Network Working Group S. Cheshire
Internet-Draft M. Krochmal
Intended status: Informational Apple Inc.
Expires: September 5, 2019 March 4, 2019

DNS Long-Lived Queries
draft-sekar-dns-llq-03

Abstract

DNS Long-Lived Queries (LLQ) is a protocol for extending the DNS protocol to support change notification, thus allowing clients to learn about changes to DNS data without polling the server. From 2007 onwards, LLQ was implemented in Apple products including Mac OS X, Bonjour for Windows, and AirPort wireless base stations. In 2019, the LLQ protocol was superseded by the IETF Standards Track RFC "DNS Push Notifications", which builds on experience gained with the LLQ protocol to create a superior replacement.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In dynamic environments, DNS Service Discovery [RFC6763] benefits significantly from clients being able to learn about changes to DNS information via a mechanism that is both more timely and more efficient than simple polling. Such a mechanism enables "live browses" that learn when a new instance of a service appears, or when an existing service disappears from the network, and allows clients to monitor changes to a service. Multicast DNS [RFC6762] supports this natively. When a host on the network publishes or deletes DNS records, these records are multicast to other hosts on the network. These hosts deliver the records to interested clients (applications running on the host). Hosts also send occasional queries to the network in case gratuitous announcements are not received due to packet loss, and to detect records lost due to their publishers crashing or having become disconnected from the network.

There is currently no equivalent in traditional unicast DNS. Queries are "one-shot" -- a name server will answer a query once, returning the results available at that instant in time. Changes could be inferred via polling of the name server. This solution is not scalable, however, as a low polling rate could leave the client with stale information, and a high polling rate would have an adverse impact on the network and server.

Therefore, an extension to DNS is required that enables a client to issue long-lived queries. This extension would allow a DNS server to notify clients about changes to DNS data.

1.1. Transition to DNS Push Notifications

The LLQ protocol enjoyed over a decade of useful operation, enabling timely live updates for the service discovery user interface in Apple's Back to My Mac [RFC6281] service.

Operational experience with LLQ informed the design of its IETF Standards Track successor, DNS Push Notifications [Push].

Because of the significant enhancements in DNS Push Notifications, all existing LLQ implementations are encouraged to migrate to using DNS Push Notifications instead.

For existing LLQ servers, they are encouraged to implement and support DNS Push Notifications, so that clients can begin migrating to the newer protocol.

For existing LLQ clients, they are encouraged to query for the "_dns-push-tls._tcp.<zone>" SRV record first, and only if DNS Push fails, then fall back to query for "_dns-llq._udp.<zone>" instead.

This will cause clients to prefer the newer protocol when possible. It is recommended that clients always attempt DNS Push Notifications first for every new request, and only if that fails, then back to using LLQ. Clients SHOULD NOT record that a given server only speaks LLQ and subsequently default to LLQ for that server, since server software gets updated, and even a server that speaks only LLQ today, may be updated to support DNS Push Notifications tomorrow.

New client and server implementations are encouraged to support only DNS Push Notifications.

2. Conventions and Terminology Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Mechanisms

DNS Long-Lived Queries (DNS-LLQ) is implemented using the standard DNS message format [RFC1035] in conjunction with an ENDS0 OPT pseudo-RR [RFC2671] with a new OPT and RDATA format proposed here. Encoding the LLQ request in an OPT RR allows for implementation of LLQ with minimal modification to a name server's front-end, and will cause servers that do not implement LLQ to automatically return an appropriate error (NOTIMPL).

Note that this protocol is designed for moderate data set sizes, and moderate change rates. Data sets in response to queries that frequently exceed a single packet, or that experience a rapid change rate, may have undesirable performance implications.

3.1. New Assigned Numbers

```
EDNS0 Option Code:
     LLQ 1

LLQ-PORT 5352

Error Codes:
      NO-ERROR    0
      SERV-FULL   1
      STATIC      2
      FORMAT-ERR  3
      NO-SUCH-LLQ 4
      BAD-VERS    5
      UNKNOWN-ERR 6

LLQ Opcodes:
     LLQ-SETUP    1
     LLQ-REFRESH  2
     LLQ-EVENT    3
```

3.2. Opt-RR Format

All OPT-RRs used in LLQs are formatted as follows:

Field Name	Field Type	Description
NAME	domain name	empty (root domain)
TYPE	u_int16_t	OPT
CLASS	u_int16_t	0*
TTL	u_int32_t	0
RDLEN	u_int16_t	describes RDATA
RDATA	octet stream	(see below)

* The CLASS field indicates, as per [RFC2671], the sender's UDP payload size. However, clients and servers need not be required to determine their reassembly buffer size, path MTU, etc. to support LLQ. Thus, the sender of an LLQ Request or Response MAY set the CLASS field to 0. The recipient MUST ignore the class field if it is set to 0.

RDATA Format:

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	LLQ
OPTION-LENGTH	u_int16_t	Length of following fields, as appropriate
VERSION	u_int16_t	Version of LLQ protocol implemented
LLQ-OPCODE	u_int16_t	Identifies LLQ operation
ERROR-CODE	u_int16_t	Identifies LLQ errors
LLQ-ID	u_int64_t	Identifier for an LLQ
LEASE-LIFE	u_int32_t	Requested or granted life of LLQ, in seconds

This data format, consisting of (OPTION-CODE, OPTION-LEN, LLQ-Metadata) tuples, may be repeated an arbitrary number of times in the RDATA section, with the RDLEN field set accordingly.

4. LLQ Address and Port Identification

A client MAY send LLQ setup and control messages to an intermediate DNS cache. If the cache serves as an intermediate LLQ proxy, it will communicate directly with the client, and with the server on behalf of one or more clients.

LLQ requests sent to a DNS Cache MUST be sent to port 53.

DNS caches not implementing LLQ proxying will return a NOTIMPL or FORMERR error to the client in the DNS message header -- the intermediate cache will not forward the request, as [RFC2671] specifies that OPT-RRs are not to be forwarded. If the client receives a NOTIMPL error from a DNS cache, the client SHOULD contact the server directly.

4.1. Server Address and Port Identification

If a client's DNS cache does not implement LLQ proxying, the client requires a mechanism to determine which server to send LLQ operations to. Additionally, some firewalls block communication directly with a name server on port 53 to avoid spoof responses. However, this direct communication is necessary for LLQs. Thus, servers MAY listen for LLQs on a different port (5352). Clients also therefore need a mechanism to determine which port to send LLQ operations to.

The client determines the server responsible for a given LLQ much as a client determines which server to send a dynamic update to. The client begins by sending a standard DNS query for the name of the LLQ, with type SOA. The server MUST answer with that SOA record in the Answer section, if the record exists. The server SHOULD include an SOA record for that name's zone in the Authority section, if the LLQ name (type SOA) does not exist. For example, a query for "_ftp._tcp.apple.com." may return an SOA record named "apple.com." in the Authority section if there is no SOA record named "_ftp._tcp.apple.com." If, in this case, the server does not include the SOA record in the Authority section, the client strips the leading label from the name and tries again, repeating until an answer is received.

Upon learning the zone (SOA), the client then constructs and sends an SRV query for the name _dns-llq._udp.<zone>, e.g., _dns-llq._udp.apple.com.

A server implementing LLQ MUST answer with an SRV record [RFC2782] for this name. The SRV RDATA is as follows:

```
PRIORITY   typically 0
WEIGHT     typically 0
PORT       typically 53 or 5352
TARGET     name of server providing LLQs for the requested zone
```

The SRV target and the SOA mname SHOULD be identical. In addition, the server SHOULD include its address record(s) in the Additional section of the response.

If the server does not include its address record in the Additional section, the client SHOULD query explicitly for the address record with the name of the SRV target.

The client MUST send all LLQ requests, refreshes, and acknowledgments to the name server specified in the SRV target, at the address contained in the address record for that target. Note that the queries described in this section (including those for SOA and SRV records) MAY be sent to an intermediate DNS cache -- they need not be sent directly to the name server.

If, on issuing the SRV query, the client receives an NXDOMAIN response indicating that the SRV record does not exist, the client SHOULD conclude that the server does not support an LLQ in the requested zone. The client then SHOULD NOT send an LLQ request for the desired name, instead utilizing the behavior for LLQ-unaware servers described in Section 5 "LLQ Setup".

4.2. Client Address and Port Identification

Servers should send all messages to the source address and port of the LLQ setup message received from the client.

5. LLQ Setup

An LLQ is initiated by a client, and is completed via a four-way handshake. This handshake provides resilience to packet loss, demonstrates client reachability, and reduces denial of service attack opportunities (see Section 8 "Security Considerations").

5.1. Setup Message Retransmission

LLQ Setup Requests and Responses sent by the client SHOULD be retransmitted if no acknowledgments are received. The client SHOULD re-try up to two more times (for a total of 3 attempts) before considering the server down or unreachable. The client MUST wait at least 2 seconds before the first retransmission and 4 seconds between the first and second retransmissions. The client SHOULD listen for a response for at least 8 seconds after the 3rd attempt before considering the server down or unreachable. Upon determining a server to be down, a client MAY periodically attempt to re-initiate an LLQ setup, at a rate of not more than once per hour.

Servers MUST NOT re-transmit acknowledgments that do not generate responses from the client. Retransmission in setup is client-driven, freeing servers from maintaining timers for incomplete LLQ setups. If servers receive duplicate messages from clients (perhaps due to the loss of the server's responses mid-flight), the server MUST re-send its reply (possibly modifying the LEASE-LIFE as described in Section 5.2.4 "ACK + Answers").

Servers MUST NOT garbage collect LLQs that fail to complete the four-way handshake until the initially granted LEASE-LIFE has elapsed.

5.2. LLQ Setup Four-Way Handshake

The four phases of the handshake include:

1) Initial Request	client to server, identifies LLQ(s) requested
2) Challenge	server to client, provides error(s) for requested LLQs, and unique identifiers for the successful requests
3) Challenge Response	client to server, echoes identifier(s), demonstrating client's reachability and willingness to participate
4) ACK + Answers	server to client, confirms setup and provides initial answers

5.2.1. Setup Request

A request for an LLQ is formatted like a standard DNS query, but with an OPT RR containing LLQ metadata in its Additional section. LLQ setup requests are identified by the LLQ-SETUP opcode and a zero-valued LLQ-ID.

The request MAY contain multiple questions to set up multiple LLQs. A request consisting of multiple questions MUST contain multiple LLQ metadata sections, one per question, with metadata sections in the same order as the questions they correspond to (i.e., the first metadata section corresponds to the first question, the second metadata section corresponds to the second question, etc.) If requesting multiple LLQs, clients SHOULD request the same LEASE-LIFE for each LLQ. Requests over UDP MUST NOT contain multiple questions if doing so would cause the message to not fit in a single packet.

A client MUST NOT request multiple identical LLQs (i.e., containing the same qname/type/class) from a single source IP address and port.

The query MUST NOT be for record type ANY (255), class ANY (255), or class NONE (0).

Setup Request OPT-RR LLQ Metadata Format:

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	LLQ (1)
OPTION-LENGTH	u_int16_t	Length of following fields (18)
VERSION	u_int16_t	Version of LLQ protocol implemented by requester (1)
LLQ-OPCODE	u_int16_t	LLQ-SETUP (1)
ERROR-CODE	u_int16_t	NOERROR (0)
LLQ-ID	u_int64_t	0
LEASE-LIFE	u_int32_t	Desired life of LLQ request

These fields MUST be repeated once for each additional query in the Question section.

5.2.2. Setup Challenge

Upon receiving an LLQ Setup Request, a server implementing LLQs will send a Setup Challenge to the requester (client). An LLQ Setup Challenge is a DNS Response, with the DNS message ID matching that of the request, and with all questions contained in the request present in the Question section of the response. Additionally, the challenge contains a single OPT-RR with an LLQ metadata section for each LLQ request, indicating the success or failure of each request. Metadata sections MUST be in the same order as the questions they correspond to. Note that some LLQs in a request containing multiple questions may succeed, while others may fail.

Setup Challenge OPT-RR RDATA Format:

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	LLQ (1)
OPTION-LENGTH	u_int16_t	Length of following fields (18)
VERSION	u_int16_t	Version of LLQ protocol implemented in server (1)
LLQ-OPCODE	u_int16_t	LLQ-SETUP (1)
ERROR-CODE	u_int16_t	[As Appropriate]
LLQ-ID	u_int64_t	[As Appropriate]
LEASE-LIFE	u_int32_t	[As Appropriate]

These fields MUST be repeated once for each query in the Questions section of the Setup Request.

LLQ Metadata field descriptions:

ERROR-CODE: Possible values include:

NO-ERROR: The LLQ Setup Request was successful.

FORMAT-ERR: The LLQ was improperly formatted. Note that if the rest of the DNS message is properly formatted, the DNS header error code MUST NOT include a format error code, as this would cause confusion between a server that does not understand the LLQ format, and a client that sends malformed LLQs.

SERV-FULL: The server cannot grant the LLQ request because it is overloaded, or the request exceeds the server's rate limit (see Section 8 "Security Considerations"). Upon returning this error, the server MUST include in the LEASE-LIFE field a time interval, in seconds, after which the client may re-try the LLQ Setup.

STATIC: The data for this name and type is not expected to change frequently, and the server therefore does not support the requested LLQ. The client MUST NOT poll for this name and type, nor should it re-try the LLQ Setup, and should instead honor the normal resource record TTLs returned. To reduce server load, an administrator MAY return this error for all records with types other than PTR and TXT as a matter of course.

BAD-VERS: The protocol version specified in the client's request is not supported by the server.

UNKNOWN-ERR: The LLQ was not granted for an unknown reason

LLQ-ID: On success, a random number generated by the server that is unique for the requested name/type/class. The LLQ-ID SHOULD be an unguessable random number. A possible method of allocating LLQ-IDs with minimal bookkeeping would be to store the time, in seconds since the Epoch, in the high 32 bits of the field, and a cryptographically generated 32-bit random integer in the low 32 bits.

On error, the LLQ-ID is set to 0.

LEASE-LIFE: On success, the actual life of the LLQ, in seconds. Value may be greater than, less than, or equal to the value requested by the client, as per the server administrator's policy. The server MAY discard the LLQ after this LEASE-LIFE expires unless the LLQ has been renewed by the client (see Section 8 "Security Considerations"). The server MUST NOT generate events (see Section 6 "Event Responses") for expired LLQs.

On SERV-FULL error, LEASE-LIFE MUST be set to a time interval, in seconds, after which the client may re-try the LLQ Setup.

On other errors, the LEASE-LIFE MUST be set to 0.

5.2.3. Challenge Response

Upon issuing a Setup Request, a client listens for a Setup Challenge (5.2.2), re-transmitting the request as necessary (5.1). After receiving a successful Challenge, the client SHOULD send a Challenge Response to the server. This Challenge Response is a DNS request with questions from the request and challenge, and a single OPT-RR in the Additional section, with the OPT-RR RDATA identical to the OPT-RR RDATA contained in the Setup Request ACK (i.e., echoing, for each set of fields, the random LLQ-ID and the granted lease life). If the challenge response contains multiple questions, the first question MUST correspond to the first OPT-RR RDATA tuple, etc.

If the Setup Request fails with a STATIC error, the client MUST NOT poll the server. The client SHOULD honor the resource record TTLs contained in the response.

If the Setup Request fails with a SERV-FULL error, the client MAY re-try the LLQ Setup Request (5.2.1) after the time indicated in the LEASE-LIFE field.

If the Setup Request fails with an error other than STATIC or SERV-FULL, or the server is determined not to support LLQ (i.e., the client receives FORMERROR or NOTIMPL in the DNS message header), the client MAY poll the server periodically with standard DNS queries, inferring Add and Remove events (see Section 8 "Security Considerations") by comparing answers to these queries. The client SHOULD NOT poll more than once every 30 minutes for a given query. The client MUST NOT poll if it receives a STATIC error code in the acknowledgment.

5.2.4. ACK + Answers

Upon receiving a Challenge Response, a server MUST return an acknowledgment, completing the LLQ setup, and provide all current answers to the question(s).

To acknowledge a successful Challenge Response, i.e., a Challenge Response in which the LLQ-ID and LEASE-LIFE echoed by the client match the values issued by the server, the server MUST send a DNS response containing all available answers to the question(s) contained in the original Setup Request, along with all additional resource records appropriate for those answers in the Additional section. The Additional section also contains an OPT-RR formatted as follows:

Successful Setup Response ACK OPT-RR RDATA Format:

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	LLQ
OPTION-LENGTH	u_int16_t	Length of following fields, as appropriate
VERSION	u_int16_t	Version of LLQ protocol implemented in server
LLQ-OPCODE	u_int16_t	LLQ-SETUP (1)
ERROR-CODE	u_int16_t	NO-ERROR
LLQ-ID	u_int64_t	Originally granted ID, echoed in client's Response
LEASE-LIFE	u_int32_t	Remaining life of LLQ, in seconds

If there is a significant delay in receiving a Setup Response, or multiple Setup Responses are issued (possibly because they were lost en route to the client, causing the client to re-send the Setup Response), the server MAY decrement the LEASE-LIFE by the time elapsed since the Setup Request ACK was initially issued.

If the setup is completed over UDP and all initially available answers to the question(s), additional records, and the OPT-RR do not fit in a single packet, some or all additional records (excluding the OPT-RR) MUST be omitted. If, after omission of all additional records, the answers still do not fit in a single message, answers MUST be removed until the message fits in a single packet. These answers not delivered in the Setup Response ACK MUST be delivered without undue delay to the client via Add Events (Section 7 "LLQ Lease-Life Expiration").

5.3. Resource Record TTLs

The TTLs of resource records contained in answers to successful LLQs SHOULD be ignored by the client. The client MAY cache LLQ answers until the client receives a gratuitous announcement (see Section 6 "Event Responses") indicating that the answer to the LLQ has changed. The client MUST NOT cache answers after the LLQs LEASE-LIFE expires without being refreshed (see Section 8 "Security Considerations"). If an LLQ request fails, the client SHOULD NOT cache answers for a period longer than the client's polling interval.

Note that resource records intended specifically to be transmitted via LLQs (e.g., DNS Service Discovery resource records) may have unusually short TTLs. This is because it is assumed that the records may change frequently, and that a client's cache coherence will be maintained via the LLQ and gratuitous responses. Short TTLs prevent stale information from residing in intermediate DNS caches that are not LLQ-aware.

TTLs of resource records included in the Additional section of an LLQ response (which do not actually answer the LLQ) SHOULD be honored by the client.

6. Event Responses

When a change ("event") occurs to a name server's zone, the server MUST check if the new or deleted resource records answer any LLQs. If so, the resource records MUST be sent to the LLQ requesters in the form of a gratuitous DNS response sent to the client, with the question(s) being answered in the Question section, and answers to these questions in the Answer section. The response also includes an OPT RR in the Additional section. This OPT RR contains, in its RDATA, an entry for each LLQ being answered in the message. Entries must include the LLQ-ID. This reduces the potential for spoof events being sent to a client.

Event Response OPT-RR RDATA Format:

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	LLQ (1)
OPTION-LENGTH	u_int16_t	Length of following fields (18)
VERSION	u_int16_t	Version of LLQ protocol implemented in server (1)
LLQ-OPCODE	u_int16_t	LLQ-EVENT (3)
ERROR-CODE	u_int16_t	0
LLQ-ID	u_int64_t	[As Appropriate]
LEASE-LIFE	u_int32_t	0

Gratuitous responses for a single LLQ MAY be batched, such that multiple resource records are contained in a single message. Responses MUST NOT be batched if this would cause a message that would otherwise fit in a single packet to be truncated. While responses MAY be deferred to provide opportunities for batching, responses SHOULD NOT be delayed, for purposes of batching, for more than 30 seconds, as this would cause an unacceptable latency for the client.

After sending a gratuitous response, the server MUST listen for an acknowledgment from the client. If the client does not respond, the server MUST re-send the response. The server MUST re-send 2 times (for a total of 3 transmissions), after which the server MUST consider the client to be unreachable and delete its LLQ. The server MUST listen for 2 seconds before re-sending the response, 4 more seconds before re-sending again, and must wait an additional 8 seconds after the 3rd transmission before terminating the LLQ.

The DNS message header of the response SHOULD include an unguessable random number in the DNS message ID field, which is to be echoed in the client's acknowledgement.

6.1. Add Events

Add events occur when a new resource record appears, usually as the result of a dynamic update [RFC2136], that answers an LLQ. This record must be sent in the Answer section of the event to the client. Records that normally accompany this record in responses MAY be included in the Additional section, as per truncation restrictions described above.

6.2. Remove Events

Remove events occur when a resource record previously sent to a client, either in an initial response, or in an Add Event, becomes invalid (normally as a result of being removed via a dynamic update). The deleted resource record is sent in the Answer section of the event to the client. The resource record TTL is set to -1, indicating that the record has been removed.

6.3. Gratuitous Response Acknowledgments

Upon receiving a gratuitous response ("event"), the client MUST send an acknowledgment to the server. This acknowledgment is a DNS response echoing the OPT-RR contained in the event, with the message ID of the gratuitous response echoed in the message header. The acknowledgment MUST be sent to the source IP address and port from which the event originated.

7. LLQ Lease-Life Expiration

7.1. Refresh Request

If the client desires to maintain the LLQ beyond the duration specified in the LEASE-LIFE field of the Request Acknowledgment (5.2), the client MUST send a Refresh Request. A Refresh Request is identical to an LLQ Challenge Response (5.3), but with the LLQ-OPCODE set to LLQ-REFRESH. Unlike a Challenge Response, a Refresh Request returns no answers.

The client SHOULD refresh an LLQ when 80% of its lease life has elapsed.

As a means of reducing network traffic, when constructing refresh messages the client SHOULD include all LLQs established with a given server, even those not yet close to expiration. However, at least one LLQ MUST have elapsed at least 80% of its original LEASE-LIFE. The client MUST NOT include additional LLQs if doing so would cause the message to no longer fit in a single packet. In this case, the LLQs furthest from expiration should be omitted such that the message fits in a single packet. (These LLQs SHOULD be refreshed in a separate message when 80% of one or more of their lease lives have elapsed.) When refreshing multiple LLQs simultaneously, the message contains multiple questions, and a single OPT-RR with multiple LLQ metadata sections, one per question, with the metadata sections in the same order as the questions they correspond to.

The client SHOULD specify the original lease life granted in the LLQ response as the desired LEASE-LIFE in the refresh request. If refreshing multiple LLQs simultaneously, the client SHOULD request the same lease life for all LLQs being refreshed (with the exception of termination requests, see below).

The client SHOULD specify a lease life of 0 to terminate an LLQ prior to its scheduled expiration (for instance, when the client terminates a DNS Service Discovery browse operation, or a client is about to go to sleep or shut down.)

The client SHOULD listen for an acknowledgment from the server. The client MAY re-try up to two more times (for a total of 3 attempts) before considering the server down or unreachable. The client MUST NOT re-try a first time before 90% of the lease life has expired, and MUST NOT re-try again before 95% of the lease life has expired. If the server is determined to be down, the client MAY periodically attempt to re-establish the LLQ via an LLQ Setup Request message. The client MUST NOT attempt the LLQ Setup Request more than once per hour.

7.2. LLQ Refresh Acknowledgment

Upon receiving an LLQ Refresh message, a server MUST send an acknowledgment of the Refresh. This acknowledgment is formatted like the Setup ACK described in 5.2.3, but with the following variations:

The LLQ-OPCODE is set to LLQ-REFRESH.

NO-SUCH-LLQ MUST be returned as an error code if the client attempts to refresh an expired or non-existent LLQ (as determined by the LLQ-ID in the request).

The LLQ-ID in the acknowledgment is set to the LLQ-ID in the request.

8. Security Considerations

Without care taken in the design of protocols such as this, servers may be susceptible to denial of service (DOS) attacks, and clients may be subjected to packet storms. Mechanisms have been added to the protocol to limit potential for these attacks.

Note: This section contains no new protocol elements -- it serves only to explain the rationale behind protocol elements described above, as they relate to security.

8.1. Server DOS

LLQs require that servers be stateful, maintaining entries for each LLQ over a potentially long period of time. If unbounded in quantity, these entries may overload the server. By returning SERV-FULL in Request Acknowledgments, the sever may limit the maximum number of LLQs it maintains. Additionally, the server may return SERV-FULL to limit the number of LLQs requested for a single name and type, or by a single client. This throttling may be in the form of a hard limit, or, preferably, by token-bucket rate limiting. Such rate limiting should occur rarely in normal use and is intended to prevent DOS attacks -- thus it is not built into the protocol explicitly, but is instead implemented at the discretion of an administrator via the SERV-FULL error and the LEASE-LIFE field to indicate a retry time to the client.

8.2. Client Packet Storms

In addition to protecting the server from DOS attacks, the protocol limits the ability of a malicious host to cause the server to flood a client with packets. This is achieved via the four-way handshake upon setup, demonstrating reachability and willingness of the client to participate, and by requiring that gratuitous responses be ACK'd by the client.

Additionally, rate-limiting by LLQ client address, as described in (8.1) serves to limit the number of packets that can be delivered to an unsuspecting client.

8.3. Spoofing

A large random ID greatly reduces the risk of spoofing either the client (by sending spoof events) or the server (by sending phony requests or refreshes).

9. Problems with the LLQ protocol

In the course of using LLQ since 2007, some problems were discovered. Since no further work is being done on the LLQ protocol, this LLQ specification will not be updated to remedy these problems.

LLQ's IETF Standards Track successor, DNS Push Notifications [Push], does not suffer from these problems, so all existing LLQ implementations are encouraged to migrate to using DNS Push Notifications, and all new implementations are encouraged to implement DNS Push Notifications instead of LLQ.

Known problems with LLQ are documented here for the record.

An LLQ "Setup Challenge" message from server to client is identical to an LLQ "ACK + Answers" message from server to client when there are no current answers for the query. If there is packet loss, retransmission, and duplication in the network, then a duplicated "Setup Challenge" message arriving late at the client would look like an "ACK + Answers" message with no answers, causing the client to clear its cache of any records matching the query.

This LLQ specification states: "Servers MUST NOT garbage collect LLQs that fail to complete the four-way handshake until the initially granted LEASE-LIFE has elapsed." This is probably a mistake, since it exposes LLQ servers to an easy resource-exhaustion denial-of-service attack. DNS Push Notifications is built using DNS Stateful Operations [RFC8490], which uses TLS over TCP, and a benefit of building on TCP is that there are already established industry best practices to guard against SYN flooding and similar attacks [SYN] [RFC4953]

LLQ is built using UDP, and because the UDP protocol has no standardized way of indicating the start and end of a session, NAT gateways tend to be fairly agressive about recycling UDP mappings that they believe to be disused [RFC4787] [RFC5382] [RFC7857]. Using a high keepalive traffic rate to maintain NAT mapping state could remedy this, but would largely defeat the purpose of using LLQ in the first place, which is to provide efficient change notification without wasteful polling. Because of this, LLQ clients use NAT Port Mapping Protocol (NAT-PMP) [RFC6886] and/or Port Control Protocol (PCP) [RFC6887] to establish longer NAT mapping lifetimes. This solves the problem, but adds extra complexity, and doesn't work with NAT gateways that don't support NAT-PMP or PCP. By using TCP instead of UDP, the DNS Push Notifications protocol benefits from better longevity of sessions through NAT gateways that don't support NAT-PMP or PCP.

10. IANA Considerations

The EDNS0 OPTION CODE 1 has already been assigned for this DNS extension. No additional IANA services are required by this document.

11. Acknowledgments

The concepts described in this document were originally explored, developed and implemented with help from Chris Sharp and Roger Pantos.

In 2005 and 2006 Kiren Sekar made significant contributions to the first two drafts of this document, and he wrote much of the code for the implementation of LLQ that shipped in Mac OS X 10.5 Leopard in 2007.

12. References

12.1. Normative References

[Push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-15 (work in progress), September 2018.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, DOI 10.17487/RFC2671, August 1999, <https://www.rfc-editor.org/info/rfc2671>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/info/rfc2782>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

12.2. Informative References

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[RFC4787] Audet, F., Ed. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January 2007, <https://www.rfc-editor.org/info/rfc4787>.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <https://www.rfc-editor.org/info/rfc4953>.

[RFC5382] Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and P. Srisuresh, "NAT Behavioral Requirements for TCP", BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008, <https://www.rfc-editor.org/info/rfc5382>.

[RFC6281] Cheshire, S., Zhu, Z., Wakikawa, R., and L. Zhang, "Understanding Apple's Back to My Mac (BTMM) Service", RFC 6281, DOI 10.17487/RFC6281, June 2011, <https://www.rfc-editor.org/info/rfc6281>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

[RFC6886] Cheshire, S. and M. Krochmal, "NAT Port Mapping Protocol (NAT-PMP)", RFC 6886, DOI 10.17487/RFC6886, April 2013, <https://www.rfc-editor.org/info/rfc6886>.

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <https://www.rfc-editor.org/info/rfc6887>.

[RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar, S., and K. Naito, "Updates to Network Address Translation (NAT) Behavioral Requirements", BCP 127, RFC 7857, DOI 10.17487/RFC7857, April 2016, <https://www.rfc-editor.org/info/rfc7857>.

[RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", BCP 14, RFC 8490, DOI 10.17487/RFC8490, October 2018, <https://www.rfc-editor.org/info/rfc8490>.

[SYN] Eddy, W., "Defenses Against TCP SYN Flooding Attacks", The Internet Protocol Journal, Cisco Systems, Volume 9, Number 4, December 2006.

Authors' Addresses

Stuart Cheshire
Apple Inc.
One Apple Park Way
Cupertino, CA 95014
United States of America

Phone: +1 (408) 996-1010
Email: cheshire@apple.com

Marc Krochmal
Apple Inc.
One Apple Park Way
Cupertino, California 95014
USA

Email: marc@apple.com