ï»¿

                    Apple's DNS Long-Lived Queries Protocol

Abstract

   Apple's DNS Long-Lived Queries (LLQ) is a mechanism for extending the
   DNS protocol to support change notification, thus allowing clients to
   learn about changes to DNS data without polling the server. From
   2005 onwards, LLQ was implemented in Apple products including Mac OS
   X, Bonjour for Windows, and AirPort wireless base stations. In 2020,
   the LLQ protocol was superseded by the IETF Standards Track RFC 8765,
   "DNS Push Notifications", which builds on experience gained with the
   LLQ protocol to create a superior replacement.

   The existing LLQ protocol deployed and used from 2005 to 2020 is
   documented here to give background regarding the operational
   experience that informed the development of DNS Push Notifications,
   and to help facilitate a smooth transition from LLQ to DNS Push
   Notifications.

Table of Contents

1. Introduction

   In dynamic environments, DNS-based 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 (a) learn when a new instance of a service appears, (b)
   learn when an existing service instance disappears from the network,
   and (c) allows clients to monitor status changes to a service
   instance (e.g., printer ink levels). Multicast DNS [RFC6762]
   supports this natively. When a device on the network publishes or
   deletes Multicast DNS records, these changes are multicast to other
   hosts on the network. Those hosts deliver the change notifications
   to interested clients (applications running on that 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.

   This document defines an Apple extension to unicast DNS that enables
   a client to issue long-lived queries that allow a unicast DNS server
   to notify clients about changes to DNS data. This is a more scalable
   and practical solution than can be achieved by polling of the name
   server, because a low polling rate could leave the client with stale
   information, while a high polling rate would have an adverse impact
   on the network and server.

   The mechanism defined in this document is now being replaced by DNS
   Push Notifications [RFC8765] as explained in Section 1.1.

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.

   However, some problems were discovered, as described in Appendix A.
   This operational experience with LLQ informed the design of its IETF
   Standards Track successor, DNS Push Notifications [RFC8765]. Since
   no further work is being done on the LLQ protocol, this LLQ
   specification will not be updated to remedy these problems.

   All existing LLQ implementations are RECOMMENDED to migrate to using
   DNS Push Notifications instead.

   Existing LLQ servers are RECOMMENDED to implement and support DNS
   Push Notifications so that clients can begin migrating to the newer
   protocol.

Existing LLQ clients are RECOMMENDED to query for the
"_dns-push-tls._tcp.<zone>" SRV record first, and then only if DNS
Push Notifications fail, fall back to query for
"_dns-llq._udp.<zone>" instead. Use of the "_dns-llq._udp.<zone>"
SRV record is described in Section 4.

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 fall 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 RECOMMENDED 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 (LLQ) are implemented using the standard DNS
message format [RFC1035] in conjunction with an EDNS(0) OPT pseudo-RR
[RFC6891] with a new OPTION-CODE and OPTION-DATA format specified
here. Encoding the LLQ request in an OPT pseudo-RR allows for
implementation of LLQ with minimal modification to a name server's
front end. If a DNS query containing an LLQ option is sent to a
server that does not implement LLQ, a server that complies with the
EDNS(0) specification [RFC6891] will silently ignore the unrecognized
option and answer the request as a normal DNS query without
establishing any long-lived state and without returning an LLQ option
in its response. If a DNS query containing an LLQ option is sent to
a server that does not implement EDNS(0) at all, the server may
silently ignore the EDNS(0) OPT pseudo-RR or it may return a nonzero
RCODE. However, in practice, this issue is mostly theoretical, since
having a zone's _dns-llq._udp.<zone> SRV record target a host that
does not implement LLQ is a configuration error.

Note that this protocol is designed for data set sizes of a few dozen
resource records at most and change rates no more than once every 10
seconds on average. Data sets that frequently exceed a single IP
packet, or that experience a rapid change rate, may have undesirable
performance implications.

3.1. Assigned Numbers

This section describes constants used in this document.

EDNS(0) OPTION-CODE (recorded with IANA):

    LLQ   1

LLQ-PORT 5352 (recorded with IANA)

LLQ Opcodes (specific to this LLQ EDNS(0) Option):

    LLQ-SETUP   1
    LLQ-REFRESH   2
    LLQ-EVENT   3

LLQ Error Codes (specific to this LLQ EDNS(0) Option):

    NO-ERROR   0
    SERV-FULL   1

```
         STATIC              2
         FORMAT-ERR          3
         NO-SUCH-LLQ         4
         BAD-VERS            5
         UNKNOWN-ERR         6
```

3.2. Opt-RR Format

As required by the EDNS(0) specification [RFC6891], all OPT
pseudo-RRs used in LLQs are formatted as follows:

Field Name	Field Type	Description
NAME	domain name	MUST be 0 (root domain)
TYPE	u_int16_t	OPT (41)
CLASS	u_int16_t	0*
TTL	u_int32_t	0
RDLEN	u_int16_t	length of all RDATA
RDATA	octet stream	(see below)

Table 1: OPT-RRs Used in LLQs

* The CLASS field indicates, as per the EDNS(0) specification
[RFC6891], the sender's UDP payload size. However, clients and
servers are not 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.

The RDATA of an EDNS(0) OPT pseudo-RR consists of zero or more
options of the form { OPTION-CODE, OPTION-LENGTH, OPTION-DATA }
packed together, with the RDLEN field set accordingly to indicate the
total size. An LLQ OPTION is illustrated below. An EDNS(0) OPT
pseudo-RR may contain zero or more LLQ OPTIONS in addition to zero or
more other EDNS(0) options.

Field Name	Field Type	Description
OPTION-CODE	u_int16_t	LLQ (1)
OPTION-LENGTH	u_int16_t	Length of following fields (18)
LLQ-VERSION	u_int16_t	Version of LLQ protocol implemented
LLQ-OPCODE	u_int16_t	Identifies LLQ operation
LLQ-ERROR	u_int16_t	Identifies LLQ errors
LLQ-ID	u_int64_t	Identifier for an LLQ
LLQ-LEASE	u_int32_t	Requested or granted life of LLQ, in seconds

Table 2: LLQ OPTION

The size and meaning of the OPTION-CODE and OPTION-LENGTH fields are
as described in the EDNS(0) specification [RFC6891]. The remainder
of the fields comprise the OPTION-DATA of the EDNS(0) LLQ OPTION.
Since for LLQ the OPTION-DATA is a fixed size, in EDNS(0) LLQ OPTIONS
the OPTION-LENGTH field always has the value 18.

In keeping with Internet convention, all multi-byte numeric

quantities (u_int16_t, u_int32_t, and u_int64_t) are represented in
big endian byte order (most significant byte first).

4. LLQ Address and Port Identification

   The client requires a mechanism to determine to which server it
   should send LLQ operations.

   Additionally, some firewalls block direct communication 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 (typically 5352). Clients, therefore, also
   need a mechanism to determine to which port to send LLQ operations.

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

   This iterative zone apex discovery algorithm is described in more
   detail in the DNS Push Notifications specification [RFC8765].

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

   An authoritative server for a zone 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 |
   +----------+--+

                         Table 3: SRV RDATA

   The server SHOULD include the address record(s) for the target host
   in the Additional section of the response.

   If the server does not include the target host's address record(s) in
   the Additional section, the client SHOULD query explicitly for the
   address record(s) 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 recursive resolver --
   they need not be sent directly to the name server.

   If, on issuing the SRV query, the client receives a negative response
   indicating that the SRV record does not exist, the client SHOULD
   conclude that the zone does not support LLQ. 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".

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
retry 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 retransmit 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
resend its reply (possibly modifying the LLQ-LEASE 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 LLQ-LEASE has elapsed.

5.2. LLQ Setup Four-Way Handshake

The four phases of the handshake include:

> 1) Setup Request   client to server, identifies LLQ(s)
>   requested
>
> 2) Setup Challenge   server to client, provides unique
>   identifiers for successful requested LLQs,
>   and error(s) for unsuccessful requested
>   LLQs.
>
> 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 pseudo-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 Setup Request consisting of multiple questions MUST contain
multiple LLQ OPTIONS, one per question, with the LLQ OPTIONS in the
same order as the questions they correspond to (i.e., the first LLQ
OPTION corresponds to the first question, the second LLQ OPTION
corresponds to the second question, etc.). If requesting multiple
LLQs, clients SHOULD request the same LLQ-LEASE for each LLQ.
Requests over UDP MUST NOT contain multiple questions if doing so
would cause the message to exceed a single IP packet.

A client MUST NOT request multiple identical LLQs (i.e., containing

the same qname/type/class) from the same source IP address and port.
This requirement is to avoid unnecessary load on servers. In the
case of multiple independent client implementations that may run on
the same device without knowledge of each other, it is allowable if
they by chance send LLQ requests for the same qname/type/class.
These independent implementations on the same client will be using
different source ports. Likewise, to the server, multiple
independent clients behind the same NAT gateway will appear as if
they were multiple independent clients using different ports on the
same host, and this is also allowable.

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

Field Name	Field Type	Description
===============	============	=================================
OPTION-CODE	u_int16_t	LLQ (1)
OPTION-LENGTH	u_int16_t	Length of following fields (18)
LLQ-VERSION	u_int16_t	Version of LLQ protocol implemented by requester (1)
LLQ-OPCODE	u_int16_t	LLQ-SETUP (1)
LLQ-ERROR	u_int16_t	NO-ERROR (0)
LLQ-ID	u_int64_t	0
LLQ-LEASE	u_int32_t	Desired life of LLQ request

                 Table 4: Setup Request LLQ OPTION Format

   The Setup Request LLQ OPTION 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 Setup Request, and with all questions contained in the Setup
   Request present in the Question section of the response.
   Additionally, the challenge contains a single OPT pseudo-RR with an
   LLQ OPTION for each LLQ request, indicating the success or failure of
   each request. The LLQ OPTIONS MUST be in the same order as the
   questions they correspond to. Note that in a Setup Request
   containing multiple questions, some LLQs may succeed while others may
   fail.

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

Table 5: Setup Challenge LLQ OPTION Format

   The Setup Challenge LLQ OPTION MUST be repeated once for each query
   in the Questions section of the Setup Challenge. Further details for
   LLQ-ERROR, LLQ-ID and LLQ-LEASE are given below.

   LLQ-ERROR:

      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, since to do so
                       would cause ambiguity between the case where a
                       client sends a valid LLQ Setup Request to a
                       server that does not understand LLQ and the case
                       where a client sends a malformed LLQ Setup
                       Request to a server that does understand LLQ.

      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 LLQ-LEASE field a
                       time interval, in seconds, after which the
                       client may retry 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 honor the resource record TTLs returned and
                       MUST NOT poll sooner than indicated by those
                       TTLs, nor should it retry the LLQ Setup for this
                       name and type.

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

      UNKNOWN-ERR: The LLQ was not granted for some other reason
                       not covered by the preceding error code values.

   LLQ-ID: On success, a random number generated by the server
                       that is unique on the server for the requested
                       name/type/class. The LLQ-ID SHOULD be an
                       unpredictable 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.

   LLQ-LEASE: 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 LLQ-LEASE expires unless
                       the LLQ has been renewed by the client (see
                       Section 7, "LLQ Lease-Life Expiration"). The
                       server MUST NOT generate events (see Section 6,
                       "Event Responses") for expired LLQs.

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

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

5.2.3. Challenge Response

Upon issuing a Setup Request, a client listens for a Setup Challenge
(Section 5.2.2) retransmitting the Setup Request as necessary
(Section 5.1). After receiving a successful Setup Challenge, the
client SHOULD send a Challenge Response to the server. This
Challenge Response is a DNS request with questions as in the Setup
Request and Setup Challenge, and a single OPT pseudo-RR in the
Additional section, with the LLQ OPTIONS corresponding to the LLQ
OPTIONS contained in the Setup Challenge (i.e., echoing, for each LLQ
OPTION, the random LLQ-ID and the granted LLQ-LEASE). If the
Challenge Response contains multiple questions, the first question
MUST correspond to the first LLQ OPTION, etc.

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

If a Setup Request fails with a SERV-FULL error, the client MAY retry
the LLQ Setup Request (Section 5.2.1) after the time indicated in the
LLQ-LEASE 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 a DNS response with a nonzero RCODE, or a DNS
response containing no LLQ option), the client MAY poll the server
periodically with standard DNS queries, inferring Add and Remove
Events (see Section 6, "Event Responses") by comparing answers to
these queries. The client SHOULD NOT poll more than once every 15
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 correct 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 LLQ-LEASE 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 LLQ OPTIONS formatted as follows:

Field Name	Field Type	Description
===============	============	=================================
OPTION-CODE	u_int16_t	LLQ (1)
OPTION-LENGTH	u_int16_t	Length of following fields (18)
LLQ-VERSION	u_int16_t	Version of LLQ protocol
		implemented in server (1)
LLQ-OPCODE	u_int16_t	LLQ-SETUP (1)
LLQ-ERROR	u_int16_t	NO-ERROR (0)
LLQ-ID	u_int64_t	Originally granted ID, echoed
		in client's Response
LLQ-LEASE	u_int32_t	Remaining life of LLQ, in
		seconds

Table 6: Successful ACK + Answers LLQ OPTION Format

If there is a significant delay in receiving a Challenge Response, or
multiple Challenge Responses are issued (possibly because they were
lost en route to the client, causing the client to resend the

Challenge Response), the server MAY decrement the LLQ-LEASE by the time elapsed since the Setup Challenge was initially issued.

If the setup is completed over UDP and all initially available answers to the question(s), additional records, and the OPT pseudo-RR do not fit in a single IP packet, some or all additional records (excluding the OPT pseudo-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 IP packet. These answers not delivered in the ACK + Answers MUST be delivered without undue delay to the client via Add Events (Section 6, "Event Responses").

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 SHOULD NOT cache answers after the LLQs LLQ-LEASE expires without being refreshed (see Section 7, "LLQ Lease-Life Expiration"). 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-based 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 recursive resolvers that are not LLQ aware.

TTLs of resource records included in the Additional section of an LLQ response (which do not directly 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 changes MUST be communicated to the LLQ requesters in the form of a gratuitous DNS response sent to the client, with the relevant question(s) in the Question section, and the corresponding answers in the Answer section. The response also includes an OPT pseudo-RR in the Additional section. This OPT pseudo-RR contains, in its RDATA, an LLQ OPTION for each LLQ being answered in the message. Each LLQ OPTION must include the LLQ-ID. This reduces the potential for spoof events being sent to a client.

Field Name	Field Type	Description
============	============	=============
OPTION-CODE	u_int16_t	LLQ (1)
OPTION-LENGTH	u_int16_t	Length of following fields (18)
LLQ-VERSION	u_int16_t	Version of LLQ protocol implemented in server (1)
LLQ-OPCODE	u_int16_t	LLQ-EVENT (3)
LLQ-ERROR	u_int16_t	NO-ERROR (0)
LLQ-ID	u_int64_t	[As Appropriate]
LLQ-LEASE	u_int32_t	0

Table 7: Event Response LLQ OPTION Format

Gratuitous responses for a single LLQ MAY be batched such that

multiple changes are communicated in a single message. Responses
MUST NOT be batched if this would cause a message that would
otherwise fit in a single IP 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 resend the response. The server MUST resend two 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 resending the response, 4 more
seconds before resending 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
unpredictable random number in the DNS message ID field, which is to
be echoed in the client's acknowledgment.

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 pseudo-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 LLQ-LEASE field of the ACK + Answers
(Section 5.2.4), the client MUST send a Refresh Request. A Refresh
Request is identical to an LLQ Challenge Response (Section 5.2.3) 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 LLQ-LEASE 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 LLQ-LEASE.
The client MUST NOT include additional LLQs if doing so would cause
the message to no longer fit in a single IP packet. In this case,
the LLQs furthest from expiration should be omitted such that the
message fits in a single IP 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 pseudo-RR with
multiple LLQ OPTIONS, one per question, with the LLQ OPTIONS in the
same order as the questions they correspond to.

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

To terminate an LLQ prior to its scheduled expiration (for instance,
when the client terminates a DNS-based Service Discovery browse
operation or when a client is about to go to sleep or shut down), the
client specifies an LLQ-LEASE value of 0.

The client MUST listen for an acknowledgment from the server. The
client MAY retry up to two more times (for a total of 3 attempts)
before considering the server down or unreachable. The client MUST
NOT retry a first time before 90% of the LLQ-LEASE has expired and
MUST NOT retry again before 95% of the LLQ-LEASE 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 "ACK + Answers" message described in Section 5.2.4, but with the
following variations:

* It contains no answers.

* 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

In datagram-based protocols (i.e., protocols running over UDP, or
directly over IP, or similar), servers may be susceptible to denial-
of-service (DoS) attacks, and clients may be subjected to packet
storms. Carefully designed mechanisms are needed 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 Setup Challenges, the server 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 LLQ-LEASE field to indicate a retry time to
the client.

8.2. Client Packet Storms

In addition to protecting the server from DoS attacks, the LLQ
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
Section 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 an off-path attacker
sending spoof packets to the client (containing spoof events) or to
the server (containing phony requests or refreshes).

9. IANA Considerations

The EDNS(0) OPTION CODE 1 has already been assigned for this DNS
extension. IANA has updated the record in the "DNS EDNS0 Option
Codes (OPT)" registry from "On-hold" to "Optional" and has set the
reference to this document.

TCP and UDP ports 5352 have already been assigned for LLQ. IANA has
added a reference to this document.

10. References

10.1. Normative References

   [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>.

   [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>.

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

   [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>.

   [RFC8765]   Pusateri, T. and S. Cheshire, "DNS Push Notifications",
               RFC 8765, DOI 10.17487/RFC8765, June 2020,
               <https://www.rfc-editor.org/info/rfc8765>.

10.2. 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",
           RFC 8490, DOI 10.17487/RFC8490, March 2019,
           <https://www.rfc-editor.org/info/rfc8490>.

[SYN]   Eddy, W., "Defenses Against TCP SYN Flooding Attacks",
           Volume 9, Number 4, The Internet Protocol Journal, Cisco
           Systems, December 2006,
           <https://www.cisco.com/web/about/ac123/ac147/
           archived_issues/ipj_9-4/ipj_9-4.pdf>.

Appendix A. Problems with the LLQ Protocol

   In the course of using LLQ since 2005, 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"
   [RFC8765], does not suffer from these problems, so all existing LLQ
   implementations are RECOMMENDED to migrate to using DNS Push
   Notifications, and all new implementations are RECOMMENDED to
   implement DNS Push Notifications instead of LLQ.

   Known problems with LLQ are documented here as a cautionary tale
   about the challenges of building an application protocol directly
   using datagrams (like IP or UDP) without the benefit of a mature and
   thoroughly reviewed intervening transport layer (such as TCP or
   QUIC).

   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.

Section 5.1 of this LLQ specification states, "Servers MUST NOT
garbage collect LLQs that fail to complete the four-way handshake
until the initially granted LLQ-LEASE has elapsed." This is probably
a mistake since it exposes LLQ servers to an easy resource-exhaustion
denial-of-service attack. LLQ's replacement, DNS Push Notifications
[RFC8765], is built using DNS Stateful Operations [RFC8490], which
uses TLS over TCP; 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].

The attempts here to pack multiple questions into a single UDP/IP
packet for efficiency are awkward and lead to error-prone programming
to deal with cases where some requests in a packet succeed and other
requests in the same packet fail. Fully specifying the correct
handling in all possible cases would be a lot of work to document, a
lot of work to implement, and even more work to thoroughly test. DNS
Push Notifications [RFC8765] avoids this problem by using an
underlying stream protocol (TLS/TCP) to deal with packing small
multiple messages into larger IP packets for efficiency.

In some cases, initial LLQ answers are delivered in the "ACK +
Answers" message, and in other cases, such as when all the initial
answers will not fit in a single IP packet, some of the initial
answers are delivered in a subsequent "Add Event" message. Having
two different ways to accomplish the same thing increases the
possibility for programming errors. DNS Push Notifications [RFC8765]
corrects this error by having only one single consistent way to
deliver results.

LLQ is built using UDP, and because UDP has no standardized way of
indicating the start and end of a session, firewalls and NAT gateways
tend to be fairly aggressive about recycling UDP mappings that they
believe to be disused [RFC4787] [RFC5382] [RFC7857]. Using a high
keepalive traffic rate to maintain firewall or 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, existing LLQ clients use
the NAT Port Mapping Protocol (NAT-PMP) [RFC6886] and/or Port Control
Protocol (PCP) [RFC6887] to establish longer port mapping lifetimes.
This solves the problem but adds extra complexity and doesn't work
with firewalls and 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 firewalls and NAT
gateways that don't support NAT-PMP or PCP.

Acknowledgments

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, CA 95014
United States of America

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