Internet Engineering Task Force
Internet Draft
draft-ietf-sip-srv-03.txt
December 24, 2001
Expires: May 2002

SIP WG
J.Rosenberg,H.Schulzrinne
dynamicsoft,Columbia U.

# SIP: Locating SIP Servers

STATUS OF THIS MEMO



Abstract

The Session Initiation Protocol (SIP) makes use of DNS procedures to allow a client to resolve a SIP URI into the IP address, port, and transport of the next hop to contact. It also uses DNS to allow a server to send a response to a backup client in the event of a failure of the primary client. This document describes those DNS procedures in detail.

## 1 Introduction

The Session Initiation Protocol (SIP) [1] is a client-server protocol used for the initiation and management of communications sessions between users. SIP end systems are called user agents, and intermediate elements are known as proxy servers. A typical SIP configuration, referred to as the SIP "trapezoid" is shown in Figure 1. In this

diagram, a caller, UA1 wishes to call joe@B. To do so, it communicates with proxy 1 in its domain (domain A). Proxy 1 forwards the request to the proxy for the domain of the called party (domain B), which is proxy 2. Proxy 2 forwards the call to the called party, UA 2.

As part of this call flow, proxy 1 needs to determine a SIP server for domain B. To do this, proxy 1 makes use of DNS procedures, using both the SRV [2] and NAPTR [3] records. This document describes the specific problems that SIP uses DNS to help solve, and provides a solution.

## 2 Problems DNS is Needed to Solve

DNS is needed to help solve several aspects of the general call flow described in the Introduction.

First off, proxy 1 needs to discover the SIP server in domain B, in order to forward the call for joe@B. Specifically, it needs to determine the IP address, port and transport for the server in domain B. Transport is particularly noteworthy. Unlike other protocols, SIP can run over a variety of transports, including TCP, UDP, TLS/TCP and SCTP. Therefore, discovery of transports for a particular domain is an important part of the processing. The proxy sending the request has a particular set of transports it supports (all proxies must implement both TCP and UDP) and a preference for using those transports. Proxy 2 has its own set of transports it supports (the minimal overlap is UDP and TCP in this case), and relative preferences for those transports. Some form of DNS procedures are needed for proxy 1 to discover the available transports for SIP services at domain B, and the relative preferences of those transports. This information can be merged with the supported transports and preferences at proxy 1, resulting in a selection of a transport.

It is important to note that DNS processing can be used multiple times throughout processing of a call. In general, an element that wishes to send a request (generally called a client) may need to perform DNS processing to determine the IP address, port, and transport of a next hop element, generally called a server (it can be a proxy or a user agent). Such processing could, in principle, occur at every hop between elements.

Since SIP is used for the establishment of interactive communications services, the time it takes to complete a transaction between a caller and called party is important. Typically, the total delay between when a user initiates the call, and when they get an indication that the called party is being alerted to the call, needs to be

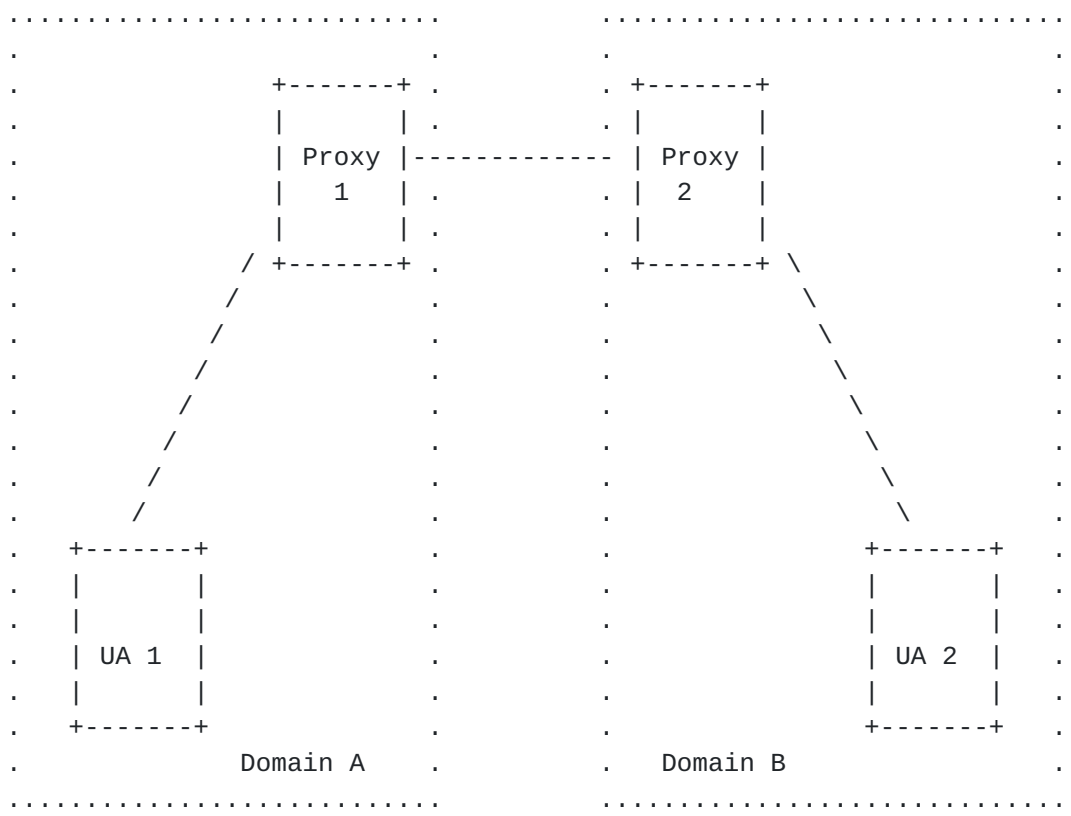


Figure 1: The SIP trapezoid

less than a few seconds. Given that there can be multiple hops, each of which is doing DNS processing in addition to other potentially time-intensive operations, the amount of time available for DNS processing at each hop is limited.

Scalability and high availability are important in SIP. SIP services scale up through clustering techniques. In a more realistic version of the network in Figure 1, proxy 2 would typically be a cluster of homogeneously configured proxies. DNS needs to provide the ability for domain B to configure a set of servers, along with prioritization and weights in order to provide a crude level of capacity based load balancing.

High availability is accomplished in SIP through detection of failures by upstream elements. For example, proxy 1 would send a request to proxy 2.1 (one of the proxies in the "cluster" proxy 2). This request would fail, and that would be detected by proxy 1. Proxy 1 would then try another of the proxies, proxy 2.2. In many cases, such as the one above, proxy 1 will not know which domains it will ultimately communicate with. That information would be known when a user actually makes a call to another user in that domain. Proxy 1 may never communicate with that domain again after the call completes. Proxy 1 could communicate with thousands of different domains within a few minutes, and proxy 2 could receive requests from thousands of different domains within a few minutes. Because of this "many-to-many" relationship, it is not generally possible for an element to perpetually maintain dynamic availability state for the proxies it will communicate with. When a proxy gets its first call with a particular domain, it will try the servers in that domain in some order until it finds one thats available. The identity of the available server would ideally be cached for some amount of time in order to reduce call setup delays of subsequent calls. However, the client cannot actively "ping" the failed servers to determine when they come back alive, because of scalability concerns. Furthermore, the availability state must eventually be flushed in order to redistribute load to recovered elements when they come back online.

It is possible for elements to fail in the middle of a transaction. For example, after proxy 2 forwards the request to UA 2, proxy 1 fails. UA 2 sends its response to proxy 2, which tries to forward it to proxy 1, which is no longer available. Ideally, we would like proxy 2 to use DNS procedures to identify a backup server for proxy 1 that it can use to forward the response. This problem is more realistic in SIP than it is in other transactional protocols. The reason is that a SIP response can take a *long* time to be generated, because a human user frequently needs to be consulted in order to generate that response. As such, it is not uncommon for tens of seconds to elapse between a call request and its acceptance.

## 3 Client Usage

Usage of DNS differs for clients and for servers. This section discusses client usage. The assumption is that the client is stateful

(either a UAC or a stateful proxy). Considerations for stateless proxies are discussed in Section 3.4.

The procedures here are invoked when a client needs to send a request to a server for which it does not already know an explicit IP address, port, and transport. This occurs when an element wishes to send a request to a server identified by a SIP URI, or when an element wishes to send a request to a specific configured server, independent of the SIP URI, but the configured server is identified by a domain name instead of a numeric IP address.

The procedures here MUST only be done once per transaction. That is, once a server has successfully been contacted (success is defined below), all retransmissions of the request and the ACK for non-2xx responses MUST be sent to the same server. Furthermore, a CANCEL for a particular request MUST be sent to the same server that the request was delivered to.

Note that, because the ACK request for 2xx responses constitutes a different transaction, there is no requirement that it be delivered to the same server that received the original request (indeed, if that server did not record-route, it will most definitely not get the ACK).

If the request is being delivered to an outbound proxy, a temporary URI, used for purposes of this specification, is constructed. That URI is of the form sip:<proxy>, where <proxy> is the domain of the outbound proxy.

The first step is to identify the TARGET. The TARGET is set to the value of the maddr parameter of the URI, if present, otherwise, the host value of the hostport construction. It represents the domain to be contacted.

## 3.1 Selecting a Transport

Next, a transport is selected.

If the URI specifies a transport, that transport MUST be used.

Otherwise, if no transport is specified, but the TARGET is a numeric IP address, the client SHOULD use UDP.

Otherwise, if no transport is specified, and the target is not a numeric IP address, the client SHOULD perform a NAPTR query. This query is for the service "SIP+D2T", which provides a mapping from a domain to a transport for contacting that domain. The transport is of the form of an SRV record, using the "S" NAPTR flag. The resource

record will contain a replacement value (not a regular expression), which is the SRV record for a particular transport. If the server supports multiple transports, there will be multiple NAPTR records, each with a different order value. The client MUST discard any records that contain an SRV value with a transport not supported by the client, but otherwise follow the processing rules of [3]. The result is that the most preferred transport of the server that is supported by the client will get used.

As an example, consider foo.com. A client wishes to contact a SIP server in foo.com. It performs a NAPTR query for that domain, and the following records are returned:

```
;;       order pref flags service           regexp  replacement
 IN NAPTR 90   50  "s"  "SIP+D2T"           ""  _sip._tcp.foo.com
 IN NAPTR 100  50  "s"  "SIP+D2T"           ""  _sip._udp.foo.com
 IN NAPTR 110  50  "s"  "SIP+D2T"           ""  _sip._tls.foo.com
```

This indicates that the server supports TCP, UDP, and TLS, in that order of preference. If the client supports UDP and TLS, UDP will be used, based on an SRV lookup of _sip._udp.foo.com.

> Somehow this doesn't seem right, since the client needs to look at the replacement values to discard entries. Perhaps the query should instead be done for sip.<domain>, and the service field is "TCP+D2T" or "UDP+D2T"?

It is STRONGLY RECOMMENDED that the domain suffixes in the replacement field (i.e., foo.com above) match the domain of the original query. Without that, backwards compatibility between RFC 2543 and this specification will not be possible.

> THis is because RFC 2543 clients will go directly to SRV records using the domain suffixes. If these are non-existent, because the NAPTR replacement used a different suffix, communication will not take place.

In the event that no NAPTR records are found, the client constructs SRV records for those transports it supports, and does a query for each. Queries are done using the service identifier "_sip". If the query is successful, it means that the particular transport is supported. The client MAY use any transport it desires which is supported by the server.

> This is a change from RFC 2543, which used to merge the priority values across different SRV records.

### 3.2 Determining port and IP

Once the transport has been determined, the next step is to determine the IP address and port.

If TARGET is a numeric IP address, use that address. If the URI also contains a port, use that port. If no port is specified, use the default port for the particular transport.

If the TARGET was not a numeric IP address, but a port is present in the URI, first check the cache to determine if a server has been previously contacted successfully for that TARGET and port. If one has been, use that server. Otherwise, perform an A or AAAA record lookup of the domain name. The result will be a list of IP address, each of which can be contacted at the specific port from the URI and transport determined previously. Processing then proceeds as described in Section 3.3.

> There is a weird case where, where the URI had a domain name and a port. SRV records will potentially be used to determine the transport, based on the algorithms above, but A records used for the actual lookup. That seems odd.

If the TARGET was not a numeric IP address, and no port was present in the URI, first check the cache to see if a server had been previously contacted successfully for that TARGET. If one had been, use that. Otherwise, perform an SRV query using the service identifier "_sip" and the transport as determined from Section 3.1, as specified in RFC 2782 [2]. The procedures of RFC 2782, as described in the Section titled "Usage rules" are followed, augmented by the additional procedures of Section 3.3.

> This is a change. Previously, if the port was explicit, but with a value of 5060, SRV records were used. Now, A records will be used. A result of this is that the URL comparison rules need to change to reflect that sip:user@foo and sip:user@foo:5060 are NOT equivalent any longer. I think this should not cause any serious interoperability issues, but further consideration is needed.

### 3.3 Details of 2782 process

RFC 2782 spells out the details of how a set of SRV records are

sorted and then tried. However, it only states that the client should "try to connect to the (protocol, address, service)" without giving any details on what happens in the event of failure. Those details, in the case of SIP, are described here.

For SIP requests, failure occurs if the transaction layer reports a 503 error response or a transport failure of some sort (generally, due to ICMP errors or TCP connection failures). Failure also occurs if the transaction layer times out without ever having received ANY response, provisional or final (i.e., timer B or timer F fires). If a failure occurs, the client SHOULD create a new request, which is identical to the previous, but has a different value of the Via branch ID than the previous (and therefore constitutes a new SIP transaction). That request is sent to the next element in the list as specified by rfc2782.

A server has been contacted "successfully" if a request sent to that server generates any kind of response, provisional or final. A mapping of the tuple (TARGET, input TRANSPORT, input PORT) to a specific server (IP address, transport, port) that was contacted successfully SHOULD be cached for a duration equal to the TTL of the A record for that server itself. Note, in the above tuple, input TRANSPORT and input PORT refer to the transport and port values from the URI itself, if present.

If a client attempts to contact the server listed in the cache, but the request fails, the server MUST be removed from the cache, and the entire DNS processing must restart by following the procedures in Section 3.1 again.

## 3.4 Consideration for Stateless Proxies

The process of the previous sections is highly stateful. When a server is contacted successfully, all requests for the transaction (plus a CANCEL for that transaction) MUST go to the same server. The identity of the successfully contacted server is a form of transaction state. This presents a challenge for stateless proxies, which still need to meet the requiretment for sending all requests in the transaction to the same server.

The requirement is not difficult to meet in the simple case where there were no failures when attempting to contact a server. Whenever the stateless proxy receives the request, it performs the appropriate DNS queries as described above. Unfortunately, the procedures of RFC 2782 and RFC 2915 are not guaranteed to be deterministic. This is because records that contain the same priority and weight (in the case of SRV) or order and preference (in the case of NAPTR) have no specified order. The stateless proxy MUST define a deterministic

order to the records in that case, using any algorithm at its disposal. One suggestion is to alphabetize them, for example. To make life easier for stateless proxies, it is RECOMMENDED that domain administrators make the weights of SRV records with equal priority different (for example, using weights of 1000 and 1001 if two servers are equivalent, rather than assigning both a weight of 1000), and similarly for NAPTR records. If the first server is contacted successfully, things are fine. However, if the first server is not contacted successfully, and a subsequent server is, the proxy cannot remain stateless for this transaction. This is because a retransmission could very well go to a different server if the failed one recovers between retransmissions. As such, whenever a proxy does not successfully contact the first server, it SHOULD act as a stateful proxy.

## 4 Server Usage

RFC 2543bis defines procedures for sending responses from a server back to the client. Typically, for unicast requests, the response is sent back to the source IP address where the request came from, using the port contained in the Via header. However, it is important to provide failover support when the client element fails between sending the request and receiving the response.

The procedures here are invoked when a server sends a response to the client and that response fails. "Fails" is defined here as any response which causes an ICMP error message to be returned, or when the transport connection the request came in on closes before the response can be sent.

In these cases, the server examines the value of the sent-by construction in the topmost Via header. If it contains a numeric IP address, the server attempts to send the response to that address, using the transport from the Via header, and the port from sent-by, if present, else the default for that transport.

If, however, the sent-by field contained a domain name and a port number, the server queries for A records with that name. It tries to send the response to each element on the resulting list of IP addresses, using the port from the Via, and the transport from the Via. As in the client processing, the next entry in the list is tred if the one before it results in a failure.

If, however, the sent-by field contained a domain name and no port, the server queries for SRV records using the service identifier "_sip" and the transport from the topmost Via header. The resulting list is sorted as described in [2], and the response is sent to the topmost element on the new list described there. If that results in a failure, the next entry on the list is tried.

## 5 Security Considerations

The authors do not believe that this specification introduces any additional security issues beyond those already described in RFC 2782 and RFC 2915.

## 6 Registration of NATPR D2T Resolution Service

Name: Domain Name to Transport
* Mnemonic: D2T
* Number of Operands: 1
* Type of Each Operand: Each operand is a domain
* Format of Each Operand: Each operand is a domain name in standard format
* Algorithm: Opaque
* Output: One or more SRV record keys
* Error Conditions:
  o No overlap in transport between client and server

* Security Considerations:

## 7 Author's Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

## 8 Bibliography

[1] J. Rosenberg, H. Schulzrinne, et al. , "SIP: Session initiation protocol," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[2] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," Request for Comments 2782, Internet Engineering Task Force, Feb. 2000.

[3] M. Mealling and R. Daniel, "The naming authority pointer (NAPTR)

DNS resource record," Request for Comments 2915, Internet Engineering Task Force, Sept. 2000.