Internet Engineering Task Force (IETF)
Request for Comments: 7766
Obsoletes: 5966
Updates: 1035, 1123
Category: Standards Track
ISSN: 2070-1721
March 2016

J. Dickinson
S. Dickinson
Sinodun
R. Bellis
ISC
A. Mankin
D. Wessels
Verisign Labs

DNS Transport over TCP - Implementation Requirements

Abstract

This document specifies the requirement for support of TCP as a transport protocol for DNS implementations and provides guidelines towards DNS-over-TCP performance on par with that of DNS-over-UDP. This document obsoletes RFC 5966 and therefore updates RFC 1035 and RFC 1123.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7766.

Copyright Notice

Table of Contents

1. Introduction

Most DNS [RFC1034] transactions take place over UDP [RFC768]. TCP [RFC793] is always used for full zone transfers (using AXFR) and is often used for messages whose sizes exceed the DNS protocol's original 512-byte limit. The growing deployment of DNS Security (DNSSEC) and IPv6 has increased response sizes and therefore the use of TCP. The need for increased TCP use has also been driven by the protection it provides against address spoofing and therefore exploitation of DNS in reflection/amplification attacks. It is now widely used in Response Rate Limiting [RRL1] [RRL2]. Additionally, recent work on DNS privacy solutions such as [DNS-over-TLS] is another motivation to revisit DNS-over-TCP requirements.

Section 6.1.3.2 of [RFC1123] states:

> DNS resolvers and recursive servers MUST support UDP, and SHOULD support TCP, for sending (non-zone-transfer) queries.

However, some implementors have taken the text quoted above to mean that TCP support is an optional feature of the DNS protocol.

The majority of DNS server operators already support TCP, and the default configuration for most software implementations is to support TCP. The primary audience for this document is those implementors whose limited support for TCP restricts interoperability and hinders deployment of new DNS features.

This document therefore updates the core DNS protocol specifications such that support for TCP is henceforth a REQUIRED part of a full DNS protocol implementation.

There are several advantages and disadvantages to the increased use of TCP (see Appendix A) as well as implementation details that need to be considered. This document addresses these issues and presents TCP as a valid transport alternative for DNS. It extends the content of [RFC5966], with additional considerations and lessons learned from research, developments, and implementation of TCP in DNS and in other Internet protocols.

Whilst this document makes no specific requirements for operators of DNS servers to meet, it does offer some suggestions to operators to help ensure that support for TCP on their servers and network is optimal. It should be noted that failure to support TCP (or the blocking of DNS over TCP at the network layer) will probably result in resolution failure and/or application-level timeouts.

2. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Terminology

- Persistent connection: a TCP connection that is not closed either by the server after sending the first response nor by the client after receiving the first response.

- Connection Reuse: the sending of multiple queries and responses over a single TCP connection.

- Idle DNS-over-TCP session: Clients and servers view application-level idleness differently. A DNS client considers an established DNS-over-TCP session to be idle when it has no pending queries to send and there are no outstanding responses. A DNS server considers an established DNS-over-TCP session to be idle when it has sent responses to all the queries it has received on that connection.

- Pipelining: the sending of multiple queries and responses over a single TCP connection but not waiting for any outstanding replies before sending another query.

- Out-of-Order Processing: The processing of queries concurrently and the returning of individual responses as soon as they are available, possibly out of order. This will most likely occur in recursive servers; however, it is possible in authoritative servers that, for example, have different backend data stores.

4. Discussion

In the absence of EDNS0 (Extension Mechanisms for DNS 0 [RFC6891]; see below), the normal behaviour of any DNS server that needs to send a UDP response that would exceed the 512-byte limit is for the server to truncate the response so that it fits within that limit and then set the TC flag in the response header. When the client receives such a response, it takes the TC flag as an indication that it should retry over TCP instead.

RFC 1123 also says:

> ... it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that applies to UDP, and hence will require TCP. Thus, resolvers and name servers should implement TCP services as a backup to UDP today, with the knowledge that they will require the TCP service in the future.

Existing deployments of DNSSEC [RFC4033] have shown that truncation at the 512-byte boundary is now commonplace. For example, a Non-Existent Domain (NXDOMAIN) (RCODE == 3) response from a DNSSEC-signed zone using NextSECure 3 (NSEC3) [RFC5155] is almost invariably larger than 512 bytes.

Since the original core specifications for DNS were written, the extension mechanisms for DNS have been introduced. These extensions can be used to indicate that the client is prepared to receive UDP responses larger than 512 bytes. An EDNS0-compatible server receiving a request from an EDNS0-compatible client may send UDP packets up to that client's announced buffer size without truncation.

However, transport of UDP packets that exceed the size of the path MTU causes IP packet fragmentation, which has been found to be unreliable in many circumstances. Many firewalls routinely block fragmented IP packets, and some do not implement the algorithms necessary to reassemble fragmented packets. Worse still, some network devices deliberately refuse to handle DNS packets containing EDNS0 options. Other issues relating to UDP transport and packet size are discussed in [RFC5625].

The MTU most commonly found in the core of the Internet is around 1500 bytes, and even that limit is routinely exceeded by DNSSEC-signed responses.

The future that was anticipated in RFC 1123 has arrived, and the only standardised UDP-based mechanism that may have resolved the packet size issue has been found inadequate.

5. Transport Protocol Selection

Section 6.1.3.2 of [RFC1123] is updated: All general-purpose DNS implementations MUST support both UDP and TCP transport.

- o Authoritative server implementations MUST support TCP so that they do not limit the size of responses to what fits in a single UDP packet.

- o Recursive server (or forwarder) implementations MUST support TCP so that they do not prevent large responses from a TCP-capable server from reaching its TCP-capable clients.

- o Stub resolver implementations (e.g., an operating system's DNS resolution library) MUST support TCP since to do otherwise would limit the interoperability between their own clients and upstream servers.

Regarding the choice of when to use UDP or TCP, Section 6.1.3.2 of RFC 1123 also says:

> ... a DNS resolver or server that is sending a non-zone-transfer query MUST send a UDP query first.

This requirement is hereby relaxed. Stub resolvers and recursive resolvers MAY elect to send either TCP or UDP queries depending on local operational reasons. TCP MAY be used before sending any UDP queries. If the resolver already has an open TCP connection to the server, it SHOULD reuse this connection. In essence, TCP ought to be considered a valid alternative transport to UDP, not purely a retry option.

In addition, it is noted that all recursive and authoritative servers MUST send responses using the same transport as the query arrived on. In the case of TCP, this MUST also be the same connection.

6. Connection Handling

6.1. Current Practices

Section 4.2.2 of [RFC1035] says:

- The server should assume that the client will initiate connection closing, and should delay closing its end of the connection until all outstanding client requests have been satisfied.

- If the server needs to close a dormant connection to reclaim resources, it should wait until the connection has been idle for a period on the order of two minutes. In particular, the server should allow the SOA and AXFR request sequence (which begins a refresh operation) to be made on a single connection. Since the server would be unable to answer queries anyway, a unilateral close or reset may be used instead of graceful close.

Other more modern protocols (e.g., HTTP/1.1 [RFC7230], HTTP/2 [RFC7540]) have support by default for persistent TCP connections for all requests. Connections are then normally closed via a 'connection close' signal from one party.

The description in [RFC1035] is clear that servers should view connections as persistent (particularly after receiving an SOA), but unfortunately does not provide enough detail for an unambiguous interpretation of client behaviour for queries other than a SOA. Additionally, DNS does not yet have a signalling mechanism for connection timeout or close, although some have been proposed.

6.1.1. Clients

There is no clear guidance today in any RFC as to when a DNS client should close a TCP connection, and there are no specific recommendations with regard to DNS client idle timeouts. However, at the time of writing, it is common practice for clients to close the TCP connection after sending a single request (apart from the SOA/AXFR case).

6.1.2. Servers

Many DNS server implementations use a long fixed idle timeout and default to a small number of TCP connections. They also offer little in the way of TCP connection management options. The disadvantages of this include:

- o Operational experience has shown that long server timeouts can easily cause resource exhaustion and poor response under heavy load.

- o Intentionally opening many connections and leaving them idle can trivially create a TCP denial of service (DoS) attack as many DNS servers are poorly equipped to defend against this by modifying their idle timeouts or other connection management policies.

- o A modest number of clients that all concurrently attempt to use persistent connections with non-zero idle timeouts to such a server could unintentionally cause the same DoS problem.

Note that this DoS is only on the TCP service. However, in these cases, it affects not only clients that wish to use TCP for their queries for operational reasons, but all clients that choose to fall back to TCP from UDP after receiving a TC=1 flag.

6.2. Recommendations

The following sections include recommendations that are intended to result in more consistent and scalable implementations of DNS-over-TCP.

6.2.1. Connection Reuse

One perceived disadvantage to DNS over TCP is the added connection setup latency, generally equal to one RTT. To amortise connection setup costs, both clients and servers SHOULD support connection reuse by sending multiple queries and responses over a single persistent TCP connection.

When sending multiple queries over a TCP connection, clients MUST NOT reuse the DNS Message ID of an in-flight query on that connection in order to avoid Message ID collisions. This is especially important if the server could be performing out-of-order processing (see Section 7).

6.2.1.1. Query Pipelining

Due to the historical use of TCP primarily for zone transfer and truncated responses, no existing RFC discusses the idea of pipelining DNS queries over a TCP connection.

In order to achieve performance on par with UDP, DNS clients SHOULD pipeline their queries. When a DNS client sends multiple queries to a server, it SHOULD NOT wait for an outstanding reply before sending the next query. Clients SHOULD treat TCP and UDP equivalently when considering the time at which to send a particular query.

It is likely that DNS servers need to process pipelined queries concurrently and also send out-of-order responses over TCP in order to provide the level of performance possible with UDP transport. If TCP performance is of importance, clients might find it useful to use server processing times as input to server and transport selection algorithms.

DNS servers (especially recursive) MUST expect to receive pipelined queries. The server SHOULD process TCP queries concurrently, just as it would for UDP. The server SHOULD answer all pipelined queries, even if they are received in quick succession. The handling of responses to pipelined queries is covered in Section 7.

6.2.2. Concurrent Connections

To mitigate the risk of unintentional server overload, DNS clients MUST take care to minimize the number of concurrent TCP connections made to any individual server. It is RECOMMENDED that for any given client/server interaction there SHOULD be no more than one connection for regular queries, one for zone transfers, and one for each protocol that is being used on top of TCP (for example, if the resolver was using TLS). However, it is noted that certain primary/secondary configurations with many busy zones might need to use more than one TCP connection for zone transfers for operational reasons (for example, to support concurrent transfers of multiple zones).

Similarly, servers MAY impose limits on the number of concurrent TCP connections being handled for any particular client IP address or subnet. These limits SHOULD be much looser than the client guidelines above, because the server does not know, for example, if a client IP address belongs to a single client, is multiple resolvers on a single machine, or is multiple clients behind a device performing Network Address Translation (NAT).

6.2.3. Idle Timeouts

To mitigate the risk of unintentional server overload, DNS clients MUST take care to minimise the idle time of established DNS-over-TCP sessions made to any individual server. DNS clients SHOULD close the TCP connection of an idle session, unless an idle timeout has been established using some other signalling mechanism, for example, [edns-tcp-keepalive].

To mitigate the risk of unintentional server overload, it is RECOMMENDED that the default server application-level idle period be on the order of seconds, but no particular value is specified. In practice, the idle period can vary dynamically, and servers MAY allow idle connections to remain open for longer periods as resources permit. A timeout of at least a few seconds is advisable for normal operations to support those clients that expect the SOA and AXFR request sequence to be made on a single connection as originally specified in [RFC1035]. Servers MAY use zero timeouts when they are experiencing heavy load or are under attack.

DNS messages delivered over TCP might arrive in multiple segments. A DNS server that resets its idle timeout after receiving a single segment might be vulnerable to a "slow-read attack". For this reason, servers SHOULD reset the idle timeout on the receipt of a full DNS message, rather than on receipt of any part of a DNS message.

6.2.4. Teardown

Under normal operation DNS clients typically initiate connection closing on idle connections; however, DNS servers can close the connection if the idle timeout set by local policy is exceeded. Also, connections can be closed by either end under unusual conditions such as defending against an attack or system failure/ reboot.

DNS clients SHOULD retry unanswered queries if the connection closes before receiving all outstanding responses. No specific retry algorithm is specified in this document.

If a DNS server finds that a DNS client has closed a TCP session (or if the session has been otherwise interrupted) before all pending responses have been sent, then the server MUST NOT attempt to send those responses. Of course, the DNS server MAY cache those responses.

7. Response Reordering

RFC 1035 is ambiguous on the question of whether TCP responses may be reordered -- the only relevant text is in Section 4.2.1, which relates to UDP:

> Queries or their responses may be reordered by the network, or by processing in name servers, so resolvers should not depend on them being returned in order.

For the avoidance of future doubt, this requirement is clarified. Authoritative servers and recursive resolvers are RECOMMENDED to support the preparing of responses in parallel and sending them out of order, regardless of the transport protocol in use. Stub and recursive resolvers MUST be able to process responses that arrive in a different order than that in which the requests were sent, regardless of the transport protocol in use.

In order to achieve performance on par with UDP, recursive resolvers SHOULD process TCP queries in parallel and return individual responses as soon as they are available, possibly out of order.

Since pipelined responses can arrive out of order, clients MUST match responses to outstanding queries on the same TCP connection using the Message ID. If the response contains a question section, the client MUST match the QNAME, QCLASS, and QTYPE fields. Failure by clients to properly match responses to outstanding queries can have serious consequences for interoperability.

8. TCP Message Length Field

DNS clients and servers SHOULD pass the two-octet length field, and the message described by that length field, to the TCP layer at the same time (e.g., in a single "write" system call) to make it more likely that all the data will be transmitted in a single TCP segment. This is for reasons of both efficiency and to avoid problems due to some DNS server implementations behaving undesirably when reading data from the TCP layer (due to a lack of clarity in previous documents). For example, some DNS server implementations might abort a TCP session if the first "read" from the TCP layer does not contain both the length field and the entire message.

To clarify, DNS servers MUST NOT close a connection simply because the first "read" from the TCP layer does not contain the entire DNS message, and servers SHOULD apply the connection timeouts as specified in Section 6.2.3.

9. TCP Fast Open

This section is non-normative.

TCP Fast Open (TFO) [RFC7413] allows data to be carried in the SYN packet, reducing the cost of reopening TCP connections. It also saves up to one RTT compared to standard TCP.

TFO mitigates the security vulnerabilities inherent in sending data in the SYN, especially on a system like DNS where amplification attacks are possible, by use of a server-supplied cookie. TFO clients request a server cookie in the initial SYN packet at the start of a new connection. The server returns a cookie in its SYN-ACK. The client caches the cookie and reuses it when opening subsequent connections to the same server.

The cookie is stored by the client's TCP stack (kernel) and persists if either the client or server processes are restarted. TFO also falls back to a regular TCP handshake gracefully.

DNS services taking advantage of IP anycast [RFC4786] might need to take additional steps when enabling TFO. From [RFC7413]:

> Servers behind load balancers that accept connection requests to the same server IP address should use the same key such that they generate identical Fast Open cookies for a particular client IP address. Otherwise, a client may get different cookies across connections; its Fast Open attempts would fall back to the regular 3WHS.

When DNS-over-TCP is a transport for DNS private exchange, as in [DNS-over-TLS], the implementor needs to be aware of TFO and to ensure that data requiring protection (e.g. data for a DNS query) is not accidentally transported in the clear. See [DNS-over-TLS] for discussion.

10. Security Considerations

Some DNS server operators have expressed concern that wider promotion and use of DNS over TCP will expose them to a higher risk of DoS attacks on TCP (both accidental and deliberate).

Although there is a higher risk of some specific attacks against TCP-enabled servers, techniques for the mitigation of DoS attacks at the network level have improved substantially since DNS was first designed.

Readers are advised to familiarise themselves with [CPNI-TCP], a security assessment of TCP that details known TCP attacks and countermeasures and that references most of the relevant RFCs on this topic.

To mitigate the risk of DoS attacks, DNS servers are advised to engage in TCP connection management. This could include maintaining state on existing connections, reusing existing connections, and controlling request queues to enable fair use. It is likely to be advantageous to provide configurable connection management options, for example:

- total number of TCP connections
- maximum TCP connections per source IP address or subnet
- TCP connection idle timeout
- maximum DNS transactions per TCP connection
- maximum TCP connection duration

No specific values are recommended for these parameters.

Operators are advised to familiarise themselves with the configuration and tuning parameters available in the TCP stack of the operating system. However, detailed advice on this is outside the scope of this document.

Operators of recursive servers are advised to ensure that they only accept connections from expected clients (for example, by the use of an Access Control List (ACL)) and do not accept them from unknown sources. In the case of UDP traffic, this will help protect against reflection attacks [RFC5358]; and in the case of TCP traffic, it will prevent an unknown client from exhausting the server's limits on the number of concurrent connections.

11. References

11.1. Normative References

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <http://www.rfc-editor.org/info/rfc1123>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <http://www.rfc-editor.org/info/rfc4786>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <http://www.rfc-editor.org/info/rfc5155>.

[RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", BCP 140, RFC 5358, DOI 10.17487/RFC5358, October 2008, <http://www.rfc-editor.org/info/rfc5358>.

[RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <http://www.rfc-editor.org/info/rfc5625>.

[RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, DOI 10.17487/RFC5966, August 2010, <http://www.rfc-editor.org/info/rfc5966>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <http://www.rfc-editor.org/info/rfc6891>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

11.2. Informative References

[Connection-Oriented-DNS]
Zhu, L., Hu, Z., Heidemann, J., Wessels, D., Mankin, A., and N. Somaiya, "Connection-Oriented DNS to Improve Privacy and Security", 2015 IEEE Symposium on Security and Privacy (SP), DOI 10.1109/SP.2015.18, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7163025>.

[CPNI-TCP]
CPNI, "Security Assessment of the Transmission Control Protocol (TCP)", 2009, <http://www.gont.com.ar/papers/tn-03-09-security-assessment-TCP.pdf>.

[DNS-over-TLS]
Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over TLS", Work in Progress, draft-ietf-dprive-dns-over-tls-06, February 2016.

[edns-tcp-keepalive]
Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The edns-tcp-keepalive EDNS0 Option", Work in Progress, draft-ietf-dnsop-edns-tcp-keepalive-03, September 2015.

[fragmentation-considered-poisonous]
Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", May 2012, <http://arxiv.org/abs/1205.4011>.

[RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, DOI 10.17487/RFC5405, November 2008, <http://www.rfc-editor.org/info/rfc5405>.

[RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824, DOI 10.17487/RFC6824, January 2013, <http://www.rfc-editor.org/info/rfc6824>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <http://www.rfc-editor.org/info/rfc7413>.

[RRL1] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL)", ISC-TN 2012-1-Draft1, April 2012, <https://ftp.isc.org/isc/pubs/tn/isc-tn-2012-1.txt>.

[RRL2] ISC Support, "Using the Response Rate Limiting Feature in BIND 9.10", ISC Knowledge Base AA-00994, June 2013, <https://kb.isc.org/article/AA-00994/>.

Appendix A. Summary of Advantages and Disadvantages to Using TCP for DNS

The TCP handshake generally prevents address spoofing and, therefore, the reflection/amplification attacks that plague UDP.

IP fragmentation is less of a problem for TCP than it is for UDP. TCP stacks generally implement Path MTU Discovery so they can avoid IP fragmentation of TCP segments. UDP, on the other hand, does not provide reassembly; this means datagrams that exceed the path MTU size must experience fragmentation [RFC5405]. Middleboxes are known to block IP fragments, leading to timeouts and forcing client implementations to "hunt" for EDNS0 reply size values supported by the network path. Additionally, fragmentation may lead to cache poisoning [fragmentation-considered-poisonous].

TCP setup costs an additional RTT compared to UDP queries. Setup costs can be amortised by reusing connections, pipelining queries, and enabling TCP Fast Open.

TCP imposes additional state-keeping requirements on clients and servers. The use of TCP Fast Open reduces the cost of closing and reopening TCP connections.

Long-lived TCP connections to anycast servers might be disrupted due to routing changes. Clients utilizing TCP for DNS need to always be prepared to re-establish connections or otherwise retry outstanding queries. It might also be possible for Multipath TCP [RFC6824] to allow a server to hand a connection over from the anycast address to a unicast address.

There are many "middleboxes" in use today that interfere with TCP over port 53 [RFC5625]. This document does not propose any solutions, other than to make it absolutely clear that TCP is a valid transport for DNS and support for it is a requirement for all implementations.

A more in-depth discussion of connection-oriented DNS can be found elsewhere [Connection-Oriented-DNS].

Appendix B. Changes to RFC 5966

This document obsoletes [RFC5966] and differs from it in several respects. An overview of the most substantial changes/updates that implementors should take note of is given below.

1. A Terminology section (Section 3) is added defining several new concepts.

2. Paragraph 3 of Section 5 puts TCP on a more equal footing with UDP than RFC 5966 does. For example, it states:

    1. TCP MAY be used before sending any UDP queries.

    2. TCP ought to be considered a valid alternative transport to UDP, not purely a fallback option.

3. Section 6.2.1 adds a new recommendation that TCP connection reuse SHOULD be supported.

4. Section 6.2.1.1 adds a new recommendation that DNS clients SHOULD pipeline their queries and DNS servers SHOULD process pipelined queries concurrently.

5. Section 6.2.2 adds new recommendations on the number and usage of TCP connections for client/server interactions.

6. Section 6.2.3 adds a new recommendation that DNS clients SHOULD close idle sessions unless using a signalling mechanism.

7. Section 7 clarifies that servers are RECOMMENDED to prepare TCP responses in parallel and send answers out of order. It also clarifies how TCP queries and responses should be matched by clients.

8. Section 8 adds a new recommendation about how DNS clients and servers should handle the 2-byte message length field for TCP messages.

9. Section 9 adds a non-normative discussion of the use of TCP Fast Open.

10. Section 10 adds new advice regarding DoS mitigation techniques.

Acknowledgements

The authors would like to thank Francis Dupont and Paul Vixie for their detailed reviews, as well as Andrew Sullivan, Tony Finch, Stephane Bortzmeyer, Joe Abley, Tatuya Jinmei, and the many others who contributed to the mailing list discussion. Also, the authors thank Liang Zhu, Zi Hu, and John Heidemann for extensive DNS-over-TCP discussions and code, and Lucie Guiraud and Danny McPherson for reviewing early draft versions of this document. We would also like to thank all those who contributed to RFC 5966.

Authors' Addresses

John Dickinson
Sinodun Internet Technologies
Magdalen Centre
Oxford Science Park
Oxford OX4 4GA
United Kingdom

Email: jad@sinodun.com
URI: http://sinodun.com

Sara Dickinson
Sinodun Internet Technologies
Magdalen Centre
Oxford Science Park
Oxford OX4 4GA
United Kingdom

Email: sara@sinodun.com
URI: http://sinodun.com

Ray Bellis
Internet Systems Consortium, Inc
950 Charter Street
Redwood City, CA 94063
United States

Phone: +1 650 423 1200
Email: ray@isc.org
URI: http://www.isc.org

Allison Mankin
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
United States

Phone: +1 301 728 7198
Email: allison.mankin@gmail.com

Duane Wessels
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
United States

Phone: +1 703 948 3200
Email: dwessels@verisign.com