Measurement study of DNS over IPv6: Problem and Solution

Linjian Song, Shengling Wang

ABSTRACT

Internet works on reliable naming and IP package transmission. As complexity added into Internet infrastructure due to security and stability consideration, some mismatch and unexpected fragility are exposed. One case is in the field of DNS (DNSSEC) and IPv6. There are some public evidence and concerns on IPv6 fragmentation issues due to larger DNS payloads over IPv6. Different from other measurement work, in this paper we proposed "glueless" measurement mechanism to analyze the end-to-end users experience worldwide. It is based on a live, real and global Ads system and shows that the IPv6 large packet drop rate is up to 38%. Our analysis shows the combination of DNSSEC, UDP and IPv6 is really not going to work very well. In the meanwhile, we advance a solution called ATR (Additional Truncated Response) as a fix on DNS server which requires no modification on users side (DNS resolver) and support incremental deployment. We also conducted a measurement which shows more than 68% impacted users can be relieved on DNS latency and failures due to large DNS response.

1 INTRODUCTION

Large DNS response is identified as a issue for a long time. There is an inherent mechanism defined in [RFC1035] to handle large DNS response (larger than 512 octets) by indicating (set TrunCation bit) the resolver to fall back to query via TCP. Due to the fear of cost of TCP, EDNS(0) [RFC6891] was proposed which encourages server to response larger response instead of falling back to TCP. However, as the increasing use of DNSSEC and IPv6, there are more public evidence and concerns on user's suffering due to packets dropping caused by IPv6 fragmentation in DNS due to large DNS response.

It is observed that some IPv6 network devices like firewalls intentionally choose to drop the IPv6 packets with fragmentation Headers[I-D.taylor-v6ops-fragdrop]. [RFC7872] reported more than 30% drop rates for sending fragmented packets. Regarding IPv6 fragmentation issue due to larger DNS payloads in response, one measurement [IPv6-frag-DNS] reported 35% of endpoints using IPv6-capable DNS resolver can not receive a fragmented IPv6 response over UDP. Moreover, most of the underlying issues with fragments are unrevealed due to good redundancy and resilience of DNS. It is hard for DNS client and server operators to trace and locate the issue when fragments are blocked or dropped. The noticeable DNS failures and latency experienced by end users are just the tip of the iceberg.

Depending on retry model, the resolver's failing to receive fragmented response may experience long latency or failure due to timeout and reties. One typical case is that the resolver finally got the answer after several retires and it falls back to TCP after deceasing the payload size in EDNS0. To avoid that issue, some authoritative servers may adopt a policy ignoring the UDP payload size in EDNS0 extension and always truncating the response when the response size is large than a expected one. However one study [Not-speak-TCP] shows that about 17% of resolvers in the samples can not ask a query in TCP when they receive truncated response. It seems a dilemma to choose hurting either the users who can not receive fragments or the users without TCP fallback capacity. There is also some voice of "moving all DNS over TCP". But It is generally desired that DNS can keep the efficiency and high performance by using DNS UDP in most of time and fallback as soon as possible to TCP if necessary for some corner case.

To relieve the problem, this memo introduces an improvement on DNS responding process by replying an Additional Truncated Response (ATR) just after a normal large response which is to be fragmented. Generally speaking ATR provides a way to decouple the EDNS0 and TCP fallback in which they can work independently according to the server operator's requirement. One goal of ATR is to relieve the hurt of users, both stub and recursive resolver, from the position of server, both authoritative and recursive server. It does not require any changes on resolver and has a deploy-and-gain feature to encourage operators to implement it to benefit their resolvers.

2 RELATED WORK(TBD)

1) measurement work

Fragmentation Considered Harmful in IPv4 (and IPv6)[9]

IP Fragmentation Considered Fragile

Many network operators filter all IPv6 fragments.[11]

two researchers utilized a measurement network to measure fragment filtering. They sent packets, fragmented to the minimum MTU of 1280, to 502 IPv6 enabled and reachable probes. They found that during any given trial period, ten percent of the probes did not receive fragmented packets. [12]

Applications That Rely on Fragmentation (draft-bonica-6man-frag-deprecate) [13]

2) solution and fix

3 A CASE STUDY OF THIS PROBLEM

We begin by studying IPv6 large DNS response issue using a case. Basically, we want to clearly demonstrate and explain the issue of combination IPv6, Large UDP Packets and the DNS. To illustrate this situation, here are two DNS queries, both made by a recursive resolver to an authoritative name server, both using UDP over IPv6.

```
Query 1:

\$ dig +bufsize=4096 +dnssec 000-4a4-000a-000
a0000-b9ec853b-241-1498607999-2a72134a.ap2.
dotnxdomain.net. @8.8.8.8
139.162.21.135
(MSG SIZE rcvd: 1190)
```

```
Query 2:

\$ dig +bufsize=4096 +dnssec 000-510-000a-000
a0000b9ec853b-241-1498607999-2a72134a.ap2.
dotnxdomain.net. @8.8.8.8
status: SERVFAIL
(MSG SIZE rcvd: 104)
```

What we see here are two almost identical DNS queries that have been passed to Google's Public DNS service to resolve. In the first case, the DNS response is 1,190 octets long, and in the second case the response is 1,346 octets long. The DNS server is an IPv6-only server, and the underlying host of this name server is configured with a local maximum packet size of 1,280 octets. Therefore, in the first case the response being sent to the Google resolver is a single, unfragmented IPv6 UDP packet, and in the second case the response is broken into two fragmented IPv6 UDP packets. And it is this single change that triggers the Google Public DNS Server to provide the intended answer in the first case, but to return a SERVFAIL failure notice in response to the fragmented IPv6 response. When the local Maximum Transmission Unit (MTU) on the server is lifted from 1,280 octets to 1,500 octets, the Google resolver returns the server DNS response in both cases.

The only difference in these two responses is IPv6 fragmentation, but there is perhaps more to it than that.

IP fragmentation in both IPv4 and IPv6 "raises the eyebrows" of firewalls. Firewalls typically use the information provided in the transport protocol header of the IP packet to decide whether to admit or deny the packet. For example, you may see firewall rules admitting packets using TCP ports 80 and 443 as a way of allowing web traffic through the firewall filter. For this process to work, the inspected packet needs to contain a TCP header and use the fields in the header to match against the filter set. Fragmentation in IP duplicates the IP portion of the packet header, but the inner IP payload, including the transport protocol header, is not duplicated in every ensuing packet fragment. Thus trailing fragments pose a conundrum to the firewall. Either all trailing fragments are admitted, a situation that has its own set of consequent risks, or all trailing fragments are discarded, a situation that also poses connection issues[5].

IPv6 adds a further factor to the picture. In IPv4 every IP packet, fragmented or not, contains IP fragmentation control fields. In IPv6 these same fragmentation control fields are included in an IPv6 Extension Header that is attached only to packets that are fragmented This 8-octet Extension Header is placed immediately after the IPv6 packet header in all fragmented packets, meaning that a fragmented IPv6 packet does not contain the Upper Level Protocol Header starting at octet offset 40 from the start of the IP packet header. But in the first packet of this set of fragmented packets, the Upper Level Protocol Header is chained off the fragmentation header, at byte offset 48, assuming that there is only a Fragmentation Extension Header in the packet. The implications of this fact are quite significant. Instead of always looking at a fixed point in a packet to determine its upper-level protocol, the packet-handling device needs to unravel the Extension Header chain, raising two rather tough questions. First, how long is the device prepared to spend unraveling this chain? And second, would the device be prepared to pass on a packet with an Extension Header that it did not recognize?

In some cases, implementers of IPv6 equipment have found it simpler to just drop all IPv6 packets that contain Extension Headers. Some measurements of this behavior are reported in RFC 7872[6]. This document reports a 38% packet-drop rate when sending fragmented IPv6 query packets to DNS Name servers. But the example provided previously is in fact the opposite case to that reported in RFC 7872, and the example illustrates a more conventional case. It's not the queries in the DNS that can readily grow to sizes that require packet fragmentation, but the responses. The relevant question concerns the anticipated probability of packet drop when sending fragmented UDP IPv6 packets as responses to DNS queries. To rephrase the question slightly, how do DNS recursive resolvers fare when the IPv6 response from the server is fragmented?

4 HOW WIDESPREAD IS THIS PROBLEM?

For a start, it appears from the cased cited here that Google's Public DNS resolvers experienced some packet-drop problem when they passed a fragmented IPv6 response (this problem was noted in mid-2017, and Google has subsequently corrected it). But was this problem limited to just one or two DNS resolvers, or do many other DNS resolvers experience a similar packet-drop issue? How widespread is this problem? Can we identify these resolvers? We tested this question using three approaches (two approaches) which measures the scenario of combination of large DNS responses, DNS resolvers and IPv6.

4.1 Measurement Setup

The measurement technique we are using is based on scripting inside online advertisement[Ref TBD]. This allows us to instrument a server and get the endpoints who are executing the measurement script to interact with the server. Our measurement approach is described as follows :

- Each endpoint is provided with a unique name string to eliminate the effects of DNS caching(The measuring Javascript put in advertisement generates a unique name string each endpoint)
- The DNS name is served from our authoritative servers
- Each DNS name contains a name creation time component (so that we can disambiguate subsequent replay from original queries)
- Resolving the DNS name requires the user's DNS resolvers to receive a fragmented IPv6 packet

We tested the system using an IPv6-only name server address response that used three response sizes as experiment parameters:

- Small: 169 octets
- Medium: 1,428 octets
- Large: 1,886 octets

We operate the experiment in IPv6 only DNS Servers, using a 1,280 octet MTU. So both the medium and large responses were fragmented. This measurement test was loaded into an online advertising campaign (When?Duration? Sample rate?). It is expected that if the client's DNS resolvers can successfully resolve the DNS name they will fetch the named web object, so the measurement here is one of the failure rate to access the web object.

Table 1: Geo-difference of Failure rate

	Americas	Europe&Africa	Asia&Oceania
SMALL	4%	5%	13%
MEDIUM	23%	49%	52%
LARGE	24%	50%	52%

It is expected that not every resolver can perform a DNS query using IPv6. There are 68,229,946 experiments collected in total, out of which 35,602,243 experiments used IPv6-capable resolvers. So the first outcome from this data is somewhat surprising. While the overall penetration of IPv6 in the larger Internet is some 15% of users when the measurement is done in August 2017, the DNS resolver result is far higher.

So our first finding is : **Some 52% of tested endpoints use DNS resolvers that are capable of using IPv6.**

Of those that did DNS over IPv6, we observed the following Loss Rates in fetching the web object:

Result - I:

- Small: 7%
- Medium: 42%
- Large: 42%

Are there regional differences? We use three different servers, and divide (roughly) clients into three geo areas: Americas, Europe & Africa, Asia & Oceania.

A IPv6-only authoritative DNS Server serving UDP fragmented DNS responses appears to have significant problems in delivering this UDP fragmented response to recursive resolvers in the Internet.

However, we cannot see what the endpoint is doing. For example, we can see from the server when we deliver a DNS response to a client, but we have no clear way to confirm that the client received the response. Normally the mechanisms are indirect, such as looking at whether or not the client then retrieved a web object that was the target of the DNS name. This measurement approach has some degree of uncertainty, as there are a number of reasons for a missing web object fetch, and the inability to resolve the DNS name is just one of these reasons. It introduce 7% loss rate for the unfragmented DNS response points to a high noise rate in the data collected using this experimental technique.

Is there a better way to measure how DNS resolvers behave? Can we identify these resolvers?

4.2 Repairing Missing "Glue" with Fragmented response

With the questions, we proposed the second approach with a DNS tricks of "glueless" delegation, a novel measurement mechanism by manipulating dynamic name on DNS name servers.

We created a "glueless" delegation in the DNS with following properties:

- The response to the query to the 'parent' lists the name servers of the 'child', but deliberately withholds the IP address of these name servers in the response. i.e. the response is missing the 'glue' records for the name servers.
- We then inflated the response of the name server record by adding pad records to the response.
- The idea is that the name will only be resolved if the resolver is capable of receiving a large response when trying to chase down the name server addresses

It is observed that a number of recursive resolvers use different query options when resolving the addresses of name servers, as distinct from resolving names. In particular, a number of resolvers, including Google's public DNS resolvers, strip all EDNS(0) query options from these name server address resolution queries. When the query has no EDNS(0) options, and in particular when there is no UDP Buffer size option in the query, then the name server responds with what will fit in 512 octets. If the response is larger, and in our case this includes the Medium and Large tests, the name server sets the Truncated Response flag in its response to indicate that the provided response is incomplete. This Truncated Response flag is a signal to the resolver that it should query again, using TCP this time.

In the case of this experiment it is observed that large amounts of queries are using TCP. This means that the observed rate of failure to resolve the name is not necessarily attributable to an inability to receive fragmented IPv6 UDP packets. But we wanted to pass the resolver a large fragmented UDP response to see if received it.

To solve the problem, we use a customized DNS name server arrangement that gratuitously fragments the small DNS response in order to always reply with a fragmented UDP response. While the IPv6 specification specifies that network Path MTU sizes should be no smaller than 1,280 octets, it does not specify a minimum size of fragmented IPv6 packets.

So we change the second approach with a little changes that :

- We fragment all response of the name server record no matter the size of the response.
- The idea is that the name will only be resolved if the resolver is capable of receiving a fragmented response when trying to chase down the name server addresses

The approach we've taken in this experiment is to use a user level packet processing system that listens on UDP port 53 and passes all incoming DNS queries to a back-end DNS server. When it receives a response from this back-end server it generates a sequence of IPv6 packets that fragments the DNS payload and uses a raw device socket to pass these packets directly to the device interface.

We are relying on the observation that IPv6 packet fragmentation occurs at the IP level in the protocol stack, so the IPv6 driver at the remote end will reassemble the fragments and pass the UDP payload to the DNS application, and if the payload packets are received by the resolver, there will be no trace that the IPv6 packets were fragmented.

As we are manipulating the response to the query for the address of the name server, we can tell if the recursive resolver has received the fragmented packets if the resolver resumes its original query sequence and queries for the terminal name.

Result - II:

- 10,851,323 experiments used IPv6 queries for the name server address
- 6,786,967 experiments queried for the terminal DNS name
- Fragmented Response: 6,786,967 / 10,851,323 = 62.54% = 37.45% Drop

This is our second result: **Some 37% of endpoints used IPv6-capable DNS resolvers that were incapable of receiving a fragmented IPv6 response.**

We used three servers for this experiment, on serving Asia Pacific, a second serving the America and the third serving Eurasia and Africa. There are some visible differences in the drop rate:

- Asia Pacific: 31% Drop
- Americas: 37% Drop
- Eurasia & Africa: 47% Drop

Given that this experiment occurs completely in the DNS, we can track each individual DNS resolver as they query for the name server record then, depending on if they receive the fragmented response, query for the terminal name. There are approximately 2 million recursive resolvers in today's Internet, but only some 15,000 individual resolvers appear to serve some 90% of all users. This implies that the behavior of the most

Table 2: DNS Resolvers with this problem

AS	Hits	% of Totle	AS Name,CC
15169	7,952,272	17.3%	GOOGLE - Google Inc., US
4761	6,521,674	14.2%	INDOSAT-INP-AP INDOSAT, ID
55644	4,313,225	9.4%	IDEANET1-IN Idea Cellular Limited, IN
22394	4,217,285	9.2%	CELLCO - Cellco Partnership DBA Verizon Wireless, US
55836	4,179,921	9.1%	RELIANCEJIO-IN Reliance Jio Infocomm Limited, IN
10507	2,939,364	6.4%	SPCS - Sprint Personal Communications Systems, US
5650	2,005,583	4.4%	FRONTIER-FRTR - Frontier Communications, US
2516	1,322,228	2.9%	KDDI KDDI CORPORATION, JP
6128	1,275,278	2.8%	CABLE-NET-1 - Cablevision Systems Corp., US
32934	1,128,751	2.5%	FACEBOOK - Facebook, US
20115	984,165	2.1%	CHARTER-NET-HKY-NC - Charter Communications, US
9498	779,603	1.7%	BBIL-AP BHARTI Airtel Ltd., IN
20057	438,137	1.0%	ATT-MOBILITY-LLC-AS20057 - AT&T Mobility LLC, US
17813	398,404	0.9%	MTNL-AP Mahanagar Telephone Nigam Ltd., IN
2527	397,832	0.9%	SO-NET So-net Entertainment Corporation, JP
45458	276,963	0.6%	SBN-AWN-AS-02-AP SBN-ISP/AWN-ISP, TH
6167	263,583	0.6%	Cellco Partnership DBA Verizon Wireless, US
8708	255,958	0.6%	RCS-RDS 73-75 Dr. Staicovici, RO
38091	255,930	0.6%	HELLONET-AS-KR CJ-HELLOVISION, KR
18101	168,164	0.4%	Reliance Communications DAKC MUMBAI, IN

intensively used resolvers has a noticeable impact on the overall picture of capabilities if DNS infrastructure for the Internet.

We saw 10,115 individual IPv6 addresses used by IPv6-capable recursive resolvers. Of this set of resolvers, we saw 3,592 resolvers that consistently behaved in a manner that was consistent with being unable to receive a fragmented IPv6 packet. As very large recursive resolvers use resolver "farms" and the queries are managed by a collection of query 'slaves'. We can group these individual resolver IPv6 addresses to their common Origin AS, and look at which networks use resolvers that show this problem with IPv6 Extension Header drops.

The table 2 now shows the preeminent position of Google's Public DNS service as the most heavily used recursive resolver, and its Extension Header drop issues, as shown in the example at the start of this article, is consistent with its position at the head of the list of networks that have DNS resolvers with this problem.

One conclusion looks starkly clear to me from these results. We can't just assume that the DNS as we know it today will just work in an all IPv6 future Internet. We must make some changes in some parts of the protocol design to get around this current widespread problem of IPv6 Extension Header packet loss in the DNS, assuming that we want to have a DNS at all in this all-IPv6 future Internet.

5 ATR MECHANISM (HALF DONE)

5.1 Methodology of ATR

As we stated in related work, we could move the DNS away from UDP and use TCP instead. That move would certainly make a number of functions a lot easier, including encrypting DNS traffic on the wire as a means of assisting with aspects of personal privacy online as well as accommodating large DNS responses.

However, the downside is that TCP imposes a far greater load overhead on servers, and while it is possible to conceive of an all-TCP DNS environment, it is more challenging to understand the economics of such a move and to understand, in particular, how name publishers and name consumers will share the costs of a more expensive name resolution environment.

If we want to continue to UDP where it's feasible, and continue to use TCP only as the 'Plan B' protocol for name resolution, then can we improve the handling of large response in UDP? Specifically, can we make this hybrid approach — of using UDP when we can, and TCP only when we must — faster and more robust?

An approach to address this challenge is first introduce inIETF draft draft-song-atr-large-resp-00. The approach

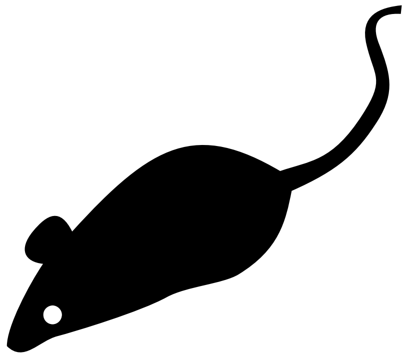

Figure 1: ATR behavior

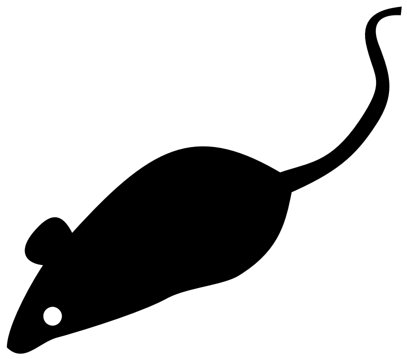

Figure 2: ATR module in DNS response process

described in this draft is simple: If a DNS server provides a response that entails sending fragmented UDP packets, then the server should wait for a 10 ms period and also back the original query as a truncated response. If the client receives and reassembles the fragmented UDP response, then the ensuing truncated response will be ignored by the client's DNS resolver as its outstanding query has already been answered. If the fragmented UDP response is dropped by the network, then the truncated response will be received (as it is far smaller), and reception of this truncated response will trigger the client to switch immediately to re-query using TCP. This behavior is illustrated in Figure 1.

5.2 ATR process or Algorithm

As show in the Figure 2 the ATR module can be implemented is right after truncation loop if the packet is not going to be fragmented.

The ATR responding process goes as follows:

- When an authoritative server receives a query and enters the responding process, it first go through the normal truncation loop to see whether the size of response surpasses the EDNS0 payload size. If yes, it ends up with responding a truncated packets. If no, it enters the ATR module.
- In ATR module, similar like truncation loop, the size of response is compared with a value called ATR payload size. If the response of a query is larger than ATR payload size, the server firstly sends the normal response and then coin a truncated response with the same ID of the query.
- The server can reply the coined truncated response in no time. But considering the possible impact of network reordering, it is suggested a timer to delay the second truncated response, for example $10\tilde{5}0$ millisecond which can be configured by local operation.

Note that the choice of ATR payload size and timer SHOULD be configured locally. And the operational consideration and guidance is discussed following sections.

There are three typical cases of ATR-unaware resolver behavior when a resolver send query to an ATR server in which the server will generate a large response with fragments:

- Case 1: a resolver (or sub-resolver) will receive both the large response and a very small truncated response in sequence. It will happily accepts the first response and drop the second one because the transaction is over.
- Case 2: In case a fragment is dropped in the middle, the resolver will end up with only receiving the small truncated response. It will retry using TCP in no time.
- Case 3: For those (probably 30%*17% of them) who can not speak TCP and sitting behind a firewall stubbornly dropping fragments. ATR can not help!

In the case authoritative server truncated all response surpass certain value , for example setting IPv6-edns-size to 1220 octets, ATR will helpful for resolver with TCP capacity, because the resolver still has a fair chance to receive the large response.

5.3 ATR timer

TBD

5.4 ATR payload size

Regarding the operational choice for ATR payload size, there are some good input from APNIC study [scoring-dns-root]on how to react to large DNS payload for authoritative server. The difference in ATR is that ATR focuses on the second response after the ordinary response.

For IPv4 DNS server, it is suggested the study that do not truncate and fragment IPv4 UDP response with a payload up to 1472 octets which is Ethernet MTU(1500) minus the sum of IPv4 header(20) and UDP header(8). The reason is to avoid gratuitously fragmenting outbound packets and TCP fallback at the source.

In the case of ATR, the first ordinary response is emitted without knowing it be to fragmented or not on the path. If a large value is set up to 1472 octets, payload size between 512 octets and the large value size will probably get fragmented by aggressive firewalls which leads losing the benefit of ATR. If ATR payload size set exactly 512 octets, in most of case ATR response and the single unfragmented packets are under a race at the risk of RO.

Given IPv4 fragmentation issue is not so serious compared to IPv6, it is suggested in this memo to set ATR payload size 1472 octets which means ATR only fit large DNS response larger than 1500 octets in IPv4.

For IPv6 DNS server, similar to IPv4, the APNIC study is suggested that do not truncate IPv6 UDP packets with a payload up to 1,452 octets which is Ethernet MTU(1500) minus the sum of IPv6 header(40) and UDP header(8). 1452 octets is chosen to avoid TCP fallback in the context that most TCP MSS in the root server is not set probably at that time.

In the case of ATR considering the second truncated response, a smaller size: 1232 octets, which is IPv6 MTU for most network devices(1280) minus the sum of IPv6 header(40) and UDP header(8), should be chosen as ATR payload size to trigger necessary TCP fallback. As a complementary requirement with ATR, the TCP MSS should be set 1220 octets to avoid Packet Too Big ICMP message as suggested in the APNIC study.

In short, it is recommended that in IPv4 ATR payload size SHOULD be 1472 octets, and in IPv6 the value SHOULD be 1232 octets.

5.5 Less aggressiveness of ATR

There is a concern ATR sends TC=1 response too aggressively especially in the beginning of adoption. ATR can be implemented as an optional and configurable feature at the disposal of authoritative server operator. One of the idea to mitigate this aggressiveness, ATR may respond TC=1 responses at a low possibility, such as 10

Another way is to reply ATR response selectively. It is observed that RO and IPv6 fragmentation issues are path specific and persistent due to the Internet components and middle box. So it is reasonable to keep a ATR "Whitelist" by counting the retries and recording the IP destination address of that large response causing many retires. ATR only acts to those queries from the IP address in the white list.

6 ATR EVALUATION (TBD)

It is worth of mentioning APNIC report[How-ATR-Work] on "How well does ATR actually work?" done by Geoff Huston and Joao Damas after 00 version of ATR draft. It was reported firstly in IEPG meeting before IETF 101 and then posted in APNIC Blog later.

It is said the test was performed over 55 million endpoints, using an on-line ad distribution network to deliver the test script across the Internet. The result is positive that ATR works! From the end users' perspective, in some 9% of IPv4 cases the use of ATR by the server will improve the speed of resolution of a fragmented UDP response by signaling to the client an immediate switch to TCP to perform a re-query. The IPv6 behavior would improve the resolution times in 15% of cases.

It also analyzed the pros and cons of ATR. On one hand, It is said that ATR certainly looks attractive if the objective is to improve the speed of DNS resolution when passing large DNS responses. And ATR is incrementally deployable in favor of decision made by each server operator. On another hand, ATR also has some negative factors. One issue is adding another DNS DDoS attack vector due to the additional packet sent by ATR, (author's note : very small adding actually.) Another issue is risk of RO by the choice of the delay timer which is discussed fully in Section 4.1.

As a conclusion, it is said that "ATR does not completely fix the large response issue. If a resolver cannot receive fragmented UDP responses and cannot use TCP to perform DNS queries, then ATR is not going to help. But where there are issues with IP fragment filtering, ATR can make the inevitable shift of the query to TCP a lot faster than it is today. But it does so at a cost of additional packets and additional DNS functionality". "If a faster DNS service is your highest priority, then ATR is worth considering", said at the end of this report

7 NETWORK RE-ORDERING CONSIDERATION (TBD)

As introduced in Section 2 ATR timer is a way to avoid the impact of network reordering(RO). The value of the timer is critical, because if the delay is too short, the ATR response may be received earlier than the fragmented response (the first piece), the resolver will fall back to TCP bearing the cost which should have been avoided. If the delay is too long, the client may timeout and retry which negates the incremental benefit of ATR. Generally speaking, the delay of the timer should be "long enough, but not too long".

To the best knowledge of author, the nature of RO is characterized as follows hopefully helping ATR users understand RO and how to operate ATR appropriately in RO context.

o RO is mainly caused by the parallelism in Internet components and links other than network anomaly [Bennett]. It was observed that RO is highly related to the traffic load of Internet components. So RO will long exists as long as the traffic load continue increase and the parallelism is used to enhance network throughput.

o The probability of RO varies largely depending on the different tests samples. Some work shown RO probability below 2[Tinta] and another work was above 90agreed that RO is site-dependent and path-dependent. It is observed in that when RO happens, it is mostly exhibited consistently in a small percentages of the paths. It is also observed that higher rates smaller packets were more prone to RO because the sending inter-spacing time was small.

o It was reported that the inter-arrival time of RO varies from a few milliseconds to multiple tens of milliseconds [Tinta]. And the larger the packet the larger the inter-arrival time, since larger packets will take longer to be transmitted.

Reasonably we can infer that firstly RO should be taken into account because it long exists due to middle Internet components which can not be avoided by end-to-end way. Secondly the mixture of larger and small packets in ATR case will increase the inter-arrival time of RO as well as the its probability. The good news is that the RO is highly site specific and path specific, and persistent which means the ATR operator is able to identify a few sites and paths, setup a tunable timer setting for them, or just put them into a blacklist without replying ATR response.

Based on the above analysis it is hard to provide a perfect value of ATR timer for all ATR users due to the diversity of networks. It seems OK to set the timer with a range from ten to hundreds ms, just below the timeout setting of typical resolver. Is suggested that a decision should be made as operator-specific according to the statistic of the RTT of their users. Some measurement shown [Brownlee][Liang] the mean of response time is below 50 ms for the sites with lots of anycast instance like L-root, .com and .net name servers. For that sites, delay less than 50 ms is appropriate.

8 SECURITY CONSIDERATION

There may be concerns on DDoS attack problem due to the fact that the ATR introduces multiple responses from authoritative server. The extra packet is pretty small. In the worst case, it's 50packets and they are small

DNS cookies [RFC7873] and RRL on authoritative may be possible solutions

9 COST ANALYSIS

TBD

10 CONCLUSION AND FUTURE WORK

TBD