Document: draft-cheshire-dnsext-multicastdns-01.txt
Category: Standards Track
Expires 20th June 2003
20th December 2002

Stuart Cheshire
Apple Computer, Inc.

Performing DNS queries via IP Multicast

<draft-cheshire-dnsext-multicastdns-01.txt>

Status of this Memo

Abstract

As networked devices become smaller, more portable, and more ubiquitous, the ability to operate with less configured infrastructure is increasingly important. In particular, the ability to look up host names and similar DNS resource record data types, in the absence of a conventional managed DNS server, is becoming essential.

Acknowledgements

This concepts described in this document have been explored and developed with help from Erik Guttman, Paul Vixie, Bill Woodcock, and others.

Table of Contents

1. Introduction

When reading this document, familiarity with the concepts of Zero Configuration Networking [ZC] and automatic link-local addressing [v4LL] [RFC 2462] is helpful.

This document proposes no change to the structure of DNS messages, and no new operation codes, response codes, or resource record types. This document simply discusses what needs to happen if DNS clients start sending DNS queries to a multicast address, and how a collection of hosts can cooperate to collectively answer those queries in a useful manner.

There has been discussion of how much burden Multicast DNS might impose on a network. It should be remembered that whenever IPv4 hosts communicate they broadcast ARP packets on the network on a regular basis, and this is not disastrous. The approximate amount of multicast traffic generated by hosts making conventional use of Multicast DNS is anticipated to be roughly the same order of magnitude as the amount of broadcast ARP traffic those hosts already generate.

New applications making new use of Multicast DNS capabilities for unconventional purposes may generate more traffic. If some of those new applications are "chatty", then work will be needed to help them become less chatty. When performing any analysis, is important to make a distinction between the application behavior and the underlying protocol behavior. If a chatty application uses UDP, that doesn't mean that UDP is chatty, or that IP is chatty, or that Ethernet is chatty. What it means is that the application is chatty. The same applies to any future applications that may decide to layer increasing portions of their functionality over Multicast DNS.

2. Conventions and Terminology Used in this Document

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 "Key words for use in RFCs to Indicate Requirement Levels" [RFC 2119].

This document uses the term "host name" in the strict sense to mean a fully qualified domain name that has an address record. It does not use the term "host name" in the commonly used but incorrect sense to mean just the first DNS label of a host's fully qualified domain name.

A DNS (or mDNS) packet contains an IP TTL in the IP header, which is effectively a hop-count limit for the packet, to guard against routing loops. Each Resource Record also contains a TTL, which is the number of seconds for which the Resource Record may be cached.

In any place where there may be potential confusion between these two types of TTL, the term "IP TTL" is used to refer to the IP header TTL (hop limit), and the term "RR TTL" is used to refer to the Resource Record TTL (cache lifetime).

When this document uses the term "Multicast DNS", it should be taken to mean: Clients performing DNS-like queries for DNS-like resource records by sending DNS-like UDP query and response packets over IP Multicast to UDP port 5353."

3. Multicast DNS Names

The DNS domain ".local." is (this document proposes) a special domain with special semantics, namely that ".local." and all its subdomains are link-local, and names within this domain are meaningful only on the link where they originate, much as IPv4 addresses in the 169.254/16 prefix are link-local and meaningful only on the link where they originate.

Any DNS query for a name ending with ".local." MUST be sent to the mDNS multicast address (224.0.0.251 or its IPv6 equivalent FF02::FB).

It is unimportant whether a name ending with ".local." occurred because the user explicitly typed in a fully qualified domain name ending in ".local.", or because the user entered an unqualified domain name and the host software appended the suffix ".local." because that suffix appears in the user's search list. The ".local." suffix could appear in the search list because the user manually configured it, or because it was received in a DHCP option, or via any other valid mechanism for configuring the DNS search list. In this respect the ".local." suffix is treated no differently to any other search domain that might appear in the DNS search list.

DNS queries for names that do not end with ".local." MAY be sent to the mDNS multicast address, if no other conventional DNS server is available. This can allow hosts on the same link to continue communicating using each other's globally unique DNS names during network outages which disrupt communication with the greater Internet. When resolving global names via local multicast, it is even more important to use DNSSEC or other security mechanisms to ensure that the response is trustworthy. Resolving global names via local multicast is a contentious issue, and this document does not discuss it in detail, instead concentrating on the issue of resolving local names using DNS packets sent to a multicast address.

A host which belongs to an organization or individual who has control over some portion of the DNS namespace can be assigned a globally unique name within that portion of the DNS namespace, for example, "cheshire.apple.com." For those of us who have this luxury, this works very well. However, the majority home customers do not have easy access to any portion of the global DNS namespace within which they have the authority to create names as they wish. This leaves the majority of home computers effectively anonymous for practical purposes.

To remedy this problem, this document allows any computer user to elect to give their computers link-local Multicast DNS host names of the form: "single-dns-label.local." For example, my Titanium PowerBook laptop computer answers to the name "sctibook.local." Any computer user is granted the authority to name their computer this way, providing that the chosen host name is not already in use on that link. Having named their computer this way, the user has the authority to continue using that name until such time as name conflict occurs on the link which is not resolved in the user's favour. If this happens, the computer (or its human user) SHOULD cease using the name, and may choose to attempt to allocate a new unique name for use on that link. Like law suits over global DNS names, these conflicts are expected to be relatively rare for people who choose reasonably imaginative names, but it is still important to have a mechanism in place to handle them when they happen.

The point made in the previous paragraph is very important and bears repeating. It is easy for those of us in the IETF community who run our own name servers at home to forget that the majority of computer users do not run their own name server and have no easy way to create their own host names. When these users wish to transfer files between two laptop computers, they are frequently reduced to typing in dotted-decimal IP addresses because they simply have no other way for one host to refer to the other by name. This is a sorry state of affairs. What is worse, most users don't even bother trying to use dotted-decimal IP addresses. Most users still move data between machines by copying it onto a floppy disk or similar removable media. In a world of gigabit Ethernet and ubiquitous wireless networking it is a sad indictment of the networking community that the preferred communication medium for most computer users is still the floppy disk.

Allowing ad-hoc allocation of single-label names in a single flat ".local." namespace may seem to invite chaos. However, operational experience with AppleTalk NBP names, which on any given link are also effectively single-label names in a flat namespace, shows that in practice name collisions happen extremely rarely and are not a problem. Groups of computer users from disparate organizations bring Macintosh laptop computers to events such as IETF Meetings, the Mac Hack conference, the Apple World Wide Developer Conference, etc., and complaints at these events about users suffering conflicts and being forced to rename their machines have never been an issue.

Enforcing uniqueness of host names (i.e. the names of DNS address records mapping names to IP addresses) is probably desirable in the common case, but this document does not mandate that. It is permissible for a collection of coordinated hosts to agree to maintain multiple DNS address records with the same name, possibly for load balancing or fault-tolerance reasons. This document does not take a position on whether that is sensible. It is important that both modes of operation are supported. The Multicast DNS protocol allows hosts to verify and maintain unique names for resource records where that behaviour is desired, and it also allows hosts to maintain multiple resource records with a single shared name where that behaviour is desired. This consideration applies to all resource records, not just address records (host names). In summary: It is required that the protocol have the ability to detect and handle name conflicts. It is not required that the user should use that ability in every case.

4. IP TTL Checks

A host sending Multicast DNS queries to a link-local destination address MUST verify that the IP TTL in response packets is 255, and silently discard any response packets where the IP TTL is not 255. Without this check, it could be possible for remote rogue hosts to send spoof answer packets (perhaps unicast to the victim host) which the receiving machine could misinterpret as having originated on the local link.

There has been some discussion that many current network programming APIs to not provide any indication of the IP TTL on received packets. This is unfortunate, and should be fixed for hosts that want to be able to guard against spoof packets arriving from off-link.

5. Reverse Address Mapping

Like ".local.", the IPv4 and IPv6 reverse-mapping domains are also defined to be link-local.

Any DNS query for a name ending with "254.169.in-addr.arpa." MUST be sent to the mDNS multicast address 224.0.0.251. Since names under this domain correspond to IPv4 link-local addresses, it is logical that the local link is the best place to find information pertaining to those names.

Likewise, any DNS query for a name ending with "0.8.e.f.ip6.arpa." MUST be sent to the IPv6 mDNS link-local multicast address FF02::FB.

6. Querying

There are three kinds of Multicast DNS Queries, one-shot queries of the kind made by today's conventional DNS clients, one-shot queries accumulating multiple responses made by multicast-aware DNS clients, and continuous ongoing Multicast DNS Queries used by IP network browser software.

A Multicast DNS Responder that is offering records that are intended to be unique on the local link MUST also implement a Multicast DNS Querier so that it can first verify the uniqueness of those records before it begins answering queries for them.

6.1 One-Shot Queries

An unsophisticated DNS client may simply send its DNS queries blindly to the 224.0.0.251 multicast address, without necessarily even being aware what a multicast address is.

Such an unsophisticated DNS client may not get ideal behaviour. Such a client may simply take the first response it receives and fail to wait to see if there are more, but in many instances this may not be a serious problem. If a user types "http://stu.local." into their Web browser and gets to see the page they were hoping for, then the protocol has met the user's needs in this case.

6.2 One-Shot Queries, Accumulating Multiple Responses

A more sophisticated DNS client should understand that Multicast DNS is not exactly the same as unicast DNS, and should modify its behaviour in some simple ways.

As described above, there are some cases, such as looking up the address associated with a unique host name, where a single response is sufficient, and moreover may be all that is expected. However, there are other DNS queries where more than one response is possible, and for these queries a more sophisticated Multicast DNS client should include the ability to wait for an appropriate period of time to collect multiple responses.

A naive DNS client retransmits its query only so long as it has received no response. A more sophisticated Multicast DNS client is aware that having received one response is not necessarily an indication that it might not receive others, and has the ability to retransmit its query an appropriate number of times at appropriate intervals until it is satisfied with the collection of responses it has gathered.

A more sophisticated Multicast DNS client that is retransmitting a query for which is has already received some responses, MAY elect to implement duplicate suppression, as described below under "Duplicate Suppression". This indicates to responders who have already replied that their responses have been received, and they don't need to send them again in response to this repeated query.

A Multicast DNS Querier MAY place more than one question into the Question Section of a Multicast DNS Query.

6.3 Continuous Querying

In One-Shot Queries, with either a single or multiple responses, the underlying assumption is that the transaction begins when the application issues a query, and ends when all the desired responses have been received. There is another type of operation which is more akin to continuous monitoring.

Macintosh users are accustomed to opening the "Chooser" window, selecting a desired printer, and then closing the Chooser window. However, when the desired printer does not appear in the list, the user will typically leave the "Chooser" window open while they go and check to verify that the printer is plugged in, powered on, connected to the Ethernet, etc. While the user jiggles the wires, hits the Ethernet hub, and so forth, they keep an eye on the Chooser window, and when the printer name appears, they know they have fixed whatever the problem was. This can be a useful and intuitive troubleshooting technique, but a user who goes home for the weekend leaving the Chooser window open places a non-trivial burden on the network.

It is important that an IP network browser window displaying live information from the network using Multicast DNS, if left running for an extended period of time, should generate significantly less multicast traffic on the network than the old AppleTalk Chooser.

A Multicast DNS Querier asking the same question repeatedly for an indefinite period of time MUST implement duplicate suppression, as described below.

7. Duplicate Suppression

A variety of techniques are used to reduce the amount of redundant traffic on the network.

7.1 Known Answer Suppression

When a Multicast DNS Querier sends a query to which it already knows some answers, it populates the Answer Section of the DNS message with those cached resource records whose remaining TTL values indicate that they will remain valid for at least the time anticipated to send this DNS query, and the next, and the one after that. For example, if the query DNS Querier is planning to wait four seconds after this query before sending the next, and then eight seconds after that, then only resource records with TTL values greater than twelve seconds should be included in the answer section. This is to ensure that when a resource record's TTL is close to expiration, the Multicast DNS Querier has *two* chances to refresh it before the cached record expires and has to be removed from the list.

A Multicast DNS Responder SHOULD NOT answer a Multicast DNS Query if the answer it would give is already included in the Answer Section with an RR TTL at least half the correct value. If the RR TTL of the answer as given in the Answer Section is less than half of the real RR TTL as known by the Multicast DNS Responder, the responder SHOULD send an answer so as to update the Querier's cache before the record becomes in danger of expiration.

A Multicast DNS Querier MUST NOT cache resource records observed in the Answer Section of other Multicast DNS Queries. The Answer Section of Multicast DNS Queries is not authoritative. By placing information in the Answer Section of a Multicast DNS Query the querier is stating that it *believes* the information to be true. It is not asserting that the information *is* true. Some of those records may have come from other hosts that are no longer on the network. Propagating that stale information to other Multicast DNS Queriers on the network would not be helpful.

7.2 Multi-Packet Known Answer Suppression

Sometimes a Multicast DNS Querier will already have too many answers to fit in the Known Answer section of its query packets. In this case, it should issue a Multicast DNS Query containing a questions as many Known Answer records as will fit. It should then set the TC (Truncated) bit in the header before sending the Query. It should then immediately follow the packet with another query containing no questions, and as many more Known Answer records as will fit. If there are still too many records remaining to fit in the packet, it again sets the TC bit and continues until all the Known Answer records have been sent.

A Multicast DNS Responder seeing a Multicast DNS Query with the TC bit set defers its response for a time period randomly selected in the interval 20-120ms. This gives the Multicast DNS Querier time to send additional Known Answer packets before the Responder responds. If the Responder sees any of its answers listed in the Known Answer lists of subsequent packets from the querying host, it should delete that answer from the list of answers it is planning to give, provided that no other host on the network is also waiting to receive the same answer record.

7.3 Duplicate Question Suppression

If a host is planning to send a query, and it sees another host on the network send a query containing the same question, and the Known Answer section of that query does not contain any records which this host would not also put in its own Known Answer section, then this host should treat its own query as having been sent. When multiple clients on the network are querying for the same resource records, there is no need for them to all be repeatedly asking the same question.

7.4 Duplicate Answer Suppression

If a host is planning to send an answer, and it sees another host on the network send a response packet containing the same answer record, and the TTL in that record is not less than the TTL this host would have given, then this host should treat its own answer as having been sent. When multiple responders on the network have the same data, there is no need for all of them to respond.

The feature is particularly useful when multiple Sleep Proxy Servers are deployed (see Section 16. "Multicast DNS and Power Management"). In future it is possible that every general-purpose OS (Mac, Windows, Linux, etc.) will implement Sleep Proxy Service as a matter of course. In this case there could be a large number of Sleep Proxy Servers on any given network, which is good for reliability and fault-tolerance, but would be bad for the network if every Sleep Proxy Server were to answer every query.

8. Responding

A Multicast DNS Responder MUST only respond when it has a positive non-null response to send. Error responses must never be sent. The non-existence of any name in a Multicast DNS Domain is ascertained by the failure of any machine to respond to the Multicast DNS query, not by NXDOMAIN errors.

Multicast DNS Responses need not contain any questions in the Question Section. Multicast DNS Queriers receiving Multicast DNS

Responses do not care what question elicited the response; they care only that the information in the response is true and accurate.

A Multicast DNS Responder on Ethernet [IEEE802] and similar shared multiple access networks SHOULD delay its responses by a random amount of time selected with uniform random distribution in the range 20-120ms. If multiple Multicast DNS Responders were all to respond immediately to a particular query, a collision would be virtually guaranteed. By imposing a small random delay, the number of collisions is dramatically reduced. 120ms is a short enough time that it is almost imperceptible to a human user, but long enough to significantly reduce the risk of Ethernet collisions. On a full-sized Ethernet using the maximum cable lengths allowed and the maximum number of repeaters allowed, an Ethernet frame is vulnerable to collisions during the transmission of its first 256 bits. On 10Mb/s Ethernet, this equates to a vulnerable time window of 25.6us.

In the case where a Multicast DNS Responder has good reason to believe that it will be the only responder on the link with a positive non-null response, it SHOULD respond immediately, without the random delay. To do this safely, it MUST have previously verified that the requested name, type and class in the DNS query are unique on this link. This may be appropriate for things like looking up the address record for a particular host name, when the host name has been previously verified unique. This is *not* appropriate for things like looking up PTR records used for DNS Service Discovery [DNS-SD], where a large number of responses may be anticipated.

Multicast DNS Responses MUST be sent to UDP port 5353 (the well-known port assigned to mDNS) on the 224.0.0.251 multicast address (or its IPv6 equivalent). Operating in a Zeroconf environment requires constant vigilance. Just because a name has been previously verified unique does not mean it will continue to be so indefinitely. By allowing all Multicast DNS Responders to constantly monitor their peers' responses, conflicts arising out of network topology changes can be promptly detected and resolved. Sending all responses by multicast also facilitates opportunistic caching by other hosts on the network.

If the source UDP port in a received Multicast DNS Query is not port 5353, this indicates that the client originating the query is a simple client that does not fully implement all of Multicast DNS. In this case, after sending the usual Multicast DNS Response to 224.0.0.251 port 5353, the Multicast DNS Responder MUST also send a second UDP response to the client, via unicast, to the query packet's source IP address and port.

Multicast DNS Responders MUST correctly handle DNS query packets containing more than one question, by answering any or all of the questions to which they have answers. Any answers generated in response to query packets containing more than one question

MUST be randomly delayed in the range 20-120ms, as described above.

9. Probing and Announcing on Startup

Whenever a Multicast DNS Responder starts up, wakes up from sleep, receives an indication of an Ethernet "Link Change" event, or has any other reason to believe that its network connectivity may have changed in some relevant way, it MUST perform two startup steps.

9.1 Probing

The first startup step is that for all those resource records that a Multicast DNS Responder desires to be unique on the local link, it MUST send a Multicast DNS Query asking for those resource records, to see if any of them are already in use. The primary example of this is its address record which maps its unique host name to its unique IP address. The ability to place more than one question in a Multicast DNS Query is useful here, because it can allow a host to use a single packet for all of its resource records instead of needing a separate packet for each. If any conflicting Multicast DNS responses are received, then the host MUST defer to the other host already using those names, and MUST select new names for its conflicting records which need to be unique. 250ms after the first query it should send a second, then 250ms after that a third. If, after a total of 750ms, no conflicting Multicast DNS responses have been received, the host may move to the second step.

The astute reader will observe that there is a race condition inherent in the previous description. If two hosts are probing for the same name simultaneously, neither will receive any response to the probe, and the hosts could incorrectly conclude that they may both proceed to use the name. To break this symmetry, each host populates the Authority Section of its queries with records giving the rdata that it would be proposing to use, should its probing be successful. The Authority Section is being used here in a way analogous to the Update section of a DNS Update packet [RFC 2136].

When a host that is probing for a record sees another host issue a query for the same record, it consults the Authority Section of that query. If it finds any resource record there which answers the query, then it compares the rdata in that resource record with its own tentative rdata. The lexicographically earlier rdata wins. This means that if the host finds that its own rdata is lexicographically earlier, it simply ignores the other host's probe. If the host finds that the rdata in the Authority Section record is lexicographically earlier, then it treats this exactly as if it had received an answer to its query, and concludes that it may not use the desired name.

The determination of 'lexicographically earlier' is performed by raw comparison of the binary content of the rdata without regard for meaning or structure. The bytes of the rdata are compared in turn until a byte is found whose value is lesser than that of its counterpart (in which case the rdata whose byte has the lesser value is deemed lexicographically earlier) or one of the resource records runs out of rdata (in which case the resource record which ran out of data first is deemed lexicographically earlier).

The following is an example of a conflict:

```
sctibook.local. A 196.254.50.100
sctibook.local. A 196.254.100.50
```

In this case 196.254.50.100 is lexicographically earlier, so is deemed the winner.

9.2 Announcing

The second startup step is that the Multicast DNS Responder MUST send a gratuitous Multicast DNS Response containing, in the Answer Section, all of its resource records. If there are too many resource records to fit in a single packet, multiple packets may be used.

In the case of shared records (e.g. the PTR records used by DNS Service Discovery [DNS-SD]) the records are simply placed as-is into the answer section of the DNS Response.

In the case of records that have been verified to be unique in the previous step, they are placed into the answer section of the DNS Response with the most significant bit of the rrclass set to one.

The most significant bit of the rrclass is the mDNS "cache flush" bit. Normally when a resource record appears answer in the section of the DNS Response, it means, "This is an assertion that this information is true." When a resource record appears answer in the section of the DNS Response with the "cache flush" bit set, it means, "This is an assertion that this information is the truth and the whole truth, and anything you may have heard before regarding records of this name/type/class is no longer valid". The "cache flush" bit is described further in Section 13.1 "Announcements to Update Cache Entries".

Up to ten of gratuitous Multicast DNS Responses may be sent, providing that the interval between gratuitous responses doubles with every response sent, and the interval between the first two gratuitous responses is not less than one second.

Whenever a Multicast DNS Responder receives any Multicast DNS response (gratuitous or otherwise) containing a conflicting resource record, the conflict MUST be resolved as described below in "Conflict Resolution".

A Multicast DNS Responder MUST NOT send announcements in the absence of information that its network connectivity may have changed in some relevant way. In particular, a Multicast DNS Responder MUST NOT send regular periodic announcements as a matter of course.

10. Conflict Resolution

A conflict occurs when two resource records with the same name, type and class have inconsistent rdata. What may be considered inconsistent is context sensitive, except that resource records with identical rdata are never considered inconsistent, even if they originate from different hosts. In the case of a host desiring to have a unique host name, another address record with the same name but a different IP address is considered inconsistent.

Whenever a Multicast DNS Responder receives any Multicast DNS response (gratuitous or otherwise) containing a conflicting resource record, the Multicast DNS Responder must immediately reset that record to probing state, and go through the startup steps described above in Section 9. "Probing and Announcing on Startup". The protocol used in the Probing phase will determine a winner and a loser, and the loser must cease using the name, and reconfigure.

In the case of a typical laptop or desktop computer with a human user, reconfiguration is achieved by displaying an error message to the user and suggesting that they choose a new name. In the case of a device with no human operator, reconfiguration is achieved by its software programmatically generating a new name. In either case, the host must then test the new name for uniqueness as described above in "Probing and Announcing on Startup".

It is important that any host that observes an apparent conflict should take action. In the case of two hosts using the same host name, where one has been configured to require a unique host name and the other has not, the one that has not been configured to require a unique host name will not perceive any conflict, and will not take any action. By reverting to Probing state, the host that desires a unique host name will go through the necessary steps to ensure that a unique host is obtained.

The examples in this section focus on address records (i.e. host names), but the same considerations apply to all resource records where uniqueness (or maintenance of some other defined constraint) is desired.

11. Special Characteristics of Multicast DNS Domains

Unlike conventional DNS names, names that end in ".local.", "254.169.in-addr.arpa." or "0.8.e.f.ip6.arpa." have only local significance. Conventional DNS seeks to provide a single unified namespace, where a given DNS query yields the same answer no matter where on the planet it is performed or to which recursive DNS server the query is sent. (However, split views, firewalls, intranets and the like have somewhat interfered with this goal of DNS representing a single universal truth.) In contrast, each IP link has its own private ".local.", "254.169.in-addr.arpa." and "0.8.e.f.ip6.arpa." namespaces, and the answer to any query for a name within those domains depends on where that query is asked.

Multicast DNS Domains are not delegated from their parent domain via use of NS records. There are no NS records anywhere in Multicast DNS Domains. Instead, all Multicast DNS Domains are delegated to the IP addresses 224.0.0.251 and FF02::FB by virtue of the individual organizations producing DNS client software deciding how to handle those names. It would be extremely valuable for the industry if this special handling were ratified and recorded by IANA, since otherwise the special handling provided by each vendor is likely to be inconsistent.

The IPv4 name server for a Multicast DNS Domain is 224.0.0.251. The IPv6 name server for a Multicast DNS Domain is FF02::FB. These are multicast addresses; therefore they identify not a single host but a collection of hosts, working in cooperation to maintain some reasonable facsimile of a competently managed DNS zone. Conceptually a Multicast DNS Domain is a single DNS zone, however its server is implemented as a distributed process running on cluster of loosely cooperating CPUs rather than as a single process running on a single CPU.

No delegation is performed within Multicast DNS Domains. Because the cluster of loosely coordinated CPUs is cooperating to administer a single zone, delegation is neither necessary nor desirable. Just because a particular host on the network may answer queries for a particular record type with the name "example.local." does not imply anything about whether that host will answer for the name "child.example.local.", or indeed for other record types with the name "example.local."

Multicast DNS Zones have no SOA record. A conventional DNS zone's SOA record contains information such as the email address of the zone administrator and the monotonically increasing serial number of the last zone modification. There is no single human administrator for any given Multicast DNS Zone, so there is no email address. Because the hosts managing any given Multicast DNS Zone are only loosely coordinated, there is no readily available monotonically increasing serial number to determine whether or not the zone contents have changed. A host holding part of the shared zone could crash or be disconnected from the network at any time without informing the other hosts. There is no reliable way to provide a zone serial number that would, whenever such a crash or disconnection occurred, immediately change to indicate that the contents of the shared zone had changed.

Zone transfers are not possible for any Multicast DNS Zone.

12. Multicast DNS for Service Discovery

This document does not describe using Multicast DNS for network browsing or service discovery. However, the mechanisms this document describes are compatible with (and support) the browsing and service discovery mechanisms proposed in "Discovering Named Instances of Abstract Services using DNS" [DNS-SD].

This document places few limitations on what DNS record types may be looked up using local multicast. One particular kind of Multicast DNS query that might be useful is a query for the SRV record named "_dns._udp.local.", yielding the port number and IP address of a conventional DNS server willing to perform general recursive DNS lookups. This could solve a particular problem facing the IPv6 community, which is that IPv6 is able to self-configure almost all of the information it needs to operate [RFC 2462], except for the address of the DNS server. Bringing in all of the mechanisms of DHCP just for that one little additional piece of information is not an attractive solution. Using DNS-format messages and DNS-format resource records to find the address of the DNS server has an elegant self-sufficiency about it. Any host that needs to know the address of the DNS server must already have code to generate and parse DNS packets, so using that same code and those same packets to find the DNS server in the first place is a simple self-reliant solution that avoids taking external dependencies on other protocols.

13. Resource Record TTL Values and Cache Coherency

The recommended TTL value for Multicast DNS resource records is 120 minutes.

A client with an active outstanding query will issue a query packet when one or more of the resource record(s) in its cache is (are) half-way to expiry. If the TTL on those records is 120 minutes, this ongoing cache maintenance process yields a steady-state query rate of one query per hour.

Any distributed cache needs a cache coherency protocol. If Multicast DNS resource records follow the recommendation and have a TTL of 120 minutes, that means that stale data could persist in the system for up to two hours. Making the default TTL significantly lower would reduce the lifetime of stale data, but would produce too much extra traffic on the network. Various techniques are available to minimize the impact of such stale data.

13.1 Announcements to Update Cache Entries

In the case where a host knows that certain resource record data is about to become invalid (for example when the host is undergoing a clean shutdown) the host sends a gratuitous announcement mDNS response packet, giving the same resource record name, type, class and rdata, but an RR TTL of zero. This has the effect of updating the TTL stored in neighbouring hosts' cache entries to zero, causing that cache entry to be promptly deleted.

Whenever a host has a resource record with potentially new data (e.g. after rebooting, waking from sleep, connecting to a new network link, changing IP address, etc.), the host sends a series of gratuitous announcements to update cache entries in its neighbour hosts. In these gratuitous announcements, if the record is one that is intended to be unique, the host sets the most significant bit of the rrclass field of the resource record. This bit, the "cache flush" bit, tells neighbouring hosts that this is not a shared record type. Instead of merging this new record additively into the cache in addition to any previous records with the same name, type and class, all old records with that name, type and class are summarily declared invalid and immediately flushed from the cache.

To accommodate the case where the set of records from one host constituting a single unique RRSet is too large to fit in a single packet, only cache records that are more than one second old are flushed. This allows the announcing host to generate a quick burst of two or more packets back-to-back on the wire, and the later packets will not immediately flush the cache records created by the earlier packets. Only cache records more than one second old will be flushed.

The "cache flush" bit is only used in Multicast DNS responses sent via multicast. The "cache flush" bit MUST NOT be set in any resource records in a response packet sent via unicast to any host.

13.2 Cache Flush on Topology change

If the hardware on a given host is able to indicate physical changes of connectivity, then when the hardware indicates such a change of connectivity, all cached records which were received on that interface should immediately be flushed.

Likewise, when a host reboots, or wakes from sleep, or undergoes some other similar discontinuous state change, its entire mDNS resource record cache should be flushed.

13.3 Cache Flush on Failure Indication

Sometimes a cache record can be determined to be stale when a client attempts to use the rdata it contains, and finds that rdata to be incorrect.

For example, the rdata in an address record can be determined to be incorrect if attempts to contact that host fail, either because ARP/ND requests for that address go unanswered (for an address on a local subnet) or because a router returns an ICMP "Host Unreachable" error (for an address on a remote subnet).

The rdata in an SRV record can be determined to be incorrect if attempts to communicate with the indicated service at the host and port number indicated are not successful.

The rdata in a DNS-SD PTR record can be determined to be incorrect if attempts to look up the SRV record it references are not successful.

In any such case, the software implementing the mDNS resource record cache should provide a mechanism so that clients detecting stale rdata can inform the cache and have that data flushed.

The end result of this is that if a printer suffers a sudden power failure or other abrupt disconnection from the network, its name may continue to appear in DNS-SD browser lists displayed on users' screens. Eventually that entry will expire from the cache naturally, but if a user tries to access the printer before that happens, the failure to successfully contact the printer will trigger the more hasty demise of its cache entries. This is a sensible trade-off between good user-experience and good network efficiency. If we were to insist that printers should disappear from the printer list within 30 seconds of becoming unavailable, for all failure modes, the only way to achieve this would be for the client to poll the printer at least every 30 seconds, or for the printer to announce its presence at least every 30 seconds, both of which would be an unreasonable burden on most networks.

14. Enabling and Disabling Multicast DNS

The option to fail-over to Multicast DNS for names not ending in ".local." SHOULD be a user-configured option, and SHOULD be disabled by default because of the possible security issues related to unintended local resolution of apparently global names.

The option to lookup unqualified (relative) names by appending ".local." (or not) is controlled by whether ".local." appears (or not) in the client's DNS search list.

No special control is needed for enabling and disabling Multicast DNS for names explicitly ending with ".local." as entered by the user. The user doesn't need a way to disable Multicast DNS for names ending with ".local.", because if the user doesn't want to use Multicast DNS, they can achieve this by simply not using those names. If a user *does* enter a name ending in ".local.", then we can safely assume the user's intention was probably that it should work. Having user configuration options that can be (intentionally or unintentionally) set so that local names don't work is just one more way of frustrating the user's ability to perform the tasks they want, perpetuating the view that, "IP networking is too complicated to configure and too hard to use." This in turn perpetuates the continued use of protocols like AppleTalk. If we want to retire AppleTalk, NetBIOS, etc., we need to offer users equivalent IP functionality that they can rely on to, "always work, like AppleTalk." A little Multicast DNS traffic may be a burden on the network, but it is an insignificant burden compared to continued widespread use of AppleTalk.

15. Considerations for Multiple Interfaces

A host should defend its host name (FQDN) on all active interfaces on which it is answering Multicast DNS queries.

In the event of a name conflict on *any* interface, a host should configure a new host name, if it wishes to maintain uniqueness of its host name.

When answering a Multicast DNS query, a multi-homed host with a link-local address (or addresses) should take care to ensure that any address going out in a Multicast DNS response is valid for use on the interface on which the response is going out.

Just as the same link-local IP address may validly be in use simultaneously on different links by different hosts, the same link-local host name may validly be in use simultaneously on different links, and this is not an error. A multi-homed host with connections to two different links may be able to communicate with two different hosts that are validly using the same name. While this kind of name duplication should be rare, it means that a host that wants to fully support this case needs network programming APIs that allow applications to specify on what interface to perform a link-local Multicast DNS query, and to discover on what interface a Multicast DNS response was received.

16. Multicast DNS and Power Management

Many modern network devices have the ability to go into a low-power mode where only a small part of the Ethernet hardware remains powered, and the device can be woken up by sending a specially formatted Ethernet frame which the device's power-management hardware recognizes.

To make use of this in conjunction with Multicast DNS, the device first uses DNS-SD to determine if Sleep Proxy Service is available on the local network. In some networks there may be more than one piece of hardware implementing Sleep Proxy Service, for fault-tolerance reasons.

If the device finds the network has Sleep Proxy Service, the device transmits two or more gratuitous mDNS announcements setting the TTL of its relevant resource records to zero, to delete them from neighbouring caches. The relevant resource records include address records and SRV records, and other resource records as may apply to a particular device. The device then communicates all of its remaining active records, plus the names, types and classes of the deleted records, to the Sleep Proxy Service(s), along with a copy of the specific "magic packet" required to wake the device up.

When a Sleep Proxy Service sees an mDNS query for one of the device's active records (e.g. a DNS-SD PTR record), it answers on behalf of the device without waking it up. When a Sleep Proxy Service sees an mDNS query for one of the device's deleted resource records, it deduces that some client on the network needs to make an active connection to the device, and sends the specified "magic packet" to wake the device up. The device then wakes up, reactivates its deleted resource records, and re-announces them to the network. The client waiting to connect sees the announcements, learns the current IP address and port number of the desired service on the device, and proceeds to connect to it.

The connecting client does not need to be aware of how Sleep Proxy Service works. Only devices that implement low power mode and wish to make use of Sleep Proxy Service need to be aware of how that protocol works.

The full specification of mDNS / DNS-SD Sleep Proxy Service is described in another document [not yet published].

17. Multicast DNS Character Set

Unicast DNS has been plagued by the lack of any support for non-US characters. Indeed, conventional DNS is usually limited to just letters, digits and hyphens, with no spaces or other punctuation. Attempts to remedy this have made slow progress because of the need to accommodate old buggy legacy implementations.

Multicast DNS is a new protocol and doesn't (yet) have old buggy legacy implementations to constrain the design choices. Accordingly, it adopts the obvious simple solution: all names in Multicast DNS are encoded using UTF-8 [RFC 2279]. For names that are restricted to letters, digits and hyphens, the UTF-8 encoding is identical to the US-ASCII encoding, so this is entirely compatible with existing host names. For characters outside the US-ASCII range, UTF-8 encoding is used.

Multicast DNS implementations MUST NOT use any other encodings apart from UTF-8 (US-ASCII being considered a compatible subset of UTF-8).

This point bears repeating: There are various baroque representations of international text being proposed for Unicast DNS. None of these representations may be used in Multicast DNS packets. Any text being represented internally in some other representation MUST be converted to canonical UTF-8 before being placed in any Multicast DNS packet.

18. Multicast DNS Message Format

This section describes specific restrictions on the allowable values for the header fields of a Multicast DNS message.

18.1. ID (Query Identifier)

Multicast DNS clients SHOULD listen for gratuitous responses issued by hosts booting up (or waking up from sleep or otherwise joining the network). Since these gratuitous responses may contain a useful answer to a question for which the client is currently awaiting an answer, Multicast DNS clients SHOULD examine all received Multicast DNS response messages for useful answers, without regard to the contents of the ID field or the question section. In multicast DNS, knowing which particular query message (if any) is responsible for eliciting a particular response message is less interesting than knowing whether the response message contains useful information.

Multicast DNS clients MAY cache any or all Multicast DNS response messages they receive, for possible future use, providing of course that normal TTL aging is performed on these cashed resource records.

In multicast query messages, the Query ID SHOULD be set to zero on transmission.

In multicast responses, including gratuitous multicast responses, the Query ID MUST be set to zero on transmission, and MUST be ignored on reception.

In unicast response messages generated specifically in response to a particular (unicast or multicast) query, the Query ID MUST match the ID from the query message.

18.2. QR (Query/Response) Bit

In query messages, MUST be zero.

In response messages, MUST be one.

18.3. OPCODE

In both multicast query and multicast response messages, MUST be zero (only standard queries are currently supported over multicast, unless other queries are allowed by future IETF Standards Action).

18.4. AA (Authoritative Answer) Bit

In query messages, the Authoritative Answer bit MUST be zero on transmission, and MUST be ignored on reception.

In response messages for Multicast Domains, the Authoritative Answer bit MUST be set to one (not setting this bit implies there's some other place where "better" information may be found) and MUST be ignored on reception.

18.5. TC (Truncated) Bit

In query messages, if the TC bit is set, it means that additional Known Answer records may be following shortly. A responder MAY choose to record this fact, and wait for those additional Known Answer records, before deciding whether to reply. If the TC bit is clear, it means that the querying host has no additional Known Answers.

In response messages, the TC bit MUST be zero on transmission, and MUST be ignored on reception.

18.6. RD (Recursion Desired) Bit

In both multicast query and multicast response messages, the Recursion Desired bit SHOULD be zero on transmission, and MUST be ignored on reception.

18.7. RA (Recursion Available) Bit

In both multicast query and multicast response messages, the Recursion Available bit MUST be zero on transmission, and MUST be ignored on reception.

18.8. Z (Zero) Bit

In both query and response messages, the Zero bit MUST be zero on transmission, and MUST be ignored on reception.

18.9. AD (Authentic Data) Bit [RFC 2535]

In query messages the Authentic Data bit MUST be zero on transmission, and MUST be ignored on reception.

In response messages, the Authentic Data bit MAY be set. Resolvers receiving response messages with the AD bit set MUST NOT trust the AD bit unless they trust the source of the message and either have a secure path to it or use DNS transaction security.

18.10. CD (Checking Disabled) Bit [RFC 2535]

In query messages, a resolver willing to do cryptography SHOULD set the Checking Disabled bit to permit it to impose its own policies.

In response messages, the Checking Disabled bit MUST be zero on transmission, and MUST be ignored on reception.

18.11. RCODE (Response Code)

In both multicast query and multicast response messages, the Response Code MUST be zero on transmission. Multicast DNS messages received with non-zero Response Codes MUST be silently ignored.

19. Choice of UDP Port Number

Arguments were made for and against using Multicast on UDP port 53. The final decision was to use UDP port 5353. Some of the arguments for and against are given below.

19.1 Arguments for using UDP port 53:

* This is "just DNS", so it should be the same port.

* There is less work to be done updating old clients to do simple mDNS queries. Only the destination address need be changed. In some cases, this can be achieved without any code changes, just by adding the address 224.0.0.251 to a configuration file.

19.2 Arguments for using a different port (UDP port 5353):

* This is not "just DNS". This is a DNS-like protocol, but different.

* Changing client code to use a different port number is not hard.

* Using the same port number makes it hard to run an mDNS Responder and a conventional unicast DNS server on the same machine. If a conventional unicast DNS server wishes to implement mDNS as well, it can still do that, by opening two sockets. Having two different port numbers is important to allow this flexibility.

* Some VPN software hijacks all outgoing traffic to port 53 and redirects it to a special DNS server set up to serve those VPN clients while they are connected to the corporate network. It is questionable whether this is the right thing to do, but it is common, and redirecting link-local multicast DNS packets to a remote server rarely produces any useful results. It does mean, for example, that the user becomes unable to access their local network printer sitting on their desk right next to their computer. Using a different UDP port eliminates this particular problem.

* On many operating systems, unprivileged clients may not send or receive packets on low-numbered ports. This means that any client sending or receiving mDNS packets on port 53 would have to run as "root", which is an undesirable security risk. Using a higher-numbered UDP port eliminates this particular problem.

20. Summary of Differences Between Multicast DNS and Unicast DNS

The value of Multicast DNS is that it shares, as much as possible, the familiar APIs, naming syntax, resource record types, etc., of Unicast DNS. There are of course necessary differences by virtue of it using Multicast, and by virtue of it operating in a community of cooperating peers, rather than a precisely defined authoritarian hierarchy controlled by a strict chain of formal delegations from the top. These differences are listed below:

Multicast DNS...
* uses multicast (of course!)
* uses UDP port 5353 instead of port 53
* operates in well-defined parts of the DNS namespace
* uses UTF-8, and only UTF-8, to encode resource record names
* allows more than one question in a query packet
* uses the Answer Section of a query to list Known Answers
* uses the TC bit in a query to indicate additional Known Answers
* uses the Authority Section of a query for probe tie-breaking
* ignores the Query ID field (except for generating legacy responses)
* uses gratuitous responses to announce new records to the peer group
* defines a "cache flush" bit in the rrclass of responses
* monitors queries to perform Duplicate Question Suppression
* monitors responses to perform Duplicate Answer Suppression...
* ... and Ongoing Conflict Detection
* ... and Opportunistic Caching

21. IPv6 Considerations

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

A dual-stack (v4/v6) host can participate in both ".local." zones, and should register its name(s) and perform its lookups both using IPv4 and IPv6. This enables it to reach, and be reached by, both IPv4-only and IPv6-only hosts. In effect this acts like a multi-homed host, with one connection to the logical "IPv4 Ethernet segment", and a connection to the logical "IPv6 Ethernet segment".

21.1 IPv6 Multicast Addresses by Hashing

Some discovery protocols use a range of multicast addresses, and determine the address to be used by a hash function of the name being sought. Queries are sent via multicast to the address as indicated by the hash function, and responses are returned to the querier via unicast. Particularly in IPv6, where multicast addresses are extremely plentiful, this approach is frequently advocated.

There are some problems with this:

* When a host has a large number of records with different names, the host may have to join a large number of multicast groups. This can place undue burden on the Ethernet hardware, which typically supports a limited number of multicast addresses efficiently. When this number is exceeded, the Ethernet hardware may have to resort to receiving all multicasts and passing them up to the host software for filtering, thereby defeating the point of using a multicast address range in the first place.

* Multiple questions cannot be placed in one packet if they don't all hash to the same multicast address.

* Duplicate Question Suppression doesn't work if queriers are not seeing each other's queries.

* Duplicate Answer Suppression doesn't work if responders are not seeing each other's responses.

* Opportunistic Caching doesn't work.

* Ongoing Conflict Detection doesn't work.

22. Security Considerations

DNSSEC [RFC 2535] should be used where the authenticity of information is important.

When DNS queries for global DNS names are sent to the mDNS multicast address (during network outages which disrupt communication with the greater Internet) it is *especially* important to use DNSSEC, because the user may have the impression that he or she is communicating with some authentic host, when in fact he or she is really communicating with some local host that is merely masquerading as that name. This is less critical for names ending with ".local.", because the user should be aware that those names have only local significance and no global authority is implied.

Most computer users neglect to type the trailing dot at the end of a fully qualified domain name, making it a relative domain name (e.g. "www.example.com"). In the event of network outage, attempts to positively resolve the name as entered will fail, resulting in application of the search list, including ".local.", if present. A malicious host could masquerade as "www.example.com" by answering the resulting Multicast DNS query for "www.example.com.local." To avoid this, a host MUST NOT append the search suffix ".local.", if present, to any relative (partially qualified) domain name containing two or more labels. Appending ".local." to single-label relative domain names is acceptable, since the user should have no expectation that a single-label domain name will resolve as-is.

23. IANA Considerations

The IANA has allocated the IPv4 link-local multicast address 224.0.0.251 for the use described in this document.

The IANA has allocated the IPv6 multicast address set FF0X::FB for the use described in this document.

When this document is published, IANA should designate a list of domains which are deemed to have only link-local significance, as described in this document.

No other IANA services are required by this document.

24. Copyright

25. Normative References

[RFC 1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987.

[RFC 1035] Mockapetris, P., "Domain Names - Implementation and Specifications", STD 13, RFC 1035, November 1987.

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

[RFC 2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

26. Informative References

[IEEE802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture. Institute of Electrical and Electronic Engineers, IEEE Standard 802, 1990.

[DNS-SD]   Cheshire, S. "DNS-Based Service Discovery", Internet-Draft (work in progress), draft-cheshire-dnsext-dns-sd-00.txt, December 2002.

[RFC 2136] Vixie, P., et al., "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC 2462] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC 2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[v4LL]   Cheshire, S., B. Aboba, and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", Internet-Draft (work in progress), draft-ietf-zeroconf-ipv4-linklocal-07.txt, August 2002.

[ZC]   Williams, A., "Requirements for Automatic Configuration of IP Hosts", Internet-Draft (work in progress), draft-ietf-zeroconf-reqts-12.txt, September 2002.

27. Author's Address

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

Phone: +1 408 974 3207
EMail: rfc@stuartcheshire.org