Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: April 26, 2019

S. Cheshire
Apple Inc.

October 23, 2018

Service Discovery Road Map
draft-cheshire-dnssd-roadmap-03

Abstract

Over the course of several years, a rich collection of technologies has developed around DNS-Based Service Discovery, described across multiple documents. This "Road Map" document gives an overview of how these related but separate technologies (and their documents) fit together, to facilitate service discovery in various environments.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 26, 2019.

1. Road Map

DNS-Based Service Discovery [RFC6763] is a component of Zero Configuration Networking [RFC6760] [ZC].

Over the course of several years, a rich collection of technologies has developed around DNS-Based Service Discovery. These various related but separate technologies are described across multiple documents. This "Road Map" document gives an overview of how these technologies (and their documents) fit together to facilitate service discovery across a broad range of operating environments, from small scale zero-configuration networks to large scale administered networks, from local area to wide area, and from low-speed wireless links in the kb/s range to high-speed wired links operating at multiple Gb/s.

Not all of the available components are necessary or appropriate in all scenarios. One goal of this "Road Map" document is to provide guidance about which components to use depending on the problem being solved.

2. Namespace of Service Types

The single most important concept in service discovery is the namespace specifying how different service types are identified. This is how a client communicates what it needs, and how a server communicates what it offers. For a client to discover a server, the client and server need to have a common language to describe what they need and what they offer. They need to use the same namespace of service types, otherwise they may actually speak the same application protocol over the air or on the wire, and may in fact be completely compatible, and yet may be unable to detect this because they are using different names to refer to the same actual service. Hence, having a consistent namespace of service types is the essential prerequisite for any useful service discovery.

IANA manages the registry of Service Types [RFC6335][STR]. This registry of Service Types can (and should) be used in any service discovery protocol as the vocabulary for describing *all* IP-based services, not only DNS-Based Service Discovery [RFC6763].

In this document we focus on the use of the IANA Service Type Registry [STR] in conjunction with DNS-Based Service Discovery, though that should not be taken in any way to imply any criticism of other service discovery protocols sharing the same namespace of service types. In different circumstances different Service Discovery protocols are appropriate.

For example, for service discovery of services potentially available via a Wi-Fi access point, prior to association with that Wi-Fi access point, when no IP communication has yet been established, a service discovery protocol may use raw 802.11 frames, not necessarily IP, UDP, or DNS-formatted messages. For Service Discovery using peer-to-peer Wi-Fi technologies, without any Wi-Fi access point at all, it may also be preferable to use raw 802.11 frames instead of IP, UDP, or DNS-formatted messages. Service Discovery using IEEE 802.15.4 radios may use yet another over-the-air protocol. What is important is that they all share the same vocabulary to describe all IP-based services. Using the same service type vocabulary means that client and server software, using agnostic APIs to consume and offer services on the network, has a common language to identify those services, independent of the medium or the particular service discovery protocol in use on that medium. Just as TCP/IP runs on many different link layers, and the concept of using an IP address to identify a particular peer is consistent across many different link layers, the concept of using a name from the IANA Service Type Registry to identify a particular service type also needs to be consistent across all IP-supporting link layers.

Originally, the IANA Service Type Registry [RFC6335][STR] used the term "Service Name" rather than "Service Type". Later it became clear that this term could be ambiguous. For a given service instance on the network, there is the machine-visible name of the type of service it provides, and the human-visible name of the particular instance of that type of service. For clarity, this document and related specifications use the term "Service Type" to denote the machine-visible name of the type of service, and the term "Instance Name" to denote the human-visible name of a particular instance.

3. Service Discovery Operational Model

The original DNS-Based Service Discovery specification [RFC6763] used the terms "register" (advertise a service), "browse" (discover service instances), and "resolve" (get IP address and port for a specific service instance). This terminology is reflective of the thinking at the time, which viewed service discovery as a new and separate step, added to existing networking code. For example, a server would first open a listening socket as it always had, and then "register" that listening socket with the service discovery engine. Similarly, a client would first "resolve" a service instance to an IP address and port, and then, having done that, "connect" to that IP address and port.

More recent thinking in this area [RFC8305] has come to the conclusion that it is preferable wherever possible to insulate application software from networking details like having to decide between IPv4 and IPv6, having to decide among multiple IP addresses of either or both address families, and having to decide among multiple available network interfaces. Consequently this document and related specifications adopt newer terminology as follows:

1. Offer
2. Enumerate
3. Use

The first step, "Offer", is when a server is offering a service using some application-layer protocol, on a listening TCP or UDP (or other transport protocol) port, and wishes to make that known to other devices. This encompasses both making a listening socket (or the equivalent concept in whatever underlying networking API is being used) and advertising the existence of that listening socket via a service discovery mechanism.

The second step, "Enumerate", is when a client device wishes to perform some action, but does not yet know which particular service instance will be used to perform that action. For example, when a user taps the "AirPrint" button on an iPhone or iPad, the iPhone or iPad knows that the user wishes to print, but not which particular printer to use. The desired *function* is known (IPP printing), but not the particular instance. In this case, the client device needs to enumerate the list of available service instances that are able to perform the desired task. In some cases this list of service instances is presented to a human user to choose from; in some cases it is software that examines the list of available service instances and determines the best one to use. This second step is the operation that was called "browsing" in the original specifications.

The third step, "Use", is when particular service instance has been selected, and the client wants to make use of that service instance. This encompasses both the "resolve" step (finding IP address(es) and port(s) for the service instance) and the subsequent steps to establish communication with it, which may include details like address family selection, interface selection, transport protocol selection, etc. Ideally, application-layer code should never be exposed to IP addresses at all, just as application-layer code today is generally not exposed to details like MAC addresses [RFC8305].

The second and third steps are intentionally separate. In the second step, a limited amount of information (typically just the name) is requested about a large number of service instances. In the third step more detailed information (e.g, target host IP address, port number, etc.) is requested about one specific service instance. Requesting all the detailed information about all available service instances would be inefficient and wasteful on the network. If the information about services on the network is imagined as a table, then the second step is requesting just one column from that table (the name column) and the third step is requesting just one row from that table (the information pertaining to just one named service instance).

To give a concrete example, clicking the "+" button in the printer settings on macOS is an operation performing the second step. It is requesting the names of all available printers. Depending on the specific use case, this step may be performed only rarely. For example, a user may do this just one once, the first time they configure their computer to use their preferred printer, and never again.

Once a desired printer has been chosen and configured, subsequent printing of documents is an operation performing the third step. This step may be done frequently, perhaps multiple times per day. This third step is important because, in a world of DHCP, IPv6 Stateless Autoconfiguration, and similar dynamic address allocation schemes, a printer's IP address could change from day to day, and to use the printer, its current address must be known. However, this third step need not be performed for every printer on the network, just the specific printer that is about to be used. Also, it is not necessary to repeat the second step again, learning the names of every printer on the network, if the client device already knows the name of the printer it intends to use.

DNS-Based Service Discovery [RFC6763] implements these three principal service discovery operations using DNS records and queries, either using Multicast DNS [RFC6762] (for queries limited to the

local link) or conventional unicast DNS [RFC1034] [RFC1035] (for queries beyond the local link).

Other service discovery protocol achieve the same semantics using different packet formats and mechanisms.

One incidental benefit of using DNS as the foundation layer for service discovery, in cases where that makes sense, is that both Multicast DNS and conventional unicast DNS are also used provide name resolution (mapping host names to IP addresses). There is some efficiency and code reuse gained by using the same underlying protocol for both service discovery and naming.

A final requirement is that the service discovery protocol should not only perform discovery at a single moment in time, but should also provide ongoing change notification (sometimes called "Publish & Subscribe"). Clients need to be notified in a timely fashion when new data of interest appears, when data of interest changes, and, equally importantly, when data of interest goes away ("goodbye packets"). Without support for ongoing change notification, clients would be forced to resort to polling to keep data up to date, which is inefficient and wasteful on the network.

Multicast DNS [RFC6762] implicitly includes change notification by virtue of announcing record creation, update, and deletion, via IP Multicast, which allows these changes to be seen by all peers on the same link (i.e., same broadcast domain).

Conventional unicast DNS [RFC1034] [RFC1035] has historically not had broad support for change notification. This capability is added via the new mechanism for DNS Push Notifications [Push].

When using DNS-Based Service Discovery [RFC6763] there are two aspects to consider: firstly how the clients determine the appropriate DNS names to query (and what query mechanisms to use) and secondly how the relevant information got into the DNS namespace in the first place, so as to be available when clients query for it.

The available namespaces are discussed broadly in Section 4 below. Client operation is then discussed in detail in Section 5, and server operation is discussed in detail in Section 6.

4. Service Discovery Namespace

When used with Multicast DNS [RFC6762] Service Discovery queries necessarily use the ".local" parent domain reserved for this purpose [SUDN].

When used with conventional unicast DNS [RFC1034] [RFC1035] some other domain must be used.

For individuals and organizations with a globally-unique domain name registered to them, their globally-unique domain name, or a subdomain of it, can be used for service discovery.

However, it would be convenient for advanced service discovery to be available even to people who haven't taken the step of registering and paying annually for a globally-unique domain name. For these people it would be useful if devices arrived preconfigured with some suitable factory-default service discovery domain, such as "services.home.arpa" [RFC8375]. Services published in this factory-default service discovery domain are not globally unique or globally resolvable, but they can have scope larger than the single link provided by Multicast DNS.

5. Client Configuration and Operation

When using DNS-Based Service Discovery [RFC6763], clients have to choose what DNS names to query.

When used with Multicast DNS [RFC6762] on the local link, queries are necessarily performed in the ".local" parent domain reserved for this purpose [SUDN].

For discovery beyond the local link, a unicast DNS domain must be used. This unicast DNS domain can be configured manually by the user, or it can be learned dynamically from the network (as has been done for many years at IETF meetings to facilitate discovery of the IETF Terminal Room printer, from outside the IETF Terminal Room). In the DNS-SD specification [RFC6763] section 11, "Discovery of Browsing and Registration Domains (Domain Enumeration)", describes how a client device learns one or more recommended service discovery domains from the network, using the special "lb._dns-sd._udp" query. All of the details from that specification are not repeated here. A walk-through describing one real-world example of how this works, using discovery of the IETF Terminal Room printer as a specific concrete case study, is given in Appendix A.

Given the service type that the user or client device is seeking (see Section 2) and one or more service discovery domains to look in, the client then sends its DNS queries, and processes the responses.

For some uses, one-shot conventional DNS queries and responses are perfectly adequate, but for service discovery, where a list may be displayed on a screen for a user to see, it is desirable to keep that list up to date without the user having to repeatedly tap a "refresh" button, and without the software repeatedly polling the network on the user's behalf.

And early solution to provide asynchronous change notifications for unicast DNS was the UDP-based protocol DNS Long-Lived Queries [DNS-LLQ]. This was used, among other things, by Apple's Back to My Mac Service [RFC6281] introduced in Mac OS X 10.5 Leopard in 2007.

A decade of operational experience has shown that an asynchronous change notification protocol built on TCP is preferable for a variety of reasons, so the IETF is has developed DNS Push Notifications [Push].

Because DNS Push Notifications is built on top of a DNS TCP connection, DNS Push Notifications adopts the conventions specified by DNS Stateful Operations [DSO] rather than inventing its own session management mechanisms.

6. Server Configuration and Operation

Section 5 above describes how clients perform their queries. The related question is how the relevant information got into the DNS namespace in the first place, so as to be available when clients query for it.

One trivial way that relevant service discovery information can get into the DNS namespace is simply via manual configuration, creating the necessary PTR, SRV and TXT records [RFC6763] by hand, and indeed this is how the IETF Terminal Room printer has been advertised to IETF meeting attendees for many years. While this is easy for the experienced network operators at the IETF, it can be onerous to others less familiar with how to set up DNS-SD records.

Hence it would be convenient to automate this process of populating the DNS namespace with relevant service discovery information. Two efforts are underway to address this need, the Service Discovery Proxy [DisProx] (see Section 6.1) and the Service Registration Protocol [RegProt] (see Section 6.4).

6.1. Service Discovery Proxy

The first technique in the direction of automatically populating the DNS namespace is the Service Discovery Proxy [DisProx]. This technology works with today's existing devices that advertise services using Multicast DNS only (such as almost all network printers sold in the last decade). A Service Discovery Proxy is a device with a presence on the same link as the devices we wish to be able to discover from afar. A remote client sends unicast queries to the Discovery Proxy, which performs local Multicast DNS queries on behalf of the remote client, and then sends back the answers it discovers.

Because the time it takes to receive Multicast DNS responses is uncertain, this mechanism benefits from being able to deliver asynchronous change notifications as new answers come in, using DNS Long-Lived Queries [DNS-LLQ] or the newer DNS Push Notifications [Push] on top of DNS Stateful Operations [DSO].

6.2. Multicast DNS Discovery Relay

As an alternative to having to be physically connected to the desired network link, a Service Discovery Proxy [DisProx] can use a Multicast DNS Discovery Relay [Relay] to give it a 'virtual' presence on a remote link. Indeed, when using Discovery Relays, a single Discovery Proxy can have a 'virtual' presence on hundreds of remote links. A single Discovery Proxy in the data center can serve the needs of an entire enterprise. This is modeled after the DHCP protocol. In simple residential scenarios the DHCP server resides in the home gateway, which is physically attached to the (single) local link. In complex enterprise networks, it is common to have a single centralized DHCP server, which resides in the data center and communicates with a multitude of simple lightweight BOOTP relay agents, implemented in the routers on each physical link.

6.3. Service Discovery Broker

Finally, when clients are communicating with multiple Service Discovery Proxies at the same time, this can be burdensome for the clients (which may be mobile and battery powered) and for the Service Discovery Proxies (which may have to serve hundreds of clients). This situation is remedied by use of a Service Discovery Broker [Broker]. A Service Discovery Broker is an intermediary between client and server. A client can issue a single query to the Service Discovery Broker and have the Service Discovery Broker do the hard work of issuing multiple queries on behalf of the client. And a Service Discovery Broker can shield a Service Discovery Proxy from excessive load by collapsing multiple duplicate queries from different client down to a single query to the Service Discovery Proxy.

6.4. Service Registration Protocol

The second technique in the direction of automatically populating the DNS namespace is the Service Registration Protocol [RegProt]. This technology is designed to enable future devices that will explicitly cooperate with the network infrastructure to advertise their services.

The Service Registration Protocol is effectively DNS Update, with some minor additions.

One addition to the basic DNS Update protocol is the introduction of a lifetime on DNS Updates, using the Dynamic DNS Update Lease EDNS(0) option [DNS-UL]. This option has similar semantics to a DHCP address lease, where a device is granted an address with with a certain DHCP lease lifetime, and if the device fails to renew the DHCP lease before it expires then the address will be reclaimed and become available to be allocated to a different device. In cases where DHCP is being used for address assignment, a device will generally request a DNS Update Lease with the same expiration time as its DHCP address lease. This way, if the device is abruptly disconnected from the network, around the same time as its address gets reclaimed its DNS records will also be garbage collected.

The second addition to the basic DNS Update protocol is the introduction of information, carried using the EDNS(0) OWNER Option [Owner], that tells the Service Registration server that the device will be going to sleep to save power, and how the Service Registration server can wake it up again on demand when needed. The use of power management information in the Service Registration messages allows devices to sleep to save power, which is especially beneficial for battery-powered devices in the home.

The use of an explicit Service Registration Protocol is beneficial in networks where multicast is expensive, inefficient, or outright blocked, such as many Wi-Fi networks. An explicit Service Registration Protocol is also beneficial in networks where multicast and broadcast are supported poorly, if at all, such as some mesh networks.

7. Security Considerations

As an informational document, this document introduces no new Security Considerations of its own. The various referenced documents each describe their own relevant Security Considerations as appropriate.

8. Informative 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>.

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

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[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>.

[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>.

[RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <https://www.rfc-editor.org/info/rfc8305>.

[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>.

[Broker] Cheshire, S. and T. Lemon, "Service Discovery Broker", drdraft-sctl-discovery-broker-00 (work in progress), July 2017.

[DisProx] Cheshire, S., "Discovery Proxy for Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-08 (work in progress), March 2018.

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

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

[DSO] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", draft-ietf-dnsop-session-signal-07 (work in progress), March 2018.

[Owner] Cheshire, S. and M. Krochmal, "EDNS0 OWNER Option", draft-cheshire-edns0-owner-option-01 (work in progress), July 2017.

[Push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-14 (work in progress), March 2018.

[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.

[STR] "Service Name and Transport Protocol Port Number Registry", <http://www.iana.org/assignments/service-names-port-numbers/>.

[SUDN] "Special-Use Domain Names Registry", <https://www.iana.org/assignments/special-use-domain-names/>.

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

Appendix A. IETF Terminal Room Printer Discovery Walk-Through

For about a decade now, the talented IETF network staff have provided off-link DNS Service Discovery for the Terminal Room printer at IETF meetings three times a year. In the case of the IETF meetings the necessary DNS records are entered manually, whereas this document advocates for increased automation of that task, but either way the process by which clients query to discover services is the same.

This appendix gives a detailed step-by step account of how this client query process works. It starts with a client joining the Wi-Fi network and doing a DHCP request, and ends with paper coming out of the printer. The reason the explanation is gives the specific details of every step is to avoid inadvertently having a hand-waving "and then a miracle occurs" part, which misses out some important detail. And one of the reasons for asking the IETF network team to set this up for IETF meetings is that operational use is an important reality check. When standing in front of a room, giving a presentation, if you miss out some vital step, people may not notice. When running an actual service used by actual people, if you miss out some vital step, no paper comes out of the printer, and everyone notices.

Using a macOS computer, at an IETF meeting, you can repeat the steps illustrated here to see exactly how it works. Or you can simply press Cmd-P in any application and see that "term-printer" appears as an available printer, to confirm that it does in fact work.

First, let's see what the macOS computer learned from the local DHCP server:

```
% scutil
> list
 ...
 subKey [74] = State:/Network/Service/21B5304C...54B28F4CA1D2/DHCP
 ...

> show State:/Network/Service/21B5304C...54B28F4CA1D2/DHCP
<dictionary> {
 Option_15 : <data> 0x6d656574696e672e696574662e6f7267
 ...
}
```

Option_15 is Domain Name. To see what domain name, we need to decode the hexadecimal data to ASCII.

```
% echo 6d656574696e672e696574662e6f7267 0A | xxd -r -p
meeting.ietf.org
```

A.1. Domain Enumeration using PTR queries

Our DHCP domain name is meeting.ietf.org. Does meeting.ietf.org recommend that we look in any Wide Area Service Discovery domains? This step is called Domain Enumeration [RFC6763], and is performed using a DNS PTR query for a name with the special prefix "lb._dns-sd._udp":

```
% dig lb._dns-sd._udp.meeting.ietf.org. ptr

; <<>> DiG 9.6-ESV-R4-P3 <<>> lb._dns-sd._udp.meeting.ietf.org. ptr
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35624
;; flags: qr aa rd ra;
                  QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;lb._dns-sd._udp.meeting.ietf.org. IN PTR

;; ANSWER SECTION:
lb._dns-sd._udp.meeting.ietf.org. 3600 IN PTR meeting.ietf.org.

...

;; Query time: 8 msec
;; SERVER: 130.129.5.6#53(130.129.5.6)
;; WHEN: Wed Mar 13 10:16:40 2013
;; MSG SIZE  rcvd: 188
```

In the middle there in the Answer Section you'll see that the answer to the PTR query is "meeting.ietf.org". In this case the answer is self-referential -- "meeting.ietf.org" is inviting us to look for services in "meeting.ietf.org", but the PTR record(s) could equally well point at any other domain, such as "services.ietf.org", or anything else.

Note that this answer does not depend on the client device being "on" the IETF meeting network, which is in any case a loosely defined concept at best. Nor does it depend on sending the DNS query to a DNS server that is "on" the IETF meeting network. Any capable DNS recursive resolver anywhere on the planet will give the same answer. We can test this by sending the same DNS PTR query to Google's 8.8.8.8 public resolver:

```
% dig @8.8.8.8 lb._dns-sd._udp.meeting.ietf.org. ptr

; <<>> DiG 9.6-ESV-R4-P3 <<>>
                    @8.8.8.8 lb._dns-sd._udp.meeting.ietf.org. ptr
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24571
;; flags: qr rd ra; QUERY:1, ANSWER:1, AUTHORITY:0, ADDITIONAL:0

;; QUESTION SECTION:
;lb._dns-sd._udp.meeting.ietf.org. IN PTR

;; ANSWER SECTION:
lb._dns-sd._udp.meeting.ietf.org. 1532 IN PTR meeting.ietf.org.

;; Query time: 21 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Wed Mar 13 10:18:27 2013
;; MSG SIZE  rcvd: 64
```

In the Answer Section you'll see that the answer is still "meeting.ietf.org".

In this example, this particular test was done at the 86th IETF in Orlando, Florida, in March 2013. The Google 8.8.8.8 public resolver still gave the correct answer, even though it was 13 hops away:

```
% traceroute -q 1 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 64 hops max, 52 byte packets
1  rtra (130.129.80.2)  1.369 ms
2  75-112-170-148.net.bhntampa.com (75.112.170.148)  14.494 ms
3  bun2.tamp20-car1.bhn.net (71.44.3.73)  19.558 ms
4  hun0-0-0-0-tamp20-cbr1.bhn.net (72.31.117.156)  20.730 ms
5  xe-8-2-0.bar1.tampa1.level3.net (4.53.172.9)  13.052 ms
6  ae-5-5.ebr1.miami1.level3.net (4.69.148.213)  27.413 ms
7  ae-1-51.edge1.miami2.level3.net (4.69.138.75)  15.552 ms
8  google-inc.edge1.miami2.level3.net (4.59.240.26)  48.852 ms
9  209.85.253.118 (209.85.253.118)  21.118 ms
10  216.239.48.192 (216.239.48.192)  21.890 ms
11  216.239.48.192 (216.239.48.192)  23.221 ms
12  *
13  google-public-dns-a.google.com (8.8.8.8)  32.961 ms
```

For the rest of this example we use the Google 8.8.8.8 public resolver for all the queries.

In the case of IETF meetings the PTR is self-referential -- meeting.ietf.org is advising us to look in meeting.ietf.org, but it could easily be set up to direct us elsewhere. However, since it's suggesting we look for services in meeting.ietf.org, we'll do that.

A.2. Instance Enumeration using PTR queries on a macOS computer

Once one or more service discovery domains have been determined, the client then looks for instances of the desired service type. This step is called Instance Enumeration and is also performed using a DNS PTR queries, using a name with a prefix indicating the type of service that is being sought.

A macOS computer with appropriate printer drivers installed will look for instances of the service type "_pdl-datastream._tcp" in the domain "meeting.ietf.org", as shown below. This is typically performed just once, the first time the macOS computer is set up to use that printer.

```
% dig +short @8.8.8.8 _pdl-datastream._tcp.meeting.ietf.org. ptr
term-printer._pdl-datastream._tcp.meeting.ietf.org.
```

There's one printing service available here, called "term-printer". That's what you see when you press the "+" button in the Print & Fax Preference Pane on macOS.

A.3. Printing from a macOS computer

When the user actually prints something, macOS sends a DNS SRV query for the printer name learned in the previous Instance Enumeration step, to learn the target host and port for the service. This DNS SRV query is then followed by address queries for the target host's IPv4 and/or IPv6 addresses. The necessary address records are usually included in the Additional Section of the reply to the SRV query, so that these address queries can be answered from the local cache, without resulting in additional packets over the air.

```
% dig +short @8.8.8.8 \
            term-printer._pdl-datastream._tcp.meeting.ietf.org. srv
0 0 9100 term-printer.meeting.ietf.org.

% dig +short @8.8.8.8 term-printer.meeting.ietf.org. AAAA
2001:df8::48:200:74ff:fee0:6cf8
```

This tells the computer that to use this printer, it must connect to [2001:df8::48:200:74ff:fee0:6cf8]:9100, using the installed printer driver, which speaks the appropriate vendor-specific printing protocol for that printer.

A.4. Instance Enumeration using PTR queries on an iOS device

Printing from an iPhone or iPad is similar, except there are no vendor-specific printer drivers installed. Instead, printing from an iPhone or iPad uses the IETF Standard IPP printing protocol, using an IPP printer that supports at least URF (Universal Raster Format). Consequently, the iOS device sends its Instance Enumeration DNS PTR queries using the prefix "_universal._sub._ipp._tcp" to indicate that it is looking for the subset of IPP printers that support Universal Raster Format.

```
% dig +short @8.8.8.8 \
                    _universal._sub._ipp._tcp.meeting.ietf.org. ptr
term-printer._ipp._tcp.meeting.ietf.org.
```

An iPhone or iPad will discover that there's one URF-capable IPP-based printing service available here, called "term-printer". It has the same name as the pdl-datastream printing service, and exists on the same physical hardware, but uses a different printing protocol.

A.5. Printing from an iOS device

When the user prints from their iPhone or iPad using AirPrint, iOS does these DNS SRV and address queries:

```
% dig +short @8.8.8.8 term-printer._ipp._tcp.meeting.ietf.org. srv
0 0 631 term-printer.meeting.ietf.org.
```

```
% dig +short @8.8.8.8 term-printer.meeting.ietf.org. aaaa
2001:df8::48:200:74ff:fee0:6cf8
```

Note that the "_ipp._tcp" service has the same target hostname and IPv6 address as the "_pdl-datastream" service from the macOS example, but is accessed at a different TCP port on that hardware device.

To use this printer, the iPhone or iPad connects to [2001:df8::48:200:74ff:fee0:6cf8]:631, and uses IPP to print.

Author's Address

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

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