Network Working Group
Request for Comments: 3486
Category: Standards Track
February 2003

G. Camarillo
Ericsson

Compressing the Session Initiation Protocol (SIP)

Status of this Memo

Copyright Notice

Abstract

This document describes a mechanism to signal that compression is desired for one or more Session Initiation Protocol (SIP) messages. It also states when it is appropriate to send compressed SIP messages to a SIP entity.

Table of Contents

1. Introduction

A SIP [1] client sending a request to a SIP server typically performs a DNS lookup for the domain name of the server. When NAPTR [4] or SRV [5] records are available for the server, the client can specify the type of service it wants. The service in this context is the transport protocol to be used by SIP (e.g., UDP, TCP or SCTP). A SIP server that supports, for instance, three different transport protocols, will have three different DNS entries.

Since it is foreseen that the number of transport protocols supported by a particular application layer protocol is not going to grow dramatically, having a DNS entry per transport seems like a scalable enough solution.

However, sometimes it is necessary to include new layers between the transport protocol and the application layer protocol. Examples of these layers are transport layer security and compression. If DNS was used to discover the availability of these layers for a particular server, the number of DNS entries needed for that server would grow dramatically.

A server that, for example, supported TCP and SCTP as transports, TLS for transport security and SigComp for signaling compression, would need the 8 DNS entries listed below:

1. TCP, no security, no compression

2. TCP, no security, SigComp

3. TCP, TLS, no compression

4. TCP, TLS, SigComp

5. SCTP, no security, no compression

6. SCTP, no security, SigComp

7. SCTP, TLS, no compression

8. SCTP, TLS, SigComp

It is clear that this way of using DNS is not scalable. Therefore, an application layer mechanism to express support of signalling compression is needed.

Note that for historical reasons both HTTP and SIP use a different port for TLS on top of TCP than for TCP alone, although at present, this solution is not considered scalable any longer.

A SIP element that supports compression will need to be prepared to receive compressed and uncompressed messages on the same port. It will perform demultiplexing based on the cookie in the topmost bits of every compressed message.

2. Overview of operation

There are two types of SIP messages; SIP requests and SIP responses. Clients send SIP requests to the host part of a URI and servers send responses to the host in the sent-by parameter of the Via header field.

We define two parameters, one for SIP URIs and the other for the Via header field. The format of both parameters is the same, as shown in the examples below:

```
sip:alice@atlanta.com;comp=sigcomp
Via: SIP/2.0/UDP server1.foo.com:5060;branch=z9hG4bK87a7;comp=sigcomp
```

The presence of this parameter (comp=sigcomp) in a URI indicates that the request has to be compressed using SigComp, as defined in [2]. The presence of comp=sigcomp in a Via header field indicates that the response has to be compressed using SigComp.

Therefore, the presence of comp=sigcomp indicates that the SIP entity identified by the URI or by the Via header field supports SigComp and is willing to receive compressed messages. Having comp=sigcomp mean "willingness" as well as "support" allows the receiver of a SIP message to influence the decision of whether or not to use SigComp at a given time.

3. SigComp implementations for SIP

Every SIP implementation that supports SigComp MUST implement the procedures described in this document.

4. Sending a Request to a Server

A request is sent to the host part of a URI. This URI, referred to as the next-hop URI, is the Request-URI of the request or an entry in the Route header field.

If the next-hop URI contains the parameter comp=sigcomp, the client SHOULD compress the request using SigComp as defined in [2].

If the next-hop URI is a SIPS URI, the request SHOULD be compressed before it is passed to the TLS layer.

A client MUST NOT send a compressed request to a server if it does not know whether or not the server supports SigComp.

Regardless of whether the request is sent compressed or not, if a client would like to receive subsequent requests within the same dialog in the UAS->UAC direction compressed, this client SHOULD add the parameter comp=sigcomp to the URI in the Contact header field if it is a user agent client. If the client is a proxy, it SHOULD add the parameter comp=sigcomp to its URI in the Record-Route header field.

If a user agent client sends a compressed request, it SHOULD add the parameter comp=sigcomp to the URI in the Contact header field. If a proxy that Record-Routes sends a compressed request, it SHOULD add comp=sigcomp to its URI in the Record-Route header field.

If a client sends a compressed request, it SHOULD add the parameter comp=sigcomp to the topmost entry of the Via header field.

If a client does not know whether or not the server supports SigComp, but in case the server supported it, it would like to receive compressed responses, this client SHOULD add the parameter comp=sigcomp to the topmost entry of the Via header field. The request, however, as stated above, will not be compressed.

4.1 Obtaining a SIP or SIPS URI with comp=sigcomp

For requests within a dialog, a next-hop URI with the comp=sigcomp parameter is obtained from a Record-Route header field when the dialog is established. A client sending a request outside a dialog can also obtain SIP URIs with comp=sigcomp in a Contact header field in a 3xx or 485 response to the request.

However, clients establishing a session will not typically be willing to wait until the dialog is established in order to begin compressing messages. One of the biggest gains that SigComp can bring to SIP is the ability to compress the initial INVITE of a dialog, when the user is waiting for the session to be established. Therefore, clients need a means to obtain a comp=sigcomp URI from their outbound proxy before the user decides to establish a session.

One solution to this problem is manual configuration. However, sometimes it is necessary to have clients configured in an automatic fashion. Unfortunately, current mechanisms for SIP client configuration (e.g., using DHCP [6]) do not allow to provide the client with URI parameters. In this case, the client SHOULD send an uncompressed OPTIONS request to its outbound proxy. The outbound proxy can provide an alternative SIP URI with the comp=sigcomp parameter in a Contact header field in a 200 OK response to the OPTIONS. The client can use this URI for subsequent requests that are sent through the same outbound proxy using compression.

RFC 3261 [1] does not define how a proxy should respond to an OPTIONS request addressed to itself. It only describes how servers respond to OPTIONS addressed to a particular user. Section 11.2 of RFC 3261 says:

> Contact header fields MAY be present in a 200 (OK) response and have the same semantics as in a 3xx response. That is, they may list a set of alternative names and methods of reaching the user.

We extend this behavior to proxy servers responding to OPTIONS addressed to them. They MAY list a set of alternative URIs to contact the proxy.

Note that receiving incoming requests (even initial INVITEs) compressed is not a problem, since user agents can REGISTER a SIP URI with comp=sigcomp in their registrar. All incoming requests for the user will be sent to this SIP URI using compression.

5. Sending a Response to a Client

A response is sent to the host in the sent-by parameter of the Via header field. If the topmost Via header field contains the parameter comp=sigcomp, the response SHOULD be compressed. Otherwise, the response MUST NOT be compressed.

In order to avoid asymmetric compression (i.e., two SIP entities exchanging compressed requests in one direction and uncompressed requests in the other direction) proxies need to rewrite their Record-Route entries in the responses. A proxy performing Record-Route inspects the Record-Route header field in the response and the Contact header field in the request that triggered this response (see example in Section 9). It looks for the URI of the next upstream (closer to the user agent client) hop in the route set. If this URI contains the parameter comp=sigcomp, the proxy SHOULD add comp=sigcomp to its entry in the Record-Route header field. If this URI does not contain the parameter comp=sigcomp, the proxy SHOULD remove comp=sigcomp (if it is present) from its entry in the Record-Route header field.

The same way, a user agent server SHOULD add comp=sigcomp to the Contact header field of the response if the URI of the next upstream hop in the route set contained the parameter comp=sigcomp.

6. Double Record-Routing

Although proxies usually add zero or one Record-Route entries to a particular request, some proxies add two of them to avoid Record-Route rewriting. A typical example of double Record-Routing is a SIP proxy that acts as a firewall between two networks. Depending on which network a request comes from, it will be received on a different interface by the proxy. The proxy adds one Record-Route entry for one interface and a second one for the other interface. This way, the proxy does not need to rewrite the Record-Route header field on the response.

Proxies that receive compressed messages from one side of the dialog (e.g., upstream) and uncompressed messages from the other side (e.g., downstream) MAY use the mechanism described above.

If a proxy detects that the next-hop proxy for a request is the proxy itself and that the request will not be sent through the network, the proxy MAY choose not to compress the request even if the URI contains the comp=sigcomp parameter.

7. Error Situations

If a compressed SIP request arrives to a SIP server that does not understand SigComp, the server will not have any means to indicate the error to the client. The message will be impossible to parse, and there will be no Via header field indicating an address to send an error response.

If a SIP client sends a compressed request and the client transaction times out without having received any response, the client SHOULD retry the same request without using compression. If the compressed request was sent over a TCP connection, the client SHOULD close that connection and open a new one to send the uncompressed request. Otherwise the server would not be able to detect the beginning of the new message.

8. Augmented BNF

This section provides the augmented Backus-Naur Form (BNF) of both parameters described above.

The compression URI parameter is a "uri-parameter", as defined by the SIP ABNF (Section 25.1 of [1]):

```
compression-param  =  "comp=" ("sigcomp" / other-compression)
other-compression  =  token
```

The Via compression parameter is a "via-extension", as defined by the SIP ABNF (Section 25.1 of [1]):

```
via-compression    =  "comp" EQUAL ("sigcomp" / other-compression)
other-compression  =  token
```

9. Example

The following example illustrates the use of the parameters defined above. The call flow of Figure 1 shows an INVITE-200 OK-ACK handshake between a UAC and a UAS through two proxies. Proxy P1 does not Record-Route but proxy P2 does. Both proxies support compression, but they do not use it by default.

```
UAC            P1            P2           UAS

 |(1)INVITE(c) |             |             |
 |------------>| (2) INVITE  |             |
 |             |------------>| (3) INVITE  |
 |             |             |------------>|
 |             |             | (4) 200 OK  |
 |             | (5) 200 OK  |<------------|
 |(6)200 OK(c) |<------------|             |
 |<------------|             |             |
 |             |  (7)ACK(c)  |             |
 |-------------------------->|   (8) ACK   |
 |             |             |------------>|
 |             |             |             |
 |             |             |             |
```

Figure 1: INVITE transaction through two proxies

Messages (1), (6) and (7) are compressed (c).

We provide a partial description of the messages involved in this call flow below. Only some parts of each message are shown, namely the Method name, the Request-URI and the Via, Route, Record-Route and

Contact header fields. We have not used a correct format for these header fields. We have rather focus on the contents of the header fields and on the presence (or absence) of the "comp=sigcomp" parameter.

```
   (1) INVITE UAS
       Via: UAC;comp=sigcomp
       Route: P1;comp=sigcomp
       Contact: UAC;comp=sigcomp
```

P1 is the outbound proxy of the UAC, and it supports SigComp. The UAC is configured to send compressed traffic to P1, and therefore, it compresses the INVITE (1). In addition, the UAC wants to receive future requests and responses for this dialog compressed. Therefore, it adds the comp=Sigcomp parameter to the Via and to the Contact header fields.

```
   (2) INVITE UAS
       Via: P1
       Via: UAC;comp=sigcomp
       Route: P2
       Contact: UAC;comp=sigcomp
```

P1 forwards the INVITE (2) to P2. P1 does not use compression by default, so it sends the INVITE uncompressed to P2.

```
   (3) INVITE UAS
       Via: P2
       Via: P1
       Via: UAC;comp=sigcomp
       Record-Route: P2
       Contact: UAC;comp=sigcomp
```

P2 forwards the INVITE (3) to the UAS. P2 supports compression, but it does not use it by default. Therefore, it sends the INVITE uncompressed. P2 wishes to remain in the signalling path and therefore it Record-Routes.

```
   (4) 200 OK
       Via: P2
       Via: P1
       Via: UAC;comp=sigcomp
       Record-Route: P2
       Contact: UAS
```

The UAS generates a 200 OK response and sends it to the host in the topmost Via, which is P2.

```
(5) 200 OK
    Via: P1
    Via: UAC;comp=sigcomp
    Record-Route: P2;comp=sigcomp
    Contact: UAS
```

P2 receives the 200 OK response. P2 Record-Routed, so it inspects the Route set for this dialog. For requests from the UAS towards the UAC (the opposite direction than the first INVITE), the next hop will be the Contact header field of the INVITE, because P1 did not Record-Route. That Contact identified the UAC:

```
Contact: UAC;comp=sigcomp
```

Since the UAC wants to receive compressed requests (Contact of the INVITE), P2 assumes that the UAC would also like to send compressed requests (Record-Route of the 200 OK). Therefore, P2 modifies its entry in the Record-Route header field of the 200 OK (5). In the INVITE (3), P2 did not used the comp=sigcomp parameter. Now it adds it in the 200 OK (5). This will allow the UAC sending compressed requests within this dialog.

```
(6) 200 OK
    Via: UAC;comp=sigcomp
    Record-Route: P2;comp=sigcomp
    Contact: UAS
```

P1 sends the 200 OK (6) compressed to the UAC because the Via header field contained the comp=sigcomp parameter.

```
(7) ACK UAS
    Via: UAC;comp=sigcomp
    Route: P2;comp=sigcomp
    Contact: UAC;comp=sigcomp
```

The UAC sends the ACK (7) compressed directly to P2 (P1 did not Record-Route).

```
(8) ACK UAS
    Via: P2
    Via: UAC;comp=sigcomp
    Contact: UAC;comp=sigcomp
```

P2 sends the ACK (8) uncompressed to the UAS.

10. Security Considerations

A SIP entity receiving a compressed message has to decompress it and to parse it. This requires slightly more processing power than only parsing a message. This implies that a denial of service attack using compressed messages would be slightly worse than an attack with uncompressed messages.

An attacker inserting the parameter comp=sigcomp in a SIP message could make a SIP entity send compressed messages to another SIP entity that did not support SigComp. Appropriate integrity mechanisms should be used to avoid this attack.

11. IANA Considerations

This document defines the "comp" uri-parameter and via-extension. New values for "comp" are registered by the IANA at http://www.iana.org/assignments/sip-parameters when new signalling compression schemes are published in standards track RFCs. The IANA Considerations section of the RFC MUST include the following information, which appears in the IANA registry along with the RFC number of the publication.

- o Name of the compression scheme.

- o Token value to be used. The token MAY be of any length, but SHOULD be no more than ten characters long.

The only entry in the registry for the time being is:

Compression scheme	Token	Reference
Signaling Compression	sigcomp	RFC 3486

12. Acknowledgements

Allison Mankin, Jonathan Rosenberg and Miguel Angel Garcia-Martin provided valuable comments on this memo.

13. 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] Price, R., Bormann, C., Christoffersson, J., Hannu, H., Liu, Z. and J. Rosenberg, "Signaling Compression (SigComp)", RFC 3320, January 2003.

[3] Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

14. Informative References

[4] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, October 2002.

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

[6] Schulzrinne, H., "DHCP option for SIP servers", Work in Progress.

15. Author’s Address

Gonzalo Camarillo
Ericsson
Advanced Signalling Research Lab.
FIN-02420 Jorvas
Finland

EMail: Gonzalo.Camarillo@ericsson.com

16. Full Copyright Statement

Acknowledgement

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