Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: July 26, 2014

S. Cheshire
Apple Inc.

January 22, 2014

Hybrid Unicast/Multicast DNS-Based Service Discovery
draft-cheshire-dnssd-hybrid-01

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) of services that are outside the local link. Using a very large local link with thousands of hosts improves service discovery, but at the cost of large amounts of multicast traffic.

Performing DNS-Based Service Discovery using purely Unicast DNS is more efficient, but requires configuration of DNS Update keys on the devices offering the services, which can be onerous for simple devices like printers and network cameras.

Hence a compromise is needed, that provides easy service discovery without requiring either large amounts of multicast traffic or onerous configuration.

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].

Section 10 ("Populating the DNS with Information") of the DNS-SD specification [RFC6763] discusses possible ways that a service’s PTR, SRV, TXT and address records can make their way into the DNS namespace, including manual zone file configuration [RFC1034] [RFC1035], DNS Update [RFC2136] [RFC3007] and proxies.

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].

Multicast DNS works between a hosts on the same link. A set of hosts is considered to be "on the same link", if:

- 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

- 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 TTL or otherwise modifies the IP header.

3. Hybrid Proxy Operation

In its simplest form, each local link in an organization is assigned a unique Unicast DNS domain name, such as "Building 1.example.com." or "4th Floor.Building 1.example.com." (Grouping multiple local links under the same 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.)

Each link in an organization has a Hybrid Proxy which serves it. This 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, multiple links. In the organization’s DNS server, NS records are used to delegate ownership of each defined link name (e.g., "Building 1.example.com.") to the Hybrid Proxy which serves that link.

Domain Enumeration PTR records [RFC6763] are also created to inform clients of available Device 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.

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

When a DNS-SD client issues a Unicast DNS query to discover services in a particular Unicast DNS (e.g., "_printer._tcp.Building 1.example.com. PTR ?") the normal DNS delegation mechanism results in that query being served from 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 by performing a Multicast DNS query (e.g., "_printer._tcp.local. PTR ?") on its local link, and then, from the data in the Multicast DNS replies it receives, generating the corresponding Unicast DNS reply.

3.1. Data Translation

Generating the corresponding Unicast DNS reply involves, at the very least, rewriting the "local" suffix to the appropriate Unicast DNS domain (e.g., "Building 1.example.com").

In addition it would be desirable to suppress Unicast DNS replies for records that are not useful outside the local link. For example, DNS A and AAAA records for IPv4 link-local addresses [RFC3927] and IPv6 link-local addresses [RFC4862] 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 DNS names with DNS SRV records that reference target host names that have no addresses usable by the requester should be also be suppressed.

The same reachability requirement for advertised services also applies to the Hybrid Proxy itself. The mechanism specified in this document only works if the Hybrid Proxy is reachable from the client making the request.

3.1.1. 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. This information is a legacy from LPR printing, because LPR does not have in-band capability negotiation, so all of this information is put in 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 replies. Therefore, a Hybrid Proxy that is aware of the specifics of an application-layer protocol such as Apple’s AirPrint (which uses IPP) can elide unnecessary key/value pairs from the DNS-SD TXT record for better network efficiency.

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.

As in other similar situations, these layer violations optional. They are done only for efficiency reasons, and are not required for correct operation. A Hybrid Proxy can operate solely at the mDNS layer, without any knowledge of DNS-SD semantics, or of any DNS-SD client semantics.

3.2. Answer Aggregation

In a simple analysis, simply gathering multicast answers and forwarding them in a unicast reply 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 replies it needs to form a complete Unicast DNS reply. If it waits too little time, then it risks its Unicast DNS reply being incomplete. If it waits too long, then it creates a poor user experience at the client end.

This dilemma is solved by use of DNS Long-Lived Queries (DNS LLQ) [I-D.sekar-dns-llq]. The Hybrid Proxy replies immediately to the Unicast DNS query 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 replies be received, these are then delivered to the client using DNS LLQ update messages. The timeliness of such LLQ updates is limited only by the timeliness of the device responding to the Multicast DNS query. If the Multicast DNS device responds quickly, then the LLQ update is delivered quickly. If the Multicast DNS device responds slowly, then the LLQ update is delivered slowly. The benefit of using LLQ is that the Hybrid Proxy can respond promptly because it doesn’t have to delay its unicast reply to allow for the expected worst-case delay for receiving all the Multicast DNS replies. 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 replies. Using LLQs solves this dilemma: even very late replies are not lost; they are delivered in subsequent LLQ update messages.

There are two factors that determine specifically how replies are generated. The first factor is whether the Hybrid Proxy already has at least one record in its cache that positively answers the question. The second factor is whether the query from the client includes the LLQ option (typical with long-lived service browsing PTR queries) or not (typical with one-shot operations like SRV or address record queries).

- No answer in cache; no LLQ option: Do local mDNS query three times, and then return NXDOMAIN if no answer after three tries.

- No answer in cache; with LLQ option: As above, do local mDNS query three times, and then return NXDOMAIN if no answer after three tries. However, the query remains active for as long as the client maintains the LLQ state, and if mDNS answers are received later, LLQ update messages are sent. (Reasoning: We don’t need to rush to send an empty answer.)

- At least one answer in cache; no LLQ option: Send reply right away to minimise delay. No local mDNS queries are performed. (Reasoning: Given RRSet TTL harmonisation, if the proxy has one answer in its cache, it should have all of them.)

- At least one answer in cache; with LLQ option: As above, send reply right away to minimise delay. However, the query remains active for as long as the client maintains the LLQ state, and if additional mDNS answers are received later, LLQ update messages are sent. (Reasoning: We want UI that is displayed very rapidly, yet continues to remain accurate even as the network environment changes.)

4. 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.

4.1. Already Implemented and Deployed

Domain enumeration discovery 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 discovery 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 doing unicast DNS queries to the IETF DNS servers.

4.2. 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. It is possible that the IETF may choose to standardize a different or better Long-Lived Query mechanism. In that case, the pragmatic deployment approach would be 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 a new IETF Standard Long-Lived Query mechanism (as the future long-term direction).

4.3. Not Yet Implemented

The translating/filtering Hybrid Proxy specified in this document. Once implemented, such a Hybrid Proxy will immediately make wide-area discovery available with today’s existing clients and devices.

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.

5. 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, neither is aware of the other’s traffic. For this reason, each physical 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.

6. Security Considerations

A service proves its presence on a local 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].

7. 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].

8. IANA Considerations

This document has no IANA Considerations.

9. Acknowledgments

Thanks to Markus Stenberg for helping develop the policy regarding the four styles of unicast reply.

10. References

10.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

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

10.2. Informative References

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

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

[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