Workgroup: Internet Engineering Task Force
Published: 26 July 2021
Intended Status: Standards Track
Expires: 27 January 2022
Authors: T. Lemon
Apple Inc.

Automatic Replication of DNS-SD Service Registration Protocol Zones

Abstract

This document describes a protocol that can be used for ad-hoc replication of a DNS zone by multiple servers where a single primary DNS authoritative server is not available and the use of stable storage is not desirable.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The DNS-SD Service Registration Protocol provides a way for network services to update a DNS zone with DNS-SD information. SRP uses unicast DNS Updates, rather than multicast DNS, to advertise services. This has several advantages over multicast DNS:

* Reduces reliance on multicast

* Reduces traffic to devices providing services, which may be constrained devices operating on battery power

* Allows the advertisement of services on one network link to consumers of such services on a different network link

1.1. Alternatives for maintaining SRP state

1.1.1. Primary authoritative DNS service

Ideally, SRP updates a primary authoritative DNS server for a particular zone. This DNS server acts as the sole source of truth for names within the DNS zone in which SRP services are published. Redundancy is provided by secondary DNS servers, if needed. However, this approach has some drawbacks.

First, it requires 100% availability on the part of a DNS primary authoritative server for the zone. If the primary server is not available for some period of time, new services appearing on the network cannot be registered until primary authoritative service is restored.

The second drawback is that there is no automatic method for managing DNS authoritative service. This means that such a service requires an operator to set it up. What it means to set up such a service is that the following capabilities are provided:

* An host must be available to act as a primary authoritative DNS server

* The zone advertised by that server must be delegated, so that the local resolver can successfully answer queries in that zone

* The local resolver must be able to provide local browsing domain advertisements [RFC6763 section 11].

1.1.2. Multicast DNS Advertising Proxy

An existing alternative to the use of DNS authoritative services for advertising SRP registrations the advertising proxy [draft-tlsc-advertising-proxy]. An advertising proxy advertises the contents of the SRP update zone using multicast DNS on links on which the need for such advertisements is anticipated. This works well for stub networks [draft-lemon-stub-networks], where services advertised on the stub network must be visible both on the stub network and on the adjacent infrastructure network, but do not generally need to be discoverable on other networks.

One drawback of the advertising proxy model, however, is that there is no shared database from which to advertise services registered by SRP. As a consequence, some of the guarantees provided by SRP, particularly first come, first served naming [draft-ietf-dnssd-srp]. Because advertising proxies are set up automatically on an ad-hoc basis, coordination between advertising proxies is not present, which means that if two devices claim the same name, but register with different SRP servers, the conflict is not detected until the service is advertised using mDNS. In practice, this results in frequent renaming of services, which means that consumers of services need to carefully follow each service that they use as the name changes over time.

An additional drawback is that, from the perspective of the SRP client, SRP service is not unified: SRP servers tend to come and go, and whenever the SRP service with which a particular client has registered goes offline, the client has to notice that this has happened, discover a new SRP server, and re-register, or else it becomes unreachable.

1.1.3. SRP Replication

This document describes a replication mechanism which eliminates the need for a single authoritative source of truth, as in the Primary Authoritative DNS model, while eliminating the drawbacks of the Advertising Proxy model. SRP Replication servers discover each other automatically. Each replication server maintains a copy of the SRP zone which is kept up to date on a best-effort basis.

SRP Replication has several benefits:

* As long as one SRP replication peer remains online at all times, SRP state is maintained when individual SRP replication peers go offline

* Name collisions when SRP clients change servers are avoided

* SRP service on a stub network can appear as an anycast service, so that SRP clients do not see an apparent change in servers and re-register when the server with which they most recently registered goes offline

1.2. Implementation

SRP Replication relies on the fact that any given client is always registering with exactly one SRP server at any given time. This means that when an SRP server receives an SRP update from a client, it can be sure that no other SRP server has a more recent version of that SRP client's registration. Consequently, that SRP server can behave as if it is the source of truth for that client's registration, and other SRP servers can safely assume that any data they have about the client that is less recent can be replaced with the new registration data.

1.2.1. Naming of a common service zone

In order for SRP replication peers to replicate a zone, they must agree upon a common name for the zone. We will describe two mechanisms for agreeing on a common zone here.

1.2.1.1. Zone name based on network name

Network names aren't guaranteed to be unique, but tend to be unique for any given site. In the case of ad-hoc (permissionless) SRP-based service, such as an advertising proxy or an authoritative service using a locally-served zone [https://www.iana.org/assignments/locally-served-dns-zones/locally-served-dns-zones.xhtml], because the DNS zone name isn't required to be globally unique, a zone name based on the network name is an easy solution to generating a unique zone name.

When generating a zone name based on a network name, the zone name could be based on a locally configured global zone name, e.g. 'example.com'. It could be based on a locally-managed locally-served name, e.g. 'home.arpa'. Or it could be based on an unmanaged locally-served name, for which we propose to use the root name 'local.arpa.' For the rest of this section we will assume that the specific setting determines which of these domains will be used, and refer to whichever domain that is as DOMAIN.

For zone names based on the network name, the network type should be used as a differentiator, in case there are two different local network types with the same name. So, for example, 'WiFi.DOMAIN.'

1.2.1.1.1. Zone name based on WiFi SSID

If the zone being represented is a WiFi network, then the zone name for the network should be constructed using the WiFi SSID followed by 'WiFi.DOMAIN'. For example, if the SSID is "Example Home" then the zone name would be 'Example Home.WiFi.DOMAIN.' Note that spaces and special characters are allowed in domain names.

1.2.1.1.2. Zone name based on Thread network name

If the zone being represented is a Thread [Thread] network, then the zone name for the network should be constructed using the Thread network name. For example, if the Thread network name is "openthread" then the zone name would be 'openthread.thread.DOMAIN.'

1.2.1.2. Zone name based on local configuration

The above examples assume that it makes sense for each separate subnet to be its own separate zone. However, since SRP guarantees name uniqueness using the first-come, first-served mechanism, it doesn't rely on mDNS's guarantee of per-link uniqueness. Consequently, it is not required that an SRP zone be constrained to the set of services advertised on a single link. For this reason, when it is possible to know that some set of links are all managed by the same set of SRP replication peers, and a name is known for that set of links, that name can be used. To avoid possible collisions, the subdomain 'srp' is used to indicate that this zone is an SRP zone. So in this case the link name would be the locally-known shared name, followed by 'srp.DOMAIN.'

An example of such a scenario would be Apple's HomeKit, in which all HomeKit accessories, regardless of which home network link they are attached to, all are shared in the same namespace. Suppose the HomeKit home's name is "Example Home". In such a situation, the domain name 'Example Home.srp.DOMAIN' could be used.

1.2.1.3. Zone name based on DNS-SD discovery

Another option for naming the local SRP Replication zone would be to use DNS-SD advertisements. This is particularly useful since each SRP replication peer advertises itself using DNS-SD, so there is a convenient place to put this information. To advertise a zone name based on DNS-SD discovery, the SRP Replication peer should add two fields to the TXT record of the service instance. The first field is the domain field: 'domain=name'. This indicates a proposed SRP replication zone name. The second is the join field. If 'join=yes' then other SRP replication servers are encouraged to use the domain name that appears in the domain field rather than creating a new domain.

1.2.2. Advertising one's own replication service

SRP replication service is advertised using DNS-SD [RFC6763]. The service name is '_srpl-tls._tcp'. Each SRP replication peer should have its own hostname, which when combined with the service instance name and the local DNS-SD domain name will produce a service instance name, for example 'example-host._srpl-tls._tcp.local.' The domain under which the service instance name appears will be 'local' for mDNS, and will be whatever domain is used for service registration in the case of a non-mDNS local DNS-SD service.

SRP replication uses DNS port 853 [RFC7858] and is based on DNS Stateful Operations [RFC8490]. Therefore, the SRV record for the example we've given would be:

```
example-host._srpl-tls._tcp.local. IN SRV 0 0 853 example-
host.local.
```

The TXT record for SRP replication advertises the domain being replicated, permission to join (if applicable), and the server identifier of the SRP replication peer. The server identifier is a 64-bit number encoded as hexadecimal ASCII, produced with a high-quality random number generator [RFC4086]. This identifier need not be persistent across SRP replication peer restarts. So in our example the TXT record might look like this:

```
#domain=openthread.thread.home.arpa.\032server-id=eb5bb51919a15cec
```

(Note that each name/value pair in the TXT record is length-encoded, so the '#' and the '\032' are the lengths of the two name/value pairs.)

1.2.3. Discovering other replication services

An SRP Replication Peer MUST maintain an ongoing DNS-SD browse on the service name '_srpl-tls._tcp' within the local browsing domain. The ongoing browse will produce two different types of events: "add" events and "remove" events. When the browse is started, it should produce an 'add' event for every SRP replication partner currently present on the network, including the peer that is doing the browsing. Whenever a partner goes offline, a 'remove' event should be produced. 'remove' events are not guaranteed, however.

When a new service is added, the SRP peer checks to see if it is in a compatible domain. If the SRP peer has a domain to advertise, it compares that domain to the domain advertised in the added service instance: if they are not the same, then this instance is not a candidate for connection, and should be ignored.

If the SRP peer does not have a domain to advertise, then when it begins to browse for partners, it sets a timer for DOMAIN_DISCOVERY_TIMEOUT seconds.

If the SRP peer does not have a domain to advertise, and is therefore willing to join an existing domain, it checks to see if the TXT record for the service indicates that joining is permitted. If so, the SRP peer adopts the provided domain name. Once it has adopted such a domain name, it updates its own TXT record to indicate that domain name, and sets the 'join=yes' key/value pair in the TXT record. It also cancels the DOMAIN_DISCOVERY_TIMEOUT timer.

If the DOMAIN_DISCOVERY_TIMEOUT timer goes off, then the SRP peer MUST propose a zone name using one of the methods mentioned previously. It advertises that zone name in its TXT record, with 'join=yes'. It then sets a new timer for DOMAIN_PROPOSE_TIMEOUT seconds.

While waiting for the DOMAIN_PROPOSE_TIMEOUT timer to go off, any new 'add' events that arrive are examined to see if they are potential domains to join. If a potential domain to join is seen, and it is the same as the proposed domain, then the peer adopts that domain and treats it as its domain to advertise. It then cancels the DOMAIN_DISCOVERY_TIMEOUT timer.

When the DOMAIN_DISCOVERY_TIMEOUT timer expires, the peer initializes the domain to be advertised using the one that it chose, and the chosen server-id to be its own. It then iterates across the list of 'add' events that have been seen. Each advertisement is examined, comparing its server-id to the chosen server-id. If the chosen server-id is numerically greater than the server-id in the advertisement, then the domain to be advertised and the chosen server-id are updated from the advertisement. At the end of this process, the peer adopts whatever domain is now set as the domain to be advertised.

Once a domain has been chosen, a list of partners in that domain can be generated from the list of add events previously seen. When a new add event is seen that advertises the peer's domain to be advertised, that partner is added to the list of partners, if not already present. When a remove event is seen, if that partner is on the list of partners, a timer is set for DOMAIN_INSTANCE_TIMEOUT seconds.

When the timer for DOMAIN_INSTANCE_TIMEOUT timer expires, if the partner that was removed has not been re-added, it is removed from the list of partners and any connection to it is dropped.

1.2.4. Discovering the addresses of peers

When a partner is discovered, two new ongoing mDNS queries are started on the hostname indicated in the SRV record of the partner: one for A records, and one for AAAA records. Each time an address 'add' event is seen, either for an 'A' record or an 'AAAA' record, the peer adds the address to the list of addresses belonging to that partner.

1.2.5. Establishing Communication with a replication peer

When an address is added to a partner's address list, the peer first checks to see if the address is one of its own addresses. If so, then the partner is marked "me", and no connection is attempted to it. This is somewhat safer than comparing hostnames, since a hostname collision can result in renaming.

If the partner is not marked 'me', then the peer checks to see if it has an existing outgoing connection to that partner. If it does not, then it checks to see whether it has disabled outgoing connections to that partner. If not, then it attempts to connect on the new address.

When a connection fails, it advances to the next address in the list, if there is one. If there are no remaining addresses, the peer sets a timer for RECONNECT_INTERVAL seconds. When this timer expires, it starts again at the beginning of the list and attempts to connect to the first address, iterating again across the list until a connection succeeds or it runs out of addresses.

Additionally, when an address is added, it is checked against the list of unidentified incoming connections. If a match is found, and the partner is marked "me," then the unidentified connection is removed from the list and dropped. Otherwise, it is attributed to the matching partner, and the protocol is started at the point of receiving an incoming connection.

When an outgoing connection succeeds, the peer sends its server ID.

1.2.6. Incoming connections

When an incoming connection is received, it is checked against the partner list based on the source address of the incoming connection. If the address appears on the list of addresses for a partner, then the connection is attributed to that partner. If no matching partner is found, a timer of UNIDENTIFIED_PARTNER_TIMEOUT seconds is set, and the incoming connection is added to the list of "unidentified" connections.

If a matching partner is found, then the peer waits for an incoming partner ID. When such an ID is received, it is compared to the peer's server-id. If the incoming server ID is the same as or greater than the peer's server ID, the connection is dropped. Otherwise, the connection proceeds to the "initial synchronization" state.

1.2.7. Eliminating extra connections

When an outgoing connection succeeds, the peer sends its server ID to the partner. When an incoming connection succeeds, the peer waits for a server ID. Because both connections are peer connections, and we only need one connection, the peer with the higher server ID acts as the client and the peer with the lower server ID acts as the server. If the server IDs are equal, then the connecting server generates a new server ID, updates its TXT record, and re-does the comparison.

1.2.8. Initial synchronization

The connecting peer begins the session by sending its server ID. The receiving peer waits for a server ID, and when it receives one, does the server ID comparison mentioned earlier. If the connection survives the comparison, then the server sends a response to the session message and waits for the client to request a list of update candidates.

The connecting peer waits for a response to the initial session message, and when it is received, requests that the server send candidates.

1.2.8.1. Sending candidates

When a peer receives a "send candidates" message that it is expecting to receive, it generates a candidate list from the list of known SRP clients. This list includes SRP clients that have registered directly with the peer, and SRP clients that have been received through SRP replication updates. Each candidate contains a hostname, a time offset, and a key identifier.

The key identifier is computed as follows:

```
uint32_t key_id(uint8_t *key_data, int key_len) {
  uint32_t key_id = 0;
  for (int i = 0; i < key_data_len; i += 4) {
     key_id += ((key_data[i] << 24) | (key_data[i + 1] << 16) |
                (key_data[i + 2] << 8) | (key_data[i + 3]));
  }
  return key_id;
}
```

When a peer receives a candidate message during the synchronization process, it first searches for an SRP registration with a hostname that matches the hostname in the candidate message. It then compares the key ID to the key ID in the candidate message. If the key ID doesn't match, it sends back a candidate response status of "conflict". If the key ID does match, it compares the time provided to the time the existing host entry was received. If the time of the update is later, it sends a "send host" response. If it is earlier or the same, it sends a "continue" response. If there is no matching host entry for the candidate message, the peer sends a "send host" response.

When a peer receives a candidate response with a status of "send host", it generates a host message, which contains the hostname, the time offset, and the SRP message that was received from the host. The peer then applies the SRP update message as if it had been received directly from the SRP client. The host update time sent by the partner is remembered as the time when the update was received from the client, for the purposes of future synchronization.

When a peer is finished iterating across its list of candidates, it sends a "send candidates" response.

When a peer receives a "send candidates" response, if it is the server, it sends its own "send candidates" message, and processes any proposed candidates.

When a peer that is a server receives a "send candidates" response, it goes into the "routine operation" state. When a peer that is a client sends its "send candidates" response, it goes into the "routine operation" state.

1.2.9. Routine Operation

During routine operation, whenever an update is successfully processed from an SRP client, the peer that received that update queues that update to be sent to each partner to which it has a connection, whether server or client. If there are no updates pending to a particular client, the update is sent immediately. Otherwise, it's send when the outstanding update is acknowledged.

When during routine operation a peer receives a host update from its partner, it immediately applies that update to its local SRP zone. This is based on the assumption that a new update is always more current than a copy of the host information in its database.

2. Protocol Details

The DNS-SD SRP Replication Protocol (henceforth SRPL) is based on DNS Stateful Operations [RFC8490]. Each SRP replication peer creates a listener on port 853, the DNS-over-TLS [RFC7858] reserved port. This listener can be used for other DNS requests as well.

Participants in the protocol are peers. To distinguish between peers, the terms "peer" and "partner" are used. "Peer" refers to the peer that is communicating or receiving communication. "Partner" refers to the other peer. Peers can be clients or servers: a peer that has established a connection to a partner is a client; a peer that has received a connection from a partner is a server.

2.1. DNS Stateful Operations considerations

DNS Stateful Operations is a DNS per-connection session management protocol. DNS Push session management includes session establishment as well as session maintenance.

2.1.1. DSO Session Establishment

An DSO session for an SRPL connection can be established either by simply sending the first SRPL message, or by sending a DSO Keepalive message. Section 5.1 of [RFC8490].

2.1.2. DSO Session maintenance

DSO sessions can be active or idle. As long as the SRPL protocol is active on a connection, the DSO state of the connection is active. DSO sessions require occasional keepalive messages. The default of fifteen seconds is adequate for SRPL.

An idle DSO session must persist for long enough that there is a chance for the browse that identifies it to succeed. Therefore, the minimum DSO session inactivity timeout is 2*UNIDENTIFIED_PARTNER_TIMEOUT seconds.

2.2. DSO Primary TLVs

Each DSO message begins with a primary TLV, and contains secondary TLVs with additional information. The primary TLVs used in the SRPL protocol are as follows:

2.2.1. SRPL Session

DSO-TYPE code: SRPLSession. Introduces the SRPL session. In addition to the header and length, the SRPL Session message includes a server ID, which is a 64-bit unsigned number in network byte order. The SRPL Session primary TLV does not include any secondary TLVs. SRPL Session requests are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.1.1. SRPL client behavior

The SRPL Session request is sent by a peer acting as a client to its partner once the TLS connection to the partner, acting as a server, has succeeded. The SRPL session message establishes the DSO connection as an SRP protocol connection. If it is the first DSO message sent by the peer acting as a client, then it also establishes the DSO session.

When the SRPL peer acting as a client receives a response to its SRPL session message, it sends an SRPL Send Candidates message.

2.2.1.2. SRPL server behavior

An SRPL peer acting as a server that receives an SRPL Session request checks to see if the connection on which it was received is already established. If so, this is a protocol error, and the SRPL peer MUST drop the connection.

It then compares the server ID sent by the partner to its own server ID. If the partner's server ID is numerically less than the server's server ID, the server MUST drop the connection.

If the server ID of the peer is identical to the partner's server ID, then the server generates a new server ID and updates its TXT record with the new server ID.

If the peer acting as a server did not drop the incoming connection, then it sends an SRPL Session response containing its current server ID.

2.2.2. SRPL Send Candidates

DSO-TYPE code: SRPLSendCandidates. Requests the peer to send its candidates list. The SRPL Send Candidates message contains no additional data. The SRPL Send Candidates primary TLV does not include any secondary TLVs. SRPL Send Candidates messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.2.1. SRPL client behavior

An SRPL peer acting as a client MUST send an SRPL Send Candidates request after it has received an SRPL Session response. It MUST NOT send this request at any other time.

An SRPL peer acting as a client expects to receive an SRPL Send Candidates message after it has received an SRPL Send Candidates response. If it receives an SRPL Send Candidates message at any other time, this is a protocol error, and the SRPL peer should drop its connection to the server.

2.2.2.2. SRPL server behavior

An SRPL peer acting as a server expects to receive an SRPL Send Candidates request after it has sent an SRPL Session response. If it receives an SRPL Candidates request at any other time, this is a protocol error, and it MUST drop the connection.

An SRPL peer acting as a server MUST send an SRPL Send Candidates request after it has sent an SRPL Send Candidates response.

An SRPL peer acting as a server MUST enter the "normal operations" state after receiving an SRPL Send Candidates response from its partner.

2.2.3. SRPL Candidate

DSO-TYPE code: SRPLCandidate. Announces the availability of a specific candidate SRP client registration. The SRPL Candidate message contains no additional data. SRPL Candidate messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.3.1. Required secondary TLVs

The SRPL Candidate request MUST include the following secondary TLVs: SRPL Hostname, SRPL Time Offset, and SRPL Key ID. If an SRPL peer receives an SRPL Candidate request that doesn't contain all of these secondary TLVs, this is a protocol error, and the peer MUST drop the connection.

The SRPL Candidate response MUST include one of the following status TLVs: SRPL Candidate Yes, SRPL Candidate No, or SRPL Conflict. If an SRPL peer receives an SRPL Candidate response which does not contain exactly one of these TLVS, this is a protocol error, and the peer MUST drop the connection.

2.2.3.2. SRPL peer common behavior

SRPL peers expect to receive SRPL Candidate messages between the time that they have sent an SRPL Send Candidates message and the time that they have received an SRPL Send Candidates response. If an SRPL Candidate message is received at any other time, this is a protocol error, and the peer MUST drop the connection.

Peers MUST NOT send SRPL Candidate requests if they have sent any SRPL Candidate or SRPL host requests that have not yet received responses. Peers receiving SRPL Candidate requests when they have not yet responded to an outstanding SRPL Candidate request or SRPL Host request MUST treat this as a protocol failure and drop the connection.

When a peer receives a valid SRPL Candidate message, it checks its SRP registration database for a host that matches both the SRPL Hostname and SRPL Key ID TLVs. If such a match is not found, the peer sends an SRPL Candidate response that includes the SRPL Candidate Yes secondary TLV.

If a match is found for the hostname, but the Key ID doesn't match, this is a conflict, and the peer sends an SRPL Candidate response with the SRPL Conflict secondary TLV.

If a match is found for the hostname, and the key ID matches, then the peer computes the update time of the candidate by subtracting the value of the SRPL Time Offset TLV from the current time in seconds. This computation should be done when the SRPL Candidate message is received to avoid clock skew. If 'candidate update time' - 'local update time' is greater than SRPL_UPDATE_SKEW_WINDOW, then the candidate update is more recent than the current SRP registration. In this case, the peer sends an SRPL Candidate response and includes the SRPL Candidate Yes secondary TLV. The reason for adding in some skew is to account for network transmission delays.

2.2.4. SRPL Host

DSO-TYPE code: SRPLHost. Provides the content of a particular SRP client registration. The SRPL Host message contains no additional data. SRPL Host messages are DSO requests: the recipient is expected to send a response TLV. Both request and response TLVs have the same format.

2.2.4.1. Required secondary TLVs

The SRPL Host request MUST include the following secondary TLVs: SRPL Hostname, SRPL Time Offset, and SRPL Key ID. If an SRPL peer receives an SRPL Candidate request that doesn't contain all of these secondary TLVs, this is a protocol error, and the peer MUST drop the connection.

2.2.4.2. SRPL peer common behavior during synchronization

SRPL peers expect to receive either zero or one SRPL Host requests after sending an SRPL Candidate response with a SRPL Candidate Yes secondary TLV. If an SRPL Host request is received at any other time during synchronization, this is a protocol error, and the peer MUST drop the connection. The only time that an SRPL Host request would *not* follow a positive SRPL Candidate response would be when the candidate host entry's lease expired after the SRPL Candidate request was sent but before the SRPL Candidate response was received.

SRPL peers send SRPL Host requests during synchronization when a valid SRPL Candidate response has been received that includes an SRPL Candidate Yes secondary TLV. The host request is generated based on the current candidate (the one for which the SRPL Candidate request being responded to was send).

2.2.4.3. SRPL peer common behavior during normal operations

When an SRPL peer during normal operations receives and has successfully validated an SRP update from an SRP client, it MUST send that update to each of its connected partners as an SRPL Host request. If the connection to a particular partner is not busy, and there are no updates already queued to be sent, it MUST send the SRPL Host message immediately. Otherwise, it MUST queue the update to send when possible. The queue MUST be first-in, first-out.

After an SRPL peer has sent an SRPL Host request to a partner, and before it receives a corresponding SRPL Host response, it MUST NOT send any more SRPL Host messages to that partner.

When an SRPL peer receives an SRPL Host request during normal operations, it MUST apply it immediately. While it is being applied, it MUST NOT send any other SRPL Host requests to that peer.

When an SRPL Host request has been successfully applied by an SRPL peer, the peer MUST send an SRPL Host response.

If an SRPL peer receives an SRPL Host request while another SRPL Host request is being processed, this is a protocol error, and the peer MUST drop the connection to its partner.

2.3. DSO Secondary TLVs

In addition to the Primary TLVs used to send requests between SRPL peers, we define secondary TLVs to carry formatter information needed for various SRPL requests.

2.3.1. SRPL Candidate Yes

DSO-TYPE code: SRPLCandidateYes. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is wanted and should be sent.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Conflict or SRPL Candidate No secondary TLVs.

2.3.2. SRPL Candidate No

DSO-TYPE code: SRPLCandidateNo. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is not wanted and should not be sent.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Conflict or SRPL Candidate Yes secondary TLVs.

2.3.3. SRPL Conflict

DSO-TYPE code: SRPLConflict. In an SRPL Candidate response, indicates to the partner that an SRPL Host message for the candidate is not wanted and should not be sent. Additionally indicates that the proposed host conflicts with local data. This indication is informative and has no effect on processing.

Appears as a secondary TLV in SRPL Candidate responses. MUST NOT appear in any other SRPL request or response. MUST NOT appear in addition to either SRPL Candidate Yes or SRPL Candidate No secondary TLVs.

2.3.4. SRPL Hostname

DSO-TYPE code: SRPLHostname. In an SRPL Candidate or SRPL Host request, indicates to the partner the hostname of an SRP registration.

Required as a secondary TLV in SRPL Candidate and SRPL Host requests. MUST NOT appear in any other SRPL request or response.

2.3.5. SRPL Host Message

DSO-TYPE code: SRPLHostMessage. In an SRPL Host request, conveys the literal contents of the SRP update that resulted in the SRP Host registration being updated. The content of the SRPL Host Message is used to update the host on the peer receiving the request. Note that the SRP message being sent can't be modified by the SRPL peer sending it, so in order to validate the message (assuming that the signature includes a nonzero time), the validation process should adjust the current time by the time offset included in the SRPL Time Offset TLV when comparing against the signature time when checking for replay attacks. The computation of the current time of signing should be done when the message is received to avoid clock skew that might result from processing delays.

Required as a secondary TLV in SRPL Host requests. MUST NOT appear in any other SRPL request or response.

2.3.6. SRPL Time Offset

DSO-TYPE code: SRPLTimeOffset. In an SRPL Candidate or SRPL Host request, conveys the difference between the time the SRP update was received from the SRP client and the current time on the peer generating the request, in seconds.

Required as a secondary TLV in SRPL Candidate and SRPL Host requests. MUST NOT appear in any other SRPL request or response.

2.3.7. SRPL Key ID

DSO-TYPE code: SRPLKeyID. In an SRPL Candidate, conveys the key ID of the SRP client.

Required as a secondary TLV in SRPL Candidate requests. MUST NOT appear in any other SRPL request or response.

3. Security Considerations

SRP replication basically relies on the trustworthiness of hosts on the local network. Since SRP itself relies on the same level of trust, SRP replication doesn't make things worse. However, when the option to have a central SRP server is available, that is likely to be more trustworthy.

4. Delegation of 'local.arpa.'

In order to be fully functional, the owner of the 'arpa.' zone must add a delegation of 'local.arpa.' in the '.arpa.' zone [RFC3172]. This delegation should be set up as was done for 'home.arpa', as a result of the specification in Section 7 of [RFC8375].

5. IANA Considerations

5.1. 'srpl-tls' Service Name

IANA is requested to add a new entry to the Service Names and Port Numbers registry for srpl-tls with a transport type of tcp. No port number is to be assigned. The reference should be to this document, and the Assignee and Contact information should reference the authors of this document. The Description should be as follows:

Availability of DNS-SD SRP Replication Service for a given domain is advertised using the "_srpl-tls._tcp.<domain>." SRV record gives the target host and port where DNS-SD SRP Replication Service is provided for the named domain.

5.2. DSO TLV type code

The IANA is requested to add the following entries to the 16-bit DSO Type Code Registry. Each type mnemonic should be replaced with an allocated type code, both in this table and elsewhere in the document. RFC-TBD should be replaced with the name of this document once it becomes an RFC.

Type	Name	Early Data	Status	Reference
SRPLSession	SRPL Session	No	STD	RFC-TBD
SRPLSendCandidates		No	STD	RFC-TBD

Type	Name	Early Data	Status	Reference
	SRPL Send Candidates			
SRPLCandidate	SRPL Candidate	No	STD	RFC-TBD
SRPLHost	SRPL Host	No	STD	RFC-TBD
SRPLCandidateYes	SRPL Candidate Yes	No	STD	RFC-TBD
SRPLCandidateNo	SRPL Candidate No	No	STD	RFC-TBD
SRPLConflict	SRPL Conflict	No	STD	RFC-TBD
SRPLHostname	SRPL Hostname	No	STD	RFC-TBD
SRPLHostMessage	SRPL Host Message	No	STD	RFC-TBD
SRPLTimeOffset	SRPL Time Offset	No	STD	RFC-TBD
SRPLKeyID	SRPL Key ID	No	STD	RFC-TBD

Table 1

5.3. Registration and Delegation of 'local.arpa' as a Special-Use Domain Name

IANA is requested to record the domain name local.arpa.' in the Special-Use Domain Names registry [SUDN]. IANA is requested, with the approval of IAB, to implement the delegation requested in Section 4.

IANA is further requested to add a new entry to the "Transport-Independent Locally-Served Zones" subregistry of the the "Locally-Served DNS Zones" registry [LSDZ]. The entry will be for the domain local.arpa.' with the description "Ad-hoc DNS-SD Special-Use Domain", listing this document as the reference.

6. Informative References

7. Normative References

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <https://www.rfc-editor.org/info/rfc3172>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport

Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC8375] Pfister, P. and T. Lemon, "Special-Use Domain 'home.arpa.'", RFC 8375, DOI 10.17487/RFC8375, May 2018, <https://www.rfc-editor.org/info/rfc8375>.

[RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <https://www.rfc-editor.org/info/rfc8490>.

[SUDN] "Special-Use Domain Names Registry", July 2012, <https://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml>.

[LSDZ] "Locally-Served DNS Zones Registry", July 2011, <https://www.iana.org/assignments/locally-served-dns-zones/locally-served-dns-zones.xhtml>.

Author's Address

Ted Lemon
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: mellon@fugue.com