sections.

# 5 Test and results

This section presents the results gathered from testing the second solution described in section 4.2. It was implemented with python by using a raw socket to intercept packets and handle them accordingly. The implementation assumes an EDNS size of 4096 bytes.



1350
1500
1500
1500
Client
Router 1
Internet
Router 2
Name Server (DNSSEC)
DNSSEC query
DNSSEC response (1400 bytes)
Packet Too Big (MTU 1280)
Path MTU is cached
Separate process resubmits query
DNSSEC response fragment #1 (1350 bytes)
DNSSEC response fragment #2 (50 bytes)

Figure 8: Implementation of second solution

Figure 8 shows how this solution would function in a normal client-server communication.

The setup for this test consisted of 427 Atlas probes, and a name server running the same version of NSD (3.2.14) as in the previous Atlas experiment. The probes queried a 1452 byte TXT record that would fit precisely in packets of 1500 bytes.

Output from the PTB-handling program showed that it successfully handled 56 sources with PMTU <1500, and only 5 probes still sent back FRTE messages. We can conclude that the 56 sources had no fragment-filtering firewalls, and 5 did. This confirms previous estimations [1] that about 10% of the destinations would filter fragments.

## 6 Conclusion

In this research, we were mainly concentrated on answering the following question: ”Is it feasible to leverage the original packet payload in ICMPv6 to increase a name server responsiveness?” We can concluded that when ICMPv6 PTB are sent for smaller links, it is feasible to use the payload to assure delivery.

We propose three approaches to make use of ICMPv6 PTB messages. In the first solution, the name server informs the client to retry the query over TCP. The second solution resubmits the query from the ICMPv6 payload. The third resubmits the query while avoiding fragmentation. Each solution can be implemented as a separate process or built into the name server. We suggest the first option because of its simplicity and portability. It is discussed that all of these solutions can aid in improving the security issues, such as DNS response ID hacking. The only security concern that arises from these solutions is the possibility of forging the PTB message by an attacker, which can result a DoS attack in clients.

Being able to use the received PTB messages, the name server does not have to fragment the responses at the minimum IPv6 MTU of 1280. Only when the path MTU to a host actually demands it, responses will be fragmented. This results in less fragmented responses and as a consequence less problems with fragment-filtering firewalls.

Using the data gathered from analyzing the SURFnet and SIDN traces, we observed the response sizes in a live environment and the prevalence of ICMPv6 error messages. We implemented the second solution and tested it using RIPE Atlas probes. This solution resulted in more clients receiving an answer without having to timeout.

## 7 Future work

We have presented three solutions that can solve the problems described. However, due to the time constraints of the project, we decided to test the second solution. It would be interesting to see the results of experiments using the first and third solution. A comparison between them could help in deciding which solution would be best to implement, and in which cases. Because of the large numbers of Administratively Prohibited messages, we suggest that they should be taken into consideration by future implementations of our proposed solutions.

We perform the test by running a simple query using RIPE Atlas probes. One can implement and test the solution in more realistic environments to obtain results similar to the SURFnet traces.

As it was shown in this paper and in previous research, fragmentation is still an issue in IPv6. However, making use of existing technologies can help solve these problems. The way in which different name servers choose to fragment their responses is also an interesting issue to look into, as it affects their deliverability.

## A Abbreviations

| | |
|---|---|
| BIND | Berkeley Internet Name Daemon |
| DF | Don't Fragment |
| DNS | Domain Name Server |
| DNSSEC | DNS Security Extensions |
| DoS | Denial Of Service |
| EDNS | Extension mechanisms for DNS |
| FRTE | Fragment Reassembly Time Exceeded |
| ICMP | Internet Control Message Protocol |
| ICMPv6 | ICMP version 6 |
| IP | Internet Protocol |
| IPv4 | IP version 4 |
| IPv6 | IP version 6 |
| MTU | Maximum Transmission Unit |
| NSD | Name Server Daemon |
| PMTU | Path MTU |
| PMTUD | Path MTU Discovery |
| PTB | Packet Too Big |
| RFC | Request For Comments |
| TCP | Transmission Control Protocol |
| TLD | Top Level Domain |
| UDP | User Datagram Protocol |

## References

[1] M. de Boer, J. Bosma, ”Discovering Path MTU black holes using RIPE Atlas”, June 8 2012. http://nlnetlabs.nl/downloads/publications/pmtu-black-holes-msc-thesis.pdf

[2] A. Conta, S. Deering, M. Gupta,”Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification”, *RFC 4443, 2006*

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

[4] G. van den Broek, R. van Rijswijk, A. Pras, A. Sperotto, ”DNSSEC Meets the Real-World: Improving Response Deliverability”, MSc.Thesis, University of Twente, The Netherlands.(Private communication) July 6th 2012

[5] M. Andrews, ”DNS and UDP Fragmentation”, *draft-andrews-dnsext-udp-fragmentation-01, 2012*

[6] V. Manojlovic, ”RIPE Atlas 2012 Year in Review”, 2013

[7] P. Ferguson, D. Senie, ”Network Ingress Filtering:Defeating Denial of Service Attacks which employ IP Source Address Spoofing”, *BCP38, 2000*