Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 7, 2016

S. Cheshire
Apple Inc.

February 4, 2016

Hybrid Unicast/Multicast DNS-Based Service Discovery
draft-ietf-dnssd-hybrid-03

Abstract

Performing DNS-Based Service Discovery using purely link-local Multicast DNS enables discovery of services that are on the local link, but not (without some kind of proxy or similar special support) discovery of services that are outside the local link. Using a very large local link with thousands of hosts facilitates service discovery, but at the cost of large amounts of multicast traffic.

Performing DNS-Based Service Discovery using purely Unicast DNS is more efficient and doesn't require excessively large multicast domains, but requires that the relevant data be available in the Unicast DNS namespace. This can be achieved by manual DNS configuration (as has been done for many years at IETF meetings to advertise the IETF Terminal Room printer) but this is labor intensive, error prone, and requires a reasonable degree of DNS expertise. The Unicast DNS namespace can be populated with the required data automatically by the devices themselves, but that requires configuration of DNS Update keys on the devices offering the services, 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.

Status of this Memo

material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 7, 2016.

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].

For a small network consisting of just a single link (or several physical links bridged together to appear as a single logical link to IP) Multicast DNS [RFC6762] is sufficient for client devices to look up the dot-local host names of peers on the same home network, and perform DNS-Based Service Discovery (DNS-SD) [RFC6763] of 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, do not cross between links. (This was a deliberate design choice for Multicast DNS, since even on a single link multicast traffic is expensive -- especially on Wi-Fi links -- and multiplying the amount of multicast traffic by flooding it across multiple links would make that problem even worse.) In this environment, Unicast DNS would be preferable to Multicast DNS. (Unicast DNS can be used either with a traditionally assigned globally unique domain name, or with a private local unicast domain name such as ".home" [HOME].)

To use Unicast DNS, the names of hosts and services need to be made available in the Unicast DNS namespace. 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.

This document specifies a type of proxy called a Hybrid 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.

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 "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

The Hybrid Proxy builds on Multicast DNS, which works between hosts on the same link. A set of hosts is considered to be "on the same link" if:

- o when any host A from that set sends a packet to any other host B 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 [802.5], 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.

3. Compatibility Considerations

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

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

Existing clients that support DNS-Based Service Discovery over Unicast DNS (Mac OS X 10.4 and later, including iPhone, iPad, and Bonjour for Windows) work with Hybrid Proxy.

4. Hybrid Proxy Operation

In a typical configuration, a Hybrid Proxy is configured to be authoritative [RFC1034] [RFC1035] for four 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.

A DNS subdomain for IPv6 Reverse Mapping records.
   This subdomain name will be a name that ends in "ip6.arpa."

A DNS subdomain for IPv4 Reverse Mapping records.
   This subdomain name will be a name that ends in "in-addr.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 [I-D.ietf-homenet-hncp].

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

4.1. Delegated Subdomain for Service Discovery Records

In its simplest form, each physical 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 a Hybrid Proxy which serves it. This Hybrid Proxy function could be performed by a router on that link, or, with appropriate VLAN configuration, a single Hybrid Proxy could have a logical presence on, and serve as the Hybrid Proxy for, many links. In the parent domain, NS records are used to delegate ownership of each defined link name (e.g., "Building 1.example.com") to the Hybrid Proxy that serves the named link. In other words, the Hybrid Proxy is the authoritative name server for that subdomain.

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 Hybrid Proxy on the link in question. Like a conventional Unicast DNS server, a Hybrid 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 Hybrid Proxy generates answers using Multicast DNS. A Hybrid Proxy does this by consulting its Multicast DNS cache and/or issuing Multicast DNS queries for the corresponding Multicast DNS name, type and class, (e.g., in this

case, "_printer._tcp.local. PTR ?"). Then, from the received Multicast DNS data, the Hybrid Proxy synthesizes the appropriate Unicast DNS response.

Naturally, the existing Multicast DNS caching mechanism is used to avoid issuing unnecessary Multicast DNS queries on the wire. The Hybrid 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.

4.2. Domain Enumeration

An DNS-SD client performs Domain Enumeration [RFC6763] via certain PTR queries. It issues unicast Domain Enumeration queries using its "home" domain (typically learned via DHCP) and using its IPv6 prefix and IPv4 subnet address. These are described below in Section 4.2.1. It also issues multicast Domain Enumeration queries in the "local" domain [RFC6762]. These are described below in Section 4.2.2. The results of all Domain Enumeration queries are combined for Service Discovery purposes.

4.2.1. Domain Enumeration via Unicast Queries

The administrator creates Domain Enumeration PTR records [RFC6763] to inform clients of available service discovery domains, e.g.,:

```
    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 "b" ("browse") records tell the client device the list of browsing domains to display for the user to select from and the "db" ("default browse") record tells the client device which domain in that list should be 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, as of 2015). 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.

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.

4.2.2. Domain Enumeration via Multicast Queries

Since a Hybrid Proxy exists on many, if not all, the links in an enterprise, it offers an additional way to provide Domain Enumeration data for clients.

A Hybrid Proxy can be configured to generate Multicast DNS responses for the following Multicast DNS Domain Enumeration queries issues 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 Hybrid Proxies to provide configuration data on a per-link granularity to DNS-SD clients. In some enterprises it may be preferable to provide this per-link configuration data in the form of Hybrid Proxy configuration, rather than populating the Unicast DNS servers with the same data (in the "ip6.arpa" or "in-addr.arpa" domains).

4.3. Delegated Subdomain for LDH Host Names

The traditional rules for host names are more restrictive than those for DNS-SD service instance names and domains.

Users typically interact with DNS-SD by viewing a list of discovered service instance names on the 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 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, with no spaces or other punctuation.

While we still 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 Hybrid Proxy SOULD support having separate subdomains delegated to it, 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).

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

```
My Printer._ipp._tcp.local. SRV 0 0 631 prnt.local.
prnt.local.                 A   10.0.1.2
```

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   10.0.1.2
```

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

4.4. Delegated Subdomain for Reverse Mapping

A Hybrid 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 Hybrid Proxy. In other words, the Hybrid Proxy becomes the authoritative name server for the reverse mapping domain.

For example, if a given link is using the IPv6 prefix 2001:0DB8/32, then the domain "8.b.d.0.1.0.0.2.ip6.arpa" is delegated to the Hybrid Proxy for that link.

If a given link is using the IPv4 subnet 10.1/16, then the domain "1.10.in-addr.arpa" is delegated to the Hybrid Proxy for that link.

When a reverse mapping query arrives at the Hybrid Proxy, it issues the identical query on its local link as a Multicast DNS query. (In the Apple "/usr/include/dns_sd.h" APIs, using ForceMulticast indicates that the DNSServiceQueryRecord() call should perform the query using Multicast DNS.) When the host owning that IPv6 or IPv4 address responds with a name of the form "something.local", the Hybrid Proxy rewrites that to use its configured LDH host name domain instead of "local" and returns the response to the caller.

For example, a Hybrid Proxy with the two subdomains "1.10.in-addr.arpa" and "bldg1.example.com" delegated to it would translate this Multicast DNS record:

```
3.2.1.10.in-addr.arpa. PTR prnt.local.
```

into this Unicast DNS response:

```
3.2.1.10.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 Hybrid Proxy, will arrive at the Hybrid 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.

4.5. 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 configured DNS Unicast domain.

Other beneficial translation and filtering operations are described below.

4.5.1. 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 neighbouring 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 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 any Long-Lived Query [I-D.sekar-dns-llq] or DNS Push Notification [I-D.ietf-dnssd-push] option) the DNS TTLs reported in the resulting Unicast DNS response SHOULD 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 5).

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 reasonble 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 contain an LLQ or DNS Push Notification option, 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 4.6.

4.5.2. Suppressing Unusable Records

A Hybrid Proxy SHOULD suppress Unicast DNS answers for records that are not useful outside the local link. For example, DNS A and AAAA records for IPv6 link-local addresses [RFC4862] and IPv4 link-local addresses [RFC3927] should be suppressed. Similarly, for sites that have multiple private address realms [RFC1918], private addresses from one private address realm should not be communicated to clients in a different private address realm.

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.

4.5.3. 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 Hybrid 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 Hybrid 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 dot-local hostname needs to be translated to an appropriate name with larger scope. Dot-local names are easily translated 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 Hybrid 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.

As is the case with NAT ALGs, protocol designers are advised to avoid communicating names and addresses in nonstandard locations, because those "hidden" names and addresses are at risk of not being translated when necessary, resulting in operational failures. 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 Hybrid 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 Hybrid Proxy MAY operate solely at the mDNS layer, without any knowledge of semantics at the DNS-SD layer or above.

4.6. 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 Hybrid 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 Hybrid 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) [I-D.sekar-dns-llq] or its newer replacement, DNS Push Notifications [I-D.ietf-dnssd-push]. (Clients and Hybrid Proxies can support both DNS LLQ and DNS Push, and when talking to a Hybrid 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 Hybrid Proxy receives a query containing a DNS LLQ or DNS Push Notification option, 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 Hybrid 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 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 Hybrid 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 included an LLQ or DNS Push Notification option (typical with long-lived service browsing PTR queries) or not (typical with one-shot operations like SRV or address record queries). Note that queries containing the LLQ or PUSH option are received directly from the client (see Section 4.6.1). Queries containing no LLQ or PUSH option are generally received via the client's configured recursive (caching) name server.

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

- o No LLQ or PUSH option; 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 an NSEC record [RFC6762]), the Hybrid 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, return a negative response to the remote client.
DNS TTLs in responses are capped to at most ten seconds.

- No LLQ or PUSH option; at least one answer in cache:
  Send response right away to minimise delay.
  DNS TTLs in responses are capped to at most ten seconds.
  No local mDNS queries are performed.
  (Reasoning: 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.)

- Query contains LLQ or PUSH option; 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. (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 state, and if mDNS answers are received later, LLQ or PUSH update messages are sent.
  DNS TTLs in responses are returned unmodified.

- Query contains LLQ or PUSH option; 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 state, and if additional mDNS answers are received later, LLQ or PUSH update 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.

Note that the "negative responses" referred to above are "no error no answer" negative responses, not NXDOMAIN. This is because the Hybrid 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.

4.6.1. Discovery of LLQ and/or PUSH Notification Service

To issue LLQ or PUSH queries, clients need to communicate directly with the authoritative Hybrid Proxy. The procedure by which the client locates the authoritative Hybrid Proxy is described in the LLQ specification [I-D.sekar-dns-llq] and the DNS Push Notifications specification [I-D.ietf-dnssd-push].

Briefly, the procedure is as follows:

To discover the LLQ service for a given domain name, a client first performs DNS zone apex discovery, and then, having discovered <apex>, the client then issues a DNS query for the SRV record with the name _dns-llq._udp.<apex> to find the target host and port for the LLQ service for that zone. By default LLQ service runs on UDP port 5352, but since SRV records are used, the LLQ service can be offered on any port.

To discover the DNS Push Notification service for a given domain name, a client first performs DNS zone apex discovery, and then, having discovered <apex>, the client then issues a DNS query for the SRV record with the name _dns-push-tls._tcp.<apex> to find the target host and port for the DNS Push Notification service for that zone. By default DNS Push Notification service runs on TCP port 5352, but since SRV records are used, the DNS Push Notification service can be offered on any port.

A client performs DNS zone apex discovery using the procedure below:

1. The client issues a DNS query for the SOA record with the given domain name.

2. A conformant recursive (caching) name server will either send a positive response, or a negative response containing the SOA record of the zone apex in the Authority Section.

3. If the name server sends a negative response that does not contain the SOA record of the zone apex, the client trims the first label off the given domain name and returns to step 1 to try again.

By this method, the client iterates until it learns the name of the zone apex, or (in pathological failure cases) reaches the root and gives up.

Normal DNS caching is used to avoid repetitive queries on the wire.

5. DNS SOA (Start of Authority) Record

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

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

The SERIAL field SHOULD contain a sequence number that increments each time the Hybrid Proxy returns an SOA record to any client. [Author's note: Or maybe it could just be zero?]

Since zone transfers are undefined for Hybrid Proxy zones, the REFRESH, RETRY and EXPIRE fields have no useful meaning for Hybrid 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 4.5.1).

[Author's note: Discussion of these recommendations is requested.]

6. 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.

6.1. 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 you Press Cmd-P on your Mac, or select AirPrint on your iPad or iPhone, and the Terminal room

printers appear, that is because your client is sending unicast DNS queries to the IETF DNS servers.

6.2. Already Implemented

A minimal portable Hybrid 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 also has an implementation that runs on any Unix/Linux. It has a RESTful interface for management and an experimental demo CLI and web interface.

6.3. 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 Hybrid Proxy deployment becomes common, then application writers will have a reason to provide better UI. Existing applications will work with the Hybrid Proxy, but will show all services in a single flat list. Applications with improved UI will group services by domain.

The Long-Lived Query mechanism [I-D.sekar-dns-llq] referred to in this specification exists and is deployed, but has not been standardized by the IETF. The IETF is considering standardizing a superior Long-Lived Query mechanism called DNS Push Notifications [I-D.ietf-dnssd-push]. The pragmatic short-term deployment approach is for vendors to produce Hybrid Proxies that implement both the deployed Long-Lived Query mechanism [I-D.sekar-dns-llq] (for today's clients) and the new DNS Push Notifications mechanism [I-D.ietf-dnssd-push] as the preferred long-term direction.

The translating/filtering Hybrid Proxy specified in this document. Implementations are under development, and operational experience with these implementations has guided updates to this document.

6.4. Not Yet Implemented

Client implementations of the new DNS Push Notifications mechanism [I-D.ietf-dnssd-push] are currently underway.

A mechanism to 'stitch' together multiple ".local." zones so that they appear as one. Such a mechanism will be specified in a future companion document.

7. IPv6 Considerations

An IPv6-only host and an IPv4-only host behave as "ships that pass in the night". Even if they are on the same Ethernet, neither is aware of the other's traffic. For this reason, each physical link may have *two* unrelated ".local." zones, one for IPv6 and one for IPv4. Since for practical purposes, a group of IPv6-only hosts and a group of IPv4-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 IPv6-only and the other IPv4-only, which are both trying to use the same name(s). Such a mechanism will be specified in a future companion document.

8. Security Considerations

8.1. 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 Hybrid Proxy mechanism should not be used, and something with stronger security should be used instead, such as authenticated secure DNS Update [RFC2136] [RFC3007].

8.2. 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 Hybrid Proxy mechanism, then when those services become visibile 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 and split-view DNS, as are customarily used today to protect the privacy of corporate DNS information.

8.3. Denial of Service

A remote attacker could use a rapid series of unique Unicast DNS queries to induce a Hybrid Proxy to generate a rapid series of corresponding Multicast DNS queries on one or more of its local links. Multicast traffic is expensive -- especially on Wi-Fi links -- which makes this attack particularly serious. To limit the damage that can be caused by such attacks, a Hybrid 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 Wi-Fi links the Multicast DNS subsystem SHOULD NOT issue more than 20 Multicast DNS query packets per second. On other link technologies like Gigabit Ethernet higher limits may be appropriate.

9. Intelectual Property Rights

Apple has submitted an IPR disclosure concerning the technique proposed in this document. Details are available on the IETF IPR disclosure page [IPR2119].

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 and Andrew Yourtchenko for their comments. [Partial list; more names to be added.]

12. References

12.1. Normative References

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

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., J. de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://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, <http://www.rfc-editor.org/info/rfc2119>.

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

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

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

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

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, December 2012.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, December 2012.

[I-D.ietf-dnssd-push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-03 (work in progress), November 2015.

12.2. Informative References

[HOME] Cheshire, S., "Special Use Top Level Domain 'home'", draft-cheshire-homenet-dot-home (work in progress), November 2015.

[IPR2119] "Apple Inc.'s Statement about IPR related to Hybrid Unicast/Multicast DNS-Based Service Discovery", <https://datatracker.ietf.org/ipr/2119/>.

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

[I-D.sekar-dns-llq]
Sekar, K., "DNS Long-Lived Queries",
draft-sekar-dns-llq-01 (work in progress), August 2006.

[I-D.ietf-homenet-hncp]
Stenberg, M., Barth, S., and P. Pfister, "Home Networking
Control Protocol", draft-ietf-homenet-hncp-09 (work in
progress), August 2015.

[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,
<http://www.rfc-editor.org/info/rfc2136>.

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

[RFC6760] Cheshire, S. and M. Krochmal, "Requirements for a Protocol
to Replace the AppleTalk Name Binding Protocol (NBP)",
RFC 6760, December 2012.

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

Author's Address

Stuart Cheshire
Apple Inc.
1 Infinite Loop
Cupertino, California 95014
USA

Phone: +1 408 974 3207
Email: cheshire@apple.com