

dnsop
Internet-Draft
Obsoletes: [5966](#) (if approved)
Intended status: Standards Track
Expires: September 10, 2015

J. Dickinson
Sinodun Internet Technologies
R. Bellis
Nominet
A. Mankin
D. Wessels
Verisign Labs
March 9, 2015

DNS Transport over TCP - Implementation Requirements
draft-ietf-dnsop-5966bis-01

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.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Requirements Terminology	3
3.	Terminology	3
4.	Discussion	3
5.	Transport Protocol Selection	4
6.	Connection Handling	5
7.	Query Pipelining	6
8.	Response Reordering	7
9.	TCP Fast Open	8
10.	IANA Considerations	8
11.	Security Considerations	8
12.	Acknowledgements	9
13.	References	9
13.1.	Normative References	9
13.2.	Informative References	10
Appendix A.	Summary of Advantages and Disadvantages to using TCP for DNS	11
Appendix B.	Changes -00 to -01	11
Appendix C.	Changes to RFC 5966	12
Authors' Addresses	12

[1.](#) Introduction

Most DNS [[RFC1034](#)] transactions take place over UDP [[RFC0768](#)]. TCP [[RFC0793](#)] is always used for full zone transfers (AXFR) and is often used for messages whose sizes exceed the DNS protocol's original 512-byte limit.

[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 failure to support TCP restricts interoperability and limits 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 as well as implementation details that need to be considered. This document addresses these issues and therefore extends the content of [\[RFC5966\]](#), with additional considerations and lessons learned from new research and implementations [\[Connection-Oriented-DNS\]](#).

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) may 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

- o Connection Reuse: the sending of multiple queries and responses over a single TCP connection.
- o Pipelining: the sending of multiple queries and responses over a single TCP connection but not waiting for any outstanding replies before sending another query.
- o Out-Of-Order Processing: The processing of queries in parallel 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) (see below), the normal behaviour of any DNS server needing 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 DNS Security (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 (EDNS0 [[RFC6891](#)]) 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 some 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

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 their interoperability with their own clients and with 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. A resolver MAY elect to send either TCP or UDP queries depending on local operational reasons. TCP MAY be used before sending any UDP queries. If it already has an open TCP connection to the server it SHOULD reuse this connection. In essence, TCP SHOULD be considered as valid a transport as UDP.

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

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

DNS currently has no connection signaling mechanism. Clients and servers may close a connection at any time. Clients MUST be prepared to retry failed queries on broken connections.

[Section 4.2.2 of \[RFC1035\]](#) says:

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 a graceful close.

Other more modern protocols (e.g., HTTP/1.1 [[RFC7230](#)]) have support for persistent TCP connections and operational experience has shown that long timeouts can easily cause resource exhaustion and poor response under heavy load. Intentionally opening many connections and leaving them dormant can trivially create a "denial-of-service" attack.

It is therefore RECOMMENDED that the default application-level idle period should be of the order of seconds, but no particular value is specified. In practice, the idle period may vary dynamically, and servers MAY allow dormant connections to remain open for longer periods as resources permit.

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. Similarly, servers MAY impose limits on the number of concurrent TCP connections being handled for any particular client. 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. The server MUST NOT enforce these rules for a particular client because it does not know if the client IP address belongs to a single client or is, for example, multiple clients behind NAT.

For reasons of efficiency, implementations SHOULD wherever possible attempt to coalesce the two byte length field and subsequent DNS payload data into a single packet.

If a server finds that a 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 server MAY cache those responses.

7. Query Pipelining

Due to the 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.

DNS servers (especially recursive) SHOULD expect to receive pipelined queries. The server should process TCP queries in parallel, just as it would for UDP. The handling of responses to pipelined queries is covered in the following section.

When pipelining queries over TCP it is very easy to send more DNS queries than there are DNS Message ID's. Implementations MUST take care to check their list of outstanding DNS Message ID's before sending a new query over an existing TCP connection. This is especially important if the server could be performing out-of-order processing. In addition, when sending multiple queries over TCP it is very easy for a name server to overwhelm its own network interface. Implementations MUST take care to manage buffer sizes or to throttle writes to the network interface.

8. 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 sending of responses in parallel and/or 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 to 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 responses may arrive out-of-order, clients must take care to match responses to outstanding queries, using the ID field, port number, query name/type/class, and any other relevant protocol features.

9. TCP Fast Open

This section is non-normative.

TCP fastopen [[RFC7413](#)] (TFO) allows data to be carried in the SYN packet. 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.

Adding support for this to existing name server implementations is relatively easy, but does require source code modifications. On the client, the call to `connect()` is replaced with a TFO aware version of `sendmsg()` or `sendto()`. On the server, TFO must be switched into server mode by changing the kernel parameter (`net.ipv4.tcp_fastopen` on Linux) to enable the server bit (Set the integer value to 2 (server only) or 3 (client and server)) and setting a socket option between the `bind()` and `listen()` calls.

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

Servers 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 regular 3WHS.

10. IANA Considerations

This memo includes no request to IANA.

11. Security Considerations

Some DNS server operators have expressed concern that wider use of DNS over TCP will expose them to a higher risk of denial-of-service (DoS) attacks.

Although there is a higher risk of such 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](#)].

Operators of recursive servers should ensure that they only accept connections from expected clients, and do not accept them from unknown sources. In the case of UDP traffic, this will help protect against reflector 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.

12. Acknowledgements

The authors would like to thank Francis Dupont for his detailed review, Liang Zhu, Zi Hu, and John Heidemann for extensive DNS-over-TCP discussions and code and Lucie Guiraud and Danny McPherson for reviewing early versions of this document. We would also like to thank all those who contributed to [RFC 5966](#).

13. References

13.1. Normative References

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, [RFC 768](#), August 1980.
- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), September 1981.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, [RFC 1034](#), November 1987.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), November 1987.
- [RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, [RFC 1123](#), October 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", [RFC 4033](#), March 2005.

- [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", [BCP 126](#), [RFC 4786](#), December 2006.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", [RFC 5155](#), March 2008.
- [RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", [BCP 140](#), [RFC 5358](#), October 2008.
- [RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", [BCP 152](#), [RFC 5625](#), August 2009.
- [RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", [RFC 5966](#), August 2010.
- [RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, [RFC 6891](#), April 2013.
- [RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", [RFC 7230](#), June 2014.

13.2. Informative References

- [CPNI-TCP]
CPNI, "Security Assessment of the Transmission Control Protocol (TCP)", 2009, <<http://www.cpni.gov.uk/Docs/tn-03-09-security-assessment-TCP.pdf>>.
- [Connection-Oriented-DNS]
Zhu, L., Hu, Z., Heidemann, J., Wessels, D., Mankin, A., and N. Somaiya, "T-DNS: Connection-Oriented DNS to Improve Privacy and Security (extended)", <<http://www.isi.edu/publications/trpublic/files/tr-693.pdf>>.
- [RFC6824] Ford, A., Raiciu, C., Handley, M., and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses", [RFC 6824](#), January 2013.
- [RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", [RFC 7413](#), December 2014.
- [fragmentation-considered-poisonous]
Herzberg, A. and H. Shulman, "Fragmentation Considered Poisonous", May 2012, <<http://arxiv.org/abs/1205.4011>>.

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 which plague UDP.

TCP does not suffer from UDP's issues with fragmentation. 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 amortized 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 re-opening TCP connections.

Long-lived TCP connections to anycast servers may be disrupted due to routing changes. Clients utilizing TCP for DNS must always be prepared to re-establish connections or otherwise retry outstanding queries. It may also be possible for TCP Multipath [[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 must be supported by all implementations.

Appendix B. Changes -00 to -01

- o Changed updates to obsoletes [RFC 5966](#).
- o Improved text in [Section 4](#) Transport Protocol Selection to change "TCP SHOULD NOT be used only for the transfers and as a fallback" to make the intention clearer and more consistent.
- o Reference to TCP FASTOPEN updated now that it is an RFC.
- o Added paragraph to say that implementations MUST NOT send the TCP framing 2 byte length field in a separate packet to the DNS message.
- o Added Terminology section.

- o Changed should and RECOMMENDED in reference to parallel processing to SHOULD in sections [7](#) and [8](#).
- o Added text to address what a server should do when a client closes the TCP connection before pending responses are sent.
- o Moved the Advantages and Disadvantages section to an appendix.

Appendix C. Changes to [RFC 5966](#)

This document differs from [RFC 5966](#) in four additions:

1. DNS implementations are recommended not only to support TCP but to support it on an equal footing with UDP
2. DNS implementations are recommended to support reuse of TCP connections
3. DNS implementations are recommended to support pipelining and out of order processing of the query stream
4. A non-normative discussion of use of TCP Fast Open is added

Authors' Addresses

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

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

Ray Bellis
Nominet
Edmund Halley Road
Oxford OX4 4DQ
UK

Phone: +44 1865 332211
Email: ray.bellis@nominet.org.uk
URI: <http://www.nominet.org.uk/>

Allison Mankin
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
US

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

Duane Wessels
Verisign Labs
12061 Bluemont Way
Reston, VA 20190
US

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

