Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2015

A. Aggarwal
Qualcomm (QCE)

July 01, 2014

Optimizing DNS-SD query using TXT records
draft-aggarwal-dnssd-optimize-query-00

Abstract

DNS-SD allows a client to find a list of named instances of a service name over a particular transport within a domain of interest using standard DNS queries. As the number of potential responders increases, DNS-SD based discovery doesn't scale well. To mitigate the scaling issues, schemes to narrow down the search context would be needed. The document proposes to include key/value pairs in the form of a DNS TXT record along with the service name in the DNS query to assist with the discovery process. The DNS TXT record can be placed in the additional section of the query without requiring any changes to the structure of DNS messages.

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 http://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 January 2, 2015.

Table of Contents

1. Introduction

DNS-SD [RFC6763] in combination with mDNS [RFC6762] provide a discovery framework for service names registered with IANA over a local link. The objective of DNS-SD was to discover service instances that implement a given service. The use of mDNS scales well when the number of service instances that implement a given service are limited in number on the local link. However, when the number of wireless devices (e.g., Wi-Fi) approach hundreds of devices in a typical link, several service instances may respond when a DNS-SD query is issued for a given service name. The number of wireless devices is slated to grow further as more devices (things) are deployed as part of the Internet of Things (IoT) era.

At the same time, the DNS-SD protocol also enables discovery in various operating environments that rely on unicast DNS. Being able to narrow down the search context beyond the service name scope will be even more critical for such DNS-SD based discovery schemes to scale.

This contribution proposes one such solution.

This document proposes no change to the structure of DNS messages, and no new operation codes, response codes, resource record types, or any other new DNS protocol values.

1.1. Sample Use Cases

Some sample use cases that might experience scaling problems are mentioned below:

- A client application is looking to find color printers on the local network

- A lighting application needs to discover lighting fixtures or bulbs from a given manufacturer before establishing a session with each device to control the fixtures

2. Background

There are two potential mechanisms that can help a DNS-SD querier narrow down the answers of interest within the scope of DNS-SD [RFC6763]:

- Placing TXT records in the response: DNS has an efficiency feature whereby a DNS server may place additional records in the additional section of the DNS message. These additional records are records that the client did not explicitly request, but the server has reasonable grounds to expect that the client might request them shortly, so including them can save the client from having to issue additional queries. DNS-SD clarifies that the intention of DNS-SD TXT records is to convey a small amount of useful additional information about a service. Ideally, it should not be necessary for a client to retrieve this additional information before it can usefully establish a connection to the service. For a well-designed application protocol, even if there is no information at all in the TXT record, it should be possible, knowing only the host name, port number, and protocol being used, to communicate with that listening process and then perform version- or feature-negotiation to determine any further options or capabilities of the service instance.

- Using subtype as part of the question: DNS-SD allows a querier to send a subtype along with the service name. It does require that the subtype be 63 octets or fewer. DNS-SD RFC further clarifies that these should be documented in the protocol specification in question and/or in the "notes" field of the registration request sent to IANA.

It can be argued that mechanisms in place to narrow down the search beyond the service name are not very flexible. While nothing prevents an application implementing DNS-SD to eventually find the service instance of interest, it results in unnecessary traffic and delay. The proposal is to enable a richer search query mechanism by

explicitly adding key/value pairs in the query to avoid having to establish sessions with all services that match the service name in the question. Since DNS-SD allows a responder to include TXT records in the additional section with key-value pairs that it thinks the client may request, session establishment with the responder can be avoided if the desired key/value pairs (from the client's perspective) were included in the response.

3. Proposed Changes

DNS-SD as defined in [RFC6763] uses DNS TXT records to store arbitrary key/value pairs conveying additional information about the named service. Each key/value pair is encoded as its own constituent string within the DNS TXT record, in the form "key=value" (without the quotation marks). The proposal is for the client to be able to query for key/value pairs along with the service name. The DNS TXT record in the additional section of the query serves to send this additional information. Since DNS messages are allowed to have an additional section, this proposal doesn't require any changes to the structure of DNS messages.

DNS TXT record is allowed to have multiple key/value pairs. If multiple keys are present in a given TXT record, they are AND'ed and the responder must match all the keys in the TXT record. At the same time, DNS query could include more than one TXT record analogous to multiple TXT records in the response. If multiple TXT records are present in the query, they are logically OR'ed while the keys of each TXT record are AND'ed as stated above.

4. Realization of the proposal

Actual key/value pairs that can be sent are specified within the application protocol specification. Some examples to aid in the understanding of the proposal are mentioned below. They correspond to the use cases introduced earlier e.g.

- A client application looking for color printer can add color=true in the DNS TXT record as part of the additional section of the query

- A lighting application looking to discover bulbs by a certain manufacturer (such as Philips), can add the DNS TXT record in the additional section of the query with manuf=Philips

- The discovery scope can be further constrained by defining additional keys within the service protocol specification. By augmenting the query with additional context, the spurious traffic

and additional delay in finding the service instance of interest is reduced.

5. Deployment Considerations

An important deployment consideration is to analyze the behavior of an existing mDNS responder and unicast DNS to the receipt of DNS-SD query with a service name in the question section and TXT records in the additional section. If mDNS responder doesn't recognize TXT records, no filtering would occur and a response will be sent only if there is a match for the service name.

Regarding the behavior of unicast DNS when the standard query carries TXT records in the additional section, the DNS will respond strictly based on the service name in the question without any filtering based on the TXT records. DNS will issue a negative response unless there is a record matching the question. In summary, unicast DNS will continue to serve DNS queries that include TXT records in the additional section.

6. API Considerations

Several high level operating systems (Android, iOS) provide service discovery APIs. For the proposed enhancement to be realized, service registration should allow for service specific key/value pairs to be registered. This capability should already exist since it is allowed as per the current DNS-SD specification. The additional impact would be for the client application to be able to query for specific key/ value pairs along with the service name over a specific transport.

7. Security Considerations

The additional data (key/value pairs signifying the search context) beyond the service name in the DNS-SD query inherently reveals more information about what the client is searching for. DNS and mDNS today do not provide confidentiality, so observers already have access to potentially sensitive information such as what names one is requesting; addressing this issue is outside the scope of this extension. Even if confidentiality were to be solved, this extension still provides more information to the actual DNS/mDNS responders themselves. A client concerned about such information disclosure can simply choose not to use this extension for such queries, and thus trade off efficiency for privacy.

8. IANA Considerations

This memo includes no request to IANA.

9. Acknowledgements

Thanks to Dave Thaler for helping develop this idea and formalizing as a contribution for DNS-SD enhancements.

10. 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.

[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.

Author’s Address

Ashutosh Aggarwal
Qualcomm (QCE)
5775 Morehouse Dr
San Diego , California 92121
USA

Phone: +1 858 658 2229
Email: aggarwal@qce.qualcomm.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

S. Cheshire
Apple Inc.

October 19, 2015

Hybrid Unicast/Multicast DNS-Based Service Discovery
draft-ietf-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) 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 a compromise is needed, that combines the ease-of-use of Multicast DNS with the efficiency and scalability of Unicast DNS.

This Internet-Draft will expire on April 21, 2016.

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].

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:

- 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 IP TTL or otherwise modifies the IP header.

3. Hybrid Proxy Operation

In a typical configuration, a Hybrid Proxy is configured to be authoritative 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."

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

3.1. 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 "4th Floor.Building 1.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.

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.

3.2. 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 learned via DHCP) and using its IPv6 prefix and IPv4 subnet address. These are described below in Section 3.2.1. It also issues multicast Domain Enumeration queries in the "local" domain [RFC6762]. These are described below in Section 3.2.2. The results of all Domain Enumeration queries are combined for Service Discovery purposes.

3.2.1. 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, today). 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:
- Owner name, encoded as a two-byte compression pointer
- Two-byte rrtype (type PTR)
- Two-byte rrclass (class IN)
- Four-byte ttl
- Two-byte rdlength
- 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.

3.2.2. 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).

3.3. 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 are often 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 software, 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 not advisable (because of the space in "Building 1").

To accomodate this difference in allowable characters, a Hybrid Proxy MUST support having separate subdomains delegated to it, one to be used for host names (names of 'A' and 'AAAA' address records), which is restricted to the traditional letter-digit-hyphen rules, and another to be used for other records (including the PTR, SRV and TXT records used by DNS-SD), which is allowed to be arbitrary Net-Unicode text [RFC5198].

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").

3.4. 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.

3.5. 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.

3.5.1. 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 requests (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. For received Unicast DNS requests 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 3.6.

3.5.2. 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.

3.5.3. 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. 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. 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.

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 are 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 semantics at the DNS-SD layer or above.

3.6. 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 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 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]. When a Hybrid Proxy recieves 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/PUSH option are received directly from the client (see Section 3.6.1). Queries containing no LLQ/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.

- No LLQ/PUSH option; no answer in cache:
  Do local mDNS query up to three times, return answers if received, otherwise return negative response if no answer after three tries.
  DNS TTLs in responses are capped to at most ten seconds.

- No LLQ/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/PUSH option; no answer in cache:
  As above, do local mDNS query up to three times, and return answers if received.
  If no answer after three tries, return negative response.
  (Reasoning: We don't need to rush to send an empty answer.)
  In both cases the query remains active for as long as the client maintains the LLQ/PUSH state, and if mDNS answers are received later, LLQ/PUSH update messages are sent.
  DNS TTLs in responses are returned unmodified.

- Query contains LLQ/PUSH option; at least one answer in cache:
  As above, send response right away to minimise delay.
  The query remains active for as long as the client maintains the LLQ/PUSH state, and if additional mDNS answers are received later, LLQ/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.

3.6.1. Discovery of LLQ or PUSH Notification Service

To issue LLQ/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.

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 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 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. 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.

4.3. 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.

5. 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.

6. Security Considerations

6.1. 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].

6.2. 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.

6.3. 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.

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 response according to what data is immediately available in the cache. Thanks to Andrew Yourtchenko for comments about privacy issues. [Partial list; more names to be added.]

10. References

10.1. 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>.

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

[I-D.ietf-dnssd-push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-02 (work in progress), October 2015.

10.2. Informative References

[HOME] Cheshire, S., "Special Use Top Level Domain 'home'", draft-cheshire-homenet-dot-home (work in progress), November 2014.

[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., 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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

T. Pusateri
Seeking affiliation
S. Cheshire
Apple Inc.

October 19, 2015

DNS Push Notifications
draft-ietf-dnssd-push-02

Abstract

The Domain Name System (DNS) was designed to return matching records efficiently for queries for data that is relatively static. When those records change frequently, DNS is still efficient at returning the updated results when polled. But there exists no mechanism for a client to be asynchronously notified when these changes occur. This document defines a mechanism for a client to be notified of such changes to DNS records, called DNS Push Notifications.

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 http://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 21, 2016.

Copyright Notice

Table of Contents

1. Introduction

DNS records may be updated using DNS Update [RFC2136]. Other mechanisms such as a Hybrid Proxy [I-D.ietf-dnssd-hybrid] can also generate changes to a DNS zone. This document specifies a protocol for Unicast DNS clients to subscribe to receive asynchronous notifications of changes to RRSets of interest. It is immediately relevant in the case of DNS Service Discovery [RFC6763] but is not limited to that use case and provides a general DNS mechanism for DNS record change notifications. Familiarity with the DNS protocol and DNS packet formats is assumed [RFC1034] [RFC1035] [RFC6195].

1.1. Requirements Language

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].

2. Motivation

As the domain name system continues to adapt to new uses and changes in deployment, polling has the potential to burden DNS servers at many levels throughout the network. Other network protocols have successfully deployed a publish/subscribe model to state changes following the Observer design pattern. XMPP Publish-Subscribe [XEP-0060] and Atom [RFC4287] are examples. While DNS servers are generally highly tuned and capable of a high rate of query/response traffic, adding a publish/subscribe model for tracking changes to DNS records can result in more timely notification of changes with reduced CPU usage and lower network traffic.

Multicast DNS [RFC6762] implementations always listen on a well known link-local IP multicast group, and new services and updates are sent for all group members to receive. Therefore, Multicast DNS already has asynchronous change notification capability. However, when DNS Service Discovery [RFC6763] is used across a wide area network using Unicast DNS (possibly facilitated via a Hybrid Proxy [I-D.ietf-dnssd-hybrid]) it would be beneficial to have an equivalent capability for Unicast DNS, to allow clients to learn about DNS record changes in a timely manner without polling.

DNS Long-Lived Queries (LLQ) [I-D.sekar-dns-llq] is an existing deployed solution to provide asynchronous change notifications. Even though it can be used over TCP, LLQ is defined primarily as a UDP-based protocol, and as such it defines its own equivalents of existing TCP features like the three-way handshake. This document builds on experience gained with the LLQ protocol, with an improved design that uses long-lived TCP connections instead of UDP (and therefore doesn’t need to duplicate existing TCP functionality), and adopts the syntax and semantics of DNS Update messages [RFC2136] instead of inventing a new vocabulary of messages to communicate DNS zone changes.

Because DNS Push Notifications impose a certain load on the responding server (though less load that rapid polling of that server) DNS Push Notification clients SHOULD exercise restraint in issuing DNS Push Notification subscriptions. A subscription SHOULD only be active when there is a valid reason to need live data (for example, an on-screen display is currently showing the results of

that subscription to the user) and the subscription SHOULD be cancelled as soon as the need for that data ends (for example, when the user dismisses that display).

A DNS Push Notification client MUST not routinely keep a DNS Push Notification subscription active 24 hours a day 7 days a week just to keep a list in memory up to date so that it will be really fast if the user does choose to bring up an on-screen display of that data. DNS Push Notifications are designed to be fast enough that there is no need to pre-load a "warm" list in memory just in case it might be needed later.

3. Overview

The existing DNS Update protocol [RFC2136] provides a mechanism for clients to add or delete individual resource records (RRs) or entire resource record sets (RRSets) on the zone's server. Adopting this existing syntax and semantics for DNS Push Notifications allows for messages going in the other direction, from server to client, to communicate changes to a zone. The client first must subscribe for Push Notifications by connecting to the server and sending DNS message(s) indicating the RRSet(s) of interest. When the client loses interest in updates to these records, it unsubscribes.

The DNS Push Notification server for a zone is any server capable of generating the correct change notifications for a name. It may be a master, slave, or stealth name server [RFC1996]. Consequently, the "_dns-push-tls._tcp.<zone>" SRV record for a <zone> MAY reference the same target host and port as that zone's "_dns-update-tls._tcp.<zone>" SRV record. When the same target host and port is offered for both DNS Updates and DNS Push Notifications, a client MAY use a single TCP connection to that server for DNS Updates, DNS Queries, and DNS Push Notification Queries.

DNS Push Notification clients are NOT required to implement DNS Update Prerequisite processing. Prerequisites are used to perform tentative atomic test-and-set type operations on the server, and that concept has no application when it comes to an authoritative server informing a client of changes to DNS records.

4. Transport

Implementations of DNS Update [RFC2136] MAY use either User Datagram Protocol (UDP) [RFC0768] or Transmission Control Protocol (TCP) [RFC0793] as the transport protocol, in keeping with the historical precedent that DNS queries must first be sent over UDP [RFC1123]. This requirement to use UDP has subsequently been relaxed [RFC5966][I-D.ietf-dnsop-5966bis]. Following that precendent, DNS Push Notification is defined only for TCP. DNS Push Notification clients MUST use TLS over TCP.

Either end of the TCP connection can terminate all of the subscriptions on that connection by simply closing the connection abruptly with a TCP FIN or RST. (An individual subscription is terminated by sending an UNSUBSCRIBE message for that specific subscription.)

If a client closes the connection, it is signaling that it is no longer interested in receiving updates to any of the records it has subscribed. It is informing the server that the server may release all state information it has been keeping with regards to this client. This may occur because the client computer has been disconnected from the network, has gone to sleep, or the application requiring the records has terminated.

If a server closes the connection, it is informing the client that it can no longer provide updates for the subscribed records. This may occur because the server application software or operating system is restarting, the application terminated unexpectedly, the server is undergoing maintenance procedures, or the server is overloaded and can no longer provide the information to all the clients that wish to receive it. The client can try to re-subscribe at a later time or connect to another server supporting DNS Push Notifications for the zone.

Connection setup over TCP ensures return reachability and alleviates concerns of state overload at the server through anonymous subscriptions. All subscribers are guaranteed to be reachable by the server by virtue of the TCP three-way handshake. Because TCP SYN flooding attacks are possible with any protocol over TCP, implementers are encouraged to use industry best practices to guard against such attacks [IPJ.9-4-TCPSYN] [RFC4953].

Transport Layer Security (TLS) [RFC5246] is well understood and deployed across many protocols running over TCP. It is designed to prevent eavesdropping, tampering, or message forgery. TLS is REQUIRED for every connection between a client subscriber and server in this protocol specification. Additional security measures such as

client authentication during TLS negotiation MAY also be employed to increase the trust relationship between client and server. Additional authentication of the SRV target using DNSSEC verification and DANE TLSA records [RFC7673] is strongly encouraged. See below in Section 9.2 for details.

5. State Considerations

Each DNS Push Notification server is capable and handling some finite number of Push Notification subscriptions. This number will vary from server to server and is based on physical machine characteristics, network bandwidth, and operating system resource allocation. After a client establishes a connection to a DNS server, each record subscription is individually accepted or rejected. Servers may employ various techniques to limit subscriptions to a manageable level. Correspondingly, the client is free to establish simultaneous connections to alternate DNS servers that support DNS Push Notifications for the zone and distribute record subscriptions at its discretion. In this way, both clients and servers can react to resource constraints. Token bucket rate limiting schemes are also effective in providing fairness by a server across numerous client requests.

6. Protocol Operation

A DNS Push Notification exchange begins with the client discovering the appropriate server, and then making a TLS/TCP connection to it. The client may then add and remove Push Notification subscriptions over this connection. In accordance with the current set of active subscriptions the server sends relevant asynchronous Push Notifications to the client. The exchange terminates when either end closes the TCP connection with a TCP FIN or RST.

A client SHOULD NOT make multiple TLS/TCP connections to the same DNS Push Notification server. A client SHOULD share a single TLS/TCP connection for all requests to the same DNS Push Notification server. This shared connection should be used for all DNS Queries and DNS Push Notification Queries queries to that server, and for DNS Update requests too when the "_dns-update-tls._tcp.<zone>" SRV record indicates that the same server also handles DNS Update requests. This is to reduce unnecessary load on the DNS Push Notification server.

However, a single client device may be home to multiple independent client software instances that don’t know about each other, so a DNS Push Notification server MUST be prepared to accept multiple connections from the same client IP address. This is undesirable from an efficiency stanpoint, but may be unavoidable in some situations, so a DNS Push Notification server MUST be prepared to accept multiple connections from the same client IP address.

6.1. Discovery

The first step in DNS Push Notification subscription is to discover an appropriate DNS server that supports DNS Push Notifications for the desired zone. The client MUST also determine which TCP port on the server is listening for connections, which need not be (and often is not) the typical TCP port 53 used for conventional DNS.

1. The client begins the discovery by sending a DNS query to the local resolver with record type SOA [RFC1035] for the name of the record it wishes to subscribe.

2. If the SOA record exists, it MUST be returned in the Answer Section of the reply. If not, the server SHOULD include the SOA record for the zone of the requested name in the Authority Section.

3. If no SOA record is returned, the client then strips off the leading label from the requested name. If the resulting name has at least one label in it, the client sends a new SOA query and

processing continues at step 2 above. If the resulting name is empty (the root label) then this is a network configuration error and the client gives up. The client MAY retry the operation at a later time.

4. Once the SOA is known, the client sends a DNS query with type SRV [RFC2782] for the record name "_dns-push-tls._tcp.<zone>", where <zone> is the owner name of the discovered SOA record.

5. If the zone in question does not offer DNS Push Notifications then SRV record MUST NOT exist and the SRV query will return a negative answer.

6. If the zone in question is set up to offer DNS Push Notifications then this SRV record MUST exist. The SRV "target" contains the name of the server providing DNS Push Notifications for the zone. The port number on which to contact the server is in the SRV record "port" field. The address(es) of the target host MAY be included in the Additional Section, however, the address records SHOULD be authenticated before use as described below in Section 9.2 [RFC7673].

7. More than one SRV record may be returned. In this case, the "priority" and "weight" values in the returned SRV records are used to determine the order in which to contact the servers for subscription requests. As described in the SRV specification [RFC2782], the server with the lowest "priority" is first contacted. If more than one server has the same "priority", the "weight" is indicates the weighted probability that the client should contact that server. Higher weights have higher probabilities of being selected. If a server is not reachable or is not willing to accept a subscription request, then a subsequent server is to be contacted.

If a server closes a DNS Push Notification subscription connection, the client SHOULD repeat the discovery process in order to determine the preferred DNS server for subscriptions at that time.

6.2. DNS Push Notification SUBSCRIBE

A DNS Push Notification client indicates its desire to receive DNS Push Notifications for a given domain name by sending a SUBSCRIBE request over the established TCP connection to the server. A SUBSCRIBE request is formatted identically to a conventional DNS QUERY request [RFC1035], except that the opcode is SUBSCRIBE (6) instead of QUERY (0). If neither QTYPE nor QCLASS are ANY (255) then this is a specific subscription to changes for the given name, type and class. If one or both of QTYPE or QCLASS are ANY (255) then this subscription matches any type and/or any class, as appropriate.

In a SUBSCRIBE request the DNS Header QR bit MUST be zero.
If the QR bit is not zero the message is not a SUBSCRIBE request.

The AA, TC, RD, RA, Z, AD, and CD bits, the ID field, and the RCODE field, MUST be zero on transmission, and MUST be silently ignored on reception.

Like a DNS QUERY request, a SUBSCRIBE request MUST contain exactly one question. Since SUBSCRIBE requests are sent over TCP, multiple SUBSCRIBE requests can be concatenated in a single TCP stream and packed efficiently into TCP segments, so the ability to pack multiple SUBSCRIBE operations into a single DNS message within that TCP stream would add extra complexity for little benefit.

ANCOUNT MUST be zero, and the Answer Section MUST be empty.
Any records in the Answer Section MUST be silently ignored.

NSCOUNT MUST be zero, and the Authority Section MUST be empty.
Any records in the Authority Section MUST be silently ignored.

ARCOUNT MUST be zero, and the Additional Section MUST be empty.
Any records in the Additional Section MUST be silently ignored.

Each SUBSCRIBE request generates exactly one SUBSCRIBE response from the server.

In the SUBSCRIBE response the RCODE indicates whether or not the subscription was accepted. Supported RCODEs are as follows:

Mnemonic	Value	Description
NOERROR	0	SUBSCRIBE successful
FORMERR	1	Server failed to process request due to a malformed request
SERVFAIL	2	Server failed to process request due to resource exhaustion
NOTIMP	4	Server does not implement DNS Push Notifications
REFUSED	5	Server refuses to process request for policy or security reasons

Table 1: Response codes

In a SUBSCRIBE response the DNS Header QR bit MUST be one.
If the QR bit is not one the message is not a SUBSCRIBE response.

The AA, TC, RD, RA, Z, AD, and CD bits, and the ID field, MUST be zero on transmission, and MUST be silently ignored on reception.

The Question Section MUST echo back the values provided by the client in the SUBSCRIBE request that generated this SUBSCRIBE response.

ANCOUNT MUST be zero, and the Answer Section MUST be empty.
Any records in the Answer Section MUST be silently ignored.
If the subscription was accepted and there are positive answers for the requested name, type and class, then these positive answers MUST be communicated to the client in an immediately following Push Notification Update, not in the Answer Section of the SUBSCRIBE response. This simplifying requirement is made so that there is only a single way that information is communicated to a DNS Push Notification client. Since a DNS Push Notification client has to parse information received via Push Notification Updates anyway, it is simpler if it does not also have to parse information received via the Answer Section of a SUBSCRIBE response.

NSCOUNT MUST be zero, and the Authority Section MUST be empty.
Any records in the Authority Section MUST be silently ignored.

ARCOUNT MUST be zero, and the Additional Section MUST be empty.

Any records in the Additional Section MUST be silently ignored.

If accepted, the subscription will stay in effect until the client revokes the subscription or until the connection between the client and the server is closed.

SUBSCRIBE requests on a given connection MUST be unique. A client MUST NOT send a SUBSCRIBE message that duplicates the name, type and class of an existing active subscription on that TLS/TCP connection. For the purpose of this matching, the established DNS case-insensitivity for US-ASCII letters applies (e.g., "foo.com" and "Foo.com" are the same). If a server receives such a duplicate SUBSCRIBE message this is an error and the server MUST immediately close the TCP connection.

DNS wildcarding is not supported. That is, a wildcard ("*") in a SUBSCRIBE message matches only a wildcard ("*") in the zone, and nothing else.

Aliasing is not supported. That is, a CNAME in a SUBSCRIBE message matches only a CNAME in the zone, and nothing else.

A client may SUBSCRIBE to records that are unknown to the server at the time of the request and this is not an error. The server MUST accept these requests and send Push Notifications if and when matches are found in the future.

Since all SUBSCRIBE operations are implicitly long-lived operations, the server MUST interpret a SUBSCRIBE request as if it contained an EDNS0 TCP Keepalive option [I-D.wouters-edns-tcp-keepalive]. A client MUST NOT include an actual EDNS0 TCP Keepalive option in the request, since it is automatic, and implied by the semantics of SUBSCRIBE. If a server receives a SUBSCRIBE request this is an error and the server MUST immediately close the TCP connection. In a SUBSCRIBE response the server MUST include an EDNS0 TCP Keepalive option specifying the idle timeout so that the client knows the frequency of keepalives it must generate to keep the connection alive. If the client receives a SUBSCRIBE response that does not contain an EDNS0 TCP Keepalive option this is an error and the client MUST immediately close the TCP connection.

6.3. DNS Push Notification UNSUBSCRIBE

To cancel an individual subscription without closing the entire connection, the client sends an UNSUBSCRIBE message over the established TCP connection to the server. The UNSUBSCRIBE message is formatted identically to the SUBSCRIBE message which created the subscription, with the exact same name, type and class, except that the opcode is UNSUBSCRIBE (7) instead of SUBSCRIBE (6).

A client MUST NOT send an UNSUBSCRIBE message that does not exactly match the name, type and class of an existing active subscription on that TLS/TCP connection. If a server receives such an UNSUBSCRIBE message this is an error and the server MUST immediately close the connection.

No response message is generated as a result of processing an UNSUBSCRIBE message.

Having being successfully revoked with a correctly-formatted UNSUBSCRIBE message, the previously referenced subscription is no longer active and the server MAY discard the state associated with it immediately, or later, at the server’s discretion.

6.4. DNS Push Notification Update Messages

Once a subscription has been successfully established, the server generates Push Notification Updates to send to the client as appropriate. An initial Push Notification Update will be sent immediately in the case that the answer set was non-empty at the moment the subscription was established. Subsequent changes to the answer set are then communicated to the client in subsequent Push Notification Updates.

The format of Push Notification Updates borrows from the existing DNS Update [RFC2136] protocol, with some simplifications.

The following figure shows the existing DNS Update header format:

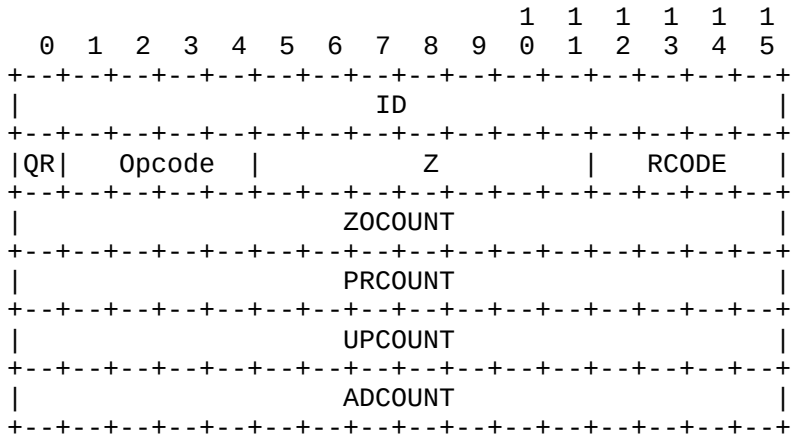

Figure 1

For DNS Push Notifications the following rules apply:

The QR bit MUST be zero, and the Opcode MUST be UPDATE (5). Messages received where this is not true are not Push Notification Update Messages and should be silently ignored for the purposes of Push Notification Update Message handling.

ID, the Z bits, and RCODE MUST be zero on transmission, and MUST be silently ignored on reception.

ZOCOUNT MUST be zero, and the Zone Section MUST be empty. Any records in the Zone Section MUST be silently ignored.

PRCOUNT MUST be zero, and the Prerequisite Section MUST be empty. Any records in the Prerequisite Section MUST be silently ignored.

ADCOUNT MUST be zero, and the Additional Data Section MUST be empty. Any records in the Additional Data Section MUST be silently ignored.

The Update Section contains the relevant change information for the client, formatted identically to a DNS Update [RFC2136]. To recap:

Delete all RRsets from a name:
TTL=0, CLASS=ANY, RDLENGTH=0, TYPE=ANY.

Delete an RRset from a name:
TTL=0, CLASS=ANY, RDLENGTH=0;
TYPE specifies the RRset being deleted.

Delete an individual RR from a name:
TTL=0, CLASS=NONE;
TYPE, RDLENGTH and RDATA specifies the RR being deleted.

Add an individual RR to a name:
TTL, CLASS, TYPE, RDLENGTH and RDATA specifies the RR being added.

Upon reception of a Push Notification Update Message, the client receiving the message MUST validate that the records being added or deleted correspond with at least one currently active subscription on that connection. Specifically, the record name MUST match the name given in the SUBSCRIBE request, subject to the usual established DNS case-insensitivity for US-ASCII letters. If the QTYPE was not ANY (255) then the TYPE of the record must match the QTYPE given in the SUBSCRIBE request. If the QCLASS was not ANY (255) then the CLASS of the record must match the QCLASS given in the SUBSCRIBE request. If a matching active subscription on that connection is not found, then that individual record addition/deletion is silently ignored. Processing of other additions and deletions in this message is not affected. The TCP connection is not closed. This is to allow for the race condition where a client sends an outbound UNSUBSCRIBE while inbound Push Notification Updates for that subscription from the server are still in flight.

In the case where a single change affects more than one active subscription, only one update is sent. For example, an update adding a given record may match both a SUBSCRIBE request with the same QTYPE and a different SUBSCRIBE request with QTYPE=ANY. It is not the case that two updates are sent because the new record matches two active subscriptions.

The server SHOULD encode change notifications in the most efficient manner possible. For example, when three AAAA records are deleted from a given name, and no other AAAA records exist for that name, the server SHOULD send a "delete an RRset from a name" update, not three

separate "delete an individual RR from a name" updates. Similarly, when both an SRV and a TXT record are deleted from a given name, and no other records of any kind exist for that name, the server SHOULD send a "delete all RRsets from a name" update, not two separate "delete an RRset from a name" updates.

All Push Notification Update Messages MUST contain an EDNS0 TCP Keepalive option [I-D.wouters-edns-tcp-keepalive] specifying the idle timeout so that the client knows the frequency of keepalives it must generate to keep the connection alive. If the client receives a Push Notification Update Message that does not contain an EDNS0 TCP Keepalive option this is an error and the client MUST immediately close the TCP connection.

Reception of a Push Notification Update Message results in no response back to the server.

The TTL of an added record is stored by the client and decremented as time passes, with the caveat that for as long as a relevant subscription is active, the TTL does not decrement below 1 second. For as long as a relevant subscription remains active, the client SHOULD assume that when a record goes away the server will notify it of that fact. Consequently, a client does not have to poll to verify that the record is still there. Once a subscription is cancelled (individually, or as a result of the TCP connection being closed) record aging resumes and records are removed from the local cache when their TTL reaches zero.

6.5. DNS RECONFIRM

Sometimes, particularly when used with a Hybrid Proxy [I-D.ietf-dnssd-hybrid], a DNS Zone may contain stale data. When a client encounters data that it believe may be stale (e.g., an SRV record referencing a target host+port that is not responding to connection requests) the client sends a DNS RECONFIRM message to request that the server re-verify that the data is still valid. For a Hybrid Proxy, this causes it to issue new Multicast DNS requests to ascertain whether the target device is still present. For other kinds of DNS server the RECONFIRM operation is currently undefined and should be sliently ignored. A RECONFIRM request is formatted similarly to a conventional DNS QUERY request [RFC1035], except that the opcode is RECONFIRM (8) instead of QUERY (0). QTYPE MUST NOT be the value ANY (255). QCLASS MUST NOT be the value ANY (255).

In a RECONFIRM request the DNS Header QR bit MUST be zero. If the QR bit is not zero the message is not a RECONFIRM request.

The AA, TC, RD, RA, Z, AD, and CD bits, the ID field, and the RCODE field, MUST be zero on transmission, and MUST be silently ignored on reception.

Like a DNS QUERY request, a RECONFIRM request MUST contain exactly one question. Since RECONFIRM requests are sent over TCP, multiple RECONFIRM requests can be concatenated in a single TCP stream and packed efficiently into TCP segments, so the ability to pack multiple RECONFIRM operations into a single DNS message within that TCP stream would add extra complexity for little benefit.

ANCOUNT MUST be nonzero, and the Answer Section MUST contain the rdata for the record(s) that the client believes to be in doubt.

NSCOUNT MUST be zero, and the Authority Section MUST be empty. Any records in the Authority Section MUST be silently ignored.

ARCOUNT MUST be zero, and the Additional Section MUST be empty. Any records in the Additional Section MUST be silently ignored.

DNS wildcarding is not supported. That is, a wildcard ("*") in a SUBSCRIBE message matches only a wildcard ("*") in the zone, and nothing else.

Aliasing is not supported. That is, a CNAME in a SUBSCRIBE message matches only a CNAME in the zone, and nothing else.

No response message is generated as a result of processing a RECONFIRM message.

If the server receiving the RECONFIRM request determines that the records are in fact no longer valid, then subsequent DNS Push Notification Update Messages will be generated to inform interested clients. Thus, one client discovering that a previously-advertised printer is no longer present has the side effect of informing all other interested clients that the printer in question is now gone.

6.6. DNS Push Notification Termination Message

If a server is low on resources it MAY simply terminate a client connection with a TCP RST. However, the likely behavour of the client may be simply to reconnect immediately, putting more burden on the server. Therefore, a server MAY instead choose to shed client load by (a) sending a DNS Push Notification Termination Message and then (b) closing the client connection with a TCP FIN instead of RST, thereby facilitating reliable delivery of the Termination Message.

The format of a Termination Message is similar to a Push Notification Update.

The following figure shows the existing DNS Update header format:

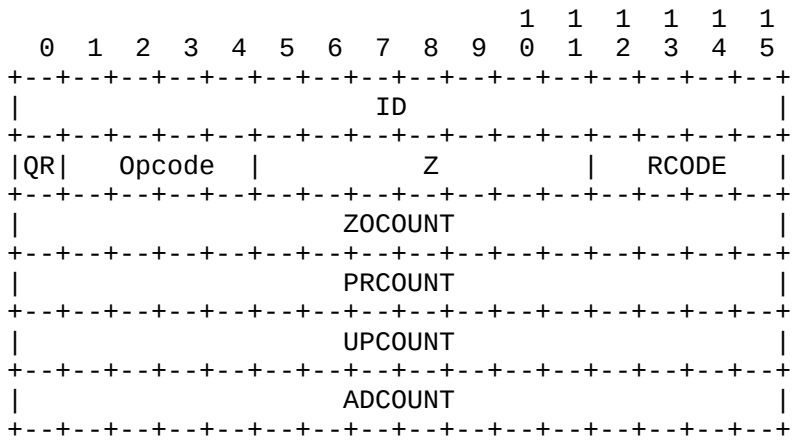

Figure 2

For Termination Messages the following rules apply:

The QR bit MUST be zero, and the Opcode MUST be UPDATE (5). Messages received where this is not true are not Termination Messages and should be silently ignored.

ID and the Z bits MUST be zero on transmission, and MUST be silently ignored on reception.

ZOCOUNT MUST be zero, and the Zone Section MUST be empty. Any records in the Zone Section MUST be silently ignored.

PRCOUNT MUST be zero, and the Prerequisite Section MUST be empty. Any records in the Prerequisite Section MUST be silently ignored.

UPCOUNT MUST be zero, and the Update Section MUST be empty.
Any records in the Update Section MUST be silently ignored.

ADCOUNT MUST be zero, and the Additional Data Section MUST be empty.
Any records in the Additional Data Section MUST be silently ignored.

The RCODE MUST contain a code giving the reason for termination.
[Codes to be determined.] The Termination Message MUST contain an EDNS0 TCP Keepalive option [I-D.wouters-edns-tcp-keepalive] where the idle timeout indicates the time the client SHOULD wait before attempting to reconnect.

7. Acknowledgements

The authors would like to thank Kiren Sekar and Marc Krochmal for previous work completed in this field. This draft has been improved due to comments from Ran Atkinson.

8. IANA Considerations

This document defines the service name: "_dns-push-tls._tcp".
It is only applicable for the TCP protocol.
This name is to be published in the IANA Service Name Registry.

This document defines two DNS OpCodes: SUBSCRIBE with (tentative) value 6 and UNSUBSCRIBE with (tentative) value 7.

9. Security Considerations

TLS support is mandatory in DNS Push Notifications. There is no provision for opportunistic encryption using a mechanism like "STARTTLS".

9.1. Security Services

It is the goal of using TLS to provide the following security services:

Confidentiality All application-layer communication is encrypted with the goal that no party should be able to decrypt it except the intended receiver.

Data integrity protection Any changes made to the communication in transit are detectable by the receiver.

Authentication An end-point of the TLS communication is authenticated as the intended entity to communicate with.

Deployment recommendations on the appropriate key lengths and cypher suites are beyond the scope of this document. Please refer to TLS Recommendations [RFC7525] for the best current practices. Keep in mind that best practices only exist for a snapshot in time and recommendations will continue to change. Updated versions or errata may exist for these recommendations.

9.2. TLS Name Authentication

As described in Section 6.1, the client discovers the DNS Push Notification server using an SRV lookup for the record name "_dns-push-tls._tcp.<zone>". The server connection endpoint SHOULD then be authenticated using DANE TLSA records for the associated SRV record. This associates the target's name and port number with a trusted TLS certificate [RFC7673]. This procedure uses the TLS Sever Name Indication (SNI) extension [RFC6066] to inform the server of the name the client has authenticated through the use of TLSA records. Therefore, if the SRV record passes DNSSEC validation and a TLSA record matching the target name is useable, an SNI extension MUST be used for the target name to ensure the client is connecting to the server it has authenticated. If the target name does not have a usable TLSA record, then the use of the SNI extension is optional.

9.3. TLS Compression

In order to reduce the chances of compression related attacks, TLS-level compression SHOULD be disabled when using TLS versions 1.2 and earlier. In the draft version of TLS 1.3 [I-D.ietf-tls-tls13], TLS-level compression has been removed completely.

9.4. TLS Session Resumption

TLS Session Resumption is permissible on DNS Push Notification servers. The server may keep TLS state with Session IDs [RFC5246] or operate in stateless mode by sending a Session Ticket [RFC5077] to the client for it to store. However, once the connection is closed, any existing subscriptions will be dropped. When the TLS session is resumed, the DNS Push Notification server will not have any subscription state and will proceed as with any other new connection.

10. References

10.1. Normative References

[I-D.ietf-dnsop-5966bis]
Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", draft-ietf-dnsop-5966bis-03 (work in progress), September 2015.

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-09 (work in progress), October 2015.

[I-D.wouters-edns-tcp-keepalive]
Wouters, P. and J. Abley, "The edns-tcp-keepalive EDNS0 Option", draft-wouters-edns-tcp-keepalive-01 (work in progress), February 2014.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[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>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <http://www.rfc-editor.org/info/rfc1123>.

[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>.

[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>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <http://www.rfc-editor.org/info/rfc4953>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, DOI 10.17487/RFC5966, August 2010, <http://www.rfc-editor.org/info/rfc5966>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <http://www.rfc-editor.org/info/rfc6066>.

[RFC6195] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", RFC 6195, DOI 10.17487/RFC6195, March 2011, <http://www.rfc-editor.org/info/rfc6195>.

[RFC7673] Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-Based Authentication of Named Entities (DANE) TLSA Records with SRV Records", RFC 7673, DOI 10.17487/RFC7673, October 2015, <http://www.rfc-editor.org/info/rfc7673>.

10.2. Informative References

[I-D.ietf-dnssd-hybrid] Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-01 (work in progress), October 2015.

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

[IPJ.9-4-TCPSYN]
Eddy, W., "Defenses Against TCP SYN Flooding Attacks", The Internet Protocol Journal, Cisco Systems, Volume 9, Number 4, December 2006.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <http://www.rfc-editor.org/info/rfc1996>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <http://www.rfc-editor.org/info/rfc4287>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <http://www.rfc-editor.org/info/rfc5077>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <http://www.rfc-editor.org/info/rfc6763>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[XEP-0060]
Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe", XSF XEP 0060, July 2010.

Authors' Addresses

Tom Pusateri
Seeking affiliation
Hilton Head Island, SC
USA

Phone: +1 843 473 7394
Email: pusateri@bangj.com

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

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

dnssd
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

D. Otis
Trend Micro
H. Rafiee
Rozanak.com

October 19, 2015

Scalable DNS-SD (SSD) Threats
draft-otis-dnssd-scalable-dns-sd-threats-02

Abstract

mDNS combined with Service Discovery (DNS-SD) extends network resource distribution beyond the reach of multicast normally limited by the MAC Bridge. Since related resources are often not authenticated, either local resources are inherently trustworthy or are subsequently verified by associated services. Resource distribution becomes complex when a hybrid scheme combines adjacent network resources into a common unicast DNS-SD structure. This document explores related security considerations.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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 http://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 21, 2016.

Table of Contents

1. Introduction

As described by [IEEE.802-1D.2004], MAC entities normally make services known via multicast announcements that do not extend beyond the Bridge as a basis for networking and layer 3 protocols. mDNS [RFC6762] allows non-centralized resource collection that can be structured as defined in DNS-SD [RFC6763]. This structure, when used in conjunction with DNS [RFC1035], provides an alternative to multicast announcement to deal with wireless links that are orders of magnitude less reliable than their wired counterparts. To improve transmission reliability, [IEEE.802-11.2012] requires positive

acknowledgement of unicast frames but does not support positive acknowledgement of multicast frames. In [IEEE.802-11.2012] wireless networks, multicast frames are transmitted at a lower data rate supported by all receivers. Multicast on wireless networks may thereby lower overall network throughput. Some network administrators block some multicast traffic or convert it to a series of link-layer unicast frames. Other types of wireless networks may impose more demanding limitations as described by [RFC4944]. As a result, it is common to observe much higher loss of multicast frames on wireless compared against wired network technologies.

A namespace structured from adjacent networks using proxy-ed mDNS resources lacks a means to quickly resolve unicast name collision. Although an expensive promiscuous mode of unicast operation at multicast destinations might replicate mDNS features within a unicast environment, not well covered in [RFC4903] are issues related to wireless upstream clients unable to operate in promiscuous mode, indeterminate latency, and PPP links requiring a NAT or IPv4 ARP proxy. As such, a non-hybrid multicast/unicast scheme would be problematic.

Scalable DNS-SD (SSD) proposes to automatically gather autonomously named mDNS [RFC6762] resources of adjacent networks within separate namespace zones or realms as defined by [RFC7368]. Realms are often contained in separate subdomains that correspond with a link-local namespace. Making routable resources visible and accessible from other networks via unicast DNS [RFC1035] structured per DNS-SD [RFC6763] mitigates the level of multicast mDNS traffic in larger networks. Reliance on DNS [RFC1035] might leverage multi-network configurations that use mDNS [RFC6762] that proxy mDNS resources into DNS-SD using [I-D.ietf-dnssd-hybrid].

1.1. Terminology and Abbreviations

o Border: A point, typically resident on a router, between two networks at which filtering and forwarding policies for different types of traffic may be applied.

o ISP: Internet Service Provider. An entity that provides access to the Internet. In this document, a service provider specifically offers Internet access using IPv6 and may also offer IPv4 Internet access. The service provider can provide such access over a variety of different transport methods such as DSL, cable, wireless, and others.

o Realm: A network delimited by a defined border. i.e. a guest network within a homenet may form a realm.

o ULA: IPv6 Unique Local Address [RFC4193].

o Global Namespace: A globally unique namespace accessible for resolution within the root domain.

o Realm Namespace: A realm specific namespace accessible for resolution referenced from a subdomain that may not be within the root domain.

o Local Namespace: A namespace accessible for link-local resolution that may be referenced from an Ambiguous Local Qualified Domain Name (ALQDN) representing a network segment or broadcast domain.

2. Scalable DNS-SD (SSD) Realm and Global Namespace

2.1. Realm and Global Names

Conflicts between realm and global DNS [RFC1035] namespaces may occur. Without adequate feedback and latency constraints, a client may be unable to determine desired service targets. Target assessment may impair network stability when a cache policy renames resources propagated into different realms. Determining actual conflicts might depend on inherent identifiers such as MAC addresses or device specific GUIDs, otherwise conflict resolution may become increasingly byzantine.

2.1.1. SSD Structures

SSD locates SRV and TXT RRsets resources in the forms:

_<sn>._<Proto>.<SrvDOM>.<ParentDOM>.

<Instance>._<sn>._<Proto>.<SrvDOM>.<ParentDOM>.

<sub>._sub._<sn>._<Proto>.<SrvDOM>.<ParentDOM>.

For DNS-SD, Proto="_udp" represents all non-TCP transports otherwise it is "_tcp".

_<sn> = IANA Registered Service Name

To facilitate browsing, DNS-SD also supports a DNS meta-query of PTR RRsets at "_services._dns-sd._udp.<Domain>" which yields service names which may vary by host along with a domain name. Only the first two labels in the PTR rdata are relevant in the construction of subsequent Service Instance Enumeration PTR queries to further discover specific service types.

[I-D.ietf-dnssd-hybrid] conveyance extends '.local.' TLD namespace into '.home.' or an Ambiguous Local Qualified Domain Name (ALQDN) space, such as '.sitelocal.' as described in Section 3.7.4 of [RFC7368] where DNS [RFC1035] can be facilitated using split horizon methods described by [RFC6950] or similar schemes described by [RFC6281]. The scheme supporting DNS should ensure queries against a sitelocal namespace is not forwarded to the Internet and to global root servers.

[I-D.ietf-dnssd-hybrid] suggests a split of traditional namespace that is restricted to letters, digits and hyphens and resolves only address resources, from the rich text namespace resolving PTR, SRV and TXT that facilitate service browsing. These resources are further bifurcated into separate link related namespace resources.

2.1.2. Scope of Discovery

As mDNS [RFC6762] is currently restricted to a single link, the scope of the advertisement is limited, by design, to the shared link between client and the device offering a service. When scaling for multi-links, the owner of the advertised service may propagate to a larger set of links or a larger realm than expected, which may result in unauthorized clients (from the perspective of the owner) connecting to the advertised service. It also discloses information (about the host and service) to a larger set of potential attackers.

If the scope of the discovery is not properly constrained, then information leaks may happen beyond the appropriate network and expose the network to various forms of attack. As such, services normally limited to local link should be assigned a separate subdomain normally not accessible from the Internet.

To reduce the amount of multicast traffic, widely distributing mDNS resources using unicast DNS-SD may scale better, but exposure of mDNS [RFC6762] derived resources to the Internet along with possibly sensitive details has proven problematic as noted by [CERTvu550620]. Protocol vulnerabilities can be found in reports published by a large number of vendors, Computer Emergency Response Teams (CERT), and Computer Security Incident Response Teams (CSIRT). With this diversity of sources, specific concerns may not be captured by Request for Comments (RFC) publications of the Internet Engineering Task Force (IETF).

Services might be sought outside the ".local." domain when applications obtain domain search lists provided by DHCP ([RFC2131] and [RFC3315] for IPv4 and IPv6 respectively or RA DNSSL [RFC6106] also for IPv6. Internet domains need to be published in DNS [RFC1035] as A-Labels [RFC3492] because IDNA2008 compliance depends on A-label enforcement by registrars. Therefore A-Labels and not U-Labels are published in DNS for Internet domains at this time.

The SRV scheme used by mDNS [RFC6762] has also been widely adopted in the Windows OS since it offered a functional replacement for Windows Internet Name Service (WINS) as their initial attempt lacked sufficient name hierarchy. Such common use may represent security considerations whenever these records might become automatically published.

2.1.2.1. Visual Spoofing

Visual selection of autonomously named resources becomes especially salient when names are not ensured to be uniquely represented. mDNS [RFC6762] only requires compliance with [RFC5198] rather than IDNA2008 [RFC5895]. This less restrictive use of namespace may impair the defense of critical services from look-alike attack. mDNS [RFC6762] does not ensure instances are visually unique and allows spaces and punctuation not permitted by IDNA2008.

To better ensure local namespace can be recognized, alternative zones might replace ASCII punctuation and spaces in SrvDOM labels with the '_' character except when located as the leftmost character. Such a convention should reduce visual confusion and handling issues related to end of string parsing, since labels in DNS [RFC1035] normally do

not contain spaces or punctuation. Nevertheless, DNS [RFC1035] is able to handle such labels within sub-domains of registered domains.

2.1.3. Restricted Distribution of Sitelocal Addresses

ULA or [RFC1918] addresses allow safer automatic publication in DNS since these addresses are unlikely to be routed beyond the site. These addresses also provide a simple scheme to ascertain which addresses should be blocked at a network boundary. The use of other addresses MUST require specific administrative confirmations. It should be noted in the Addendum example, the Brother printer published a Globally routable address.

When doing so, address translation or overlays using Unique Local Addresses, ULAs [RFC4193] can offer a significant level of protection since typical link-local addresses are not usable from other networks. Although ULAs are to be treated as being globally routable, both ULA or [RFC1918] addresses typically indicate site local. Section 3.2 of [RFC4193] are locally defined and handled as Global addresses although not intended to be routed beyond the site or to those not having explicit routing provisions.

Section 4.1 of [RFC4193] indicates the default behavior of exterior routing protocol sessions between administrative routing regions must be to ignore receipt of and not advertise prefixes in the FC00::/7 block. A network operator may specifically configure prefixes longer than FC00::/7 for inter-site communication. Specifically, these prefixes are not designed to aggregate. Routers by default do not block ULA prefixes which makes it important to confirm how ULA traffic is handled by the access provider.

ULA or [RFC1918] addresses are not normally routed over the Internet where their use provides a degree of isolation. For either home or enterprise networks, ULAs as an overlay network avoids network address translations and permits local routing isolated from direct Internet access. ULAs also permit local communications to remain unaffected by Internet related link failures or scope limitations imposed by use of multicast protocols.

ULAs avoid a need to renumber internal-only private nodes when changing ISPs, or when ISPs restructure their address allocations. In these situations, use of ULA offers an effective tool for protecting internal-only nodes. As such, more than just the security considerations discussed in mDNS [RFC6762] and DNS-SD [RFC6763] are needed. For example, DNS-SD [RFC6763] states the following: "Since DNS-SD is just a specification for how to name and use records in the existing DNS, it has no specific additional security requirements over and above those that already apply to DNS queries and DNS

updates." This simply overlooks that many devices are not automatically published in DNS nor can it be assumed they are able to handle the access that DNS might permit.

Current BTMM [RFC6281] only publishes ULAs of hosts in DNS able to authenticate when setting up an overlay network. Remaining devices, such as printers, are accessed as services offered by authenticating hosts. DNS resources should never be considered to offer privacy even in split-horizon configurations. DNS is unable to authenticate incoming queries nor can it offer application layer protection. Since many prefixes are expected to be in use within environments served by [I-D.ietf-dnssd-hybrid], errors related to network boundary detections becomes critical. As such, DNS SHOULD NOT publish addresses of devices unable to authenticate sessions traversing the Internet.

2.1.4. Confirming Valid Resources

[RFC6950] Source Address Validation Improvement (SAVI) for DHCP as specified by [RFC7513] may help administrators qualify resources published in DNS. DNS-SD [RFC6763] recommends additional DNS records such as associated PTR and TXT SHOULD be generated to improve network efficiency for both unicast and multicast DNS-SD responses. This behavior further increases some risks related to query/response ratios and the likelihood of exposure of security sensitive information.

This new routable namespace also lacks the benefit of registrar involvement and may not afford an administrator an ability to mitigate nefarious activity, such as spoofing and phishing, without requisite controls having been first carefully established. When a device has access to different realms on multiple interfaces, it is not even clear how simple conflict resolution avoids threatening network stability while resolving names conveyed over disparate technologies.

2.1.5. Selective Forwarding based on IGMP or MLD snooping

Internet Group Management Protocol (IGMP) [RFC3376] supports multicast on IPv4 networks. Multicast Listener Discovery (MLD) [RFC3810] supports multicast management on IPv6 networks using ICMPv6 messaging in contrast to IGMP’s bare IP encapsulation. This management allows routers to announce their multicast membership to neighboring routers. To optimize which LANs receive forwarded multicast frames, IGMP or MLD snooping can be used to determine the presence of listeners as a means to permit selective forwarding of multicast frames as well.

2.1.6. VLAN

Use of VLAN such as [RFC5517] can selectively extend multicast forwarding beyond Bridge limitations. While not a general solution, use of VLAN can both isolate and unite specific networks.

2.1.7. DHCP

IP address assignment and host registration might use a single or forwarded DHCP [RFC2131] or [RFC3315] server for IPv4 and IPv6 respectively that responds to interconnected networks as a means to register hosts and addresses. DHCP does not ensure against name or address conflict nor is it intended to configure routers.

2.2. Exfiltration and Poisoning

IP addresses made visible by DNSSEC [RFC4033] or DNS [RFC1035] that conform with DNS-SD [RFC6763] might be used, but the automated population of information into DNS [RFC1035] should be limited to administrative systems.

Automated conversion of mDNS [RFC6762] into unicast DNS [RFC1035] can be problematic from a security standpoint as can widespread propagation of multicast frames. mDNS [RFC6762] only requires compliance with [RFC5198] rather than IDNA2008 [RFC5895]. This means mDNS [RFC6762] will not ensure instances are visually unique and may contain spaces and punctuation not permitted by IDNA2008. As such, this might cause users into becoming misled about the associated service.

SSD MUST include requisite filtering necessary to prevent data ex-filtration or the interception of sensitive services. Any exchanged data must first ensure locality, limit the resources gathered, resolved, and propagated to just those elements that can be effectively administrated. It is critical to ensure normal network protection is not lost for hosts that depend on link-local addressing and exclusion of routable traffic. A printer would be one such example of a host that can not be upgraded.

2.3. Amplification Concerns

It is unknown whether sufficient filtering of mDNS [RFC6762] to expose just those services likely needed will provide sufficient network protection. The extent of using IGMP or MLD for selective forwarding to mitigate otherwise spurious traffic is unknown.

Instance names and <SrvDOM> intended to correspond with link-local domains may use Unicode for Network Interchange [RFC5198] encoding

but excludes ASCII control characters while also allowing escaped periods "\." and other punctuation and spaces.

For DNS-SD, Proto="_udp" represents all non-TCP transports otherwise it is "_tcp".

_<sn> = IANA Registered Service Name

Optional service browsing and various RRsets could result in large responses limited only by an MTU that may become fairly large in various HomeNet networking protocols.

Increased reliance on Resource Record Sets (RRsets) for discovery increases DDoS amplification concerns when overall RRset size is overlooked. The extent of this amplification had been constrained by the minimum MTU first established by [RFC0791] and noted by [RFC1191] of 576 bytes which accommodates 512 byte UDP DNS messages. Most Internet links are now able to handle much larger MTUs. Per [RFC2460], the minimum 1280 byte MTU is specified for IPv6.

To ensure minimal latency, DNS queries are first made using UDP. When a response becomes truncated, TCP is then normally attempted. Reliance on UDP has been relaxed by [RFC5966]. The size of a PTR RRset can be fairly large and result in UDP amplification issues when carried within a large minimum MTU. The potential query/response ratio may have a large impact on ISPs and in turn impact a large number of users.

At each of the DNS-SD SRV and TXT Resource Record Sets locations that offer instance and service enumerations, administration of the resulting RRsets must ensure these resources are suitable for distribution and the DNS-SD query to response ratio is suitable for Internet access.

DNS-SD [RFC6763] should not be viewed as a catalog structure of desired services suitable for Internet use. [I-D.ietf-dnssd-hybrid] is to be used to bridge adjacent networks but this risks conveying resources of hosts unable to safely facilitate Internet access. Since [I-D.ietf-dnssd-hybrid] should opt for the most conservative address mode when selecting addresses to be distributed, ULAs or [RFC1918] address should represent a default option rather than selecting GUAs.

Browsing change notification facilitated with [I-D.ietf-dnssd-push] uses the message structure defined by [RFC2136] but is based on TCP. TCP eliminates spoofed source query attacks and congestion issues. If neither QTYPE nor QCLASS are ANY (255) then this is a specific subscription to changes for the given name. When QTYPE or QCLASS are

ANY (255) then this becomes a wildcard subscription to changes of the given name for any type and/or class, as appropriate.

Browsing resource synchronization should use [I-D.ietf-dnssd-push] instead of depending on expanded RRsets or UDP transactions. Directly using DNS when overloaded would be much slower. This is because DNS [RFC1035] recommends 5 second timeouts with a doubling on two subsequent retries for a total of 35 seconds.

2.3.1. Resource Exhaustion Threats

DNS is currently vulnerable whenever responses are much larger than associated queries which could occur when browsing a domain offering services from a large number of hosts. To mitigate specific problematic query sources, an experimental mode of DNS operation is described in a technical note: DNS Response Rate Limiting [ISC-TN-2012-1-Draft1]. Additional information is available at [RedBarn].

Another experiment is [I-D.ietf-dnsop-cookies] which reduces reliance on DNS Response Rate Limiting and minimizes resources needed to handle random initial exchanges in a manner as described by [RFC6013] for forged sources of initial TCP <Syn> where servers keep client state within encrypted cookies.

3. Protection of SSD related interchange

SSD protocols may require additional steps to ensure against the poisoning of resource collection where close attention should be given to the scope of a ULA or [RFC1918] where the related resources are not to be directly exchanged with the Internet.

3.1. Link-Local

[RFC3927] provides an overview of IPv4 address complexities related to dealing with multiple segments and interfaces. IPv6 introduces new paradigms in respect to interface address assignments which offer scoping as explained in [RFC4291].

3.2. Authorization Issues

DNSSEC [RFC4033] can assert the validity but not the veracity of records in a zone file. The trust model of the global DNS [RFC1035] relies on the fact that human administrators either a) manually enter resource records into a zone file, or b) configure the DNS [RFC1035] server to authenticate a trusted device (e.g., a DHCP server) that can automatically maintain such records.

An imposter may register on the local link and appear as a legitimate service. Such "rogue" services may then be automatically registered in wide area DNS-SD [RFC6763].

3.3. Authentication Issues

Up to now, the "plug-and-play" nature of mDNS [RFC6762] devices have relied only on physical connectivity to the local network. If a device is visible via mDNS [RFC6762], it had been assumed to be trusted. When multiple networks are involved, verifying a host is local using mDNS [RFC6762] is no longer possible so other verification schemes must be used.

3.4. Privacy Considerations

Mobile devices such as smart phones that can expose the location of their owners by registering services in arbitrary zones pose a risk to privacy. Such devices must not register their services in arbitrary zones without the approval of their operators. However, it should be possible to configure one or more "safe" zones, e.g., based on subnet prefix, in which mobile devices may automatically register their services.

As noted in [CERTvu550620] private security information is leaked in many cases. This includes hostnames and MACs, networking details, service related details such as those for Printers and NAS devices. Many consumer printers can not authenticate users or block addresses when connected with IPv6. Once this information is leaked, malefactors are thereby given unlimited access.

4. IANA Considerations

This document requires no IANA consideration.

5. Acknowledgements

The authors wish to acknowledge valuable contributions from the following: Dave Rand, John C. Klensin, Dan York, Harald Albrecht, and Paul Vixie

6. References

6.1. Normative References

[I-D.ietf-dnsop-cookies]
Eastlake, D. and M. Andrews, "Domain Name System (DNS) Cookies", draft-ietf-dnsop-cookies-05 (work in progress), August 2015.

[I-D.ietf-dnssd-hybrid]
Cheshire, S., "Hybrid Unicast/Multicast DNS-Based Service Discovery", draft-ietf-dnssd-hybrid-00 (work in progress), November 2014.

[I-D.ietf-dnssd-push]
Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-00 (work in progress), March 2015.

[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., 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>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

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

[RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <http://www.rfc-editor.org/info/rfc3587>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

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

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[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>.

[RFC5895] Resnick, P. and P. Hoffman, "Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008", RFC 5895, DOI 10.17487/RFC5895, September 2010, <http://www.rfc-editor.org/info/rfc5895>.

[RFC5966] Bellis, R., "DNS Transport over TCP - Implementation Requirements", RFC 5966, DOI 10.17487/RFC5966, August 2010, <http://www.rfc-editor.org/info/rfc5966>.

[RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <http://www.rfc-editor.org/info/rfc6106>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <http://www.rfc-editor.org/info/rfc6763>.

6.2. References - Informative

[CERTvu550620] Seaman, C., "CERT Vulnerability Note VU#550620", March 2015, <https://www.kb.cert.org/vuls/id/550620>.

[IEEE.802-11.2012]
"Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", IEEE Standard 802.11, February 2012, <http://standards.ieee.org/getieee802/download/802.11-2012.pdf>.

[IEEE.802-1D.2004]
Institute of Electrical and Electronics Engineers, "Information technology - Telecommunications and information exchange between systems - Local area networks - Media access control (MAC) bridges", IEEE Standard 802.1D, February 2004, <http://standards.ieee.org/getieee802/download/802.1D-2004.pdf>.

[IEEE.802-3.2012]
"Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications"", IEEE Standard 802.3, August 2012, <http://standards.ieee.org/getieee802/download/802.3-2012_section1.pdf>.

[ISC-TN-2012-1-Draft1]
Vixie, P. and Rhyolite, "DNS Response Rate Limiting (DNS RRL)", April 2012, <http://ss.vix.su/~vixie/isc-tn-2012-1.txt>.

[RedBarn] Vixie, P. and Rhyolite, "Response Rate Limiting in the Domain Name System (DNS RRL)", June 2012, <http://www.redbarn.org/dns/ratelimits>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <http://www.rfc-editor.org/info/rfc1112>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <http://www.rfc-editor.org/info/rfc1191>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[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>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <http://www.rfc-editor.org/info/rfc3376>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <http://www.rfc-editor.org/info/rfc3810>.

[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>.

[RFC4043] Pinkas, D. and T. Gindin, "Internet X.509 Public Key Infrastructure Permanent Identifier", RFC 4043, DOI 10.17487/RFC4043, May 2005, <http://www.rfc-editor.org/info/rfc4043>.

[RFC4510] Zeilenga, K., Ed., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, DOI 10.17487/RFC4510, June 2006, <http://www.rfc-editor.org/info/rfc4510>.

[RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <http://www.rfc-editor.org/info/rfc4541>.

[RFC4903] Thaler, D., "Multi-Link Subnet Issues", RFC 4903, DOI 10.17487/RFC4903, June 2007, <http://www.rfc-editor.org/info/rfc4903>.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <http://www.rfc-editor.org/info/rfc4944>.

[RFC5517] HomChaudhuri, S. and M. Foschiano, "Cisco Systems' Private VLANs: Scalable Security in a Multi-Client Environment", RFC 5517, DOI 10.17487/RFC5517, February 2010, <http://www.rfc-editor.org/info/rfc5517>.

[RFC6013] Simpson, W., "TCP Cookie Transactions (TCPCT)", RFC 6013, DOI 10.17487/RFC6013, January 2011, <http://www.rfc-editor.org/info/rfc6013>.

[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, <http://www.rfc-editor.org/info/rfc6281>.

[RFC6895] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", BCP 42, RFC 6895, DOI 10.17487/RFC6895, April 2013, <http://www.rfc-editor.org/info/rfc6895>.

[RFC6950] Peterson, J., Kolkman, O., Tschofenig, H., and B. Aboba, "Architectural Considerations on Application Features in the DNS", RFC 6950, DOI 10.17487/RFC6950, October 2013, <http://www.rfc-editor.org/info/rfc6950>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <http://www.rfc-editor.org/info/rfc7217>.

[RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", RFC 7368, DOI 10.17487/RFC7368, October 2014, <http://www.rfc-editor.org/info/rfc7368>.

[RFC7513] Bi, J., Wu, J., Yao, G., and F. Baker, "Source Address Validation Improvement (SAVI) Solution for DHCP", RFC 7513, DOI 10.17487/RFC7513, May 2015, <http://www.rfc-editor.org/info/rfc7513>.

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

Appendix A. mDNS Example of Device Resolution Information

```
dns-sd -L "Brother MFC-9560CDW" _printer._tcp local
  Lookup Brother MFC-9560CDW._printer._tcp.local

16:00:26.965  Brother\032MFC-9560CDW._printer._tcp.local.
 can be reached at BRN30066C239958.local.:515
(interface 4) Flags: 2 txtvers=1 qtotal=1
pdl=application/vnd.hp-PCL,application/vnd.brother-hbp
rp=duerqxesz5090 ty=Brother\ MFC-9560CDW\
product=\(Brother\ MFC-9560CDW\)
adminurl=http://BRN30066C239958.local./
priority=75 usb_MFG=Brother usb_MDL=MFC-9560CDW
Color=T Copies=T Duplex=F PaperCustom=T Binary=T Transparent=T TBCP=F

Timestamp    A/R Flg if Hostname         Address                  TTL
16:14:34.855 Add  3  4 BRN30066C239958.local.
                       192.168.99.99                            245
16:14:34.856 Add  2  4 BRN30066C239958.local.
                       2699:9999:7300:1510:3205:5CFF:FE23:9958%<0>245

dns-sd -L "Canon MX920 series" _printer._tcp local.
Lookup Canon MX920 series._printer._tcp.local.

16:47:09.676  Canon\032MX920\032series._printer._tcp.local.
 can be reached at 9299990000.local.:515 (interface 4) Flags: 2
 txtvers=1 rp=auto note= qtotal=1 priority=60 ty=Canon\ MX920
 \ series product=\(Canon\ MX920\ series\)
 pdl=application/octet-stream adminurl=http://929999000000.local.
 usb_MFG=Canon usb_MDL=MX920\ series
 usb_CMD= UUID=00000000-0000-1000-8000-F4813999999
 Color=T Duplex=T Scan=T Fax=F mac=F4:81:39:99:99:99

dns-sd -G v4v6 "9299999000000.local."
Timestamp    A/R Flg if Hostname         Address                  TTL
17:07:12.460 Add  3  4 929999000000.local.
                       FE80:0000:0000:0000:F681:39FF:FE92:9999%en0 65
17:07:12.461 Add   2 4 929999000000.local.
                        192.168.99.108                           65
```

Appendix B. Uncontrolled Access Example

The risk is that adequate IPv6 filtering is simply not available on either current printers, scanners, cameras and other devices never intended to be used directly on the Internet.

For example, in the case of a printer:

ftp [DNS entry]

```
Trying 2699:9999:7300:1510:3205:5cff:fe23:9958...
Connected to [DNS entry]
220 FTP print service:V-1.13/Use the network password for the ID if
updating.
Name (BRN30066C239958.local.:dlr): ftp
230 User ftp logged in.
ftp> ls
229 Entering Extended Passive Mode (|||62468|)
150 Transfer Start
total 1
-r--r--r--   1 root     printer   4096 Sep 28  2001 CFG-PAGE.TXT
----------   1 root     printer      0 Sep 28  2001 Toner-Low-------
226 Data Transfer OK.
ftp>
```

From here, I can print a file with no further authentication. But the printer also now appears on the Internet with TCP ports 21,23,25,80,515,631 and 9100 active. I can scan a document that was left in the flatbed. I can send a fax. Or I can print many copies of black pages if I want to do a physical DOS. And, thanks to the globally routable address present, I can reach this from anywhere in the world.

Authors' Addresses

Douglas Otis
Trend Micro
10101 N. De Anza Blvd
Cupertino, CA 95014
USA

Phone: +1.408.257-1500
Email: doug_otis@trendmicro.com

Hosnieh Rafiee
Rozanak.com
Munich
Germany

Phone: +49 (0)176 57587575
Email: ietf@rozanak.com