Network Working Group
Request for Comments: 3263
Category: Standards Track
Obsoletes: 2543

Rosenberg/Schulzrinne
dynamicsoft/Columbia U.
June 2002

Session Initiation Protocol (SIP): Locating SIP Servers

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The Session Initiation Protocol (SIP) uses DNS procedures to allow a client to resolve a SIP Uniform Resource Identifier (URI) into the IP address, port, and transport protocol of the next hop to contact. It also uses DNS to allow a server to send a response to a backup client if the primary client has failed. This document describes those DNS procedures in detail.

Contents

1 Introduction

The Session Initiation Protocol (SIP) (RFC 3261 [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 in domain A (UA1) wishes to call Joe in domain B (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 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 two aspects of the general call flow described in the Introduction. The first is for proxy 1 to discover the SIP server in domain B, in order to forward the call for joe@B. The second is for proxy 2 to identify a backup for proxy 1 in the event it fails after forwarding the request.

For the first aspect, proxy 1 specifically needs to determine the IP address, port, and transport protocol for the server in domain B. The choice of transport protocol is particularly noteworthy. Unlike many other protocols, SIP can run over a variety of transport protocols, including TCP, UDP, and SCTP. SIP can also use TLS. Currently, use of TLS is defined for TCP only. Thus, clients need to be able to automatically determine which transport protocols are available. The proxy sending the request has a particular set of transport protocols it supports and a preference for using those transport protocols. Proxy 2 has its own set of transport protocols it supports, and relative preferences for those transport protocols. All proxies must implement both UDP and TCP, along with TLS over TCP, so that there is always an intersection of capabilities. Some form of DNS procedures are needed for proxy 1 to discover the available transport protocols for SIP services at domain B, and the relative preferences of those transport protocols. Proxy 1 intersects its list of supported transport protocols with those of proxy 2 and then chooses the protocol preferred by proxy 2.

It is important to note that DNS lookups can be used multiple times throughout the processing of a call. In general, an element that wishes to send a request (called a client) may need to perform DNS processing

```
 ...........................          ..............................
 .                         .          .                            .
 .                +-------+.          . +-------+                  .
 .                |       |.          . |       |                  .
 .                | Proxy |-------------| Proxy |                  .
 .                |   1   |.          . |  2    |                  .
 .                |       |.          . |       |                  .
 .              / +-------+.          . +-------+ \\               .
 .             /           .          .            \\              .
 .            /            .          .             \\             .
 .           /             .          .              \\            .
 .          /              .          .               \\           .
 .         /               .          .                \\          .
 .        /                .          .                 \\         .
 .       /                 .          .                  \\        .
 .   +-------+             .          .                +-------+   .
 .   |       |             .          .                |       |   .
 .   |       |             .          .                |       |   .
 .   | UA 1  |             .          .                | UA 2  |   .
 .   |       |             .          .                |       |   .
 .   +-------+             .          .                +-------+   .
 .              Domain A   .          .  Domain B                  .
 ...........................          ..............................
```

Figure 1: The SIP trapezoid

to determine the IP address, port, and transport protocol of a next hop element, 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 time from when the caller initiates a call until the time the called party is alerted should be no more than a few seconds. Given that there can be multiple hops, each of which is doing DNS lookups in addition to other potentially time-intensive operations, the amount of time available for DNS lookups at each hop is limited.

Scalability and high availability are important in SIP. SIP services scale up through clustering techniques. Typically, in a realistic version of the network in Figure 1, proxy 2 would 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.

SIP assures high availability by having upstream elements detect failures. For example, assume that proxy 2 is implemented as a cluster of two proxies, proxy 2.1 and proxy 2.2. If proxy 1 sends a request to proxy 2.1 and the request fails, it retries the request by sending it to proxy 2.2. In many cases, 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 may 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, and the possibly long intervals between communications between a pair of domains, it is not generally possible for an element to 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 that is 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. The client cannot query a failed server continuously to determine when it becomes available again, since this does not scale. 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. The second aspect of the flow in the introduction for which DNS is needed, is for proxy 2 to identify a backup for proxy 1 that it can send the response to. This problem is more realistic in SIP than it is in other transactional protocols. The reason is that some SIP responses 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 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [4] and indicate requirement levels for compliant SIP implementations.

4 Client Usage

Usage of DNS differs for clients and for servers. This section discusses client usage. We assume that the client is stateful (either a User Agent Client (UAC) or a stateful proxy). Stateless proxies are discussed in Section 4.4.

The procedures here are invoked when a client needs to send a request to a resource identified by a SIP or SIPS (secure SIP) URI. This URI can identify the desired resource to which the request is targeted (in which case, the URI is found in the Request-URI), or it can identify an intermediate hop towards that resource (in which case, the URI is found in the Route header). The procedures defined here in no way affect this URI (i.e., the URI is not rewritten with the result of the DNS lookup), they only result in an IP address, port and transport protocol where the request can be sent. RFC 3261 [1] provides guidelines on determining which URI needs to be resolved in DNS to determine the host that the request needs to be sent to. In some cases, also documented in [1], the request can be sent to a specific intermediate proxy not identified by a SIP URI, but rather, by a hostname or numeric IP address. In that case, a temporary URI, used for purposes of this specification, is constructed. That URI is of the form sip:¡proxy¿, where ¡proxy¿ is the FQDN or numeric IP address of the next-hop proxy. As a result, in all cases, the problem boils down to resolution of a SIP or SIPS URI in DNS to determine the IP address, port, and transport of the host to which the request is to be sent.

The procedures here MUST be done exactly once per transaction, where transaction is as defined in [1]. That is, once a SIP server has successfully been contacted (success is defined below), all retransmissions of the SIP request and the ACK for non-2xx SIP responses to INVITE MUST be sent to the same host.

Furthermore, a CANCEL for a particular SIP request MUST be sent to the same SIP server that the SIP request was delivered to.

Because the ACK request for 2xx responses to INVITE 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 not get the ACK).

We define TARGET as the value of the maddr parameter of the URI, if present, otherwise, the host value of the hostport component of the URI. It identifies the domain to be contacted. A description of the SIP and SIPS URIs and a definition of these parameters can be found in [1].

We determine the transport protocol, port and IP address of a suitable instance of TARGET in Sections 4.1 and 4.2.

4.1 Selecting a Transport Protocol

First, the client selects a transport protocol.

If the URI specifies a transport protocol in the transport parameter, that transport protocol SHOULD be used.

Otherwise, if no transport protocol is specified, but the TARGET is a numeric IP address, the client SHOULD use UDP for a SIP URI, and TCP for a SIPS URI. Similarly, if no transport protocol is specified, and the TARGET is not numeric, but an explicit port is provided, the client SHOULD use UDP for a SIP URI, and TCP for a SIPS URI. This is because UDP is the only mandatory transport in RFC 2543 [6], and thus the only one guaranteed to be interoperable for a SIP URI. It was also specified as the default transport in RFC 2543 when no transport was present in the SIP URI. However, another transport, such as TCP, MAY be used if the guidelines of SIP mandate it for this particular request. That is the case, for example, for requests that exceed the path MTU.

Otherwise, if no transport protocol or port is specified, and the target is not a numeric IP address, the client SHOULD perform a NAPTR query for the domain in the URI. The services relevant for the task of transport protocol selection are those with NAPTR service fields with values "SIP+D2X" and "SIPS+D2X", where X is a letter that corresponds to a transport protocol supported by the domain. This specification defines D2U for UDP, D2T for TCP, and D2S for SCTP. We also establish an IANA registry for NAPTR service name to transport protocol mappings.

These NAPTR records provide a mapping from a domain to the SRV record for contacting a server with the specific transport protocol in the NAPTR services field. The resource record will contain an empty regular expression and a replacement value, which is the SRV record for that particular transport protocol. If the server supports multiple transport protocols, there will be multiple NAPTR records, each with a different service value. As per RFC 2915 [3], the client discards any records whose services fields are not applicable. For the purposes of this specification, several rules are defined. First, a client resolving a SIPS URI MUST discard any services that do not contain "SIPS" as the protocol in the service field. The converse is not true, however. A client resolving a SIP URI SHOULD retain records with "SIPS" as the protocol, if the client supports TLS. Second, a client MUST discard any service fields that identify a resolution service whose value is not "D2X", for values of X that indicate transport protocols supported by the client. The NAPTR processing as described in RFC 2915 will result in the discovery of the most preferred transport protocol of the server that is supported by the client, as well as an SRV record for the server. It will also allow the client to discover if TLS is available and its preference for its usage.

As an example, consider a client that wishes to resolve `sip:user@example.com`. The client performs a NAPTR query for that domain, and the following NAPTR records are returned:

```
;          order pref flags service      regexp  replacement
   IN NAPTR 50   50  "s"  "SIPS+D2T"     ""  _sips._tcp.example.com.
   IN NAPTR 90   50  "s"  "SIP+D2T"      ""  _sip._tcp.example.com
   IN NAPTR 100  50  "s"  "SIP+D2U"      ""  _sip._udp.example.com.
```

This indicates that the server supports TLS over TCP, TCP, and UDP, in that order of preference. Since the client supports TCP and UDP, TCP will be used, targeted to a host determined by an SRV lookup of `_sip._tcp.example.com`. That lookup would return:

```
;;          Priority Weight Port   Target
    IN SRV  0        1      5060   server1.example.com
    IN SRV  0        2      5060   server2.example.com
```

If a SIP proxy, redirect server, or registrar is to be contacted through the lookup of NAPTR records, there MUST be at least three records - one with a "SIP+D2T" service field, one with a "SIP+D2U" service field, and one with a "SIPS+D2T" service field. The records with SIPS as the protocol in the service field SHOULD be preferred (i.e., have a lower value of the order field) above records with SIP as the protocol in the service field. A record with a "SIPS+D2U" service field SHOULD NOT be placed into the DNS, since it is not possible to use TLS over UDP.

It is not necessary for the domain suffixes in the NAPTR replacement field to match the domain of the original query (i.e., `example.com` above). However, for backwards compatibility with RFC 2543, a domain MUST maintain SRV records for the domain of the original query, even if the NAPTR record is in a different domain. As an example, even though the SRV record for TCP is `_sip._tcp.school.edu`, there MUST also be an SRV record at `_sip._tcp.example.com`.

> RFC 2543 will look up the SRV records for the domain directly. If these do not exist because the NAPTR replacement points to a different domain, the client will fail.

For NAPTR records with SIPS protocol fields, (if the server is using a site certificate), the domain name in the query and the domain name in the replacement field MUST both be valid based on the site certificate handed out by the server in the TLS exchange. Similarly, the domain name in the SRV query and the domain name in the target in the SRV record MUST both be valid based on the same site certificate. Otherwise, an attacker could modify the DNS records to contain replacement values in a different domain, and the client could not validate that this was the desired behavior or the result of an attack.

If no NAPTR records are found, the client constructs SRV queries for those transport protocols it supports, and does a query for each. Queries are done using the service identifier "_sip" for SIP URIs and "_sips" for SIPS URIs. A particular transport is supported if the query is successful. The client MAY use any transport protocol it desires which is supported by the server.

> This is a change from RFC 2543. It specified that a client would lookup SRV records for all transports it supported, and merge the priority values across those records. Then, it would choose the most preferred record.

If no SRV records are found, the client SHOULD use TCP for a SIPS URI, and UDP for a SIP URI. However, another transport protocol, such as TCP, MAY be used if the guidelines of SIP mandate it for this particular request. That is the case, for example, for requests that exceed the path MTU.

4.2 Determining Port and IP Address

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

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

If the TARGET was not a numeric IP address, but a port is present in the URI, the client performs an A or AAAA record lookup of the domain name. The result will be a list of IP addresses, each of which can be contacted at the specific port from the URI and transport protocol determined previously. The client SHOULD try the first record. If an attempt should fail, based on the definition of failure in Section 4.3, the next SHOULD be tried, and if that should fail, the next SHOULD be tried, and so on.

> This is a change from RFC 2543. Previously, if the port was explicit, but with a value of 5060, SRV records were used. Now, A or AAAA records will be used.

If the TARGET was not a numeric IP address, and no port was present in the URI, the client performs an SRV query on the record returned from the NAPTR processing of Section 4.1, if such processing was performed. If it was not, because a transport was specified explicitly, the client performs an SRV query for that specific transport, using the service identifier "_sips" for SIPS URIs. For a SIP URI, if the client wishes to use TLS, it also uses the service identifier "_sips" for that specific transport, otherwise, it uses "_sip". If the NAPTR processing was not done because no NAPTR records were found, but an SRV query for a supported transport protocol was successful, those SRV records are selected. Irregardless of how the SRV records were determined, the procedures of RFC 2782, as described in the section titled "Usage rules" are followed, augmented by the additional procedures of Section 4.3 of this document.

If no SRV records were found, the client performs an A or AAAA record lookup of the domain name. The result will be a list of IP addresses, each of which can be contacted using the transport protocol determined previously, at the default port for that transport. Processing then proceeds as described above for an explicit port once the A or AAAA records have been looked up.

4.3 Details of RFC 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 are described here for SIP.

For SIP requests, failure occurs if the transaction layer reports a 503 error response or a transport failure of some sort (generally, due to fatal ICMP errors in UDP or connection failures in TCP). 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 in RFC 3261 [1] 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 RFC 2782.

4.4 Consideration for Stateless Proxies

The process of the previous sections is highly stateful. When a server is contacted successfully, all retransmissions of the request for the transaction, as well as ACK for a non-2xx final response, and CANCEL requests 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 requirement for sending all requests in the transaction to the same server.

The problem is similar, but different, to the problem of HTTP transactions within a cookie session getting routed to different servers based on DNS randomization. There, such distribution is not a problem. Farms of servers generally have common back-end data stores, where the session data is stored. Whenever a server in the farm receives an HTTP request, it takes the session identifier, if present, and extracts the needed state to process the request. A request without a session identifier creates a new one. The problem with stateless proxies is at a lower layer; it is retransmitted requests within a transaction that are being potentially spread across servers. Since none of these retransmissions carries a "session identifier" (a complete dialog identifier in SIP terms), a new dialog would be created identically at each server. This could, for example result in multiple phone calls to be made to the same phone. Therefore, it is critical to prevent such a thing from happening in the first place.

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 detailsDNS queries as described above. However, the procedures of RFC 2782 are not guaranteed to be deterministic. This is because records that contain the same priority 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, or, more generally, sort them by ASCII-compatible encoding. To make processing 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, the proxy can remain stateless. However, if the first server is not contacted successfully, and a subsequent server is, the proxy cannot remain stateless for this transaction. If it were stateless, 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.

Unfortunately, it is still possible for a stateless proxy to deliver retransmissions to different servers, even if it follows the recommendations above. This can happen if the DNS TTLs expire in the middle of a transaction, and the entries had changed. This is unavoidable. Network implementors should be aware of this limitation, and not use stateless proxies that access DNS if this error is deemed critical.

5 Server Usage

RFC 3261 [1] defines procedures for sending responses from a server back to the client. Typically, for unicast UDP requests, the response is sent back to the source IP address where the request came from, using the port contained in the Via header. For reliable transport protocols, the response is sent over the connection the request arrived on. However, it is important to provide failover support when the client element fails between sending the request and receiving the response.

A server, according to RFC 3261 [1], will send a response on the connection it arrived on (in the case of reliable transport protocols), and for unreliable transport protocols, to the source address of the request, and the port in the Via header field. The procedures here are invoked when a server attempts to send to that location and that response fails (the specific conditions are detailed in RFC 3261). "Fails" is defined as any closure of the transport connection the request came in on before the response can be sent, or communication of a fatal error from the transport layer.

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 protocol from the Via header, and the port from sent-by, if present, else the default for that transport protocol. The transport protocol in the Via header can indicate “TLS”, which refers to TLS over TCP. When this value is present, the server MUST use TLS over TCP to send the response.

If, however, the sent-by field contained a domain name and a port number, the server queries for A or AAAA 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 protocol from the Via (again, a value of TLS refers to TLS over TCP). As in the client processing, the next entry in the list is tried 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 at that domain name using the service identifier “_sips” if the Via transport is “TLS”, “_sip” otherwise, and the transport from the topmost Via header (“TLS” implies that the transport protocol in the SRV query is TCP). 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.

6 Constructing SIP URIs

In many cases, an element needs to construct a SIP URI for inclusion in a Contact header in a REGISTER, or in a Record-Route header in an INVITE. According to RFC 3261 [1], these URIs have to have the property that they resolve to the specific element that inserted them. However, if they are constructed with just an IP address, for example:

```
sip:1.2.3.4
```

then should the element fail, there is no way to route the request or response through a backup.

SRV provides a way to fix this. Instead of using an IP address, a domain name that resolves to an SRV record can be used:

```
sip:server23.provider.com
```

The SRV records for a particular target can be set up so that there is a single record with a low value for the priority field (indicating the preferred choice), and this record points to the specific element that constructed the URI. However, there are additional records with higher values of the priority field that point to backup elements that would be used in the event of failure. This allows the constraint of RFC 3261 [1] to be met while allowing for robust operation.

7 Security Considerations

DNS NAPTR records are used to allow a client to discover that the server supports TLS. An attacker could potentially modify these records, resulting in a client using a non-secure transport when TLS is in fact available and preferred.

This is partially mitigated by the presence of the sips URI scheme, which is always sent only over TLS. An attacker cannot force a bid down through deletion or modification of DNS records. In the worst case, they can prevent communication from occurring by deleting all records. A sips URI itself is generally exchanged within a secure context, frequently on a business card or secure web page, or within a SIP message which has already been secured with TLS. See RFC 3261 [1] for details. The sips URI is therefore preferred when security is truly needed, but we allow TLS to be used for requests resolved by a SIP URI to allow security that is better than no TLS at all.

The bid down attack can also be mitigated through caching. A client which frequently contacts the same domain SHOULD cache whether or not its NAPTR records contain SIPS in the services field. If such records were present, but in later queries cease to appear, it is a sign of a potential attack. In this case, the client SHOULD generate some kind of alert or alarm, and MAY reject the request.

An additional problem is that proxies, which are intermediaries between the users of the system, are frequently the clients that perform the NAPTR queries. It is therefore possible for a proxy to ignore SIPS entries even though they are present, resulting in downgraded security. There is very little that can be done to prevent such attacks. Clients are simply dependent on proxy servers for call completion, and must trust that they implement the protocol properly in order for security to be provided. Falsifying DNS records can be done by tampering with wire traffic (in the absence of DNSSEC), whereas compromising and commandeering a proxy server requires a break-in, and is seen as the considerably less likely downgrade threat.

8 The Transport Determination Application

This section more formally defines the NAPTR usage of this specification, using the Dynamic Delegation Discovery System (DDDS) framework as a guide [7]. DDDS represents the evolution of the NAPTR resource record. DDDS defines applications, which can make use of the NAPTR record for specific resolution services. This application is called the Transport Determination Application, and its goal is to map an incoming SIP or SIPS URI to a set of SRV records for the various servers that can handle the URI.

The following is the information that DDDS requests an application to provide:

Application Unique String: The Application Unique String (AUS) is the input to the resolution service. For this application, it is the URI to resolve.

First Well Known Rule: The first well known rule extracts a key from the AUS. For this application, the first well known rule extracts the host portion of the SIP or SIPS URI.

Valid Databases: The key resulting from the first well known rule is looked up in a single database, the DNS [8].

Expected Output: The result of the application is an SRV record for the server to contact.

9 IANA Considerations

The usage of NAPTR records described here requires well known values for the service fields for each transport supported by SIP. The table of mappings from service field values to transport protocols is to be maintained by IANA. New entries in the table MAY be added through the publication of standards track RFCs, as described in RFC 2434 [5].

The registration in the RFC MUST include the following information:

Service Field: The service field being registered. An example for a new fictitious transport protocol called NCTP might be "SIP+D2N".

Protocol: The specific transport protocol associated with that service field. This MUST include the name and acronym for the protocol, along with reference to a document that describes the transport protocol. For example - "New Connectionless Transport Protocol (NCTP), RFC 5766".

Name and Contact Information: The name, address, email address and telephone number for the person performing the registration.

The following values have been placed into the registry:

```
Services Field                Protocol
SIP+D2T                        TCP
SIPS+D2T                       TCP
SIP+D2U                        UDP
SIP+D2S                        SCTP (RFC 2960)
```

10 Acknowledgements

The authors would like to thank Randy Bush, Leslie Daigle, Patrik Faltstrom, Jo Hornsby, Rohan Mahy, Allison Mankin, Michael Mealling, Thomas Narten, and Jon Peterson for their useful comments.

Normative References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: session initiation protocol," RFC 3261, Internet Engineering Task Force, June 2002.

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

[3] M. Mealling and R. Daniel, "The naming authority pointer (NAPTR) DNS resource record," RFC 2915, Internet Engineering Task Force, Sept. 2000.

[4] A. Ramos, "Request for comments summary RFC numbers 2100-2199," RFC 2199, Internet Engineering Task Force, Jan. 1998.

[5] T. Narten and H. Alvestrand, "Guidelines for writing an IANA considerations section in RFCs," RFC 2434, Internet Engineering Task Force, Oct. 1998.

Informative References

[6] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," RFC 2543, Internet Engineering Task Force, Mar. 1999.

[7] M. Mealling, "Dynamic delegation discovery system (DDDS) part one: The comprehensive DDDS standard," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

[8] M. Mealling, "Dynamic delegation discovery system (DDDS) part three: The DNS database," Internet Draft, Internet Engineering Task Force, May 2002. Work in progress.

11 Authors' Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936

EMail: jdrosen@dynamicsoft.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.
New York, NY 10027-7003

EMail: schulzrinne@cs.columbia.edu

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.