Network Working Group
Request for Comments: 3581
Category: Standards Track

J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia University

August 2003

An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing

Status of this Memo

Copyright Notice

Abstract

The Session Initiation Protocol (SIP) operates over UDP and TCP, among others. When used with UDP, responses to requests are returned to the source address the request came from, and to the port written into the topmost Via header field value of the request. This behavior is not desirable in many cases, most notably, when the client is behind a Network Address Translator (NAT). This extension defines a new parameter for the Via header field, called "rport", that allows a client to request that the server send the response back to the source IP address and port from which the request originated.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [1] operates over UDP and TCP. When used with UDP, responses to requests are returned to the source address the request came from, and to the port written into the topmost Via header field value of the request. This results in a "hybrid" way of computing the destination of the response. Half of the information (specifically, the IP address) is taken from the IP packet headers, and the other half (specifically, the port) from the SIP message headers. SIP operates in this manner so that a server can listen for all messages, both requests and responses, on a single IP address and port. This helps improve scalability. However, this behavior is not desirable in many cases, most notably, when the client is behind a NAT. In that case, the response will not properly traverse the NAT, since it will not match the binding established with the request.

Furthermore, there is currently no way for a client to examine a response and determine the source port that the server saw in the corresponding request. Currently, SIP provides the client with the source IP address that the server saw in the request, but not the port. The source IP address is conveyed in the "received" parameter in the topmost Via header field value of the response. This information has proved useful for basic NAT traversal, debugging

purposes, and support of multi-homed hosts. However, it is incomplete without the port information.

This extension defines a new parameter for the Via header field, called "rport", that allows a client to request that the server send the response back to the source IP address and port where the request came from. The "rport" parameter is analogous to the "received" parameter, except "rport" contains a port number, not the IP address.

2. 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 BCP 14, RFC 2119 [2] and indicate requirement levels for compliant implementations.

3. Client Behavior

The client behavior specified here affects the transport processing defined in Section 18.1 of SIP (RFC 3261) [1].

A client, compliant to this specification (clients include UACs and proxies), MAY include an "rport" parameter in the top Via header field value of requests it generates. This parameter MUST have no value; it serves as a flag to indicate to the server that this extension is supported and requested for the transaction.

When the client sends the request, if the request is sent using UDP, the client MUST be prepared to receive the response on the same IP address and port it used to populate the source IP address and source port of the request. For backwards compatibility, the client MUST still be prepared to receive a response on the port indicated in the sent-by field of the topmost Via header field value, as specified in Section 18.1.1 of SIP [1].

When there is a NAT between the client and server, the request will create (or refresh) a binding in the NAT. This binding must remain in existence for the duration of the transaction in order for the client to receive the response. Most UDP NAT bindings appear to have a timeout of about one minute. This exceeds the duration of non-INVITE transactions. Therefore, responses to a non-INVITE request will be received while the binding is still in existence. INVITE transactions can take an arbitrarily long amount of time to complete. As a result, the binding may expire before a final response is received. To keep the binding fresh, the client SHOULD retransmit its INVITE every 20 seconds or so. These retransmissions will need to take place even after receiving a provisional response.

A UA MAY execute the binding lifetime discovery algorithm in Section 10.2 of RFC 3489 [4] to determine the actual binding lifetime in the NAT. If it is longer than 1 minute, the client SHOULD increase the interval for request retransmissions up to half of the discovered lifetime. If it is shorter than one minute, it SHOULD decrease the interval for request retransmissions to half of the discovered lifetime. Note that discovery of binding lifetimes can be unreliable. See Section 14.3 of RFC 3489 [4].

4. Server Behavior

The server behavior specified here affects the transport processing defined in Section 18.2 of SIP [1].

When a server compliant to this specification (which can be a proxy or UAS) receives a request, it examines the topmost Via header field value. If this Via header field value contains an "rport" parameter with no value, it MUST set the value of the parameter to the source port of the request. This is analogous to the way in which a server will insert the "received" parameter into the topmost Via header field value. In fact, the server MUST insert a "received" parameter containing the source IP address that the request came from, even if it is identical to the value of the "sent-by" component. Note that this processing takes place independent of the transport protocol.

When a server attempts to send a response, it examines the topmost Via header field value of that response. If the "sent-protocol" component indicates an unreliable unicast transport protocol, such as UDP, and there is no "maddr" parameter, but there is both a "received" parameter and an "rport" parameter, the response MUST be sent to the IP address listed in the "received" parameter, and the port in the "rport" parameter. The response MUST be sent from the same address and port that the corresponding request was received on. This effectively adds a new processing step between bullets two and three in Section 18.2.2 of SIP [1].

The response must be sent from the same address and port that the request was received on in order to traverse symmetric NATs. When a server is listening for requests on multiple ports or interfaces, it will need to remember the one on which the request was received. For a stateful proxy, storing this information for the duration of the transaction is not an issue. However, a stateless proxy does not store state between a request and its response, and therefore cannot remember the address and port on which a request was received. To properly implement this specification, a stateless proxy can encode the destination address and port of a request into the Via header field value that it inserts. When the response arrives, it can extract this information and use it to forward the response.

5. Syntax

The syntax for the "rport" parameter is:

```
response-port = "rport" [EQUAL 1*DIGIT]
```

This extends the existing definition of the Via header field parameters, so that its BNF now looks like:

```
via-params        =  via-ttl / via-maddr
                     / via-received / via-branch
                     / response-port / via-extension
```

6. Example

A client sends an INVITE to a proxy server which looks like, in part:

```
INVITE sip:user@example.com SIP/2.0
Via: SIP/2.0/UDP 10.1.1.1:4540;rport;branch=z9hG4bKkjshdyff
```

This INVITE is sent with a source port of 4540 and a source IP address of 10.1.1.1. The proxy is at 192.0.2.2 (proxy.example.com), listening on both port 5060 and 5070. The client sends the request to port 5060. The request passes through a NAT on the way to the proxy, so that the source IP address appears as 192.0.2.1 and the source port as 9988. The proxy forwards the request, but not before appending a value to the "rport" parameter in the proxied request:

```
INVITE sip:user@example.com SIP/2.0
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4bKkjsh77
Via: SIP/2.0/UDP 10.1.1.1:4540;received=192.0.2.1;rport=9988
 ;branch=z9hG4bKkjshdyff
```

This request generates a response which arrives at the proxy:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy.example.com;branch=z9hG4bKkjsh77
Via: SIP/2.0/UDP 10.1.1.1:4540;received=192.0.2.1;rport=9988
 ;branch=z9hG4bKkjshdyff
```

The proxy strips its top Via header field value, and then examines the next one. It contains both a "received" parameter and an "rport" parameter. The server follows the rules specified in Section 4 and sends the response to IP address 192.0.2.1, port 9988, and sends it from port 5060 on 192.0.2.2:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.1.1.1:4540;received=192.0.2.1;rport=9988
 ;branch=z9hG4bKkjshdyff
```

This packet matches the binding created by the initial request. Therefore, the NAT rewrites the destination address of this packet back to 10.1.1.1, and the destination port back to 4540. It forwards this response to the client, which is listening for the response on that address and port. The client properly receives the response.

7. Security Considerations

When a server uses this specification, responses that it sends will now include the source port where the request came from. In some instances, the source address and port of a request are sensitive information. If they are sensitive, requests SHOULD be protected by using SIP over TLS [1]. In such a case, this specification does not provide any response routing functions (as these only work with TCP); it merely provides the client with information about the source port as seen by the server.

It is possible that an attacker might try to disrupt service to a client by acting as a man-in-the-middle, modifying the "rport" parameter in a Via header in a request sent by a client. Removal of this parameter will prevent clients from behind NATs from receiving service. The addition of the parameter will generally have no impact. Of course, if an attacker is capable of launching a man-in-the-middle attack, there are many other ways of denying service, such as merely discarding the request. Therefore, this attack does not seem significant.

8. IANA Considerations

There are no IANA considerations associated with this specification.

9. IAB Considerations

The IAB has studied a class of protocols referred to as Unilateral Self Address Fixing (UNSAF) protocols [5]. These protocols allow a client behind a NAT to learn the IP address and port that a NAT will allocate for a particular request, in order to use this information in application layer protocols. An example of an UNSAF protocol is the Simple Traversal of UDP Through NATs (STUN) [4].

Any protocol is an UNSAF protocol if it reveals, to a client, the source IP address and port of a packet sent through that NAT. Although not designed for that purpose, this specification can be used as an UNSAF protocol. Using the "rport" parameter (defined here) and the "received" parameter (defined in RFC 3261 [1]) in the topmost Via header field value of a response, a client sending a request can learn its address as it was seen by the server which sent the response.

There are two uses of this information. The first is for registrations. Consider a client behind a NAT wishing to register with a proxy/registrar on the other side of the NAT. The client must provide, in its registration, the address at which it should receive incoming SIP requests from the proxy. However, since the client is located behind a NAT, none of the addresses on any of its interfaces will be reachable from the proxy. If the client can provide the proxy with an address that the proxy can reach, the client can receive incoming requests. Using this specification, a client behind a NAT can learn its address and port as seen by the proxy which receives a REGISTER request. The client can then perform an additional registration, using this address in a Contact header. This would allow a client to receive incoming requests, such as INVITE, on the IP address and port it used to populate the source IP address and port of the registration it sent. This approach will only work when servers send requests to a UA from the same address and port on which the REGISTER itself was received.

In many cases, the server to whom the registration is sent won't be the registrar itself, but rather a proxy which then sends the request to the registrar. In such a case, any incoming requests for the client must traverse the proxy to whom the registration was directly sent. The Path header extension to SIP [3] allows the proxy to indicate that it must be on the path of such requests.

The second usage is for record routing, to address the same problem as above, but between two proxies. A proxy behind a NAT which forwards a request to a server can use OPTIONS, for example, to learn its address as seen by that server. This address can be placed into the Record-Route header field of requests sent to that server. This would allow the proxy to receive requests from that server on the same IP address and port it used to populate the source IP address and port of the OPTIONS request.

Because of this potential usage, this document must consider the issues raised in [5].

9.1. Problem Definition

From [5], any UNSAF proposal must provide:

> Precise definition of a specific, limited-scope problem that is to be solved with the UNSAF proposal. A short term fix should not be generalized to solve other problems; this is why "short term fixes usually aren't".

This specification is primarily aimed at allowing SIP responses to be received when a request is sent through a NAT. In this primary application, this specification is not an UNSAF proposal. However, as a side effect of this capability, this specification can be used as an UNSAF protocol. In that usage, it would address two issues:

- Provide a client with an address that it could use in the Contact header of a REGISTER request when it is behind a NAT.
- Provide a proxy with an address that it could use in a Record-Route header in a request, when it is behind a NAT.

9.2. Exit Strategy

From [5], any UNSAF proposal must provide:

> Description of an exit strategy/transition plan. The better short term fixes are the ones that will naturally see less and less use as the appropriate technology is deployed.

The SIP working group has recognized that the usage of this specification to support registrations and record-routing through NATs is not appropriate. It has a number of known problems which are documented below. The way to eliminate potential usage of this specification for address fixing is to provide a proper solution to the problems that might motivate the usage of this specification for address fixing. Specifically, appropriate solutions for registrations and record-routing in the presence of NATs need to be developed. These solutions would not rely on address fixing.

Requirements for such solutions are already under development [6].

Implementors of this specification are encouraged to follow this work for better solutions for registrations and record-routing through NAT.

9.3. Brittleness Introduced by this Specification

From [5], any UNSAF proposal must provide:

> Discussion of specific issues that may render systems more "brittle". For example, approaches that involve using data at multiple network layers create more dependencies, increase debugging challenges, and make it harder to transition.

This specification, if used for address fixing, introduces several points of brittleness into a SIP system:

- If used for UDP registrations, a client will need to frequently re-register in order to keep the NAT bindings fresh. In many cases, these registrations will need to take place nearly one hundred times more frequently than the typical refresh interval of a registration. This introduces load into the system and hampers scalability.

- A client cannot accurately determine the binding lifetime of a NAT it is registering (or record-routing) through. Therefore, there may be periods of unreachability that occur between the time a binding expires and the next registration or OPTIONS refresh is sent. This may result in missed calls, messages, or other information.

- If the NAT is of the symmetric variety [4], a client will only be able to use its address to receive requests from the server it has sent the request to. If that server is one of many servers in a cluster, the client may not be able to receive requests from other servers in the cluster. This may result in missed calls, messages, or other information.

- If the NAT is of the symmetric variety [4], a client will only be able to use its address to receive requests if the server sends requests to the client from the same address and port the server received the registrations on. This server behavior is not mandated by RFC 3261 [1], although it appears to be common in practice.

- If the registrar and the server to whom the client sent its REGISTER request are not the same, the approach will only work if the server uses the Path header field [3]. There is not an easy and reliable way for the server to determine that the Path header should be used for a registration. Using Path when the address in the topmost Via header field is a private address will usually work, but may result in usage of Path when it is not actually needed.

9.4. Requirements for a Long Term Solution

From [5], any UNSAF proposal must provide:

> Identify requirements for longer term, sound technical solutions -- contribute to the process of finding the right longer term solution.

The brittleness described in Section 9.3 has led us to the following requirements for a long term solution:

The client should not need to specify its address. Registrations and record routing require the client to specify the address at which it should receive requests. A sound technical solution should allow a client to explicitly specify that it wants to receive incoming requests on the connection over which the outgoing request was sent. In this way, the client does not need to specify its address.

The solution must deal with clusters of servers. In many commercially deployed SIP systems, there will be multiple servers, each at different addresses and ports, handling incoming requests for a client. The solution must explicitly consider this case.

The solution must not require increases in network load. There cannot be a penalty for a sound technical solution.

9.5. Issues with Existing NAPT Boxes

From [5], any UNSAF proposal must provide:

> Discussion of the impact of the noted practical issues with existing, deployed NA[P]Ts and experience reports.

To our knowledge, at the time of writing, there is only very limited usage of this specification for address fixing. Therefore, no specific practical issues have been raised.

10. Acknowledgements

The authors would like to thank Rohan Mahy and Allison Mankin for their comments and contributions to this work.

11. References

11.1. Normative References

[1] 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.

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

[3] Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP) Extension Header Field for Registering Non-Adjacent Contacts", RFC 3327, December 2002.

[4] Rosenberg, J., Weinberger, J., Huitema, C. and R. Mahy, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)", RFC 3489, March 2003.

11.2. Informative References

[5] Daigle, L., Ed., and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[6] Mahy, R., "Requirements for Connection Reuse in the Session Initiation Protocol (SIP)", Work in Progress.

12. Intellectual Property Statement

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

13. Authors' Addresses

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

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

EMail: schulzrinne@cs.columbia.edu
URI: http://www.cs.columbia.edu/~hgs

14. Full Copyright Statement

Acknowledgement

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