                Discovery Proxy for Multicast DNS-Based Service Discovery
                          draft-ietf-dnssd-hybrid-10

Abstract

   This document specifies a network proxy that uses Multicast DNS to
   automatically populate the wide-area unicast Domain Name System
   namespace with records describing devices and services found on the
   local link.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Multicast DNS [RFC6762] and its companion technology DNS-based
   Service Discovery [RFC6763] were created to provide IP networking
   with the ease-of-use and autoconfiguration for which AppleTalk was
   well known [RFC6760] [ZC] [Roadmap].

   For a small home network consisting of just a single link (or a few
   physical links bridged together to appear as a single logical link
   from the point of view of IP) Multicast DNS [RFC6762] is sufficient
   for client devices to look up the ".local" host names of peers on the
   same home network, and to use Multicast DNS-Based Service Discovery
   (DNS-SD) [RFC6763] to discover services offered on that home network.

   For a larger network consisting of multiple links that are
   interconnected using IP-layer routing instead of link-layer bridging,
   link-local Multicast DNS alone is insufficient because link-local
   Multicast DNS packets, by design, are not propagated onto other
   links.

   Using link-local multicast packets for Multicast DNS was a conscious
   design choice [RFC6762]. Even when limited to a single link,
   multicast traffic is still generally considered to be more expensive
   than unicast, because multicast traffic impacts many devices, instead
   of just a single recipient. In addition, with some technologies like
   Wi-Fi [IEEE-11], multicast traffic is inherently less efficient and
   less reliable than unicast, because Wi-Fi multicast traffic is sent
   at lower data rates, and is not acknowledged [Mcast]. Increasing the
   amount of expensive multicast traffic by flooding it across multiple
   links would make the traffic load even worse.

   Partitioning the network into many small links curtails the spread of
   expensive multicast traffic, but limits the discoverability of
   services. At the opposite end of the spectrum, using a very large
   local link with thousands of hosts enables better service discovery,
   but at the cost of larger amounts of multicast traffic.

   Performing DNS-Based Service Discovery using purely Unicast DNS is
   more efficient and doesn't require large multicast domains, but does
   require that the relevant data be available in the Unicast DNS
   namespace. The Unicast DNS namespace in question could fall within a
   traditionally assigned globally unique domain name, or could use a
   private local unicast domain name such as ".home.arpa" [RFC8375].

   In the DNS-SD specification [RFC6763], Section 10 ("Populating the
   DNS with Information") discusses various possible ways that a
   service's PTR, SRV, TXT and address records can make their way into
   the Unicast DNS namespace, including manual zone file configuration

[RFC1034] [RFC1035], DNS Update [RFC2136] [RFC3007] and proxies of various kinds.

Making the relevant data available in the Unicast DNS namespace by manual DNS configuration is one option. This option has been used for many years at IETF meetings to advertise the IETF Terminal Room printer. Details of this example are given in Appendix A of the Roadmap document [Roadmap]. However, this manual DNS configuration is labor intensive, error prone, and requires a reasonable degree of DNS expertise.

Populating the Unicast DNS namespace via DNS Update by the devices offering the services themselves is another option [RegProt] [DNS-UL]. However, this requires configuration of DNS Update keys on those devices, which has proven onerous and impractical for simple devices like printers and network cameras.

Hence, to facilitate efficient and reliable DNS-Based Service Discovery, a compromise is needed that combines the ease-of-use of Multicast DNS with the efficiency and scalability of Unicast DNS.

This document specifies a type of proxy called a "Discovery Proxy" that uses Multicast DNS [RFC6762] to discover Multicast DNS records on its local link, and makes corresponding DNS records visible in the Unicast DNS namespace.

In principle, similar mechanisms could be defined using other local service discovery protocols, to discover local information and then make corresponding DNS records visible in the Unicast DNS namespace. Such mechanisms for other local service discovery protocols could be addressed in future documents.

The design of the Discovery Proxy is guided by the previously published requirements document [RFC7558].

In simple terms, a descriptive DNS name is chosen for each link in an organization. Using a DNS NS record, responsibility for that DNS name is delegated to a Discovery Proxy physically attached to that link. Now, when a remote client issues a unicast query for a name falling within the delegated subdomain, the normal DNS delegation mechanism results in the unicast query arriving at the Discovery Proxy, since it has been declared authoritative for those names. Now, instead of consulting a textual zone file on disk to discover the answer to the query, as a traditional DNS server would, a Discovery Proxy consults its local link, using Multicast DNS, to find the answer to the question.

For fault tolerance reasons there may be more than one Discovery
Proxy serving a given link.

Note that the Discovery Proxy uses a "pull" model. The local link is
not queried using Multicast DNS until some remote client has
requested that data. In the idle state, in the absence of client
requests, the Discovery Proxy sends no packets and imposes no burden
on the network. It operates purely "on demand".

An alternative proposal that has been discussed is a proxy that
performs DNS updates to a remote DNS server on behalf of the
Multicast DNS devices on the local network. The difficulty with this
is is that Multicast DNS devices do not routinely announce their
records on the network. Generally they remain silent until queried.
This means that the complete set of Multicast DNS records in use on a
link can only be discovered by active querying, not by passive
listening. Because of this, a proxy can only know what names exist
on a link by issuing queries for them, and since it would be
impractical to issue queries for every possible name just to find out
which names exist and which do not, there is no reasonable way for a
proxy to programmatically learn all the answers it would need to push
up to the remote DNS server using DNS Update. Even if such a
mechanism were possible, it would risk generating high load on the
network continuously, even when there are no clients with any
interest in that data.

Hence, having a model where the query comes to the Discovery Proxy is
much more efficient than a model where the Discovery Proxy pushes the
answers out to some other remote DNS server.

A client seeking to discover services and other information achieves
this by sending traditional DNS queries to the Discovery Proxy, or by
sending DNS Push Notification subscription requests [Push].

How a client discovers what domain name(s) to use for its service
discovery queries, (and consequently what Discovery Proxy or Proxies
to use) is described in Section 5.2.

The diagram below illustrates a network topology using a Discovery
Proxy to provide discovery service to a remote client.

```
 +--------+   Unicast      +-----------+  +---------+ +---------+
 | Remote | Communcation   | Discovery |  | Network | | Network |
 | Client |---- . . . -----|   Proxy   |  | Printer | | Camera  |
 +--------+                +-----------+  +---------+ +---------+
                                  |            |           |
                            ------------------------------------------
                            Multicast-capable LAN segment (e.g., Ethernet)
```
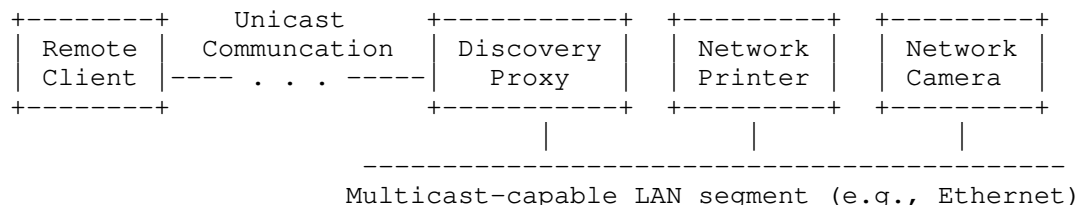
2. Operational Analogy

   A Discovery Proxy does not operate as a multicast relay, or multicast
   forwarder. There is no danger of multicast forwarding loops that
   result in traffic storms, because no multicast packets are forwarded.
   A Discovery Proxy operates as a *proxy* for a remote client,
   performing queries on its behalf and reporting the results back.

   A reasonable analogy is making a telephone call to a colleague at
   your workplace and saying, "I'm out of the office right now. Would
   you mind bringing up a printer browser window and telling me the
   names of the printers you see?" That entails no risk of a forwarding
   loop causing a traffic storm, because no multicast packets are sent
   over the telephone call.

   A similar analogy, instead of enlisting another human being to
   initiate the service discovery operation on your behalf, is to log
   into your own desktop work computer using screen sharing, and then
   run the printer browser yourself to see the list of printers. Or log
   in using ssh and type "dns-sd -B _ipp._tcp" and observe the list of
   discovered printer names. In neither case is there any risk of a
   forwarding loop causing a traffic storm, because no multicast packets
   are being sent over the screen sharing or ssh connection.

   The Discovery Proxy provides another way of performing remote
   queries, except using a different protocol instead of screen sharing
   or ssh.

   When the Discovery Proxy software performs Multicast DNS operations,
   the exact same Multicast DNS caching mechanisms are applied as when
   any other client software on that Discovery Proxy device performs
   Multicast DNS operations, whether that be running a printer browser
   client locally, or a remote user running the printer browser client
   via a screen sharing connection, or a remote user logged in via ssh
   running a command-line tool like "dns-sd", or a remote user sending
   DNS requests that cause a Discovery Proxy to perform discovery
   operations on its behalf.

3. 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 "Key words for use in RFCs to Indicate Requirement Levels",
   when, and only when, they appear in all capitals, as shown here
   [RFC2119] [RFC8174].

   The Discovery Proxy builds on Multicast DNS, which works between
   hosts on the same link. For the purposes of this document a set of
   hosts is considered to be "on the same link" if:

   o when any host from that set sends a packet to any other host in
      that set, using unicast, multicast, or broadcast, the entire link-
      layer packet payload arrives unmodified, and

   o a broadcast sent over that link, by any host from that set of
      hosts, can be received by every other host in that set.

   The link-layer *header* may be modified, such as in Token Ring Source
   Routing [IEEE-5], but not the link-layer *payload*. In particular,
   if any device forwarding a packet modifies any part of the IP header
   or IP payload then the packet is no longer considered to be on the
   same link. This means that the packet may pass through devices such
   as repeaters, bridges, hubs or switches and still be considered to be
   on the same link for the purpose of this document, but not through a
   device such as an IP router that decrements the IP TTL or otherwise
   modifies the IP header.

4. Compatibility Considerations

   No changes to existing devices are required to work with a Discovery
   Proxy.

   Existing devices that advertise services using Multicast DNS work
   with Discovery Proxy.

   Existing clients that support DNS-Based Service Discovery over
   Unicast DNS work with Discovery Proxy. Service Discovery over
   Unicast DNS was introduced in Mac OS X 10.4 in April 2005, as is
   included in Apple products introduced since then, including iPhone
   and iPad, as well as products from other vendors, such as Microsoft
   Windows 10.

   An overview of the larger collection of related Service Discovery
   technologies, and how Discovery Proxy relates to those, is given in
   the Service Discovery Road Map document [Roadmap].

5. Discovery Proxy Operation

   In a typical configuration, a Discovery Proxy is configured to be
   authoritative [RFC1034] [RFC1035] for four or more DNS subdomains,
   and authority for these subdomains is delegated to it via NS records:

   A DNS subdomain for service discovery records.
      This subdomain name may contain rich text, including spaces and
      other punctuation. This is because this subdomain name is used
      only in graphical user interfaces, where rich text is appropriate.

   A DNS subdomain for host name records.
      This subdomain name SHOULD be limited to letters, digits and
      hyphens, to facilitate convenient use of host names in command-
      line interfaces.

   One or more DNS subdomains for IPv4 Reverse Mapping records.
      These subdomains will have names that ends in "in-addr.arpa."

   One or more DNS subdomains for IPv6 Reverse Mapping records.
      These subdomains will have names that ends in "ip6.arpa."

   In an enterprise network the naming and delegation of these
   subdomains is typically performed by conscious action of the network
   administrator. In a home network naming and delegation would
   typically be performed using some automatic configuration mechanism
   such as HNCP [RFC7788].

   These three varieties of delegated subdomains (service discovery,
   host names, and reverse mapping) are described below in Section 5.1,
   Section 5.3 and Section 5.4.

   How a client discovers where to issue its service discovery queries
   is described below in Section 5.2.

5.1. Delegated Subdomain for Service Discovery Records

   In its simplest form, each link in an organization is assigned a
   unique Unicast DNS domain name, such as "Building 1.example.com" or
   "2nd Floor.Building 3.example.com". Grouping multiple links under a
   single Unicast DNS domain name is to be specified in a future
   companion document, but for the purposes of this document, assume
   that each link has its own unique Unicast DNS domain name. In a
   graphical user interface these names are not displayed as strings
   with dots as shown above, but something more akin to a typical file
   browser graphical user interface (which is harder to illustrate in a
   text-only document) showing folders, subfolders and files in a file
   system.

```
+---------------+-------------+------------+------------------+
| *example.com* | Building 1  | 1st Floor  | Alice's printer  |
|               | Building 2  | *2nd Floor*| Bob's printer    |
|               | *Building 3*| 3rd Floor  | Charlie's printer|
|               | Building 4  | 4th Floor  |                  |
|               | Building 5  |            |                  |
|               | Building 6  |            |                  |
+-------------+-------------+-----------+------------------+
```

                      Figure 1: Illustrative GUI

   Each named link in an organization has one or more Discovery Proxies
   which serve it. This Discovery Proxy function for each link could be
   performed by a device like a router or switch that is physically
   attached to that link. In the parent domain, NS records are used to
   delegate ownership of each defined link name
   (e.g., "Building 1.example.com") to the one or more Discovery Proxies
   that serve the named link. In other words, the Discovery Proxies are
   the authoritative name servers for that subdomain. As in the rest of
   DNS-Based Service Discovery, all names are represented as-is using
   plain UTF-8 encoding, and, as described in Section 5.5.4, no text
   encoding translations are performed.

   With appropriate VLAN configuration [IEEE-1Q] a single Discovery
   Proxy device could have a logical presence on many links, and serve
   as the Discovery Proxy for all those links. In such a configuration
   the Discovery Proxy device would have a single physical Ethernet
   [IEEE-3] port, configured as a VLAN trunk port, which would appear to
   software on that device as multiple virtual Ethernet interfaces, one
   connected to each of the VLAN links.

   As an alternative to using VLAN technology, using a Multicast DNS
   Discovery Relay [Relay] is another way that a Discovery Proxy can
   have a 'virtual' presence on a remote link.

When a DNS-SD client issues a Unicast DNS query to discover services in a particular Unicast DNS subdomain (e.g., "_printer._tcp.Building 1.example.com. PTR ?") the normal DNS delegation mechanism results in that query being forwarded until it reaches the delegated authoritative name server for that subdomain, namely the Discovery Proxy on the link in question. Like a conventional Unicast DNS server, a Discovery Proxy implements the usual Unicast DNS protocol [RFC1034] [RFC1035] over UDP and TCP. However, unlike a conventional Unicast DNS server that generates answers from the data in its manually-configured zone file, a Discovery Proxy generates answers using Multicast DNS. A Discovery Proxy does this by consulting its Multicast DNS cache and/or issuing Multicast DNS queries, as appropriate, according to the usual protocol rules of Multicast DNS [RFC6762], for the corresponding Multicast DNS name, type and class, with the delegated zone part of the name replaced with ".local" (e.g., in this case, "_printer._tcp.local. PTR ?"). Then, from the received Multicast DNS data, the Discovery Proxy synthesizes the appropriate Unicast DNS response, with the ".local" top-level label replaced with with the name of the delegated zone. How long the Discovery Proxy should wait to accumulate Multicast DNS responses before sending its unicast reply is described below in Section 5.6.

The existing Multicast DNS caching mechanism is used to minimize unnecessary Multicast DNS queries on the wire. The Discovery Proxy is acting as a client of the underlying Multicast DNS subsystem, and benefits from the same caching and efficiency measures as any other client using that subsystem.

Note that the contents of the delegated zone, generated as it is by performing ".local" Multicast DNS queries, mirrors the records available on the local link via Multicast DNS very closely, but not precisely. There is not a full bidirectional equivalence between the two. Certain records that are available via Multicast DNS may not have equivalents in the delegated zone, possibly because they are invalid or not relevant in the delegated zone, or because they are being supressed because they are unusable outside the local link (see Section 5.5.2). Conversely, certain records that appear in the delegated zone may not have corresponding records available on the local link via Multicast DNS. In particular there are certain administrative SRV records (see Section 6) that logically fall within the delegated zone, but semantically represent metadata *about* the zone rather than records *within* the zone, and consequently these administrative records in the delegated zone do not have any corresponding counterparts in the Multicast DNS namespace of the local link.

5.2. Domain Enumeration

   A DNS-SD client performs Domain Enumeration [RFC6763] via certain PTR
   queries, using both unicast and multicast. If it receives a Domain
   Name configuration via DHCP option 15 [RFC2132], then it issues
   unicast queries using this domain. It issues unicast queries using
   names derived from its IPv4 subnet address(es) and IPv6 prefix(es).
   These are described below in Section 5.2.1. It also issues multicast
   Domain Enumeration queries in the "local" domain [RFC6762]. These
   are described below in Section 5.2.2. The results of all the Domain
   Enumeration queries are combined for Service Discovery purposes.

5.2.1. Domain Enumeration via Unicast Queries

   The administrator creates Domain Enumeration PTR records [RFC6763] to
   inform clients of available service discovery domains. Two varieties
   of such Domain Enumeration PTR records exist; those with names
   derived from the domain name communicated to the clients via DHCP,
   and those with names derived from IPv4 subnet address(es) and IPv6
   prefix(es) in use by the clients. Below is an example showing the
   name-based variety:

       b._dns-sd._udp.example.com.   PTR   Building 1.example.com.
                                        PTR   Building 2.example.com.
                                        PTR   Building 3.example.com.
                                        PTR   Building 4.example.com.

       db._dns-sd._udp.example.com.   PTR   Building 1.example.com.

       lb._dns-sd._udp.example.com.   PTR   Building 1.example.com.

   The meaning of these records is defined in the DNS Service Discovery
   specification [RFC6763] but for convenience is repeated here. The
   "b" ("browse") records tell the client device the list of browsing
   domains to display for the user to select from. The "db" ("default
   browse") record tells the client device which domain in that list
   should be selected by default. The "db" domain MUST be one of the
   domains in the "b" list; if not then no domain is selected by
   default. The "lb" ("legacy browse") record tells the client device
   which domain to automatically browse on behalf of applications that
   don't implement UI for multi-domain browsing (which is most of them,
   at the time of writing). The "lb" domain is often the same as the
   "db" domain, or sometimes the "db" domain plus one or more others
   that should be included in the list of automatic browsing domains for
   legacy clients.

   Note that in the example above, for clarity, space characters in
   names are shown as actual spaces. If this data is manually entered

into a textual zone file for authoritative server software such as
BIND, care must be taken because the space character is used as a
field separator, and other characters like dot ('.'), semicolon
(';'), dollar ('$'), backslash ('\'), etc., also have special
meaning. These characters have to be escaped when entered into a
textual zone file, following the rules in Section 5.1 of the DNS
specification [RFC1035]. For example, a literal space in a name is
represented in the textual zone file using '\032', so "Building
1.example.com." is entered as "Building\0321.example.com."

DNS responses are limited to a maximum size of 65535 bytes. This
limits the maximum number of domains that can be returned for a
Domain Enumeration query, as follows:

A DNS response header is 12 bytes. That's typically followed by a
single qname (up to 256 bytes) plus qtype (2 bytes) and qclass
(2 bytes), leaving 65275 for the Answer Section.

An Answer Section Resource Record consists of:

o Owner name, encoded as a two-byte compression pointer
o Two-byte rrtype (type PTR)
o Two-byte rrclass (class IN)
o Four-byte ttl
o Two-byte rdlength
o rdata (domain name, up to 256 bytes)

This means that each Resource Record in the Answer Section can take
up to 268 bytes total, which means that the Answer Section can
contain, in the worst case, no more than 243 domains.

In a more typical scenario, where the domain names are not all
maximum-sized names, and there is some similarity between names so
that reasonable name compression is possible, each Answer
Section Resource Record may average 140 bytes, which means that the
Answer Section can contain up to 466 domains.

It is anticipated that this should be sufficient for even a large
corporate network or university campus.

5.2.2. Domain Enumeration via Multicast Queries

   In the case where Discovery Proxy functionality is widely deployed
   within an enterprise (either by having a Discovery Proxy on each
   link, or by having a Discovery Proxy with a remote 'virtual' presence
   on each link using VLANs or Multicast DNS Discovery Relays [Relay])
   this offers an additional way to provide Domain Enumeration data for
   clients.

   A Discovery Proxy can be configured to generate Multicast DNS
   responses for the following Multicast DNS Domain Enumeration queries
   issued by clients:

       b._dns-sd._udp.local.   PTR   ?
       db._dns-sd._udp.local.   PTR   ?
       lb._dns-sd._udp.local.   PTR   ?

   This provides the ability for Discovery Proxies to indicate
   recommended browsing domains to DNS-SD clients on a per-link
   granularity. In some enterprises it may be preferable to provide
   this per-link configuration data in the form of Discovery Proxy
   configuration, rather than populating the Unicast DNS servers with
   the same data (in the "ip6.arpa" or "in-addr.arpa" domains).

   Regardless of how the network operator chooses to provide this
   configuration data, clients will perform Domain Enumeration via both
   unicast and multicast queries, and then combine the results of these
   queries.

5.3. Delegated Subdomain for LDH Host Names

   DNS-SD service instance names and domains are allowed to contain
   arbitrary Net-Unicode text [RFC5198], encoded as precomposed UTF-8
   [RFC3629].

   Users typically interact with service discovery software by viewing a
   list of discovered service instance names on a display, and selecting
   one of them by pointing, touching, or clicking. Similarly, in
   software that provides a multi-domain DNS-SD user interface, users
   view a list of offered domains on the display and select one of them
   by pointing, touching, or clicking. To use a service, users don't
   have to remember domain or instance names, or type them; users just
   have to be able to recognize what they see on the display and touch
   or click on the thing they want.

   In contrast, host names are often remembered and typed. Also, host
   names have historically been used in command-line interfaces where
   spaces can be inconvenient. For this reason, host names have
   traditionally been restricted to letters, digits and hyphens (LDH),
   with no spaces or other punctuation.

   While we do want to allow rich text for DNS-SD service instance names
   and domains, it is advisable, for maximum compatibility with existing
   usage, to restrict host names to the traditional letter-digit-hyphen
   rules. This means that while a service name
   "My Printer._ipp._tcp.Building 1.example.com" is acceptable and
   desirable (it is displayed in a graphical user interface as an
   instance called "My Printer" in the domain "Building 1" at
   "example.com"), a host name "My-Printer.Building 1.example.com" is
   less desirable (because of the space in "Building 1").

   To accomodate this difference in allowable characters, a Discovery
   Proxy SHOULD support having two separate subdomains delegated to it
   for each link it serves, one whose name is allowed to contain
   arbitrary Net-Unicode text [RFC5198], and a second more constrained
   subdomain whose name is restricted to contain only letters, digits,
   and hyphens, to be used for host name records (names of 'A' and
   'AAAA' address records). The restricted names may be any valid name
   consisting of only letters, digits, and hyphens, including Punycode-
   encoded names [RFC3492].

For example, a Discovery Proxy could have the two subdomains
"Building 1.example.com" and "bldg1.example.com" delegated to it.
The Discovery Proxy would then translate these two Multicast DNS
records:

```
My Printer._ipp._tcp.local. SRV 0 0 631 prnt.local.
prnt.local.                 A   203.0.113.2
```

into Unicast DNS records as follows:

```
My Printer._ipp._tcp.Building 1.example.com.
                            SRV 0 0 631 prnt.bldg1.example.com.
prnt.bldg1.example.com.     A   203.0.113.2
```

Note that the SRV record name is translated using the rich-text
domain name ("Building 1.example.com") and the address record name is
translated using the LDH domain ("bldg1.example.com").

A Discovery Proxy MAY support only a single rich text Net-Unicode
domain, and use that domain for all records, including 'A' and 'AAAA'
address records, but implementers choosing this option should be
aware that this choice may produce host names that are awkward to use
in command-line environments. Whether this is an issue depends on
whether users in the target environment are expected to be using
command-line interfaces.

A Discovery Proxy MUST NOT be restricted to support only a letter-
digit-hyphen subdomain, because that results in an unnecessarily poor
user experience.

As described above in Section 5.2.1, for clarity, space characters in
names are shown as actual spaces. If this data were to be manually
entered into a textual zone file (which it isn't) then spaces would
need to be represented using '\032', so
"My Printer._ipp._tcp.Building 1.example.com." would become
"My\032Printer._ipp._tcp.Building\0321.example.com."
Note that the '\032' representation does not appear in the network
packets sent over the air. In the wire format of DNS messages,
spaces are sent as spaces, not as '\032', and likewise, in a
graphical user interface at the client device, spaces are shown as
spaces, not as '\032'.

5.4. Delegated Subdomain for Reverse Mapping

   A Discovery Proxy can facilitate easier management of reverse mapping
   domains, particularly for IPv6 addresses where manual management may
   be more onerous than it is for IPv4 addresses.

   To achieve this, in the parent domain, NS records are used to
   delegate ownership of the appropriate reverse mapping domain to the
   Discovery Proxy. In other words, the Discovery Proxy becomes the
   authoritative name server for the reverse mapping domain. For fault
   tolerance reasons there may be more than one Discovery Proxy serving
   a given link.

   If a given link is using the IPv4 subnet 203.0.113/24,
   then the domain "113.0.203.in-addr.arpa"
   is delegated to the Discovery Proxy for that link.

   For example, if a given link is using the
   IPv6 prefix 2001:0DB8:1234:5678/64,
   then the domain "8.7.6.5.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa"
   is delegated to the Discovery Proxy for that link.

   When a reverse mapping query arrives at the Discovery Proxy, it
   issues the identical query on its local link as a Multicast DNS
   query. The mechanism to force an apparently unicast name to be
   resolved using link-local Multicast DNS varies depending on the API
   set being used. For example, in the "dns_sd.h" APIs
   (available on macOS, iOS, Bonjour for Windows, Linux and Android),
   using kDNSServiceFlagsForceMulticast indicates that the
   DNSServiceQueryRecord() call should perform the query using Multicast
   DNS. Other APIs sets have different ways of forcing multicast
   queries. When the host owning that IPv4 or IPv6 address responds
   with a name of the form "something.local", the Discovery Proxy
   rewrites that to use its configured LDH host name domain instead of
   "local", and returns the response to the caller.

For example, a Discovery Proxy with the two subdomains
"113.0.203.in-addr.arpa" and "bldg1.example.com" delegated to it
would translate this Multicast DNS record:

    2.113.0.203.in-addr.arpa. PTR prnt.local.

into this Unicast DNS response:

    2.113.0.203.in-addr.arpa. PTR prnt.bldg1.example.com.

Subsequent queries for the prnt.bldg1.example.com address record,
falling as it does within the bldg1.example.com domain, which is
delegated to the Discovery Proxy, will arrive at the Discovery Proxy,
where they are answered by issuing Multicast DNS queries and using
the received Multicast DNS answers to synthesize Unicast DNS
responses, as described above.

Note that this design assumes that all addresses on a given IPv4
subnet or IPv6 prefix are mapped to hostnames using the Discovery
Proxy mechanism. It would be possible to implement a Discovery Proxy
that can be configured so that some address-to-name mappings are
performed using Multicast DNS on the local link, while other address-
to-name mappings within the same IPv4 subnet or IPv6 prefix are
configured manually.

5.5. Data Translation

   Generating the appropriate Multicast DNS queries involves,
   at the very least, translating from the configured DNS domain
   (e.g., "Building 1.example.com") on the Unicast DNS side to "local"
   on the Multicast DNS side.

   Generating the appropriate Unicast DNS responses involves translating
   back from "local" to the appropriate configured DNS Unicast domain.

   Other beneficial translation and filtering operations are described
   below.

5.5.1. DNS TTL limiting

   For efficiency, Multicast DNS typically uses moderately high DNS TTL
   values. For example, the typical TTL on DNS-SD PTR records is 75
   minutes. What makes these moderately high TTLs acceptable is the
   cache coherency mechanisms built in to the Multicast DNS protocol
   which protect against stale data persisting for too long. When a
   service shuts down gracefully, it sends goodbye packets to remove its
   PTR records immediately from neighboring caches. If a service shuts
   down abruptly without sending goodbye packets, the Passive
   Observation Of Failures (POOF) mechanism described in Section 10.5 of
   the Multicast DNS specification [RFC6762] comes into play to purge
   the cache of stale data.

   A traditional Unicast DNS client on a distant remote link does not
   get to participate in these Multicast DNS cache coherency mechanisms
   on the local link. For traditional Unicast DNS queries (those
   received without using Long-Lived Query [LLQ] or DNS Push
   Notification subscriptions [Push]) the DNS TTLs reported in the
   resulting Unicast DNS response MUST be capped to be no more than ten
   seconds.

   Similarly, for negative responses, the negative caching TTL indicated
   in the SOA record [RFC2308] should also be ten seconds (Section 6.1).

   This value of ten seconds is chosen based on user-experience
   considerations.

   For negative caching, suppose a user is attempting to access a remote
   device (e.g., a printer), and they are unsuccessful because that
   device is powered off. Suppose they then place a telephone call and
   ask for the device to be powered on. We want the device to become
   available to the user within a reasonable time period. It is
   reasonable to expect it to take on the order of ten seconds for a
   simple device with a simple embedded operating system to power on.

Once the device is powered on and has announced its presence on the network via Multicast DNS, we would like it to take no more than a further ten seconds for stale negative cache entries to expire from Unicast DNS caches, making the device available to the user desiring to access it.

Similar reasoning applies to capping positive TTLs at ten seconds. In the event of a device moving location, getting a new DHCP address, or other renumbering events, we would like the updated information to be available to remote clients in a relatively timely fashion.

However, network administrators should be aware that many recursive (caching) DNS servers by default are configured to impose a minimum TTL of 30 seconds. If stale data appears to be persisting in the network to the extent that it adversely impacts user experience, network administrators are advised to check the configuration of their recursive DNS servers.

For received Unicast DNS queries that use LLQ [LLQ] or DNS Push Notifications [Push], the Multicast DNS record's TTL SHOULD be returned unmodified, because the Push Notification channel exists to inform the remote client as records come and go. For further details about Long-Lived Queries, and its newer replacement, DNS Push Notifications, see Section 5.6.

5.5.2. Suppressing Unusable Records

A Discovery Proxy SHOULD offer a configurable option, enabled by default, to suppress Unicast DNS answers for records that are not useful outside the local link. When the option to suppress unusable records is enabled:

o DNS A and AAAA records for IPv4 link-local addresses [RFC3927] and IPv6 link-local addresses [RFC4862] SHOULD be suppressed.

o Similarly, for sites that have multiple private address realms [RFC1918], in cases where the Discovery Proxy can determine that the querying client is in a different address realm, private addresses SHOULD NOT be communicated to that client.

o IPv6 Unique Local Addresses [RFC4193] SHOULD be suppressed in cases where the Discovery Proxy can determine that the querying client is in a different IPv6 address realm.

o By the same logic, DNS SRV records that reference target host names that have no addresses usable by the requester should be suppressed, and likewise, DNS PTR records that point to unusable SRV records should be similarly be suppressed.

5.5.3. NSEC and NSEC3 queries

Multicast DNS devices do not routinely announce their records on the network. Generally they remain silent until queried. This means that the complete set of Multicast DNS records in use on a link can only be discovered by active querying, not by passive listening. Because of this, a Discovery Proxy can only know what names exist on a link by issuing queries for them, and since it would be impractical to issue queries for every possible name just to find out which names exist and which do not, a Discovery Proxy cannot programmatically generate the traditional NSEC [RFC4034] and NSEC3 [RFC5155] records which assert the nonexistence of a large range of names.

When queried for an NSEC or NSEC3 record type, the Discovery Proxy issues a qtype "ANY" query using Multicast DNS on the local link, and then generates an NSEC or NSEC3 response with a Type Bit Map signifying which record types do and do not exist for just the specific name queried, and no other names.

Multicast DNS NSEC records received on the local link MUST NOT be forwarded unmodified to a unicast querier, because there are slight differences in the NSEC record data. In particular, Multicast DNS NSEC records do not have the NSEC bit set in the Type Bit Map, whereas conventional Unicast DNS NSEC records do have the NSEC bit set.

5.5.4. No Text Encoding Translation

A Discovery Proxy does no translation between text encodings. Specifically, a Discovery Proxy does no translation between Punycode encoding [RFC3492] and UTF-8 encoding [RFC3629], either in the owner name of DNS records, or anywhere in the RDATA of DNS records (such as the RDATA of PTR records, SRV records, NS records, or other record types like TXT, where it is ambiguous whether the RDATA may contain DNS names). All bytes are treated as-is, with no attempt at text encoding translation. A client implementing DNS-based Service Discovery [RFC6763] will use UTF-8 encoding for its service discovery queries, which the Discovery Proxy passes through without any text encoding translation to the Multicast DNS subsystem. Responses from the Multicast DNS subsystem are similarly returned, without any text encoding translation, back to the requesting client.

5.5.5. Application-Specific Data Translation

   There may be cases where Application-Specific Data Translation is
   appropriate.

   For example, AirPrint printers tend to advertise fairly verbose
   information about their capabilities in their DNS-SD TXT record. TXT
   record sizes in the range 500-1000 bytes are not uncommon. This
   information is a legacy from LPR printing, because LPR does not have
   in-band capability negotiation, so all of this information is
   conveyed using the DNS-SD TXT record instead. IPP printing does have
   in-band capability negotiation, but for convenience printers tend to
   include the same capability information in their IPP DNS-SD TXT
   records as well. For local mDNS use this extra TXT record
   information is inefficient, but not fatal. However, when a Discovery
   Proxy aggregates data from multiple printers on a link, and sends it
   via unicast (via UDP or TCP) this amount of unnecessary TXT record
   information can result in large responses. A DNS reply over TCP
   carrying information about 70 printers with an average of 700 bytes
   per printer adds up to about 50 kilobytes of data. Therefore, a
   Discovery Proxy that is aware of the specifics of an application-
   layer protocol such as AirPrint (which uses IPP) can elide
   unnecessary key/value pairs from the DNS-SD TXT record for better
   network efficiency.

   Also, the DNS-SD TXT record for many printers contains an "adminurl"
   key something like "adminurl=http://printername.local/status.html".
   For this URL to be useful outside the local link, the embedded
   ".local" hostname needs to be translated to an appropriate name with
   larger scope. It is easy to translate ".local" names when they
   appear in well-defined places, either as a record's name, or in the
   rdata of record types like PTR and SRV. In the printing case, some
   application-specific knowledge about the semantics of the "adminurl"
   key is needed for the Discovery Proxy to know that it contains a name
   that needs to be translated. This is somewhat analogous to the need
   for NAT gateways to contain ALGs (Application-Specific Gateways) to
   facilitate the correct translation of protocols that embed addresses
   in unexpected places.

   To avoid the need for application-specific knowledge about the
   semantics of particular TXT record keys, protocol designers are
   advised to avoid placing link-local names or link-local IP addresses
   in TXT record keys, if translation of those names or addresses would
   be required for off-link operation. In the printing case, the
   operational failure of failing to translate the "adminurl" key
   correctly is that, when accessed from a different link, printing will
   still work, but clicking the "Admin" UI button will fail to open the
   printer's administration page. Rather than duplicating the host name

from the service's SRV record in its "adminurl" key, thereby having
the same host name appear in two places, a better design might have
been to omit the host name from the "adminurl" key, and instead have
the client implicitly substitute the target host name from the
service's SRV record in place of a missing host name in the
"adminurl" key. That way the desired host name only appears once,
and it is in a well-defined place where software like the Discovery
Proxy is expecting to find it.

Note that this kind of Application-Specific Data Translation is
expected to be very rare. It is the exception, rather than the rule.
This is an example of a common theme in computing. It is frequently
the case that it is wise to start with a clean, layered design, with
clear boundaries. Then, in certain special cases, those layer
boundaries may be violated, where the performance and efficiency
benefits outweigh the inelegance of the layer violation.

These layer violations are optional. They are done primarily for
efficiency reasons, and generally should not be required for correct
operation. A Discovery Proxy MAY operate solely at the mDNS layer,
without any knowledge of semantics at the DNS-SD layer or above.

5.6. Answer Aggregation

   In a simple analysis, simply gathering multicast answers and
   forwarding them in a unicast response seems adequate, but it raises
   the question of how long the Discovery Proxy should wait to be sure
   that it has received all the Multicast DNS answers it needs to form a
   complete Unicast DNS response. If it waits too little time, then it
   risks its Unicast DNS response being incomplete. If it waits too
   long, then it creates a poor user experience at the client end. In
   fact, there may be no time which is both short enough to produce a
   good user experience and at the same time long enough to reliably
   produce complete results.

   Similarly, the Discovery Proxy -- the authoritative name server for
   the subdomain in question -- needs to decide what DNS TTL to report
   for these records. If the TTL is too long then the recursive
   (caching) name servers issuing queries on behalf of their clients
   risk caching stale data for too long. If the TTL is too short then
   the amount of network traffic will be more than necessary. In fact,
   there may be no TTL which is both short enough to avoid undesirable
   stale data and at the same time long enough to be efficient on the
   network.

   Both these dilemmas are solved by use of DNS Long-Lived Queries
   (DNS LLQ) [LLQ] or its newer replacement, DNS Push Notifications
   [Push].

   Clients supporting unicast DNS Service Discovery SHOULD implement DNS
   Push Notifications [Push] for improved user experience.

   Clients and Discovery Proxies MAY support both DNS LLQ and DNS Push,
   and when talking to a Discovery Proxy that supports both, the client
   may use either protocol, as it chooses, though it is expected that
   only DNS Push will continue to be supported in the long run.

   When a Discovery Proxy receives a query using DNS LLQ or DNS Push
   Notifications, it responds immediately using the Multicast DNS
   records it already has in its cache (if any). This provides a good
   client user experience by providing a near-instantaneous response.
   Simultaneously, the Discovery Proxy issues a Multicast DNS query on
   the local link to discover if there are any additional Multicast DNS
   records it did not already know about. Should additional Multicast
   DNS responses be received, these are then delivered to the client
   using additional DNS LLQ or DNS Push Notification update messages.
   The timeliness of such update messages is limited only by the
   timeliness of the device responding to the Multicast DNS query. If
   the Multicast DNS device responds quickly, then the update message is
   delivered quickly. If the Multicast DNS device responds slowly, then

the update message is delivered slowly. The benefit of using update
messages is that the Discovery Proxy can respond promptly because it
doesn't have to delay its unicast response to allow for the expected
worst-case delay for receiving all the Multicast DNS responses. Even
if a proxy were to try to provide reliability by assuming an
excessively pessimistic worst-case time (thereby giving a very poor
user experience) there would still be the risk of a slow Multicast
DNS device taking even longer than that (e.g., a device that is not
even powered on until ten seconds after the initial query is
received) resulting in incomplete responses. Using update message
solves this dilemma: even very late responses are not lost; they are
delivered in subsequent update messages.

There are two factors that determine specifically how responses are
generated:

The first factor is whether the query from the client used LLQ or DNS
Push Notifications (used for long-lived service browsing PTR queries)
or not (used for one-shot operations like SRV or address record
queries). Note that queries using LLQ or DNS Push Notifications are
received directly from the client. Queries not using LLQ or DNS Push
Notifications are generally received via the client's configured
recursive (caching) name server.

The second factor is whether the Discovery Proxy already has at least
one record in its cache that positively answers the question.

o Not using LLQ or Push Notifications; no answer in cache:
   Issue an mDNS query, exactly as a local client would issue an mDNS
   query on the local link for the desired record name, type and
   class, including retransmissions, as appropriate, according to the
   established mDNS retransmission schedule [RFC6762]. As soon as
   any Multicast DNS response packet is received that contains one or
   more positive answers to that question (with or without the Cache
   Flush bit [RFC6762] set), or a negative answer (signified via a
   Multicast DNS NSEC record [RFC6762]), the Discovery Proxy
   generates a Unicast DNS response packet containing the
   corresponding (filtered and translated) answers and sends it to
   the remote client. If after six seconds no Multicast DNS answers
   have been received, cancel the mDNS query and return a negative
   response to the remote client. Six seconds is enough time to
   transmit three mDNS queries, and allow some time for responses to
   arrive.
   DNS TTLs in responses MUST be capped to at most ten seconds.
   (Reasoning: Queries not using LLQ or Push Notifications are
   generally queries that that expect an answer from only one device,
   so the first response is also the only response.)

o Not using LLQ or Push Notifications; at least one answer in cache:
   Send response right away to minimise delay.
   DNS TTLs in responses MUST be capped to at most ten seconds.
   No local mDNS queries are performed.
   (Reasoning: Queries not using LLQ or Push Notifications are
   generally queries that that expect an answer from only one device.
   Given RRSet TTL harmonisation, if the proxy has one Multicast DNS
   answer in its cache, it can reasonably assume that it has all of
   them.)

o Using LLQ or Push Notifications; no answer in cache:
   As in the case above with no answer in the cache, perform mDNS
   querying for six seconds, and send a response to the remote client
   as soon as any relevant mDNS response is received.
   If after six seconds no relevant mDNS response has been received,
   return negative response to the remote client (for LLQ; not
   applicable for Push Notifications).
   (Reasoning: We don't need to rush to send an empty answer.)
   Whether or not a relevant mDNS response is received within six
   seconds, the query remains active for as long as the client
   maintains the LLQ or Push Notification state, and if mDNS answers
   are received later, LLQ or Push Notification messages are sent.
   DNS TTLs in responses are returned unmodified.

o Using LLQ or Push Notifications; at least one answer in cache:
   As in the case above with at least one answer in cache, send
   response right away to minimise delay.
   The query remains active for as long as the client maintains the
   LLQ or Push Notification state, and results in transmission of
   mDNS queries, with appropriate Known Answer lists, to determine if
   further answers are available. If additional mDNS answers are
   received later, LLQ or Push Notification messages are sent.
   (Reasoning: We want UI that is displayed very rapidly, yet
   continues to remain accurate even as the network environment
   changes.)
   DNS TTLs in responses are returned unmodified.

The "negative responses" referred to above are "no error no answer"
negative responses, not NXDOMAIN. This is because the Discovery
Proxy cannot know all the Multicast DNS domain names that may exist
on a link at any given time, so any name with no answers may have
child names that do exist, making it an "empty nonterminal" name.

Note that certain aspects of the behavior described here do not have
to be implemented overtly by the Discovery Proxy; they occur
naturally as a result of using existing Multicast DNS APIs.

For example, in the first case above (no LLQ or Push Notifications,
and no answers in the cache) if a new Multicast DNS query is
requested (either by a local client, or by the Discovery Proxy on
behalf of a remote client), and there is not already an identical
Multicast DNS query active, and there are no matching answers already
in the Multicast DNS cache on the Discovery Proxy device, then this
will cause a series of Multicast DNS query packets to be issued with
exponential backoff. The exponential backoff sequence in some
implementations starts at one second and then doubles for each
retransmission (0, 1, 3, 7 seconds, etc.) and in others starts at one
second and then triples for each retransmission (0, 1, 4, 13 seconds,
etc.). In either case, if no response has been received after six
seconds, that is long enough that the underlying Multicast DNS
implementation will have sent three query packets without receiving
any response. At that point the Discovery Proxy cancels its
Multicast DNS query (so no further Multicast DNS query packets will
be sent for this query) and returns a negative response to the remote
client via unicast.

The six-second delay is chosen to be long enough to give enough time
for devices to respond, yet short enough not to be too onerous for a
human user waiting for a response. For example, using the "dig" DNS
debugging tool, the current default settings result in it waiting a
total of 15 seconds for a reply (three transmissions of the query
packet, with a wait of 5 seconds after each packet) which is ample
time for it to have received a negative reply from a Discovery Proxy
after six seconds.

The statement that for a one-shot query (i.e., no LLQ or Push
Notifications requested), if at least one answer is already available
in the cache then a Discovery Proxy should not issue additional mDNS
query packets, also occurs naturally as a result of using existing
Multicast DNS APIs. If a new Multicast DNS query is requested
(either locally, or by the Discovery Proxy on behalf of a remote
client), for which there are relevant answers already in the
Multicast DNS cache on the Discovery Proxy device, and after the
answers are delivered the Multicast DNS query is then cancelled
immediately, then no Multicast DNS query packets will be generated
for this query.

6. Administrative DNS Records

6.1. DNS SOA (Start of Authority) Record

   The MNAME field SHOULD contain the host name of the Discovery Proxy
   device (i.e., the same domain name as the rdata of the NS record
   delegating the relevant zone(s) to this Discovery Proxy device).

   The RNAME field SHOULD contain the mailbox of the person responsible
   for administering this Discovery Proxy device.

   The SERIAL field MUST be zero.

   Zone transfers are undefined for Discovery Proxy zones, and
   consequently the REFRESH, RETRY and EXPIRE fields have no useful
   meaning for Discovery Proxy zones. These fields SHOULD contain
   reasonable default values. The RECOMMENDED values are: REFRESH 7200,
   RETRY 3600, EXPIRE 86400.

   The MINIMUM field (used to control the lifetime of negative cache
   entries) SHOULD contain the value 10. The value of ten seconds is
   chosen based on user-experience considerations (see Section 5.5.1).

   In the event that there are multiple Discovery Proxy devices on a
   link for fault tolerance reasons, this will result in clients
   receiving inconsistent SOA records (different MNAME, and possibly
   RNAME) depending on which Discovery Proxy answers their SOA query.
   However, since clients generally have no reason to use the MNAME or
   RNAME data, this is unlikely to cause any problems.

6.2. DNS NS Records

   In the event that there are multiple Discovery Proxy devices on a
   link for fault tolerance reasons, the parent zone MUST be configured
   with NS records giving the names of all the Discovery Proxy devices
   on the link.

   Each Discovery Proxy device MUST be configured to answer NS queries
   for the zone apex name by giving its own NS record, and the NS
   records of its fellow Discovery Proxy devices on the same link, so
   that it can return the correct answers for NS queries.

   The target host name in the RDATA of an NS record MUST NOT reference
   a name that falls within any zone delegated to a Discovery Proxy.
   Apart from the zone apex name, all other host names that fall within
   a zone delegated to a Discovery Proxy correspond to local Multicast
   DNS host names, which logically belong to the respective Multicast
   DNS hosts defending those names, not the Discovery Proxy. Generally
   speaking, the Discovery Proxy does not own or control the delegated
   zone; it is merely a conduit to the corresponding ".local" namespace,
   which is controlled by the Multicast DNS hosts on that link. If an
   NS record were to reference a manually-determined host name that
   falls within a delegated zone, that manually-determined host name may
   inadvertently conflict with a corresponding ".local" host name that
   is owned and controlled by some device on that link.

6.3. DNS Delegation Records

   Since the Multicast DNS specification [RFC6762] states that there can
   be no delegation (subdomains) within a ".local" namespace, this
   implies that any name within a zone delegated to a Discovery Proxy
   (except for the zone apex name itself) cannot have any answers for
   any DNS queries for RRTYPEs SOA, NS, or DS. Consequently:

   o for any query for the zone apex name of a zone delegated to a
      Discovery Proxy, the Discovery Proxy MUST generate the appropriate
      immediate answers as described above, and

   o for any query for RRTYPEs SOA, NS, or DS, for any name within a
      zone delegated to a Discovery Proxy, other than the zone apex
      name, instead of translating the query to its corresponding
      Multicast DNS ".local" equivalent, a Discovery Proxy MUST generate
      an immediate negative answer.

6.4. DNS SRV Records

   There are certain special DNS records that logically fall within the
   delegated unicast DNS subdomain, but rather than mapping to their
   corresponding ".local" namesakes, they actually contain metadata
   pertaining to the operation of the delegated unicast DNS subdomain
   itself. They do not exist in the corresponding ".local" namespace of
   the local link. For these queries a Discovery Proxy MUST generate
   immediate answers, whether positive or negative, to avoid delays
   while clients wait for their query to be answered. For example, if a
   Discovery Proxy does not implement Long-Lived Queries [LLQ] then it
   MUST return an immediate negative answer to tell the client this
   without delay, instead of passing the query through to the local
   network as a query for "_dns-llq._udp.local.", and then waiting
   unsuccessfully for answers that will not be forthcoming.

   If a Discovery Proxy implements Long-Lived Queries [LLQ] then it MUST
   positively respond to "_dns-llq._udp.<zone> SRV" queries,
   "_dns-llq._tcp.<zone> SRV" queries, and
   "_dns-llq-tls._tcp.<zone> SRV" queries as appropriate, else it MUST
   return an immediate negative answer for those queries.

   If a Discovery Proxy implements DNS Push Notifications [Push] then it
   MUST positively respond to "_dns-push-tls._tcp.<zone>" queries, else
   it MUST return an immediate negative answer for those queries.

   A Discovery Proxy MUST return an immediate negative answer for
   "_dns-update._udp.<zone> SRV" queries, "_dns-update._tcp.<zone> SRV"
   queries, and "_dns-update-tls._tcp.<zone> SRV" queries, since using
   DNS Update [RFC2136] to change zones generated dynamically from local
   Multicast DNS data is not possible.

7. DNSSEC Considerations

7.1. On-line signing only

   The Discovery Proxy acts as the authoritative name server for
   designated subdomains, and if DNSSEC is to be used, the Discovery
   Proxy needs to possess a copy of the signing keys, in order to
   generate authoritative signed data from the local Multicast DNS
   responses it receives. Off-line signing is not applicable to
   Discovery Proxy.

7.2. NSEC and NSEC3 Records

   In DNSSEC NSEC [RFC4034] and NSEC3 [RFC5155] records are used to
   assert the nonexistence of certain names, also described as
   "authenticated denial of existence".

   Since a Discovery Proxy only knows what names exist on the local link
   by issuing queries for them, and since it would be impractical to
   issue queries for every possible name just to find out which names
   exist and which do not, a Discovery Proxy cannot programmatically
   synthesize the traditional NSEC and NSEC3 records which assert the
   nonexistence of a large range of names. Instead, when generating a
   negative response, a Discovery Proxy programmatically synthesizes a
   single NSEC record assert the nonexistence of just the specific name
   queried, and no others. Since the Discovery Proxy has the zone
   signing key, it can do this on demand. Since the NSEC record asserts
   the nonexistence of only a single name, zone walking is not a
   concern, so NSEC3 is not necessary.

   Note that this applies only to traditional immediate DNS queries,
   which may return immediate negative answers when no immediate
   positive answer is available. When used with a DNS Push Notification
   subscription [Push] there are no negative answers, merely the absence
   of answers so far, which may change in the future if answers become
   available.

8. IPv6 Considerations

   An IPv4-only host and an IPv6-only host behave as "ships that pass in
   the night". Even if they are on the same Ethernet [IEEE-3], neither
   is aware of the other's traffic. For this reason, each link may have
   two unrelated ".local." zones, one for IPv4 and one for IPv6.
   Since for practical purposes, a group of IPv4-only hosts and a group
   of IPv6-only hosts on the same Ethernet act as if they were on two
   entirely separate Ethernet segments, it is unsurprising that their
   use of the ".local." zone should occur exactly as it would if they
   really were on two entirely separate Ethernet segments.

   It will be desirable to have a mechanism to 'stitch' together these
   two unrelated ".local." zones so that they appear as one. Such
   mechanism will need to be able to differentiate between a dual-stack
   (v4/v6) host participating in both ".local." zones, and two different
   hosts, one IPv4-only and the other IPv6-only, which are both trying
   to use the same name(s). Such a mechanism will be specified in a
   future companion document.

   At present, it is RECOMMENDED that a Discovery Proxy be configured
   with a single domain name for both the IPv4 and IPv6 ".local." zones
   on the local link, and when a unicast query is received, it should
   issue Multicast DNS queries using both IPv4 and IPv6 on the local
   link, and then combine the results.

9. Security Considerations

9.1. Authenticity

   A service proves its presence on a link by its ability to answer
   link-local multicast queries on that link. If greater security is
   desired, then the Discovery Proxy mechanism should not be used, and
   something with stronger security should be used instead, such as
   authenticated secure DNS Update [RFC2136] [RFC3007].

9.2. Privacy

   The Domain Name System is, generally speaking, a global public
   database. Records that exist in the Domain Name System name
   hierarchy can be queried by name from, in principle, anywhere in the
   world. If services on a mobile device (like a laptop computer) are
   made visible via the Discovery Proxy mechanism, then when those
   services become visible in a domain such as "My House.example.com"
   that might indicate to (potentially hostile) observers that the
   mobile device is in my house. When those services disappear from
   "My House.example.com" that change could be used by observers to
   infer when the mobile device (and possibly its owner) may have left
   the house. The privacy of this information may be protected using
   techniques like firewalls, split-view DNS, and Virtual Private
   Networks (VPNs), as are customarily used today to protect the privacy
   of corporate DNS information.

   The privacy issue is particularly serious for the IPv4 and IPv6
   reverse zones. If the public delegation of the reverse zones points
   to the Discovery Proxy, and the Discovery Proxy is reachable
   globally, then it could leak a significant amount of information.
   Attackers could discover hosts that otherwise might not be easy to
   identify, and learn their hostnames. Attackers could also discover
   the existence of links where hosts frequently come and go.

   The Discovery Proxy could also provide sensitive records only to
   authenticated users. This is a general DNS problem, not specific to
   the Discovery Proxy. Work is underway in the IETF to tackle this
   problem [RFC7626].

9.3. Denial of Service

   A remote attacker could use a rapid series of unique Unicast DNS
   queries to induce a Discovery Proxy to generate a rapid series of
   corresponding Multicast DNS queries on one or more of its local
   links. Multicast traffic is generally more expensive than unicast
   traffic -- especially on Wi-Fi links -- which makes this attack
   particularly serious. To limit the damage that can be caused by such

attacks, a Discovery Proxy (or the underlying Multicast DNS subsystem which it utilizes) MUST implement Multicast DNS query rate limiting appropriate to the link technology in question. For today's 802.11b/g/n/ac Wi-Fi links (for which approximately 200 multicast packets per second is sufficient to consume approximately 100% of the wireless spectrum) a limit of 20 Multicast DNS query packets per second is RECOMMENDED. On other link technologies like Gigabit Ethernet higher limits may be appropriate. A consequence of this rate limiting is that a rogue remote client could issue an excessive number of queries, resulting in denial of service to other legitimate remote clients attempting to use that Discovery Proxy. However, this is preferable to a rogue remote client being able to inflict even greater harm on the local network, which could impact the correct operation of all local clients on that network.

10. IANA Considerations

   This document has no IANA Considerations.

11. Acknowledgments

   Thanks to Markus Stenberg for helping develop the policy regarding the four styles of unicast response according to what data is immediately available in the cache. Thanks to Anders Brandt, Ben Campbell, Tim Chown, Alissa Cooper, Spencer Dawkins, Ralph Droms, Joel Halpern, Ray Hunter, Joel Jaeggli, Warren Kumari, Ted Lemon, Alexey Melnikov, Kathleen Moriarty, Tom Pusateri, Eric Rescorla, Adam Roach, David Schinazi, Markus Stenberg, Dave Thaler, and Andrew Yourtchenko for their comments.

12. References

12.1. Normative References

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

   [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>.

   [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.,
              and E. Lear, "Address Allocation for Private Internets",
              BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996,
              <https://www.rfc-editor.org/info/rfc1918>.

   [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>.

   [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS
              NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998,
              <https://www.rfc-editor.org/info/rfc2308>.

   [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO
              10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November
              2003, <https://www.rfc-editor.org/info/rfc3629>.

   [RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic
              Configuration of IPv4 Link-Local Addresses", RFC 3927,
              DOI 10.17487/RFC3927, May 2005, <https://www.rfc-
              editor.org/info/rfc3927>.

   [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S.
              Rose, "Resource Records for the DNS Security Extensions",
              RFC 4034, DOI 10.17487/RFC4034, March 2005,
              <https://www.rfc-editor.org/info/rfc4034>.

   [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless
              Address Autoconfiguration", RFC 4862,
              DOI 10.17487/RFC4862, September 2007, <https://www.rfc-
              editor.org/info/rfc4862>.

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

   [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network
              Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008,
              <https://www.rfc-editor.org/info/rfc5198>.

   [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>.

   [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>.

   [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>.

   [Push]   Pusateri, T. and S. Cheshire, "DNS Push Notifications",
              draft-ietf-dnssd-push-19 (work in progress), March 2019.

12.2. Informative References

   [Roadmap] Cheshire, S., "Service Discovery Road Map", draft-
              cheshire-dnssd-roadmap-03 (work in progress), October
              2018.

   [DNS-UL]   Sekar, K., "Dynamic DNS Update Leases", draft-sekar-dns-
              ul-01 (work in progress), August 2006.

   [LLQ]   Cheshire, S. and M. Krochmal, "DNS Long-Lived Queries",
              draft-sekar-dns-llq-03 (work in progress), March 2019.

   [RegProt] Cheshire, S. and T. Lemon, "Service Registration Protocol
              for DNS-Based Service Discovery", draft-sctl-service-
              registration-00 (work in progress), July 2017.

   [Relay]   Cheshire, S. and T. Lemon, "Multicast DNS Discovery
              Relay", draft-sctl-dnssd-mdns-relay-04 (work in progress),
              March 2018.

   [Mcast]   Perkins, C., McBride, M., Stanley, D., Kumari, W., and J.
              Zuniga, "Multicast Considerations over IEEE 802 Wireless
              Media", draft-ietf-mboned-ieee802-mcast-problems-04 (work
              in progress), November 2018.

   [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor
              Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997,
              <https://www.rfc-editor.org/info/rfc2132>.

   [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>.

   [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic
              Update", RFC 3007, DOI 10.17487/RFC3007, November 2000,
              <https://www.rfc-editor.org/info/rfc3007>.

   [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode
              for Internationalized Domain Names in Applications
              (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003,
              <https://www.rfc-editor.org/info/rfc3492>.

   [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast
              Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005,
              <https://www.rfc-editor.org/info/rfc4193>.

   [RFC6760] Cheshire, S. and M. Krochmal, "Requirements for a Protocol
              to Replace the AppleTalk Name Binding Protocol (NBP)",
              RFC 6760, DOI 10.17487/RFC6760, February 2013,
              <https://www.rfc-editor.org/info/rfc6760>.

   [RFC7558] Lynn, K., Cheshire, S., Blanchet, M., and D. Migault,
              "Requirements for Scalable DNS-Based Service Discovery
              (DNS-SD) / Multicast DNS (mDNS) Extensions", RFC 7558,
              DOI 10.17487/RFC7558, July 2015, <https://www.rfc-
              editor.org/info/rfc7558>.

   [RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626,
              DOI 10.17487/RFC7626, August 2015, <https://www.rfc-
              editor.org/info/rfc7626>.

   [RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking
              Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April
              2016, <https://www.rfc-editor.org/info/rfc7788>.

   [RFC8375] Pfister, P. and T. Lemon, "Special-Use Domain
              'home.arpa.'", RFC 8375, DOI 10.17487/RFC8375, May 2018,
              <https://www.rfc-editor.org/info/rfc8375>.

   [ohp]   "Discovery Proxy (Hybrid Proxy) implementation for
              OpenWrt", <https://github.com/sbyx/ohybridproxy/>.

   [ZC]   Cheshire, S. and D. Steinberg, "Zero Configuration
              Networking: The Definitive Guide", O'Reilly Media, Inc. ,
              ISBN 0-596-10100-7, December 2005.

   [IEEE-1Q] "IEEE Standard for Local and metropolitan area networks --
              Bridges and Bridged Networks", IEEE Std 802.1Q-2014,
              November 2014, <http://standards.ieee.org/getieee802/
              download/802-1Q-2014.pdf>.

   [IEEE-3]   "Information technology – Telecommunications and
              information exchange between systems – Local and
              metropolitan area networks – Specific requirements – Part
              3: Carrier Sense Multiple Access with Collision Detection
              (CMSA/CD) Access Method and Physical Layer
              Specifications", IEEE Std 802.3-2008, December 2008,
              <http://standards.ieee.org/getieee802/802.3.html>.

   [IEEE-5]   Institute of Electrical and Electronics Engineers,
              "Information technology – Telecommunications and
              information exchange between systems – Local and
              metropolitan area networks – Specific requirements – Part
              5: Token ring access method and physical layer
              specification", IEEE Std 802.5-1998, 1995.

   [IEEE-11] "Information technology – Telecommunications and
              information exchange between systems – Local and
              metropolitan area networks – Specific requirements – Part
              11: Wireless LAN Medium Access Control (MAC) and Physical
              Layer (PHY) Specifications", IEEE Std 802.11-2007, June
              2007, <http://standards.ieee.org/getieee802/802.11.html>.

Appendix A. Implementation Status

   Some aspects of the mechanism specified in this document already
   exist in deployed software. Some aspects are new. This section
   outlines which aspects already exist and which are new.

A.1. Already Implemented and Deployed

   Domain enumeration by the client (the "b._dns-sd._udp" queries) is
   already implemented and deployed.

   Unicast queries to the indicated discovery domain is already
   implemented and deployed.

   These are implemented and deployed in Mac OS X 10.4 and later
   (including all versions of Apple iOS, on all iPhone and iPads), in
   Bonjour for Windows, and in Android 4.1 "Jelly Bean" (API Level 16)
   and later.

   Domain enumeration and unicast querying have been used for several
   years at IETF meetings to make Terminal Room printers discoverable
   from outside the Terminal room. When an IETF attendee presses Cmd-P
   on a Mac, or selects AirPrint on an iPad or iPhone, and the Terminal
   room printers appear, that is because the client is sending unicast
   DNS queries to the IETF DNS servers. A walk-through giving the
   details of this particular specific example is given in Appendix A of
   the Roadmap document [Roadmap].

A.2. Already Implemented

   A minimal portable Discovery Proxy implementation has been produced
   by Markus Stenberg and Steven Barth, which runs on OS X and several
   Linux variants including OpenWrt [ohp]. It was demonstrated at the
   Berlin IETF in July 2013.

   Tom Pusateri has an implementation that runs on any Unix/Linux. It
   has a RESTful interface for management and an experimental demo CLI
   and web interface.

   Ted Lemon also has produced a portable implementation of Discovery
   Proxy, which is available in the mDNSResponder open source code.

   The Long-Lived Query mechanism [LLQ] referred to in this
   specification exists and is deployed, but was not standardized by the
   IETF. The IETF has developed a superior Long-Lived Query mechanism
   called DNS Push Notifications [Push], which is built on DNS Stateful
   Operations [RFC8490]. The pragmatic short-term deployment approach
   is for vendors to produce Discovery Proxies that implement both the

deployed Long-Lived Query mechanism [LLQ] (for today's clients) and the new DNS Push Notifications mechanism [Push] as the preferred long-term direction.

A.3. Partially Implemented

The current APIs make multiple domains visible to client software, but most client UI today lumps all discovered services into a single flat list. This is largely a chicken-and-egg problem. Application writers were naturally reluctant to spend time writing domain-aware UI code when few customers today would benefit from it. If Discovery Proxy deployment becomes common, then application writers will have a reason to provide better UI. Existing applications will work with the Discovery Proxy, but will show all services in a single flat list. Applications with improved UI will group services by domain.

Author's Address

Stuart Cheshire
Apple Inc.
One Apple Park Way
Cupertino, California 95014
USA

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