dnsop
Internet-Draft
Intended status: Standards Track
Expires: July 9, 2016

P. Wouters
Red Hat
J. Abley
Dyn, Inc.
S. Dickinson
Sinodun
R. Bellis
ISC

January 6, 2016

The edns-tcp-keepalive EDNS0 Option
draft-ietf-dnsop-edns-tcp-keepalive-05

Abstract

DNS messages between clients and servers may be received over either UDP or TCP. UDP transport involves keeping less state on a busy server, but can cause truncation and retries over TCP. Additionally, UDP can be exploited for reflection attacks. Using TCP would reduce retransmits and amplification. However, clients commonly use TCP only for retries and servers typically use idle timeouts on the order of seconds.

This document defines an EDNS0 option ("edns-tcp-keepalive") that allows DNS servers to signal a variable idle timeout. This signalling encourages the use of long-lived TCP connections by allowing the state associated with TCP transport to be managed effectively with minimal impact on the DNS transaction time.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DNS messages between clients and servers may be received over either UDP or TCP [RFC1035]. Historically, DNS clients used API's that only facilitated sending and receiving a single query over either UDP or TCP. New APIs and deployment of DNSSEC validating resolvers on hosts that in the past were using stub resolving only is increasing the DNS client base that prefer using long lived TCP connections. Long-lived TCP connections can result in lower request latency than the case where UDP transport is used and truncated responses are received. This is because clients that retry over TCP following a truncated UDP response typically only use the TCP session for a single (request, response) pair, continuing with UDP transport for subsequent queries.

UDP transport is stateless, and hence presents a much lower resource burden on a busy DNS server than TCP. An exchange of DNS messages over UDP can also be completed in a single round trip between communicating hosts, resulting in optimally-short transaction times. UDP transport is not without its risks, however.

A single-datagram exchange over UDP between two hosts can be exploited to enable a reflection attack on a third party. Response Rate Limiting [RRL] is designed to help mitigate such attacks against authoritative-only servers. One feature of RRL is to let some amount of responses "slip" through the rate limiter. These are returned with the TC (truncation) bit set, which causes legitimate clients to re-query using TCP transport.

[RFC1035] specified a maximum DNS message size over UDP transport of 512 bytes. Deployment of DNSSEC [RFC4033] and other protocols subsequently increased the observed frequency at which responses exceed this limit. EDNS0 [RFC6891] allows DNS messages larger than 512 bytes to be exchanged over UDP, with a corresponding increased incidence of fragmentation. Fragmentation is known to be problematic in general, and has also been implicated in increasing the risk of cache poisoning attacks [fragmentation-considered-poisonous].

TCP transport is less susceptible to the risks of fragmentation and reflection attacks. However, TCP transport as currently deployed has expensive setup overhead.

The overhead of the three-way TCP handshake for a single DNS transaction is substantial, increasing the transaction time for a single (request, response) pair of DNS messages from 1 x RTT to 2 x RTT. There is no such overhead for a session that is already established therefore the overhead of the initial TCP handshake is minimised when the resulting session is used to exchange multiple DNS message pairs over a single session. The extra RTT time for session

setup can be represented as the equation (1 + N)/N, where N represents the number of DNS message pairs that utilize the session and the result approaches unity as N increases.

With increased deployment of DNSSEC and new RRtypes containing application specific cryptographic material, there is an increase in the prevalence of truncated responses received over UDP with retries over TCP. The overhead for a DNS transaction over UDP truncated due to RRL is 3x RTT, higher than the overhead imposed on the same transaction initiated over TCP.

The use of TCP transport requires state to be retained on DNS servers. If a server is to perform adequately with a significant query load received over TCP, it must manage its available resources to ensure that all established TCP sessions are well-used, and idle connections are closed after an appropriate amount of time.

This document proposes a signalling mechanism between DNS clients and servers that encourages the use of long-lived TCP connections by allowing the state associated with TCP transport to be managed effectively with minimal impact on the DNS transaction time.

This mechanism will be of benefit both for stub-resolver and resolver-authoritative TCP connections. In the latter case the persistent nature of the TCP connection can provide improved defence against attacks including DDoS.

The reduced overhead of this extension adds up significantly when combined with other EDNS0 extensions, such as [CHAIN-QUERY] and [DNS-over-TLS]. For example, the combination of these EDNS0 extensions make it possible for hosts on high-latency mobile networks to natively and efficiently perform DNSSEC validation and encrypt queries.

2. Requirements Notation

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. The edns-tcp-keepalive Option

This document specifies a new EDNS0 [RFC6891] option, edns-tcp-keepalive, which can be used by DNS clients and servers to signal a willingness to keep an idle TCP session open to conduct future DNS transactions, with the idle timeout being specified by the server. This specification does not distinguish between different types of DNS client and server in the use of this option.

3.1. Option Format

The edns-tcp-keepalive option is encoded as follows:

```
                        1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-------------------------------+-------------------------------+
   !         OPTION-CODE           !         OPTION-LENGTH         !
   +-------------------------------+-------------------------------+
   |           TIMEOUT             !
   +-------------------------------+
```

where:

OPTION-CODE: the EDNS0 option code assigned to edns-tcp-keepalive, TBD1

OPTION-LENGTH: the value 0 if the TIMEOUT is omitted, the value 2 if it is present;

TIMEOUT: an idle timeout value for the TCP connection, specified in units of 100 milliseconds, encoded in network byte order.

3.2. Use by DNS Clients

3.2.1. Sending Queries

DNS clients MUST NOT include the edns-tcp-keepalive option in queries sent using UDP transport.

DNS clients MAY include the edns-tcp-keepalive option in the first query sent to a server using TCP transport to signal their desire to keep the connection open when idle.

DNS clients MAY include the edns-tcp-keepalive option in subsequent queries sent to a server using TCP transport to signal their continued desire to keep the connection open when idle.

Clients MUST specify an OPTION-LENGTH of 0 and omit the TIMEOUT value.

3.2.2. Receiving Responses

A DNS client that receives a response using UDP transport that includes the edns-tcp-keepalive option MUST ignore the option.

A DNS client that receives a response using TCP transport that includes the edns-tcp-keepalive option MAY keep the existing TCP

session open when it is idle. It SHOULD honour the timeout received in that response (overriding any previous timeout) and initiate close of the connection before the timeout expires.

A DNS client that receives a response that includes the edns-tcp-keepalive option with a TIMEOUT value of 0 SHOULD send no more queries on that connection and initiate closing the connection as soon as it has received all outstanding responses.

A DNS client that sent a query containing the edns-keepalive-option but receives a response that does not contain the edns-keepalive-option SHOULD assume the server does not support keepalive and behave following the guidance in [DRAFT-5966bis]. This holds true even if a previous edns-keepalive-option exchange occurred on the existing TCP connection.

3.3. Use by DNS Servers

3.3.1. Receiving Queries

A DNS server that receives a query using UDP transport that includes the edns-tcp-keepalive option MUST ignore the option.

A DNS server that receives a query using TCP transport that includes the edns-tcp-keepalive option MAY modify the local idle timeout associated with that TCP session if resources permit.

3.3.2. Sending Responses

A DNS server that receives a query sent using TCP transport that includes an OPT RR MAY include the edns-tcp-keepalive option in the response to signal the expected idle timeout on a connection. Servers MUST specify the TIMEOUT value that is currently associated with the TCP session. It is reasonable for this value to change according to local resource constraints or in consideration of intermediary behaviour (for example TCP middleboxes or NATs). The DNS server SHOULD send a edns-tcp-keepalive option with a timeout of 0 if it deems its local resources are too low to service more TCP keepalive sessions, or if it wants clients to close currently open connections.

3.4. TCP Session Management

Both DNS clients and servers are subject to resource constraints which will limit the extent to which TCP sessions can persist. Effective limits for the number of active sessions that can be maintained on individual clients and servers should be established, either as configuration options or by interrogation of process limits

imposed by the operating system. Servers that implement edns-tcp-keepalive should also engage in TCP connection management by recycling existing connections when appropriate, closing connections gracefully and managing request queues to enable fair use.

In the event that there is greater demand for TCP sessions than can be accommodated, servers may reduce the TIMEOUT value signalled in successive DNS messages to minimise idle time on existing sessions. This also allows, for example, clients with other candidate servers to query to establish new TCP sessions with different servers in expectation that an existing session is likely to be closed, or to fall back to UDP.

Based on TCP session resources servers may signal a TIMEOUT value of 0 to request clients to close connections as soon as possible. This is useful when server resources become very low or a denial-of-service attack is detected and further maximises the shifting of TIME_WAIT state to well-behaved clients.

However it should be noted that RCF6891 states:

> Lack of presence of an OPT record in a request MUST be taken as an indication that the requestor does not implement any part of this specification and that the responder MUST NOT include an OPT record in its response.

Since servers must be faithful to this specification even on a persistent TCP connection it means that (following the initial exchange of timeouts) a server may not be presented with the opportunity to signal a change in the idle timeout associated with a connection if the client does not send any further requests containing EDNS0 OPT RRs. This limitation makes persistent connection handling via an initial idle timeout signal more attractive than a mechanism that establishes default persistence and then uses a connection close signal (in a similar manner to HTTP 1.1 [RFC7320]).

If a client includes the edns-tcp-keepalive option in the first query, it SHOULD include an EDNS0 OPT RR periodically in any further messages it sends during the TCP session. This will increase the chance of the client being notified should the server modify the timeout associated with a session. The algorithm for choosing when to do this is out of scope of this document and is left up to the implementor and/or operator.

DNS clients and servers MAY close a TCP session at any time in order to manage local resource constraints. The algorithm by which clients

and servers rank active TCP sessions in order to determine which to close is not specified in this document.

3.5. Non-Clean Paths

Many paths between DNS clients and servers suffer from poor hygiene, limiting the free flow of DNS messages that include particular EDNS0 options, or messages that exceed a particular size. A fallback strategy similar to that described in [RFC6891] section 6.2.2 SHOULD be employed to avoid persistent interference due to non-clean paths.

3.6. Anycast Considerations

DNS servers of various types are commonly deployed using anycast [RFC4786].

Changes in network topology between clients and anycast servers may cause disruption to TCP sessions making use of edns-tcp-keepalive more often than with TCP sessions that omit it, since the TCP sessions are expected to be longer-lived. It might be possible for anycast servers to avoid disruption due to topology changes by making use of TCP multipath [RFC6824] to anchor the server side of the TCP connection to an unambiguously-unicast address.

4. Intermediary Considerations

It is RECOMMENDED that DNS intermediaries which terminate TCP connections implement edns-tcp-keepalive. An intermediary that does not implement edns-tcp-keepalive but sits between a client and server that both support edns-tcp-keepalive might close idle connections unnecessarily.

5. Security Considerations

The edns-tcp-keepalive option can potentially be abused to request large numbers of long-lived sessions in a quick burst. When a DNS Server detects abusive behaviour, it SHOULD immediately close the TCP connection and free the resources used.

Servers could choose to monitor client behaviour with respect to the edns-tcp-keepalive option to build up profiles of clients that do not honour the specified timeout.

Readers are advised to familiarise themselves with the security considerations outlined in [DRAFT-5966bis]

6. IANA Considerations

The IANA is directed to assign an EDNS0 option code for the edns-tcp-keepalive option from the DNS EDNS0 Option Codes (OPT) registry as follows:

Value	Name	Status	Reference
TBD1	edns-tcp-keepalive	Standard	[This document]

7. Acknowledgements

The authors acknowledge the contributions of Jinmei TATUYA and Mark Andrews. Thanks to Duane Wessels for detailed review and the many others who contributed to the mailing list discussion.

8. References

8.1. Normative References

[DRAFT-5966bis]
Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", draft-ietf-dnsop-5966bis (work in progress), December 2015.

[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>.

[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>.

[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>.

[RFC7320] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 7320, DOI 10.17487/RFC7320, July 2014, <http://www.rfc-editor.org/info/rfc7320>.

8.2. Informative References

[CHAIN-QUERY]
Wouters, P., "Chain Query requests in DNS", draft-ietf-dnsop-edns-chain-query (work in progress), November 2015.

[DNS-over-TLS]
Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "TLS for DNS: Initiation and Performance Considerations", draft-ietf-dprive-dns-over-tls (work in progress), December 2015.

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

[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>.

[RRL] Vixie, P. and V. Schryver, "DNS Response Rate Limiting (DNS RRL)", ISC-TN 2012-1-Draft1, April 2012, <http://ss.vix.su/~vixie/isc-tn-2012-1.txt>.

Appendix A. Editors' Notes

A.1. Abridged Change History

[Note to RFC Editor: please remove this section prior to publication.]

A.1.1. draft-ietf-dnsop-edns-tcp-keepalive-05

Reword Abstract and paragraph 9 in Introduction to remove discussion on balancing UDP/TCP and talk about encouraging use of long-lived TCP sessions.

Section 3.2.2: should -> SHOULD

Changed draft-ietf-dnsop-5966bis to be a normative reference, therefore adding a dependancy on publication of that as RFC.

Reword sentence referring to RFC6824 since it is informational.

Update IANA option to Standard.

Remove last sentence from 1st paragraph of introduction.

Reword paragraph 6 in Introduction, merge paragraph 7 and 8.

Reword Section 3, first sentence to clarify the timeout is specified by the server.

Correct missing URIs in 2 references.

Clarify statement in Section 3.2.2 as how clients should handle updating the timeout when receiving a response.

Reworded first paragraph of Introduction discussing TCP vs (UDP + retry over TCP). Changed 'fallback' to 'retry' in 2 places.

A.1.2. draft-ietf-dnsop-edns-tcp-keepalive-04

Adding wording to sections 3.2.1 and 3.4 to clarify client behaviour on subsequent queries on a TCP connection.

Changed the should to a SHOULD in section 3.2.2

Changed Nameserver to DNS server in section 5.

Updated references.

Changed reference to RFC6824 to be informative.

Corrected reference to requested EDNS0 option code to be 'TBD1'.

A.1.3. draft-ietf-dnsop-edns-tcp-keepalive-03

Clarified that a response to a query with any OPT RR may contain the ends-tcp-keepalive option.

Corrected TIMEOUT length from 4 to 2 in the diagram.

Updated references, including name change of STARTTLS -> DNS-over-TLS and adding reference for cache poisoning.

Updated wording in section on Intermediary Considerations.

Updated wording describing RRL.

Added paragraph to security section describing client behaviour profiles.

Added wording to introduction on use case for stub/resolver/ authoritative.

A.1.4. draft-ietf-dnsop-edns-tcp-keepalive-02

Changed timeout value to idle timeout and re-phrased document around this.

Changed units of timeout to 100ms to allow values less than 1 second.

Change specification to remove use of the option over UDP. This is potentially confusing, could cause issues with ALG's and adds only limited value.

Changed semantics so the client no longer sends a timeout. The client timeout is of limited value as servers should be managing connections based on their view of their resources, not on client requests as this is open to abuse. Additionally this identifies cases were the option is simply being reflected back.

Changed semantics for the meaning of a server sending a timeout of 0. The maximum timeout value of 6553.5s (~1.8h) is already large and a distinct 'connection close'-like signal is potentially more useful.

Added more detail on server side requirements when supporting keepalive in terms of resource and connection management.

Added discussion of EDNS0 per-message limitation and implications of this.

Added reference to STARTTLS draft and RFC7320.

A.1.5. draft-ietf-dnsop-edns-tcp-keepalive-01

Version bump with no changes

A.1.6. draft-ietf-dnsop-edns-tcp-keepalive-00

Clarifications, working group adoption.

A.1.7. draft-wouters-edns-tcp-keepalive-01

Also allow clients to specify KEEPALIVE timeout values, clarify motivation of document.

A.1.8. draft-wouters-edns-tcp-keepalive-00

Initial draft.

Authors' Addresses

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Joe Abley
Dyn, Inc.
470 Moore Street
London, ON N6C 2C2
Canada

Phone: +1 519 670 9327
Email: jabley@dyn.com

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

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

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

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