February 2024
Geoff Huston

DNS and Truncation in UDP

I'll press on with another item within an overall theme of some current work in DNS behaviours with a report of a recent measurement on the level of compliance of DNS resolvers with one aspect of standard-defined DNS behaviour: truncation of DNS over UDP responses.

The DNS leverages the UDP transport protocol to maximise its efficiency. No state is held in a DNS server, and each received query is processed without reference to preceding queries. It represents a relatively good analogy to a simple transaction protocol. In the case of authoritative servers, incoming queries cause the server to look up its local zone data and respond on the basis of whether it can find a local data item that matches the query or not. The actions of a recursive resolver are not all that different, except that the local data store is built using a cache of recent responses, and if there is no match between the query and the local data store, then the recursive resolver will use the DNS itself to query authoritative servers to find an answer.

UDP has no control channel. There are no acknowledgements in the UDP protocol itself, nor any support for a reliable transport. An application may add its own control fields to support a reliable communication (which is essentially what happens with QUIC), but UDP itself is a simple instantiation of the underlying IP datagram paradigm. That means that a UDP client uses a local timer to decide that a query, or its response, has been dropped. The client's choice of this timer value is critical to the performance of the application. Using a conservative timer value means that the client application will appear to be unresponsive to incidents of network level packet drop, while a shorter value may make the client application overly aggressive, needlessly generating duplicate queries, loading both the server and potentially the network itself. In any case, the use of a UDP transport allows servers to support a far higher UDP query load as compared to TCP queries and responses.

For more detail, see Petr Špaček's 2020 investigation of the throughput capacity of DNS resolvers comparing workloads presented over UDP with the same query workload presented over TCP and various flavours of TLS - https://indico.dns-oarc.net/event/34/contributions/782/attachments/769/1308/shotgun.pdf

The choice of UDP for the DNS is one represents a basic design decision in favour of a lightweight and efficient protocol platform.

However, it is a compromise in many ways, and the trade-offs that were the case in the late 1980's in the evolution of the DNS protocol. UDP does not perform source address validation, so DNS-based denial-of-service attacks can be mounted through a technique of substituting the intended victim's IP address in DNS queries over UDP. The DNS is not an encrypted protocol, so if these UDP exchanges can be inspected, the DNS content can be readily observed. All of these weaknesses have been exploited in various ways, and these days DNS implementations have a number of defences, including response rate management, increasing the entropy in DNS query response exchanges to counter third party injection.

A recent report of the defensive measures used by Google's Public DNS Service can be found in a presentation by Google's Puneet Sood to DNS OARC 38.
https://indico.dns-oarc.net/event/43/contributions/917/attachments/883/1640/Cache%20Poisoning%20Protection%20for%20Authoritative%20Queries.pdf

In addition, there is the issue of IP packet fragmentation. IP was designed to be inter-network overlay protocol, intended to operate across a diversity of underlying packet switched networks. These networks have their own limitations on permissible packet sizes, such as Ethernet's 1,500 bytes or FDDI's 4,478 bytes. IP version 4 accommodated this diversity by supporting *fragmentation on the fly*, where a IP router, when attempting to forward a packet which was too large to be passed to the next hop network, was allowed to divide a packet into multiple smaller *fragments*, duplicating the necessary parts of the original IP packet header in each IP packet fragment. These fragments were independently forwarded to their IP destination, which then had the task to reassemble the original IP packet from the received fragments.

Packet Fragmentation is seen to present a number of security vulnerabilities, including injection attacks and resource exhaustion denial of service attacks, so it's no surprise to observe that many security firewalls discard trailing IP packet fragments.

In its efforts to "improve" IP fragmentation the IPv6 protocol disallowed *fragmentation on the fly*, and instead uses an ICMP6 control signal passed back to the packet's source address indicating that the packet is too large for the next hop. There are some issues with this approach, including the assumption that the switching units in the interior of the end-to-end network path can reach the original source (tunnels and other form of packet encapsulation add layers of complexity to this task). For the UDP transport protocol there is the additional issue of how to handle this ICMP6 signal. As we've noted UDP is not a positive acknowledgement protocol and does not retain a copy of unacknowledged sent packets, so there is little a UDP sender can do when receiving such a ICMP6 Packet Too Big control message. Conventionally, the sender should store the ICMP6-notified MTU size in the local forwarding table for a period, and if the host should ever need to send another UDP packet to this destination then it will be able to fragment at source.

At best, this IPv6 behaviour is time consuming and clumsy. The application-level protocol, in this case the DNS, must rely on the original transaction initiator timing out when awaiting a response and re-sending the query.

An underlying problem here is that if the DNS response using UDP is large enough to require IP packet fragmentation, and the path contains an active element that discards packet fragments due to the heightened security risk, then there needs to be a different DNS response to ensure transmission of the DNS data. The DNS' response is to require implementations of the DNS to support both UDP and TCP transports. DNS transactions should use UDP by preference, but if there is a signal in the DNS response to indicate that the entire response cannot be loaded into UDP, then the querier should re-query using TCP. This signal is the setting of the Truncation flag (the "TC" bit) in the DNS response (Figure 1).

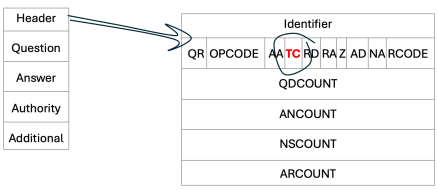

Figure 1 – DNS Packet Format and the Truncation (TC) bit

RFC 2818 has some clarifying advice on what a standards-compliant implementation should do when it receives such a response:

> "When a DNS client receives a reply with TC set, it should ignore that response, and query again, using a mechanism, such as a TCP connection, that will permit larger replies." RFC 2191, Clarifications to the DNS Specification, July 1997.

This RFC does not provide any reasons why the clarification was considered to be necessary. It is likely that there was a view at the time that some recursive resolver implementations were trying to improve their responsiveness by acting in an opportunistic manner with truncated DNS responses. Even if the truncated bit flag was set, if there was a usable answer section in the response then the resolver might use the provided data and move on with the resolution task if possible. This optimisation applies particularly to DNS referral responses where the answer section is complete and intact and the glue records in the additional sections in the response may be truncated.

This leads to two questions that relate to the current DNS resolution environment:

- How prevalent is the behaviour of using the data in a truncated DNS response?
- Do all name resolution systems successfully followup with a re-query using TCP after receiving a truncated UDP DNS responses?

Measuring DNS Behaviour and Glueless Delegation

We've used the APNIC Labs Ad-based measurement environment to provide an answer to these two questions.

This system uses online ads to enrol users to test particular behaviours from the perspective of the end user. The script in the ad performs a number of fetches of URLs. Each URL to be fetched uses a unique DNS name, and there is only a single authoritative DNS server to resolve this name, and a single web server to serve the named web object. We cannot instrument the end user browser that is running the ad script, but we can instrument the DNS and Web servers to record their end of each measurement transaction. Each fetch within as single ad script can be used to measure a particular behaviour or attribute, such as IPv6-only capability, use of DNSSEC to validate domain name resolution, the use of QUIC, and the adoption of network behaviours that drop routes have invalid ROAs. The only control framework of the script on the user side measures the elapsed time to perform each URL fetch and passes this information back to an ad controller by performing a final URL fetch with the individual time values encoded as attributes to this closing report.

The ad system is configured to present some 15M – 20M ad impressions per day, with an ad configuration that attempts to secure as wide a diversity of end users as possible. On the server side we use a distributed network of five server-side platform clusters, located approximately on each continent to try and minimise the network delays when each user connects to the experiment's DNS and Web servers.

To perform this measurement of DNS resolver handling of truncated responses and the related ability to switch to use TCP we've used a *glueless* DNS technique. This allows us to use the DNS itself to detect whether a DNS resolution environment can resolve a DNS name that is constructed using a particular DNS behaviour.

Generically, the technique is to use a target DNS name that is itself a delegated DNS name, and in the DNS delegation data the glue records are deliberately omitted. This is shown in Figure 1.

```
1. example.com zone
       child NS ns1.oob.example.net.

2. oob.example.net zone
       ns1  IN   A   192.0.2.1

3. child.example.com zone
       .    IN    A   203.0.113.1
```

Figure 1 – Example of Glueless Delegation

In this example, to resolve the DNS name `child.example.com`, the recursive resolver will ask the name server for the `example.com` zone, and the server will response with a referral record, indicating that the name is defined in the zone `child.example.com`, and the name server for that zone is `ns1.oob.example.net`. However, the referral response does not contain any glue records that provide an IP address for this name, so the recursive resolver must set aside its primary task (resolving `child.example.com`) and start a new task to resolve the name `ns1.oob.example.net`. If it is successful, the resolver now has an IP address for the name server of the original target zone, and it can ask the final question. It will only ask this final query if the resolution of the nameserver name is successful.

In this case we have modified the behaviour of the DNS server for the second zone (`oob.example.net`), such that all UDP reponses to queries for name server names in this zone are truncated. We also use an analogous categorization of these names to an odd/even system, such one half of experiment's unique name set uses a field in the query name that causes the nameserver to generate a truncated response (TC=1) with an empty answer section and the other half of the query names generate a perfectly normal complete DNS response, but with the TC bit set, indicating (incorrectly in this case) that the UDP response has been truncated.

If the DNS resolver is using the contents of a truncated UDP response, then it will be able to obtain the IP address of the nameserver and make the third query without needing to re-query using TCP. A standards-compliant resolver will ignore the contents of the UDP response that had the truncated bit set and re-query using TCP and use the response to make the third query.

The behaviours are determined by performing a full recording of all packets that arrive at and leave our servers, and then analysing these packets to determine the DNS resolution query and response sequences for each individual experiment.

How do DNS Resolvers behave with Truncation?

We conducted this test over a 7-day period in February 2024. The results are shown in Table 1

Tests	67,469,670
xTC	33,026,054
xTC-noTCP	306,271
Query Target	78,777
Rate	0.24%

Table 1 – Test of DNS Truncation

Over a 7-day period the test was conducted across some 67M end users, and of those some 33M were handed a DNS UDP response that contained the complete DNS answer but had the truncated bit set (called "xTC" in Table 1). Of these, some 9% of users, or some 306,000 did not re-query using TCP, yet in 78,777 cases the user's resolver(s) queried the target server. Each user is presented with unique domain names, so the only way the user's recursive resolvers could learn the IP address of the target name server was to use the information provided in the truncated DNS response.

A value of 0.24% is a relatively small rate, and it's possible that this is within the realm of experimental error, and noise factors in the DNS. However, if the locations of the users who are exhibiting this behaviour of using the data in a truncated response are concentrated in a small number of networks, then it is reasonable to conclude that this is systematic behaviour in the resolvers used on these users' networks.

Rank	AS	CC	Count	Rate	AS Name
1	4837	CN	35,555	45.13	CHINA UNICOM Backbone
2	17816	CN	33,699	42.78	China Unicom Guangdong
3	4134	CN	2,607	3.31	CHINANET Backbone

Table 2 – Networks that use Truncated Responses

As shown in Table 2, some 72,000 cases (91% of all such cases) where the resolver appears to be using truncated DNS response data occur for users located in just three networks, all located in China. It is reasonable to conclude that this is not within the bounds of experimental error but is systematic behaviour in the DNS resolvers used in these networks.

Finally, is this an issue for all resolvers within these three networks, or a more isolated set of behaviours? Let's add a couple of columns to Table 2, looking at the total number of samples drawn from each of these three networks, and the proportion of these samples that were observed to make use of the data contained in truncated responses. This is shown in Table 3.

Rank	AS	CC	TC Count	Sample Count	Rate	AS Name
1	4837	CN	35,555	1,968,824	1.8%	CHINA UNICOM Backbone
2	17816	CN	33,699	125,125	26.9%	China Unicom Guangdong
3	4134	CN	2,607	2,879,125	0.1%	CHINANET Backbone

Table 3 – Networks that use Truncated Responses as a proportion of per network test count

It appears that there is some systematic behaviour within AS17816, China Unicom Guandong Province, where the data in truncated responses is being used by the resolver, but in the other two networks the incidence is low enough that it may be a case of anomalous behaviour in resolvers located in edge networks rather than the ISP's own resolvers.

How reliable is Re-Query using TCP?

The advice in RFC 2181 has a second part. Not only is the information in a truncated response to be discarded by a resolver, but the resolver must re-query using TCP. Let's look at the comparison of experiments that were passed truncated UDP responses and experiments that re-queried using TCP.

The results of this analysis are shown in Table 4.

Tests	67,469,670
TC	62,471,679
TC-noTCP	562,334
No Query Target	483,557
Rate	0.77%

Table 4 – Test of DNS TCP Followup

Again, this is an encouragingly low number, where some 483,000 experiments did not followup the truncated UDP response with a TCP session. The distribution of where these cases occurred in terms of the host network is shown in Table 5 for the top 10 such networks.

Rank	AS	CC	Count	Rate	AS Name
1	45609	IN	98,527	20.4%	Bharti Airtel, India
2	7418	CL	96,826	20.1%	Telefonica Chile
3	52341	CL	71,103	14.7%	WOM Chile
4	17816	CN	36,285	7.5%	China Unicom Guangdong, China
5	27995	CL	33,474	6.9%	Claro Chile
6	4837	CN	32,273	6.7%	China Unicom Backbone, China
7	6535	CL	15,841	3.3%	Telmex Servicios, Chile
8	6849	UA	13,681	2.8%	UKRTELNET, Ukraine
9	43197	TJ	10,097	2.1%	PJSC TT mobile, Tajikistan
10	4134	CN	9,114	1.9%	CHINANET Backbone, China

Table 5 – Networks that contain non-TCP DNS resolvers

The occurrence of recursive resolvers in Chile that appear to be unable to perform a DNS query over TCP is curious. It may be that these resolvers are using some common configuration that prevents the use of TCP, or our TCP server, located in a Linode server farm in North America, in inaccessible from Chile, but this is in the realm of conjecture given that we cannot instrument the user side with diagnostics to identify the root cause here. Resolver software issues seems to be the likely explanation, and throttling the number of concurrent TCP sessions as a means of protecting the resolver engine from resource exhaustion is a plausible explanation, but why are ISPs in Chile featured so prominently in this list?

Again, is this an issue for all resolvers within these ten networks, or a more isolated set of behaviours?

Rank	AS	CC	TC Count	Sample Count	Rate	AS Name
1	45609	IN	98,527	1,428,411	6.9%	Bharti Airtel, India
2	7418	CL	96,826	128,305	75.5%	Telefonica Chile
3	52341	CL	71,103	78,023	91.1%	WOM Chile
4	17816	CN	36,285	125,125	29.0%	China Unicom Guangdong, China
5	27995	CL	33,474	36,524	91.6%	Claro Chile
6	4837	CN	32,273	1,968,824	1.6%	China Unicom Backbone, China
7	6535	CL	15,841	17,331	91.4%	Telmex Servicios, Chile
8	6849	UA	13,681	14,640	93.4%	UKRTELNET, Ukraine
9	43197	TJ	10,097	11,801	85.6%	PJSC TT mobile, Tajikistan
10	4134	CN	9,114	2,879,125	0.3%	CHINANET Backbone, China

Table 6 – Networks that use Truncated Responses as a proportion of per network test count

The four Chilean networks all have a high non-TCP count, and the anomalous behaviour is more likely to be in the ISP's resolver environment than in the edge networks that connect to the ISP. The same is likely the case for AS6849, URTELNET in the Ukraine and AS 43197 PISC TT in Tajikistan. The intermediate rates seen in AS45609, Bharti Airtel, and AS 17816, China Unicom Guangdong, point to a large-scale distributed DNS resolver infrastructure where some parts of this resolver environment cannot perform TCP queries while the rest of the infrastructure can perform satisfactorily.

Conclusions

The overall picture of DNS handling of truncated responses in UDP and the consequent failover to TCP is looking surprisingly good.

The use of data in truncated responses appears to be confined to some resolver systems deployed in networks located in China, and in this measurement exercise, spanning some 67M end users drawn from over the entire public Internet over a period of one week it was observed that only 0.24% of users were using DNS resolution environments which used the data held in truncated DNS responses over UDP.

A similar outcome was observed when looking for a failure to re-query using TCP upon receipt of a truncated DNS response over TCP. The observed failure rate was 0.77% of users.

This measurement exercise does not attempt to identify individual recursive resolvers. Modern high-capacity recursive resolver systems are compound systems composed of a number of DNS resolution engines. Some engines may use only UDP, while TCP tasks may be handed to other engines that are specifically configured to manage the somewhat different TCP load profile. Without undertaking an effort to identify the modes of behaviour of these compound systems, identifying individual resolver systems by their IP address is not overly useful when trying to identify systemic behaviour anomalies.

It's also noted that this is a measurement exercise that has been undertaken in a dual-stack context. The DNS is relatively persistent in attempting to secure a response, and in this case a dual protocol environment can work to the advantage of the end user client, in that even if there are systemic issues with DNS behaviour in one protocol (such as Packet Fragmentation in IPv6) a DNS resolver in a dual-stacked environment will typically attempt to use the other protocol, as well as other servers, in its attempt to find a usable response. We will look

at the issues of using a single protocol environment with IPv6 and the DNS when using truncation in the next article in this set of DNS measurement reports, in an effort to understand to what extent a dual-protocol is able to provide a more resilient service than that of a single protocol environment.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.

Author

Geoff Huston AM, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net