Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 5 October 2025

T. Lemon
Apple Inc.
E. Dijk
IOTconsultancy.nl

3 April 2025

Multicast DNS conflict resolution using the Time Since Received (TSR) EDNS option
draft-ietf-dnssd-tsr-00

Abstract

This document specifies a new conflict resolution mechanism for DNS, for use in cases where the advertisement is being proxied, rather than advertised directly, e.g. when using a combined DNS-SD Advertising Proxy and SRP registrar. A new EDNS option is defined that communicates the time at which the set of resource records on a particular DNS owner name was most recently updated.

Table of Contents

1. Introduction

Unlike the Domain Name System [RFC1034], with its authority servers and delegation of authority, Multicast DNS has no single source of authority. Because of this, mDNS has a mechanism, conflict resolution (Section 9 of [RFC6762]) for detecting and fixing conflicts in mDNS advertisements.

The current goal of mDNS conflict resolution is to prevent a newly advertised service from taking the place of an existing service with the same name that is already being advertised. This goal, however, assumes that the entity advertising an mDNS service is in fact authoritative for that service. In the case of an Advertising Proxy

[I-D.sctl-advertising-proxy], this is not the case: the source of truth for the service being advertised is an SRP [I-D.ietf-dnssd-srp] client.

On a link with more than one SRP registrar, an SRP client may register with one SRP registrar, and then subsequently update its registration on a different SRP registrar. Both SRP registrars may be acting as advertising proxies. If so, the original server may still be advertising the old SRP registration using mDNS. If the information in the new SRP registration is identical to that in the old registration, this is often not a problem. However if some information has changed (e.g., a new IP address has been added, or a TXT record updated), then the new registration will be seen to be in conflict with the old registration. In addition, the method used in mDNS to detect conflicts can sometimes produce apparent conflicts where no actual conflict exists because of the way records in mDNS packets are marshalled.

In the case of such an apparent conflict, the current behavior of mDNS is for the older (stale) registration to win, and the newer (current) information to be discarded. This behavior, which is entirely correct for services that are advertising on their own behalf, is exactly wrong when a service registration is being proxied.

1.1. Current Behavior

When a new service is to be advertised, the requestor (the entity requesting the registration) typically registers the service with a central mDNS registrar on the host on which it is running. This mDNS registrar may have an internal database of services already registered, and may detect a conflict with one of those services. This can be true whether the conflicting database entry is data for which the mDNS registrar is authoritative, or data it has received via mDNS and cached.

In the case of such a conflict, no network transaction is required: the mDNS registrar detects it locally. It addresses the conflict in one of two ways. The first alternative is that the mDNS registrar will report the conflict to the requestor as an error, which it must fix. Alternatively, the server requestor have indicated that the mDNS mDNS registrar should automatically choose a new name for it, in which case the mDNS registrar does so automatically, without notifying the requestor.

Once any locally-detectable conflicts have been resolved, the mDNS registrar probes (see Section 8.1 of [RFC6762]) local network to see if any other host has already registered a service the conflicts with

the proposed new service. If such a service is present on the network, the mDNS registrar follows the same process previously described, either reporting the error to the requestor or automatically choosing a new name.

The effect of this approach is that generally whichever requestor first registers a service under a particular name wins. If a requestor comes along later and registers the same service with conflicting information, the newcomer's information is rejected.

1.2. Problem Statement

The current behavior works well for requestors registering on their own behalf. However, for example in the case of an SRP registrar, it works poorly: an SRP registrar acting as an advertising proxy publishes the contents of its registration dataset(s) using mDNS. The source of truth for information in such datasets is the SRP requestor not the SRP registrar (which is acting in proxy as the mDNS requestor) itself.

In the case of an advertising proxy publishing an SRP dataset, what we want is not the oldest information, but the newest. When the SRP requestor is able to continue registering with the same SRP registrar, this works well: stale data is automatically removed and replaced with current data. However, if more than one SRP registrar is available, the requestor may wind up registering with a different SRP registrar. This can happen as a result of a network partition, or in cases where the SRP server is advertised on a anycast address.

When the SRP requestor registers with a different SRP registrar, the behavior we get with the current conflict resolution approach is that the SRP client will be given a new name, and both the old (stale) advertisement (A) and the new (more recent) advertisement (A') will be discoverable as separate services.

This creates a new burden on consumers of such services: they need to parse through the whole list of services of their type, using metadata from the TXT record in the service instance data, if possible, to determine that service A and service A' are the same service. If no such information is present in the TXT record, the only way to determine that one of these two registrations is stale is to attempt to use the advertised service, which may no longer be reachable if, for example, the change that produced the conflict was an IP address change. When the SRP lease for the stale service expires, that service's advertisement will be removed, and the service will no longer be discoverable under the original name, even if the IP address hasn't changed.

This document proposes an enhancement to the current conflict resolution algorithm for mDNS, which allows an mDNS proxy to report the time at which it received the registration it is newly advertising, and the source from which it was received. This is done using a new Time Since Received EDNS option, for which there must be exactly one per name being registered by the proxy.

2. Time Since Received EDNS Option

Each Time Since Received (TSR) EDNS option is applicable to exactly one DNS owner name. So all the records for that owner name that appear in the answer, authority and/or additional sections of an mDNS message would be covered by a single TSR option.

The TSR EDNS option consists of three fields: the RR index (two byte integer in network byte order), a key checksum (four bytes), and a time of registration (four bytes).

The RR index is the number of the RR in the mDNS packet. Question RRs are not counted. So if the message includes two answer RRs, one authority RR and two additional RRs, an index of 0 would refer to the first answer, an index of 1 to the second answer, and index of 2 to the single authority record, and so on. Questions are excluded because they have no data associated with them, and so it makes no sense for them to have TSR records associated with them.

If there is more than one record in the mDNS Message with the same owner name, only one TSR option is emitted for that name, and it applies to every RR in the mDNS Message with that owner name. It is not possible in the SRP protocol for two updates at two different times to contain records that apply to the same name: in such a situation, the second update completely replaces the first, so all data in the first update is then rendered stale.

The second field, the key checksum, is simple 32-bit checksum of the public key that the owner of the data (for example the SRP requestor) used to authenticate itself. The key checksum is computed by treating the key as a series of 32-bit unsigned integers in network byte order, and adding these integers together to produce a 32-bit unsigned checksum. Overflow is not considered. This checksum need not be cryptographically secure: mDNS messages are not authenticated, so an attacker on the local link can always cause problems with mDNS by providing spurious responses. The purpose of the checksum is simply to notice whether, for a specific owner name, two different authoritative sources have provided information.

The TSR time offset field contains the difference, in seconds, between the the time at which the TSR record is being generated and the time of receipt for recorded for that owner name.

The time of registration is represented in the mDNS message as a time in seconds relative to the time when the mDNS message is sent. If this difference is greater than seven days (7 * 24 * 60 * 60), the mDNS registrar MUST use a value of seven days rather than the larger value. The relative time represented in the TSR option is converted to an absolute time when stored in a cache or authority database on an mDNS registrar, and is converted to a relative time whenever an mDNS message is generated from local data.

3. mDNS Registrar Behavior

3.1. Validating requested local RR registrations that include a TSR option

When a local mDNS requestor asks an mDNS registrar to register one or more records on an owner name, and provides TSR data for that name, the mDNS requestor first checks to see if there are any records either in cache or from other local registrations on that owner name. If no such data exists, the mDNS registrar puts the record(s) in this registration in the probing state.

When such data exists, the registrar MUST check to see if it has TSR data for that owner name. If it does not, or if there is TSR data on that name but the key checksum does not match, the registrar MUST treat this registration as a conflict and return an appropriate error to the requestor.

If such data exists and the key checksums match, there are three possibilities based on the known TSR time and the proposed TSR time:

Known time is more recent In this case, the registrar MUST treat the new registration as stale, and return an indication to the requestor that its registration is stale. This indication must be different than the conflict indication.

Both times are the same In this case, the new record is added to the local registration database and put in the probing state.

Proposed time is more recent In this case, all cached data on the

name is discarded. The requestor for any existing locally-registered data is notified that the data they have registered is stale is stale, and the data is removed from the local registration database. The new data is added and put in the probing state, and the TSR data is updated with the proposed TSR data.

It is in principle possible for two different mDNS requestors to ask the same mDNS registrar to publish different RRs on the same name, some of which are shared and some of which are unique (see Section 2 of [RFC6762]). If an mDNS requestor registers an RR on a name for which the registrar already has data, cached or authoritative, on the same name, whether of the same type or a different type, for which there is no TSR data, or for which the key checksum in the TSR data being registered does not match what is already known, the registrar MUST treat this as an immediate conflict, and MUST NOT probe.

As with any local mDNS registration, the mDNS registrar treats all of the records in the registration as tentative (that is, in the probing state) until they have been probed and no conflicting answers have been received.

3.2. Probing resource records on names for which TSR data has been proposed

Section 8.1 of [RFC6762] describes how an mDNS registrar probes to ensure that there is no conflicting data for records in the probing state. The behavior for records that are in the probing state on names to which no TSR data applies is unchanged. When there is TSR data on a name for which records are being probed, the mDNS registar MUST include TSR options for each such name as described in Section 2. Handling of responses is described in Section 3.4.

3.3. Processing questions for which TSR data exists

When processing a question for which local TSR data is present, the mDNS registrar MUST first check to see if there is corresponding data in the mDNS message being processed. If there is, the question is part of a probe. In this case, before constructing a response, the mDNS registrar MUST process the non-question records in the packet, since this could result in stale data being flushed. Processing is performed as described in Section 3.4

Once all non-question records have been processed, the responder MUST respond to any questions that match locally-registered resource records for which a known answer is not present in the query. Responses are constructed as described in Section 2.

3.4. Processing messages containing TSR options

For each TSR option in an mDNS message, the mDNS registrar first determines the owner name of the TSR option by assigning an index to each non-question resource record in the mDNS message. The index of each TSR option is then matched to the index of a resource record, and the owner name for that resource record is applied to the TSR option. The time on the TSR option is then computed by taking the current local clock time and subtracting from it the time offset in the TSR record.

If there is a TSR option in an mDNS message for which there is no matching resource record in the mDNS message, the mDNS registrar MUST ignore that TSR option. The mDNS registrar MUST NOT use the index from the TSR option to search across the mDNS Packet since such an index can easily be out of bounds.

Now, for each record in the mDNS message, the mDNS registrar first determines whether the record is an OPT record, is in the question section, or is a known answer (QD bit = 0 and it’s a record in the answer section). For all such records, no special processing is done for TSRs, since no TSR should exist in the mDNS message.

For each remaining resource record in the mDNS message, the mDNS registrar MUST check to see if there is a TSR option in the mDNS message for that owner name. If there is not, the mDNS registrar MUST check to see if there is TSR data with that owner name locally. If there is not, the record is processed normally.

If there is local TSR data for the record’s owner name, the mDNS registrar checks to see if there are any resource records in the local registration database (that is, not just in the cache) on that name. If there are, the record is treated as a conflict. This conflict exists even if the locally registered records are all shared records. In cases where there are records on the name in the cache, those records are all discarded, because they are in conflict with the new data.

In the case that there is TSR data for the record in the mDNS packet, and no local TSR record, this always means that any data is in conflict. How that conflict is addressed depends on the data. First, note that resource records in the answer section of an mDNS Query (QR bit in the header is 0) are "known answers" and therefore are not relevant when adding data to the mDNSResponder cache. Such records can never have TSR options associated with them. However, resource records in the authority and additional sections of a query do need to be processed (but in the case of authority records, are not added to the cache).

In cases where the TSR data for a particular name is present both locally and in the mDNS message, the mDNS responder MUST compare the key checksums. If they are different, then the records are always in conflict, and are handled according to the context of the conflict, as described in Section 9 of [RFC6762]

In cases where the key checksums match, the mDNS registrar MUST compare the times. When the TSR time from the mDNS Message is more recent than the local TSR time, local data in the cache is flushed and registered data is removed and reported to the requestor that registered it as stale.

When the TSR times are the same, any resource records on that name in the answer section and additional section are added to the cache.

When the local TSR time is more recent, the data in the message is not added to the cache, and no action is taken with respect to any locally-registered data.

3.5. Constructing a mDNS message with TSR options

For each non-question record that is added to the mDNS message, one of three things must be true:

* The mDNS server is has resource records locally registered on that owner name, which may or may not be in the probing state.

* It is sending an answer which is either an announcement or a response containing data it has already validated and for which it is authoritative

* The message is a query (QD=0) and the record is in the answer section, and is therefore a "known answer."

As described in Section 7.1 of [RFC6762], an mDNS registrar asking a question about one or more RRs on a particular name populates the answer section of its mDNS message with the answers it already knows, to avoid unnecessary responses. However, in this case it can’t also be probing for records on the same name, because probes are only done for unique (non-shared) records.

The requirements in Section 3.1 mean that there can never be an mDNS probe that contains known answers on an owner name for which any RR is being probed to which a TSR option applies.

This means that for any particular owner name that might be represented in an mDNS packet, it must be the case either that it is not a known answer, or that it is a known answer and no other records

exist in the mDNS packet with the same owner name to which a TSR record would apply. That is, one of two things must be true about the set of all records with a particular owner name being added to the mDNS packet: either a TSR option applies to all of the records, or it applies to none of the records. Furthermore, either a record is a known answer from cache, or it is a locally-registred record.

When constructing an mDNS message, the registrar maintains a set of names and associated TSR data. Initially this set is empty. When the registrar adds a record to the mDNS message, if that record is locally registered, and if the registrar has TSR data for that name, it first checks to see if it has already added TSR data for that name to the set. If not, then it adds a new entry to the set containing the TSR data for the owner name of the RR. The data added consists of the owner name, the index of the record being added (since it is the first), the key checksum, and the time of receipt.

Once the mDNS responder has added all of the resource records it intends to to the mDNS message that is being constructed, it emits an OPT record in the additional section. To this OPT record it adds a TSR record for every name in the set that was generated when adding resource records to the message. The time of receipt is subtracted from the current time to prodiuce the time difference, and this is clamped to a maximum of seven days.

4. The effect of network latency on time computations

Because TSR computations are affected by network latency, comparisons can’ t be considered accurate. It is therefore necessary to tolerate some amount of error. In practice, however, it should generally not be the case that two advertising proxies receive SRP updates from the same SRP client at nearly the same time. So it should always be the case either that there is a clear ordering to the timestamps, or that there is no conflict in the data. For example with anycast, a retransmission could go to a different SRP registrar, but in this case both servers would simultaneously receive identical data, so the close ordering or even equality of the timestamps should not affect the outcome.

5. Internal Handling of TSR data

The TSR option that is sent on the wire is expressed in seconds relative to the time of receipt of the registration. In order to derive the time to send in a TSR option, the registrar must remember the time at which the registration occurred. This time is recorded as an absolute time, not a relative time. We refer to this as the time of receipt. When constructing a TSR option, the registrar computes the difference between the current time and the time of

receipt, which must always be in the past. This difference, which should be a positive integer, is converted to seconds, and that unsigned value is then used to synthesize the TSR RR.

6. Timeliness of Conflict Resolution

It is expected that if a conflict exists, it will be recent, and will be resolved quickly. Different hosts may be able to record shorter or longer time differences. However, because of this expectation of recentness, mDNS registrars should never need to report a TSR of longer than seven days. It' s reasonable to expect that every mDNS implementation should be able to remember time intervals of at least seven days.

7. Legacy Behavior

y mDNS registrars and queriers that do not support the TSR option are expected to ignore the option, so they will behave as if no TSR option was sent. This may result in such registrars temporarily caching stale data. However, in the normal course of processing, more recent data will win. In cases where it does not, the Reconfirm process which is part of [RFC6762] already works to clear stale data: since we expect SRP servers to implement TSR, by the time a Reconfirm is attempted, all authoritative stale data should have been cleared.

8. When to Use TSR

TSR is only relevant for mDNS proxies. Regular (non-proxy) mDNS registrants are not expected to use it, since it will produce the wrong behavior for this use case. An mDNS registrant that is a proxy MUST explicitly request that a TSR be used for conflict resolution. mDNS registrars MUST NOT record a time of receipt unless the registrant has specifically requested it.

9. Registrant API considerations

When an mDNS proxy registers a service and requests the use of a time of receipt, the proxy MUST specify when it received the registration. In order to support this, the API is required not only to allow the registrant to specify that TSR conflict resolution is wanted, but must also provide a way for the proxy to specify an absolute time at which the registration was received, and the key checksum used to identify the entity that's actually authoritative for the data.

This is important, for example, in the case of SRP Replication [I-D.lemon-srp-replication], where an SRP registrar may receive a registration from a peer during startup synchronization. This registration will have occurred at some significant amount of time in

the past, and so it would be incorrect for the mDNS proxy receiving the registration to use the time that the mDNS proxy registers the service as the time of receipt.

10. Security Considerations

The TSR option is an optimization: it ameliorates an edge case for mDNS proxies. A malicious host on the same link could use the TSR option to win conflict resolution processes. However, because TSR is only used by proxies, this technique will not work for normal mDNS service registrations: in that case, normal mDNS conflict resolution is done, and the attacker gains no benefit from using TSR.

Whether or not an mDNS registration has a recorded time of receipt, an attacker can deny service by announcing its own conflicting data and then answering the subsequent probe as described in Section 9 of [RFC6762]. Because it does not include a TSR record in its authority section, it can win the simultaneous conflict resolution process that follows its bogus announcement.

So the TSR-based conflict resolution process creates no new vulnerability. Addressing the existing vulnerability is out of scope for this document. Protocols that rely on mDNS MUST NOT assume that mDNS service is secure or private. If security (authentication, authorization and/or secrecy) are needed, these must be provided at the application layer, or by using DNSSEC rather than mDNS for service discovery.

11. IANA Considerations

IANA is requested to allocate a new OPT RR option code from the DNS EDNS0 Option Codes (OPT) registry for the ’Time Since Received’ Option. The Name shall be ’mDNS-TSR’. The value shall be allocated by IANA. The meaning shall be ’Multicast DNS Time Since Received". Reference shall refer to this document, once published. IANA shall determine the registration date.

12. Informative References

13. Normative References

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

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

[I-D.lemon-srp-replication]
Lemon, T., Keshavarzian, A., and J. Hui, "Automatic Replication of DNS-SD Service Registration Protocol Zones", Work in Progress, Internet-Draft, draft-lemon-srp-replication-03, 22 February 2023, <https://datatracker.ietf.org/doc/html/draft-lemon-srp-replication-03>.

[I-D.sctl-advertising-proxy]
Cheshire, S. and T. Lemon, "Advertising Proxy for DNS-SD Service Registration Protocol", Work in Progress, Internet-Draft, draft-sctl-advertising-proxy-02, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-sctl-advertising-proxy-02>.

[I-D.ietf-dnssd-srp]
Lemon, T. and S. Cheshire, "Service Registration Protocol for DNS-Based Service Discovery", Work in Progress, Internet-Draft, draft-ietf-dnssd-srp-27, 18 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-dnssd-srp-27>.

Authors' Addresses

Ted Lemon
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America
Email: mellon@fugue.com

Esko Dijk
IOTconsultancy.nl
Email: esko.dijk@iotconsultancy.nl