Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2009

J. Lee
H. Schulzrinne
Columbia U.
W. Kellerer
Z. Despotovic
DoCoMo Euro

July 12, 2008

SIP URI Service Discovery using DNS-SD
draft-lee-sip-dns-sd-uri-03

Status of this Memo

Copyright Notice

Abstract

This document describes how to use the DNS-based Service Discovery (DNS-SD), better known as Apple Bonjour, for advertising Session Initiation Protocol (SIP) URIs in local area networks. Using this mechanism, a SIP user agent (UA) can communicate with another UA even when no SIP registrar is available, as in a wireless ad-hoc network for example.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [RFC3261], an application-layer protocol for controlling multimedia sessions such as voice-over-IP calls, uses SIP Uniform Resource Identifiers (SIP URIs) to represent who to contact or where to place a call. There are two types of SIP URIs. An Address-of-Record (AOR) represents the user's logical identity, analogous to an email address. A contact URI, on the other hand, indicates the network location of a host machine or a communication device where the user can be currently reached. The mappings between AORs and contact URIs are stored using SIP servers called registrars, and they are used by SIP servers called proxy servers to route calls (Section 10, [RFC3261]). For example, Carol, whose AOR is sip:carol@chicago.com, registers sip:carol@cube2214a.chicago.com as the current contact location using the registrar for the chicago.com domain. When the proxy server for the chicago.com domain receives a call request for sip:carol@chicago.com, it looks up the binding and routes the request to cube2214a.chicago.com.

Server-based mechanisms are not suitable for all types of networks, however. Consider, for example, a wireless ad-hoc network formed temporarily to address a specific situation, such as disaster recovery. In this case, it is clearly impractical to deploy SIP servers and configure user agents (UAs) to use the servers. What is needed here is a mechanism for the SIP UAs to learn the identities and locations of each other without using any server.

DNS-based Service Discovery (DNS-SD) [I-D.cheshire-dnsext-dns-sd] and Multicast DNS (mDNS) [I-D.cheshire-dnsext-multicastdns], better known as Apple Bonjour, provide a generic multicast-based solution for discovering services available in a local network without requiring any server deployment. DNS-SD/mDNS defines a set of naming rules for certain DNS record types that it uses for advertising and discovering services. PTR records are used to enumerate service instances of a given service type. A service instance name is mapped to a host name and a port number using a SRV record. If a service instance has more information to advertise than the host name and port number contained in its SRV record, the additional information is carried in a TXT record.

Those DNS records are not stored in a conventional unicast DNS server. Instead, they are stored in a collection of mDNS daemons, which are limited-functionality DNS servers running on each host in a local subnet. The mDNS daemons collectively manage a special top-level domain, ".local.", which is used for names that are meaningful only in a local subnet. The queries and answers are sent via link-local multicast using UDP port 5353 instead of 53, the conventional port for DNS. Thus, an application can advertise a network service to the local subnet by creating appropriate DNS records and depositing them into the mDNS daemon running on the same host. The mDNS daemon will then respond with these records when it hears a multicast query for a matching service. We will see an example of this process in Section 3. Note that creating DNS records and storing them with mDNS are usually done by invoking API calls in a DNS-SD/mDNS client library implementation. A hardware device can also advertise its network service using DNS-SD/mDNS, in which case a stripped-down implementation of mDNS customized for the specific service is usually embedded in the device.

This document specifies how the SIP UAs in the same subnet use DNS-SD/mDNS to advertise and discover the identities and locations of each other, enabling the communications between them without using SIP servers. Section 3 describes this process using a simple example of a UA discovering a SIP URI advertised by another UA. Section 4 defines the format of the SIP URI advertisement, and Section 5 specifies the behavior of a UA that discovers a SIP URI in the local network and wants to initiate a call.

It should be noted that the DNS-SD/mDNS mechanism described in this document and the SIP server mechanism in [RFC3261] are not mutually exclusive. Implementing SIP URI discovery via DNS-SD/mDNS will merely augment the functionality of a SIP UA, making it more useful in an ad-hoc network where the SIP servers are unavailable. Section 6 discusses how such enhancements can be incorporated to the existing user interface of a UA.

The generic nature of DNS-SD/mDNS make it a good candidate for the discovery mechanism of other SIP resources such as server location and capability. While we hope that the usage of DNS-SD/mDNS will become more pervasive in the SIP ecosphere, the scope of this document is limited to the discovery of SIP URIs among the UAs in a local network.

2. Requirements Notation

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

3. Overview of Operation

This section gives an overview of SIP URI advertisement in DNS-SD/mDNS environment using a simple example scenario.

Consider two users, Alice and Bob, who are running SIP UAs on their laptop computers on the same subnet. Assume that a DNS-SD/mDNS implementation, such as Bonjour or Avahi, is installed and running on both computers. Further assume that SIP registrar and proxy server are not available to the UAs (there is no SIP server in the network or the UAs are not configured with the local servers, for instance), but the UAs are equipped with an implementation of the mechanism described in this document. Alice, knowing that Bob is in the vicinity with his computer connected probably to the same subnet as hers, would like to make a SIP call to Bob.

Bob's UA advertises sip:bob@example.com, Bob's AOR, to the local subnet. This is done by the UA invoking an API function in a DNS-SD/mDNS client library, which in turn connects to the mDNS daemon process running on the same computer using an IPC mechanism and sends the following DNS records:

```
_sipuri._udp.local.  PTR  sip:bob@example.com._sipuri._udp.local.

sip:bob@example.com._sipuri._udp.local.
                     SRV  0 0 5060 bobs-machine.local.

bobs-machine.local.  A    192.168.0.100
```

The mDNS daemon receives and stores these records, and starts listening for multicast DNS queries that might ask for those records. The PTR record says that there is a service instance called "sip:bob@example.com", that it is of the service type "sipuri", that it uses UDP transport, and that it is available in the local subnet (".local."). The SRV record maps the service instance to the host, "bobs-machine.local.", and the port, 5060. Finally the A record provides the IP address. In short, using these records, Bob's UA is advertising that it is listening on the UDP port 5060 at 192.168.0.100 for a SIP request addressed to sip:bob@example.com.

Alice's UA tries to discover the SIP URIs being advertised in the local subnet by multicasting a PTR query for "_sipuri._udp.local.". (It will probably send out queries for _sipuri._tcp.local. and _sipuri._sctp.local. as well, so it can also discover the UAs advertising those transport protocols. But we will limit our scenario to UDP for simplicity.)

Alice's UA receives the following two answers to its multicast PTR query:

```
_sipuri._udp.local.  PTR  sip:bob@example.com._sipuri._udp.local.
_sipuri._udp.local.  PTR  sip:carol@chicago.com._sipuri._udp.local.
```

Those two answers came from two different machines. The mDNS daemons running on Bob's and Carol's computers both heard the multicast query and each replied to Alice's UA with its PTR record. Alice's UA then communicates to the user, Alice, that it found the two SIP URI advertisement in the local subnet. It does this perhaps by displaying sip:bob@example.com and sip:carol@chicago.com in a window titled "Local Users". (Section 6 discusses the user interface in further detail.)

Alice decides that she wants to call Bob and clicks on Bob's URI displayed in the window presented by her UA. Alice's UA then sends another series of multicast queries, a SRV query followed by an A query, to which it receives the following responses from Bob's mDNS:

```
sip:bob@example.com._sipuri._udp.local.
                     SRV  0 0 5060 bobs-machine.local.

bobs-machine.local.  A    192.168.0.100
```

From these answers, Alice's UA obtains the IP address and port number, to which it sends a SIP INVITE addressed to sip:bob@example.com.

It should be noted that, for clarity of exposition, our example glosses over many details of how the mDNS daemons actually exchange DNS packets. The mDNS specification ([I-D.cheshire-dnsext-multicastdns]) spends considerable efforts to provide immediate and on-going discovery of services while reducing network traffic. For example, a mDNS daemon will send out a gratuitous multicast DNS answer packet containing all of its resource records whenever it starts up, wakes from sleep, or detects a change in network configuration. The other mDNS daemons in the subnet will receive the gratuitous packet and cache the information contained in it. Thus, in our example of Alice and Bob, what actually happened could have been that Alice's mDNS might have already cached the gratuitous packet sent by Bob's mDNS even before Alice launched her UA, and when Alice wanted to call Bob, her mDNS could have provided all the necessary DNS records from its cache without sending any multicast query. DNS-SD/mDNS also tracks newly arriving services using these gratuitous announcements as other UAs enter and leave the subnet.

In many cases, an application or a device advertising a service has more information to convey than what a SRV record can carry, namely, the host name and port number. For example, a printer advertising a LPR port should convey a queue name as well. The necessary additional data can be carried in a TXT record with the same name as the SRV record. In our previous example, Bob's UA could have included the following TXT record in the service advertisement:

```
sip:bob@example.com._sipuri._udp.local.  TXT
  txtvers=1 name=Bob contact=<sip:bob@192.168.0.100:5060>;audio;video
```

Under the same name as the SRV record, it defines several name/value attributes that the other UAs would find useful. Using the "name" attribute, the "Local Users" window of Alice's UA can now display "Bob (sip:bob@example.com)" rather than just the URI. The "contact" attribute contains a contact URI that includes the IP address and port number (thereby obviating the SRV and A queries) and the supported media types. The formats of the attributes are defined in Section 4.2.

4. SIP URI Advertisement Format

This section specifies the format of a SIP URI advertisement. Section 4.1 specifies the format of the service instance name, which is the name of the SRV and TXT resource records. Section 4.2 defines the attribute names that can be stored in the TXT record, and specifies the rules for their values.

4.1. SIP URI Service Instance Name

Section 4.1 of [I-D.cheshire-dnsext-dns-sd] specifies that a service instance name in DNS-SD has the following structure:

    <Instance> . <Service> . <Domain>

The <Domain> portion specifies the DNS sub-domain where the service instance is registered. It may be "local.", indicating the mDNS local domain, or it may be a conventional domain name such as "example.com.".

The <Service> portion of the SIP URI service instance name MUST be "_sipuri._udp", "_sipuri._tcp", or "_sipuri._sctp", depending on the transport protocol desired by the UA advertising the service instance. If a UA supports multiple protocols, it SHOULD advertise multiple service instances. Note that, while this usage with the protocol part is in agreement with DNS SRV RR definition ([RFC2782]) and with the previous usage of SRV RR in SIP (Section 4.1, [RFC3263]), it does not agree with the DNS-SD guideline. This is discussed further in Section 8.

The <Instance> portion is a DNS label, containing UTF-8-encoded text, limited to 63 octets in length. It is meant to be a user-friendly description of the service instance, suitable for a menu-like user interface display. Thus it can contain any characters including spaces, punctuation, and non-Latin characters as long as they can be encoded in UTF-8.

For the SIP URI service instance, however, there is a required format. The <Instance> portion of the SIP URI service instance MUST start with a valid SIP or SIPS URI, optionally followed by a space character and an arbitrary text further describing the URI. In Augmented BNF (ABNF) [RFC2234], this is expressed as follows:

```
instance = ( SIP-URI / SIPS-URI ) [ SP description ]
```

The definition and the ABNF for SIP-URI and SIPS-URI are given in Section 19.1 and Section 25.1 of [RFC3261]. SP denotes a space character and "description" is an arbitrary UTF-8-encoded text string. The entire instance string cannot be more than 63 octets in length.

For example, the SIP URI service instance names for Bob's two SIP devices may be:

    sip:bob@example.com - Softphone._sipuri._udp.local.

    sip:bob@example.com - PDA._sipuri._udp.local.

This scheme is also compatible with the automatic name conflict resolution of Apple's mDNS implementation, which appends a numerical suffix such as " (2)" to a name in order to distinguish it from another instance with the same name in the same subnet. If both of Bob's devices advertise themselves as "sip:bob@example.com" in such an environment, the resulting service instance names will be:

    sip:bob@example.com._sipuri._udp.local.

    sip:bob@example.com (2)._sipuri._udp.local.

which are both valid SIP URI service instance names. Since an advertiser is free to choose any arbitrary text for the description following the URI, the other UAs discovering the service instance should not attempt to parse the text for any specific information. The text is meant to be displayed to the human user as is, in a menu listing the discovered service instances for example.

The reason for requiring that the instance name begins with a valid SIP URI is that having a SIP URI available in the name makes the service advertisement contain sufficient information for a UA to initiate a call. The UA resolves the service instance name and obtains the IP address and the port number. (This is done by issuing an SRV query. See Section 5, [I-D.cheshire-dnsext-dns-sd].) Then it can send a SIP request using the SIP URI from the service name as the Request-URI. This makes the information from the TXT record (described in the next section) optional, in accordance with the recommendation that the TXT record should be viewed as a performance optimization (Section 6.2, [I-D.cheshire-dnsext-dns-sd]).

The SIP or SIPS URI in the service instance name SHOULD be an Address-of-Record (AOR). It is conceivable that a UA may not be configured with an AOR. A group of UAs in an ad-hoc network may be configured only with user names, for example. In such cases, the UA host names or IP addresses may be used to form a valid SIP URI for service advertisement.

4.2. TXT Record Attributes

In addition to the service instance name, IP address and the port number, DNS-SD provides a way to publish other information pertinent to the service being advertised. The additional data can be stored as name/value attributes in a TXT record with the same name as the SRV record for the service. Each name/value pair within the TXT record is preceded by a single length byte, thereby limiting the length of the pair to 255 bytes. (See Section 6 of [I-D.cheshire-dnsext-dns-sd] and Section 3.3.14 of [RFC1035] for details.)

The following subsections describe the attributes defined for the SIP URI service. Note that, while the presence of any of these attributes in a SIP URI advertisement is optional, the presence of certain attributes affects the behavior of the UA processing the service instance. (See Section 5 for detail.)

4.2.1. txtvers

The "txtvers" attribute defines the version number of the TXT record specification as recommended in Section 6.7, [I-D.cheshire-dnsext-dns-sd]. If present, this attribute MUST be the first name/value pair in the TXT record. For this specification, it MUST be "txtvers=1".

4.2.2. name

The "name" attribute contains the display name of the user. For example, "name=John Doe".

It MUST conform to the "display-name" ABNF element in Section 25.1, [RFC3261], so that it can be used in the "To" SIP header, as in "To: John Doe <sip:john@example.com>".

4.2.3. contact

The "contact" attribute contains a SIP or SIPS URI that represents a direct route to the user. The URI usually contains a fully qualified domain name (FQDN) or an IP address indicating the physical contact location of the user. For example, "contact=sip:carol@cube2214a.chicago.com".

Note that, while this attribute has the same semantics as the "Contact" SIP header defined in [RFC3261], the attribute does not allow the full syntax of the SIP header. First, only SIP or SIPS URIs are allowed in the attribute, whereas non-SIP URIs are allowed in the Contact header. Non-SIP URIs are not applicable in the SIP

URI service discovery. Second, the attribute can contain only a single URI, whereas the Contact header can contain multiple URIs in a comma-separated list. We argue that multiple contact locations can (and should) be advertised as multiple service instances.

[RFC3261] also defines two Contact parameters "q" and "expires". The "q" parameter is only applicable when there are multiple Contact locations. The "expires" parameter is also not relevant in this environment since the service instance must be created and removed according to the rules of the underlying service discovery system.

The attribute name/value pair has the following syntax ABNF:

```
contact-attr  =  "contact="
                 ( name-addr / uri ) *( SEMI contact-extension )
name-addr     =  [ display-name ] LAQUOT uri RAQUOT
uri           =  SIP-URI / SIPS-URI
```

SEMI, LAQUOT and RAQUOT denote ";", "<" and ">", respectively. Note that whitespace is often allowed around these characters. The contact attribute value has nearly the same syntax as the "contact-param" element in Section 25.1 of [RFC3261]. The difference is that the contact attribute syntax disallows non-SIP URIs and it omits the "q" and "expires" parameters. See Section 25.1 of [RFC3261] for the other syntax elements that are not expanded here, such as contact-extension and display-name. Also see Section 20.10 and the last paragraph of Section 20 of [RFC3261] for the important information regarding the Contact header parsing rules, which are equally applicable to the contact attribute.

The attribute syntax allows one or more contact-extension elements, which are generic name/value parameter provisions for future extensions. Currently, [RFC3840] defines a mechanism by which SIP UAs can exchange information about their capabilities and characteristics through these parameters. Such a mechanism is particularly germane to service discovery.

5. Making a Call Using Discovered SIP URI

This section specifies the behavior of the UA that sends a SIP request using the discovered SIP URI service instance. In particular, it specifies how to form the Request-URI and the "To" header of the request, and how to determine the destination host to which the SIP request should be transported. Beyond that, Section 8.1 and 18.1 of [RFC3261] describe in detail the behavior of a UA generating and sending a SIP request.

5.1. Forming the SIP Request

The "To" header MUST be formed using the SIP or SIPS URI from the service instance name. The URI is either the first DNS label of the service instance name if it contains no space, or the longest prefix in the first DNS label that does not include the first space character. (See Section 4.1.)

If the "name" attribute of the TXT record is available, it SHOULD be used as the "display-name" in the "To" header according to the formatting rules outlined in Section 20.10 of [RFC3261].

The Request-URI MUST be formed using the SIP or SIPS URI from the "contact" attribute of the TXT record. If the "contact" attribute is not available, the Request-URI MUST be set to the same value as the "To" header.

5.2. Sending the SIP Request

First, the UA determines the transport protocol from the service type portion of the discovered service instance name. The three possible values are "_sipuri._udp", "_sipuri._tcp", or "_sipuri._sctp", for which the UDP, TCP, or SCTP transport protocol MUST be used, respectively.

Next, the UA determines the IP address and port number to which to send the request. This procedure differs depending on whether the "contact" attribute is available in the TXT record.

If the "contact" attribute is available, the IP address and the port number of the destination host MUST be determined from the SIP/SIPS URI in the attribute as follows. The value of the maddr parameter of the URI, if present, becomes the destination host. Otherwise, the host value of the hostport component of the URI becomes the destination host. (See [RFC3261] for the description of the maddr parameter and the definition of the SIP/SIPS URI.) If the destination host is already in the form of a numeric IP address, the UA uses that address. If not, the UA performs an A or AAAA record lookup of the description host to obtain the IP address. The A/AAAA query should be sent to the local mDNS daemon if the destination host name belongs in the ".local." domain. The DNS-SD/mDNS implementations usually modify the resolver configuration of the operating system to direct all .local queries to mDNS, so the UA can simply make a DNS query as usual without worrying about whether it is for a local name or not. If a port number is present in the SIP/SIPS URI, the UA uses that port. Otherwise, the UA uses the default port for the particular transport protocol.

If the "contact" attribute is not available, the UA MUST resolve the service instance name to obtain the host name and port number to which to send the request. The service instance name is resolved by sending a SRV query or by calling the equivalent API routine in the DNS-SD library implementation (Section 5, [I-D.cheshire-dnsext-dns-sd]). The host name is then further resolved to an IP address by performing an A/AAAA lookup.

We should note that the host name and the port number in the SRV record are ignored when the "contact" attribute is present. Normally the destination obtained from the contact URI will be the same as that from the SRV record. But a UA has an option to advertise a contact URI pointing to a host or device different from its own, perhaps in order to redirect incoming calls to voice mail or to have the calls go through a proxy server for accounting reasons, for example.

As described above, the host name from the contact URI or from the SRV record is resolved by performing an A/AAAA query. This is in contrast with [RFC3263], which requires additional steps using NAPTR and SRV lookups in resolving a host name in a SIP URI. (The SRV lookups in [RFC3263] are plain SRV lookups described in [RFC2782], so they should not be confused with the DNS-SD's use of SRV records that we have been discussing.) The flexibility provided by the additional levels of indirection specified in [RFC3263] is of limited value in the usual serverless setting of DNS-SD/mDNS, so it was deemed not a strong enough reason to deviate from the normal DNS-SD convention of resolving a host name using an A/AAAA query.

6. User Interface Guidelines

This section considers the user interface of a UA that implements the behavior specified in Section 5. As a model for our discussion, let us consider a typical graphical UA that presents three user interface elements: an address book window containing the AORs manually maintained by the human user, another window listing the SIP URIs currently available through DNS-SD, and a text edit box in which the user can directly type a URI not listed in either window.

The address book entries and the DNS-SD entries SHOULD be presented in a way that makes it clear to the user that they are two separate lists. When the user selects an entry from the DNS-SD list, the UA MUST follow the behavior outlined in Section 5.

When the user selects an entry from the address book window, the UA MUST follow the normal user agent client behavior specified in Section 8.1 of [RFC3261]. This means that the SIP request is routed either using a configured outbound proxy or using the SIP server location mechanisms described in [RFC3263]. If such an effort fails, due to a network outage or a server failure for example, and there is a DNS-SD entry with the same URI as the address book entry that the user has selected, then (and only then) the UA MAY try the DNS-SD entry with the same URI, following the behavior in Section 5. In this case, the address book entry might indicate that the URI is also being announced via DNS-SD advertisement. The reason for requiring that the UA first follows the server mechanisms when processing an address book entry is discussed in Section 9.

A URI directly typed in by the user MUST be processed as if it has been selected from the address book window.

7. Other Related Mechanisms

7.1. SIP Multicast

The previous SIP specification [RFC2543] included sending INVITE requests via multicast. The intended purpose was to provide the mechanism where a UA can send an INVITE message to a logical entity comprised of multiple hosts serving a single function such as a help desk. Due to the complexity of the mechanism, the multicast INVITE has been removed from the current specification. Currently the use of multicast is limited to "single-hop-discovery-like" services such as registrations. (Section 10.2.6 and 18.1.1, [RFC3261])

Multicast REGISTER requests provide another way to discover peer locations. When using multicast REGISTER, UAs send the REGISTER requests to the SIP multicast address (sip.mcast.net or 224.0.1.75). They would also listen to that address and keep a local database of peer locations as they encounter REGISTER requests. This may seem similar to the SIP URI advertisement using DNS-SD/mDNS as described in this document. The most important difference is that the multicast REGISTER method provides passive discovery only. Unlike in the DNS-SD/mDNS environment where a UA can simply make a query, in the multicast REGISTER setting a newly arriving UA would not discover the existing UAs until their registrations are refreshed, which could introduce up to an hour delay even if we assume no packet loss. This makes multicast REGISTER unsuitable for high-churn environments such as wireless ad-hoc networks.

7.2. "sip" DNS-SD Service Type

There is another DNS-SD service type related to SIP. The "sip" service type is primarily used for server advertisements. Most notably, it is used by Asterisk, a popular open-source software system for IP PBX (<http://www.asterisk.org/>). In contrast, the "sipuri" service type described in this document is intended for user agent advertisements.

Some UA implementations are currently using the "sip" service type for user agent advertisements. This is not ideal because the TXT attributes defined for the "sip" type are geared towards server announcements, and thus are not suitable for user agent advertisements. This leads the UAs using the "sip" type to ignore the TXT attributes, or even worse, to define their own set of attributes. Ekiga softphone (<http://www.ekiga.org/>) uses the "sip" type, but introduces a number of TXT attributes not defined for the "sip" type. Gizmo Project (<http://www.gizmoproject.com/>) uses the "sip" service type to advertise SIP URIs in the same way as the "sipuri" type advertisement described in this document: it uses the

SIP URI as the name of the SRV resource record. But it does not use any TXT attribute. We encourage the UA implementations to use the "sipuri" service type described in this document, which defines a set of TXT attributes that are suitable for user agent advertisements.

7.3. Peer-to-Peer SIP

The IETF Peer-to-peer SIP (P2PSIP) working group has been formed recently. The working group's goal is to develop protocols and mechanisms to replace or augment the centralized SIP servers with the services provided by the peer-to-peer network of SIP endpoints.

This may seem similar to the DNS-SD/mDNS setting considered in this document. The difference is that while DNS-SD/mDNS is primarily for local area networks, P2PSIP is concerned with the peer-to-peer overlays of SIP endpoints spanning the globe. In fact, its charter (<http://www.ietf.org/html.charters/p2psip-charter.html>) specifically excludes multicast and dynamic DNS based approaches from the scope of its work.

8. Transport Protocol in Service Instance Name

Section 7 of [I-D.cheshire-dnsext-dns-sd] states:

> The "_tcp" or "_udp" should be regarded as little more than boilerplate text, and care should be taken not to attach too much importance to it. Some might argue that the "_tcp" or "_udp" should not be there at all, but this format is defined by RFC 2782, and that's not going to change. In addition, the presence of "_tcp" has the useful side-effect that it provides a convenient delegation point to hand off responsibility for service discovery to a different DNS server, if so desired.

The web site for DNS-SD service type registration (see Section 10) goes further and says:

> Protocols that can run over either UDP or TCP (e.g. NFS) are usually advertised using whichever transport is considered the 'normal' or 'primary' mode of operation (and clients should attempt communication with the service using either or both transports, as appropriate for the client).

This interpretation and policy are reasonable for those application protocols that have clear "primary" transport protocols, but they present difficulty in a protocol such as SIP that supports multiple transports without favoring any particular one. [RFC3263] specifies how NAPTR and SRV records are used to resolve a SIP URI into the IP address, port, and transport protocol of the request destination. The transport label in the SRV record ("_udp", "_tcp", or "_sctp") plays an important role in determining which transport protocol should be used.

It would be inconsistent and confusing for a SIP UA to interpret the transport labels in SRV records differently depending on whether it is processing a DNS-SD service or not. This document follows the conventional SRV record interpretation that treats the transport label as indicating the desired transport protocol (Section 4.1). We believe the DNS-SD interpretation is an oversight and hope to see a change in the subsequent iterations of [I-D.cheshire-dnsext-dns-sd].

9. Security Considerations

In a DNS-SD/mDNS environment, there is no restriction on who can advertise what services. An attacker who has gained access to a local area network, such as an unsecured wireless network, can impersonate any SIP URI simply by advertising it using DNS-SD. At a minimum, a UA must be careful to present the URIs discovered through DNS-SD in a way clearly distinguishable from the ones in the user's address book. The discovered URIs and the address book entries SHOULD be presented to the user in two separate lists. Moreover, the DNS-SD entries can use the display names rather than the advertised URIs in order to further indicate the fact that the URIs are not authenticated in any way. This security concern underlies the user interface guidelines in Section 6.

When it is important to verify the authenticity of the advertised AORs, SIPS URIs should be used. Ideally a UA advertising a SIPS URI should authenticate itself using a certificate signed by a certificate authority (CA), but the burden of obtaining a CA-signed certificate may not be justifiable for a few SIP end-points communicating directly with each other in a local area network. In such cases, self-signed certificates can be used to obtain most of the security benefits provided by TLS without having to acquire a CA-signed certificate. A self-signed certificate provides no authentication when the connection is made for the first time. The UA SHOULD present a clear warning to the user indicating that the SIPS URI that the user wants to contact is using a self-signed certificate, therefore it is unauthenticated, and encouraging the user to verify the authenticity in some other way. Once the user has successfully made the first connection (perhaps after checking the authenticity by external means), the UA can store the self-signed certificate in its local database so that all the subsequent connections can be authenticated by comparing the certificate being presented to the one stored in the local database. The usefulness of this mechanism is clearly demonstrated by the widespread adoption of SSH, which uses essentially the same mechanism for authenticating the servers. (Section 4.1, [RFC4251])

Since this document is essentially a naming and usage convention within the framework of DNS-SD and mDNS, the security considerations for those systems apply here as well. [I-D.cheshire-dnsext-dns-sd] recommends the use of DNSSEC [RFC2535] when the authenticity of information is important. [I-D.cheshire-dnsext-multicastdns] suggests, among other things, IPSEC and/or DNSSEC signatures when it is desirable to distinguish a group of cooperating nodes from other (possibly) antagonistic ones operating on the same physical link.

10. IANA Considerations

Currently, DNS-SD service type names are not managed by IANA. Section 19 of [I-D.cheshire-dnsext-dns-sd] proposes an IANA allocation policy for unique application protocol or service type names. Until the proposal is adopted and in force, Section 19 points to <http://www.dns-sd.org/ServiceTypes.html> for instruction on how to register a unique service type name for DNS-SD.

The service type "sipuri" for the discovery method presented in this document has been registered according to that instruction.

11. Normative References

[I-D.cheshire-dnsext-dns-sd]
Krochmal, M. and S. Cheshire, "DNS-Based Service Discovery", draft-cheshire-dnsext-dns-sd-04 (work in progress), August 2006.

[I-D.cheshire-dnsext-multicastdns]
Cheshire, S. and M. Krochmal, "Multicast DNS", draft-cheshire-dnsext-multicastdns-06 (work in progress), August 2006.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[RFC2543] Handley, M., Schulzrinne, H., Schooler, E., and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

Authors' Addresses

Jae Woo Lee
Columbia University
Dept. of Computer Science
1214 Amsterdam Avenue
New York, NY 10027
US

Email: jae@cs.columbia.edu

Henning Schulzrinne
Columbia University
Dept. of Computer Science
1214 Amsterdam Avenue
New York, NY 10027
US

Email: schulzrinne@cs.columbia.edu

Wolfgang Kellerer
DoCoMo Communications Laboratories Europe
Landsberger Str. 312
Munich 80687
Germany

Email: kellerer@docomolab-euro.com

Zoran Despotovic
DoCoMo Communications Laboratories Europe
Landsberger Str. 312
Munich 80687
Germany

Email: despotovic@docomolab-euro.com

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).