Workgroup: Internet Engineering Task Force
Published: 11 July 2022
Intended Status: Standards Track
Expires: 12 January 2023
Authors: T. Lemon (Apple Inc.)
A. Keshavarzian (Google)
J. Hui (Google)

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 12 January 2023.

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 (Section 11 of [RFC6763]).

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 partner remains online at all times, SRP state is maintained when individual SRP replication partners 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

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.

2.1. Naming of a common service zone

In order for SRP replication partners 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.

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.'

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.

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.'

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 partners, 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.

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 partner 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 partner 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.

2.2. Advertising one's own replication service

An SRP replication service operating in the "routine operation" state (Section 2.10) advertises its replication service.

SRP replication service is advertised using DNS-SD [RFC6763]. The service name is '_srpl-tls._tcp'. Each SRP replication partner 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 following fields:

did a 64-bit number encoded as hexadecimal ASCII. The dataset ID used by SRP servers to establish a common SRP dataset for a domain as described in Section 2.4.2.

join 'yes' or 'no'. Indicates whether other SRP replication servers are invited to join in replicating the dataset.

pid a 64-bit number encoded as hexadecimal ASCII. The partner ID to uniquely identify a SRP partner, as described in Section 2.6.

dn the domain name that this dataset is intended to represent

So in our example, the TXT record might look like this:

\031dn=openthread.thread.home.arpa.
\020did=eb5bb51919a15cec\020pid=2cde2bed200126af\008join=yes

(Note that each name/value pair in the TXT record is length-encoded, so the '\031`, the two '\020', and the '\008' are the lengths of the name/value pairs.)

2.3. Discovering other replication services

SRP Replication is a cooperative process. In order to ensure cooperation between SRP replication partners on a link, it is necessary that each replication partner be aware of other potential

partners. This is accomplished by maintaining a continuous browse for services of the service type "_srpl-tls._tcp".

An SRP Replication Partner 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 partner 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 partner checks to see if it is in a compatible domain. If the SRP partner 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.

2.4. Startup

When a partner starts SRP Replication, it enters the "startup" state and picks a random discovery time interval uniformly selected from the range [MIN_PARNTER_DISCOVERY_INTERVAL, MAX_PARTNER_DISCOVERY_INTERVAL]. The recommended minimum and maximum values are 4 and 7.5 seconds, respectively.

When the discovery time interval expires, the SRPL partner transitions to the "routine operation" state (Section 2.10) after it has either successfully synchronized with or NUM_DISCOVERY_SYNC_ATTEMPTS have occurred with each discovered partner. If the partner did not discover any other SRPL partners to synchronize with, it immediately transitions to the "routine operation" state.

2.4.1. Determining an SRP zone name

An SRPL partner attempts to synchronize with other partners advertising the same domain. If the SRPL partner is not configured with a domain name, and is therefore willing to join an existing domain, it adopts the first domain name with joining enabled it discovers. During the remainder of the discovery time interval, the SRPL partner attempts to synchronize with other partners advertising the same domain name. When the SRPL partner transitions to the "routine operation" state, it will advertise the adopted domain name and sets the 'join=yes' key/value pair in the TXT record.

When the discovery time interval expires, if the SRPL partner is not configured with a domain name and was not able to adopt a domain name, it MUST select a zone name using one of the methods mentioned previously in Section 2.1.

2.4.2. Determining a Dataset ID

The dataset ID is a 64-bit number that identifies the set of data replicated by a set of cooperating SRPL partners. Ideally, there should be exactly one dataset ID per domain. However, it is possible for several independent sets of SRPL partners to replicate a particular domain, for example when some SRPL partners are unable to discover each other. When the SRPL partners are able to discover each other again, the sets of partners must converge on a single dataset using the dataset ID.

If an SRPL partner does not discover any other partners advertising the same domain in the "startup" state Section 2.4, it generates a new dataset ID when entering the "routine operation" state. SRPL partners MUST persist the highest (most significant byte or MSB) of the dataset ID in non-volatile memory. When generating a new dataset ID, the partner MUST increment the MSB of last used dataset ID to use as MSB of new dataset ID and populate the lower 56 bits randomly using a high-quality random number generator [RFC4086]. If there is no previously saved ID, then the partner randomly generates the entire 64-bit ID.

When multiple dataset IDs exist for a given domain, the largest dataset ID is the preferred dataset ID. Implementations MUST follow the Serial Number Arithmetic as defined in [RFC1982] when comparing two dataset IDs. When using Serial Number Arithmetic with three or more IDs, it is possible that the largest value may not be well-defined (it is possible to have three IDs each being "larger" than another one). In this case, the dataset ID that is largest using absolute value comparison is the preferred dataset ID.

If at any time (regardless of "startup" or "routine operation" state) an SRPL partner discovers that it is synchronizing with a non-preferred dataset ID, it MUST abandon that dataset, re-enter the "startup" state, and attempt to synchronize with the (newly discovered) preferred dataset id.

2.5. Discovering the addresses of partners

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 partner adds the address to the list of addresses belonging to that partner.

2.6. Partner ID

The partner ID is a 64-bit number associated with an SRPL partner. Upon entering the "startup" state, the partner MUST generate a

random ID using a high-quality random number generator [RFC4086]. If a partner restarts SRP replication operation it MUST select a new random ID.

The likelihood of partner ID conflict is small. However, if an SRPL partner discovers another partner using the same ID, the partner MUST restart SRP replication operation, which triggers it to pick a new random ID.

2.7. Establishing Communication with a replication partner

When an address is added to a partner's address list, the partner 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 partner checks to see if it has an existing 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 determines whether it should initiate a connection on the new address.

While a partner is in the "startup" state (Section 2.4) it initiates connections with the other discovered partners. In the "startup" state, the partner does not advertise its SRP replication service so other partners cannot discover it over DNS-SD.

If the partner is in the "routine operation" state, it uses the partner ID to determine which partner should initiate the connection. If the partner determines that its own partner ID is larger, it MUST initiate the connection. Otherwise, the partner MUST NOT initiate the connection. Implementations MUST follow the Serial Number Arithmetic as defined in [RFC1982] when comparing two partner IDs.

When a connection fails and if there are multiple addresses associated with partner, the connecting partner advances to the next address in the list. If there are no remaining addresses, the partner 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.

2.8. 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 there is already an existing connection attributed to the same partner, the partner processing the incoming connection MUST immediately close the existing connection.

2.9. Initial synchronization

The connecting partner begins the session by sending an SRPL Session message, which includes its partner ID and indicates whether or not it is in the "startup" state. The receiving partner waits to receive the SRPL Session message. If the message indicates that the connecting partner is in the "startup" state, the receiving partner accepts the connection. Otherwise, the receiving partner compares the partner IDs as described in Section 2.7 and accepts the connection if its partner ID is smaller than the connecting partner's ID. If the receiving partner accepts the connection, it MUST send a response to the SRPL Session message and waits for the connecting partner to request a list of update candidates.

The connecting partner waits for a response to the SRPL Session message, and when it is received, requests that the server send candidates.

2.9.1. Sending candidates

When a partner 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 partner, 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 partner 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 partner sends a "send host" response.

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

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

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

2.10. Routine Operation

During routine operation, whenever an update is successfully processed from an SRP client, the partner 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 partner 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.

3. Protocol Details

The DNS-SD SRP Replication Protocol (henceforth SRPL) is based on DNS Stateful Operations [RFC8490]. Each SRP replication partner

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 partners. To distinguish between partners, the terms "partner" and "remote partner" are used. "Partner" refers to the partner that is communicating or receiving communication. "Remote partner" refers to the other partner. Partners can be clients or servers: a partner that has established a connection to another one is a client; a partner that has received a connection from another one is a server.

3.1. 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.

3.1.1. 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].

3.1.2. 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.

3.2. SRPL Messages

SRPL uses the DSO message format. Each message begins with a Primary TLV and may contain secondary TLVs with additional information (Section 5.4 of [RFC8490]). SRPL uses DSO request and response message types. A partner receiving a DSO request is expected to send a DSO response with same Primary TLV. SRPL does not define or use any unidirectional DSO message.

To ensure future compatibility, when processing a received SRPL message, any unrecognized secondary TLVs or additional value in currently defined SRPL TLVs MUST be silently ignored, and the message is interpreted and handled as if the unrecognized parts were not present (Section 5.4.5 of [RFC8490]). However, if a partner receives an SRPL message that does not satisfy the message format

and processing rules specified below, it MUST treat this as a "fatal error" and forcibly abort the connection immediately (Section 5.3.1 of [RFC8490]).

The SRPL messages and their corresponding Primary TLVs are as follows:

3.2.1. SRPL Session

DSO-TYPE code: SRPLSession. Introduces the SRPL session. The SRPL session TLV contains the partner ID as an unsigned 64-bit value. A SRPL Session request may include an SRPL New Partner secondary TLV. Inclusion of the SRPL New Partner TLV indicates whether the partner is in the "startup" state.

3.2.1.1. SRPL client behavior

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

When the SRPL partner acting as a client receives a response to its SRPL Session message from server, it MUST send an SRPL Send Candidates request.

3.2.1.2. SRPL server behavior

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

If the SRPL Session request is establishing a new session, the server MUST send a SRPL Session response if either of the following conditions are satisfied:

- The SRPL Session request contains an SRPL New Partner secondary TLV.

- The SRPL Session request contains a partner ID that is larger than the server's (see Section 2.7).

If none of these conditions are satisfied, this is a protocol error, and the SRPL partner MUST drop the connection.

3.2.2. SRPL Send Candidates

DSO-TYPE code: SRPLSendCandidates. Requests the remote partner to send its candidates list. The SRPL Send Candidates TLV contains no value. The SRPL Send Candidates message does not include any secondary TLVs.

3.2.2.1. SRPL client behavior

An SRPL partner 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 partner 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 partner should drop its connection to the server.

3.2.2.2. SRPL server behavior

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

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

3.2.3. SRPL Candidate

DSO-TYPE code: SRPLCandidate. Announces the availability of a specific candidate SRP client registration. The SRPL Candidate TLV contains no value.

3.2.3.1. Required secondary TLVs

The SRPL Candidate request MUST include the following secondary TLVs: SRPL Hostname, SRPL Time Offset, and SRPL Key ID.

The SRPL Candidate response MUST include exactly one of the following status TLVs: SRPL Candidate Yes, SRPL Candidate No, or SRPL Conflict.

3.2.3.2. SRPL partner common behavior

SRPL partners expect to receive SRPL Candidate messages between the time that they have sent an SRPL Send Candidates request 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 partner MUST drop the connection.

Partners MUST NOT send SRPL Candidate requests if they have sent any SRPL Candidate or SRPL host requests that have not yet received responses. Partners 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 partner 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 partner 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 partner 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 partner 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 partner 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.

3.2.4. SRPL Host

DSO-TYPE code: SRPLHost. Provides the content of a particular SRP client registration. The SRPL Host TLV contains no value.

3.2.4.1. Required secondary TLVs

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

The SRPL Host request MUST always include at least one SRPL Host Message TLV, which contains the most recent update the SRP server has received for that host. However, in some cases an update for a host may update some, but not all, service instances that reference that host; in this case, the SRPL Host request MUST include all of the previously received SRP updates that would be required to reconstruct the current state of the host registration on the server sending the SRPL Host request. SRPL Host Message TLVs MUST be ordered starting with the largest time offset and ending with the smallest time offset.

3.2.4.2. SRPL partner common behavior during synchronization

SRPL partners 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 initial synchronization, this is a protocol error, and the partner 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 partners 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).

3.2.4.3. SRPL partner common behavior during routine operations

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

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

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

3.3. DSO Secondary TLVs

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

3.3.1. SRPL Candidate Yes

DSO-TYPE code: SRPLCandidateYes. It contains no value. 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.

3.3.2. SRPL Candidate No

DSO-TYPE code: SRPLCandidateNo. It contains no value. 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.

3.3.3. SRPL Conflict

DSO-TYPE code: SRPLConflict. It contains no value. 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.

3.3.4. SRPL Hostname

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

registration. The TLV value is the hostname represented in DNS wire format Section 3.1 of [RFC1035].

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

3.3.5. SRPL Host Message

DSO-TYPE code: SRPLHostMessage. In an SRPL Host request, conveys four data objects in order:

- the lease time and key lease time returned to the client, represented as two unsigned 32-bit numbers in units of seconds.

- the time offset at which the message was received, represented as a 32-bit unsigned number of seconds. The time offset is computed as the difference between the time when the SRPL Host Message TLV is being constructed for transmission, and the time when the SRP update contained in the SRPL Host Message was received.

- the SRP Update message received from the SRP client. This contains the contents of the message, but not any IP, UDP, TCP or TLS headers that may have encapsulated it.

The content of the SRPL Host Message is used to update the host on the partner receiving the request. Note that the SRP message being sent can't be modified by the SRPL partner 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.

3.3.6. SRPL Time Offset

DSO-TYPE code: SRPLTimeOffset. In an SRPL Candidate request, conveys the difference between the time the SRP update was received from the SRP client and the current time on the partner generating the request, in seconds. The time offset value is represented as an unsigned 32-bit value

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

3.3.7. SRPL Key ID

DSO-TYPE code: SRPLKeyID. In an SRPL Candidate request, conveys the key ID of the SRP client. The value is an unsigned 32-bit number and calculated as described in Section 2.9.1.

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

3.3.8. SRPL New Partner

DSO-TYPE code: SRPLNewPartnter. It contains no value. When included in an SRPL Session request, indicates that the partner is in the "startup" state.

Can be optionally included as a secondary TLV in SRPL Session requests. MUST NOT appear in any other SRPL request or response.

4. 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.

5. 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].

6. IANA Considerations

6.1. '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.

6.2. 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	SRPL Send Candidates	No	STD	RFC-TBD
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
SRPLNewPartnter	SRPL New Partner	No	STD	RFC-TBD

Table 1

6.3. 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 5.

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.

7. Informative References

8. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

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

[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>.

Authors' Addresses

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

Email: mellon@fugue.com

Abtin Keshavarzian
Google
1600 Amphitheatre Parkway CA 94043
Mountain View, California 94043
United States of America

Email: abtink@google.com

Jonathan Hui
Google
1600 Amphitheatre Parkway CA 94043
Mountain View, California 94043
United States of America

Email: jonhui@google.com