The path of SIP signalling messages

Emin Gabrielyan
2007-04-12
Switzernet

1. Introduction

SIP transactions are short exchanges of SIP messages (based on a request and replies). SIP dialog, representing a phone call, consists of several transactions. A phone call should contain a call setup transaction enveloping the following four messages INVITE / 100 (Trying) / 180 (Ringing) / 200 (OK), and the call disconnection transaction: BYE / 200 (OK). A SIP message from one endpoint UA to another endpoint UA may traverse several proxy servers. Within the scope of one transaction SIP requests and all replies follow the same signalling path. It is ensured by a stack of Via fields. When processing SIP request messages each intermediary proxy adds its own Via field on top of the stack of Via header fields. Each Via field contains the IP address of the proxy and a branch parameter identifying the transaction (i.e. all messages in the transaction have the same branch parameter). The replies are transmitted by the destination UA with the same stack of Via fields as they received by UA in the request message (recall that the stack of Via fields is summed up on the way from the originating UA to the destination UA). The top most Via field of reply message dictates the next hop where the message should be transmitted (i.e. the last proxy of the request before it arrived to UA). At each hop the proxy server removes its own Via field from the top and forwards the message according to the top most Via field of the reduced stack.

Usually, in simple cases, once the INVITE transaction is accomplished, endpoint User Agents learn each others direct addresses (from the Contact field transferred back and forth during the first transaction) and carry out the subsequent transactions directly bypassing the intermediary proxies.

However, the proxies can request to maintain the signalling path during the dialog. Record-Route header is designed for these purposes. A stack of Record-Route headers works similarly as the stack of Via headers but is a little bit more complex. Each proxy server (wishing to stay in the signalling path) adds on top of the Record-Route headers stack its IP address (or its domain name). Record-Route headers are usually

added by proxies only in the first request of the first transaction (i.e. in the INVITE message). With the first request of the first transaction the destination UA receives a stack of Record-Route headers which is similar to the stack of Via headers. The destination UA will copy the stack of Record-Route headers from the request to all responses replied within the scope of the transaction. Recall that replies are delivered back through a path dictated by the stack of Via headers. The Record-Route headers are therefore not required for determining the path of replies. The Record-Route headers will not be removed (like the Via header) as the reply traverses back to the originating UA. Therefore the originating and destination User Agents will be in possession of the same stack of Record-Route headers at the end of the SIP dialog's first transaction.

The endpoint User Agents form Route header stacks from the received Record-Route header stacks. The Route header stack of the destination UA is the copy of the Record-Route header stack (only the header name is changed from "Record-Route:" to "Route:"). The Route header stack of the originating UA is the copy of the Record-Route header stack but in the reversed order. The Route header stack added to request messages dictates the path of the message. Each time an intermediary proxy receives a message with its own IP in the top most Route header field, the proxy will remove its Route field from the stack and will forward the request to the next proxy listed in the stack. The routing according to the Route fields with removals of top most Route fields is called loose routing. Using the stack of Route fields (for maintaining the signalling path) during the phone conversation is the obligation of User Agents. They follow that during the entire duration of the phone conversation the request messages will be transmitted by both User Agents with the stacks of Route fields (associated to the current phone conversation). The proxies only provide the Record-Route headers once in the call setup transaction.

2. The configuration of the experiment

We show here an example of the application of Record-Route headers. We use OpenSER proxy server in our configuration. The latest version 1.2.0 is used for flexible transformations of pseudo-variables (i.e. for easier manipulation of text information obtained from SIP message headers). In our example we have two Budge Tone-100 SIP phones and one OpenSER proxy server. In a typical case after the call setup transaction, the subsequent transactions and individual messages communicate directly between the SIP phones. In our example we establish a signalling path which starts at one UA, passes through the proxy server, loops four times through the same proxy server, and finishes with the second UA. The signalling path between two SIP phones (looped in the proxy server) will be maintained for the duration of the entire call.

In OpenSER there is an official record_route() function for adding the Record-Route header of the current proxy. We cannot force looping with the record_route() function since it can add the header of the current proxy only once. Instead of using record_route() function, in our configuration file we insert the Record-Route headers

manually. Record-Route headers are inserted only for the INVITE message (i.e. within the first request of the first transaction):

```
  if(method=="INVITE") {
    xlog("L_INFO","$CbxAdding Record-Route headers$Cxx\n");
    #record_route();
    $var(rr-1)="Record-Route: <sip:192.168.1.15;lr=on;hop=first-
proxy>\r\n";
    $var(rr-2)="Record-Route: <sip:192.168.1.15;lr=on;hop=second-
proxy>\r\n";
    $var(rr-3)="Record-Route: <sip:192.168.1.15;lr=on;hop=third-
proxy>\r\n";
    $var(rr-4)="Record-Route: <sip:192.168.1.15;lr=on;hop=fourth-
proxy>\r\n";
    insert_hf(
      "$var(rr-4)$var(rr-3)$var(rr-2)$var(rr-1)",
      "From"
    );
  }
```

All four Record-Route headers contain the same IP address of our proxy server (192.168.1.15). The “hop” is an arbitrarily chosen name for a parameter indicating us the route number. These hop parameters will be ignored by loose routing but will be copied in Route headers by the endpoint User Agents so we can see them in subsequent requests. Note that Record-Route headers are inserted in the same order as they would appear in the header if the request passes sequentially through the first, second, third, and forth proxies (i.e. the first proxy will be in the bottom of the stack and the last proxy on the top).

For subsequent request messages we do not insert Record-Route header anymore, but we now check the presence of Route headers (added by endpoint User Agents). While Record-Route headers are for informing User Agents about the desired route, the Route headers are routing instructions for intermediary nodes. The result of our check is a couple of P-hint headers added to the message. P-hint headers are added only for our monitoring. This check (and P-hint headers) can be removed without any impact on the signalling path or on the functionality of the message routing:

```
  if($hdr(Route[1])!="")
  {
    xlog("L_INFO","$CyxRoute header fields need processing$Cxx\n");
    if(!search("^P-hint: "))
      $var(hint-id)=1;
    else
      $var(hint-id)=$(hdr(P-hint[-
1]){s.select,0,]}{s.select,1,[}{s.int}) + 1;
    $var(lr-cleanup)="[" + $var(hint-id) + "] lr-cleanup " +
$hdr(Route[0]);
    $var(rr-enforce)="[" + $var(hint-id) + "] rr-enforce " +
$hdr(Route[1]);
    append_hf("P-hint: $var(lr-cleanup)\r\nP-hint: $var(rr-
enforce)\r\n");
  }
  else
  {
    xlog("L_INFO","$CrxRoute header fields are expired$Cxx\n");
  }
```

If there are at least two Route headers (upon the reception of a request), then the message needs to be further forwarded. The top most Route[0] header corresponds to the current proxy and will be removed by loose_route(), the following Route[1]

header corresponds to the next hop. Using pseudo-variables and the transformation functions available in version 1.2.0 of OpenSER, we read the last P-hint header (if there is one), we analyze the number (added by ourselves at the previous processing), and we create two P-hint headers with an incremented number. This procedure demonstrates the transformations of pseudo-variables. It is needed only for better readability of P-hint headers. In one P-hint headers we inform which route is cleaned (Route[0]) and in the second P-hint header we inform which route is enforced at the next hop (Route[1]). Then the loose_route() is called for real cleaning and forwarding:

```
  if(loose_route()) {
    xlog("L_INFO","$CbxLoose Route$Cxx\n");
    route(1);
  }
```

The configuration file logs on the screen the contents of all message text buffers upon their arrival. The P-hint headers added by proxy will be seen upon the reception of the message at the next hop.

Our Budge Tone-100 SIP phones have static IP address. One SIP phone has address 192.168.1.10 and a phone number 308, and the second one has address 192.168.1.11 and a phone number 309.

3. The SIP messages

Below is the sequence of SIP signalling messages passing through our proxy server (at 192.168.1.15) when we make a call from UA 308 (at 192.168.1.10) to UA 309 (at 192.168.1.11) and transmit two DTMF signals during the call. The phone call is hung-up by the callee party. The messages are listed in the order of their arrival. The message bodies and some headers are skipped. The full printout is available [htm], [txt], [doc].

```
[  Method INVITE from 192.168.1.10:5060  ]
INVITE sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK717a94a8462afd78
From: <sip:308@192.168.1.15>;tag=6f8fc7ff539f6c5a
To: <sip:309@192.168.1.15>
Contact: <sip:308@192.168.1.10>
{the rest of headers and the SDP body are skipped}
[  End of Request  ]
```

We receive the invite, add four Record-Route headers, and relay the message.

```
[  Reply 100 (Trying) from 192.168.1.11:5060 concerning INVITE  ]
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK717a94a8462afd78
From: <sip:308@192.168.1.15>;tag=6f8fc7ff539f6c5a
To: <sip:309@192.168.1.15>
{skipped}
[  End of Reply  ]
```

```
[  Reply 180 (Ringing) from 192.168.1.11:5060 concerning INVITE  ]
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK717a94a8462afd78
Record-Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
Record-Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Record-Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Record-Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
From: <sip:308@192.168.1.15>;tag=6f8fc7ff539f6c5a
To: <sip:309@192.168.1.15>;tag=c12e534f3f82359f
{skipped}
[  End of Reply  ]
```

The 180 (Ringing) reply from UA 309 contains the copy of the Record-Route headers stack. This copy is intended for the UA 308 which originated the call. The calling UA 308 will be unaware of the requested signalling path (asked by intermediary proxies) until such message is received.

```
[  Reply 200 (OK) from 192.168.1.11:5060 concerning INVITE  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK717a94a8462afd78
Record-Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
Record-Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Record-Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Record-Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
From: <sip:308@192.168.1.15>;tag=6f8fc7ff539f6c5a
To: <sip:309@192.168.1.15>;tag=c12e534f3f82359f
{skipped}
Contact: <sip:309@192.168.1.11>
{the remaining headers and the SDP body are skipped}
[  End of Reply  ]
```

The 200 (OK) reply also contains the copy of Record-Route headers intended for UA 308. UA 308 needs only one copy of Record-Route headers stack. This or the previous copy is redundant. Note that replies carry the Record-Route headers in the same order. It means that the originating UA 308 must take into account that it deals with a sequence of Record-Route headers as it was received at the other end.

Note that the reply messages do not need Record-Route headers to be delivered back. The replies within the scope of one transaction (namely the current one initiated by INVITE) contain a stack of Via messages (summed up within the request message while it travelled toward the UA). The Via headers are used to route the replies back (and not the Record-Route headers) via the arrival path of the request message.

At this point (as soon as the originating UA 308 receives its own copy of Record-Route headers) the exchange of the signalling path information between the endpoints is accomplished. Now both User Agents know that there are four intermediary proxies who wish to remain in the signalling path. Although the callee phone 309 learned the direct contact of the originating phone 308 from the Contact field of the INVITE request, and the calling phone 308 learned the direct contact of the callee phone from the Contact field of the 200 OK reply (see the message above), the User Agents give a priority to Record-Route instructions and will not use the available direct contact information.

```
[  Method ACK from 192.168.1.10:5060  ]
```

```
ACK sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK4f1baaf825d9c4d8
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
From: <sip:308@192.168.1.15>;tag=6f8fc7ff539f6c5a
To: <sip:309@192.168.1.15>;tag=c12e534f3f82359f
{the remaining headers and the SDP body are skipped}
[  End of Request  ]
```

The calling phone 308, sends an ACK. This ACK is out of scope of the INVITE transaction (see the different branch parameter in the Via header fields). First of all we see that the calling phone well took care of adding the stack of Route headers. We observe that the stack is inverted compared to the stack of Record-Route headers received by the callee phone 309 (and then retransmitted to our calling phone 308). The stack can be executed from the top. The fact that we see this message in our proxy already means that SIP phone 308 executed the first Route instruction and forwarded the request to 192.168.1.15. Note that the first ACK is received from 192.168.1.10.

We are going to see this ACK message yet another three times, because now our proxy will remove the top most Route header and will execute the next routing instruction (this procedure is carried out by loose_route()). Before executing the loose_route() function, our proxy adds some P-hint headers for information. The P-hint headers added here can be seen at the next arrival of the message in the proxy.

```
[  Method ACK from 192.168.1.15:5060  ]
ACK sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.2
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK4f1baaf825d9c4d8
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{skipped}
P-hint: [1] lr-cleanup <sip:192.168.1.15;lr=on;hop=first-proxy>
P-hint: [1] rr-enforce <sip:192.168.1.15;lr=on;hop=second-proxy>
[  End of Request  ]
```

In the second ACK we see the changes done during the first passage through the proxy. The loose_route() function removed the first Route header. Now the top most Route header represents the second-proxy. This Route is already executed in the previous hop (so we are currently within the second-proxy). The loose_route() of the second-proxy must remove the top most Route field before forwarding the message to the third-proxy. The processing at the previous hop also added a pair of P-hint headers informing us that first-proxy Route header is cleaned and that the message is forwarded to the second-proxy (now our current location). We now add the new P-hint headers and execute the loose_routing().

```
[  Method ACK from 192.168.1.15:5060  ]
ACK sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.2
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.2
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK4f1baaf825d9c4d8
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{skipped}
```

```
P-hint: [1] lr-cleanup <sip:192.168.1.15;lr=on;hop=first-proxy>
P-hint: [1] rr-enforce <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] lr-cleanup <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] rr-enforce <sip:192.168.1.15;lr=on;hop=third-proxy>
 [  End of Request  ]
```

We received the third ACK in so called third-proxy. We see the P-hint messages added by previous two proxies (recall that all these proxies, namely first-proxy, second-proxy, third-proxy, and fourth-proxy, represent the same computer). It remains last forwarding to the fourth-proxy.

```
[  Method ACK from 192.168.1.15:5060  ]
ACK sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.2
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.2
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKcaa9.a80a1492.2
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK4f1baaf825d9c4d8
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{skipped}
P-hint: [1] lr-cleanup <sip:192.168.1.15;lr=on;hop=first-proxy>
P-hint: [1] rr-enforce <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] lr-cleanup <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] rr-enforce <sip:192.168.1.15;lr=on;hop=third-proxy>
P-hint: [3] lr-cleanup <sip:192.168.1.15;lr=on;hop=third-proxy>
P-hint: [3] rr-enforce <sip:192.168.1.15;lr=on;hop=fourth-proxy>
[  End of Request  ]
```

Here is the last appearance of the ACK message. No other Route headers left for execution (the remaining one represents the current location and must be removed). The ACK will be now forwarded to the callee phone 309. We see the pair of P-hint messages [3] added at the previous hop.

Observe the Via headers. The stack of the Via headers represent the complete path traversed by the request. This Via headers could have been used for a reply (but for this case there will be no reply, since ACK does not have replies). Replies do not need Route headers. They will mimic the path of the arrived request thanks to Via headers.

The endpoint phones include the stack of Route headers not only in ACK message but for in requests transmitted within the scope of the current phone conversation.

The following INFO message is generated from 308 during the phone conversation. It informs the phone 309, that user of 308 pressed digit 5 (see Send DTMF settings of phones 308 and 309).

```
[  Method INFO from 192.168.1.10:5060  ]
INFO sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{skipped}
Content-Type: application/dtmf-relay
Content-Length: 23

Signal=5
Duration=1920{end}
[  End of Request  ]
```

```
[  Method INFO from 192.168.1.15:5060  ]
INFO sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.95739653.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{skipped}
P-hint: [1] lr-cleanup <sip:192.168.1.15;lr=on;hop=first-proxy>
P-hint: [1] rr-enforce <sip:192.168.1.15;lr=on;hop=second-proxy>
{body is skipped}
[  End of Request  ]


[  Method INFO from 192.168.1.15:5060  ]
INFO sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.a5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.95739653.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{skipped}
P-hint: [1] lr-cleanup <sip:192.168.1.15;lr=on;hop=first-proxy>
P-hint: [1] rr-enforce <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] lr-cleanup <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] rr-enforce <sip:192.168.1.15;lr=on;hop=third-proxy>
{body is skipped}
[  End of Request  ]


[  Method INFO from 192.168.1.15:5060  ]
INFO sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.b5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.a5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.95739653.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{skipped}
P-hint: [1] lr-cleanup <sip:192.168.1.15;lr=on;hop=first-proxy>
P-hint: [1] rr-enforce <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] lr-cleanup <sip:192.168.1.15;lr=on;hop=second-proxy>
P-hint: [2] rr-enforce <sip:192.168.1.15;lr=on;hop=third-proxy>
P-hint: [3] lr-cleanup <sip:192.168.1.15;lr=on;hop=third-proxy>
P-hint: [3] rr-enforce <sip:192.168.1.15;lr=on;hop=fourth-proxy>
{body is skipped}
[  End of Request  ]
```

The INFO messages sent from SIP phone 308 behave exactly as the ACK messages (also sent from phone 308). It undergoes the same modifications and follows the same path. In the initial stack of Route headers the first-proxy is on the top and the fourth-proxy is at the bottom. At the last appearance of INFO message, we see that Via fields stack contains the full information for backward transmission of responses. For both, ACK and INFO messages, note that the green bar shows that the message is arrived directly from phone 308 at 192.168.1.10 only when it appeared first time. The subsequent three messages arrive from 192.168.1.15, i.e. from the proxy itself.

Below is the reply which travels from phone 309 back to phone 308.

```
[  Reply 200 (OK) from 192.168.1.11:5060 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.c5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.b5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.a5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.95739653.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.b5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.a5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.95739653.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.a5739653.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.95739653.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK9aa9.95739653.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK8473675e920112eb
{skipped}
[  End of Reply  ]
```

We see the same reply four times. The first time proxy tells us (in blue bars) that it is received from 192.168.1.11 (i.e. from phone 308). For the following three passages of the reply, the proxy reports that the reply is received from itself (192.168.1.15). We see the content of the reply messages as they arrive in proxy. Before sending them out, proxy removes the top most Via header. This removal is seen at the next passage.

Below is an INFO message transferred from the callee phone 309 to the calling phone 308 (appeared four times) and its 200 (OK) reply (appeared four times). An attention should be paid on the fact that the phone 309 generated Route headers stack where the fourth-proxy is at the top and the first-proxy is at the bottom. The Route header stacks of two User Agents are reversed with respect to each other. In the below printout we do not show the P-hint headers added by the proxy server at each passage.

```
[  Method INFO from 192.168.1.11:5060  ]
INFO sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{the remaining headers and the DTMF-relay body are skipped}
[  End of Request  ]
```

```
[  Method INFO from 192.168.1.15:5060  ]
INFO sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.c5a8c967.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{the remaining headers including the P-hint headers, and the DTMF-
relay body are skipped}
 [  End of Request  ]


[  Method INFO from 192.168.1.15:5060  ]
INFO sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.d5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.c5a8c967.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{the remaining headers including the P-hint headers, and the DTMF-
relay body are skipped}
 [  End of Request  ]


[  Method INFO from 192.168.1.15:5060  ]
INFO sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.e5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.d5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.c5a8c967.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{the remaining headers including the P-hint headers, and the DTMF-
relay body are skipped}
 [  End of Request  ]


[  Reply 200 (OK) from 192.168.1.10:5060 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.f5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.e5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.d5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.c5a8c967.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.e5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.d5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.c5a8c967.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning INFO  ]
SIP/2.0 200 OK
```

```
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.d5a8c967.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.c5a8c967.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
{skipped}
[  End of Reply  ]
 0(9666)

[  Reply 200 (OK) from 192.168.1.15:5060 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK42e1.c5a8c967.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK33ba31c02d94bb8a
{skipped}
[  End of Reply  ]
```

The same scenario is repeated for the BYE message transferred from 192.168.1.11 (phone 309) and for its 200 (OK) reply from 192.168.1.10 (phone 308):

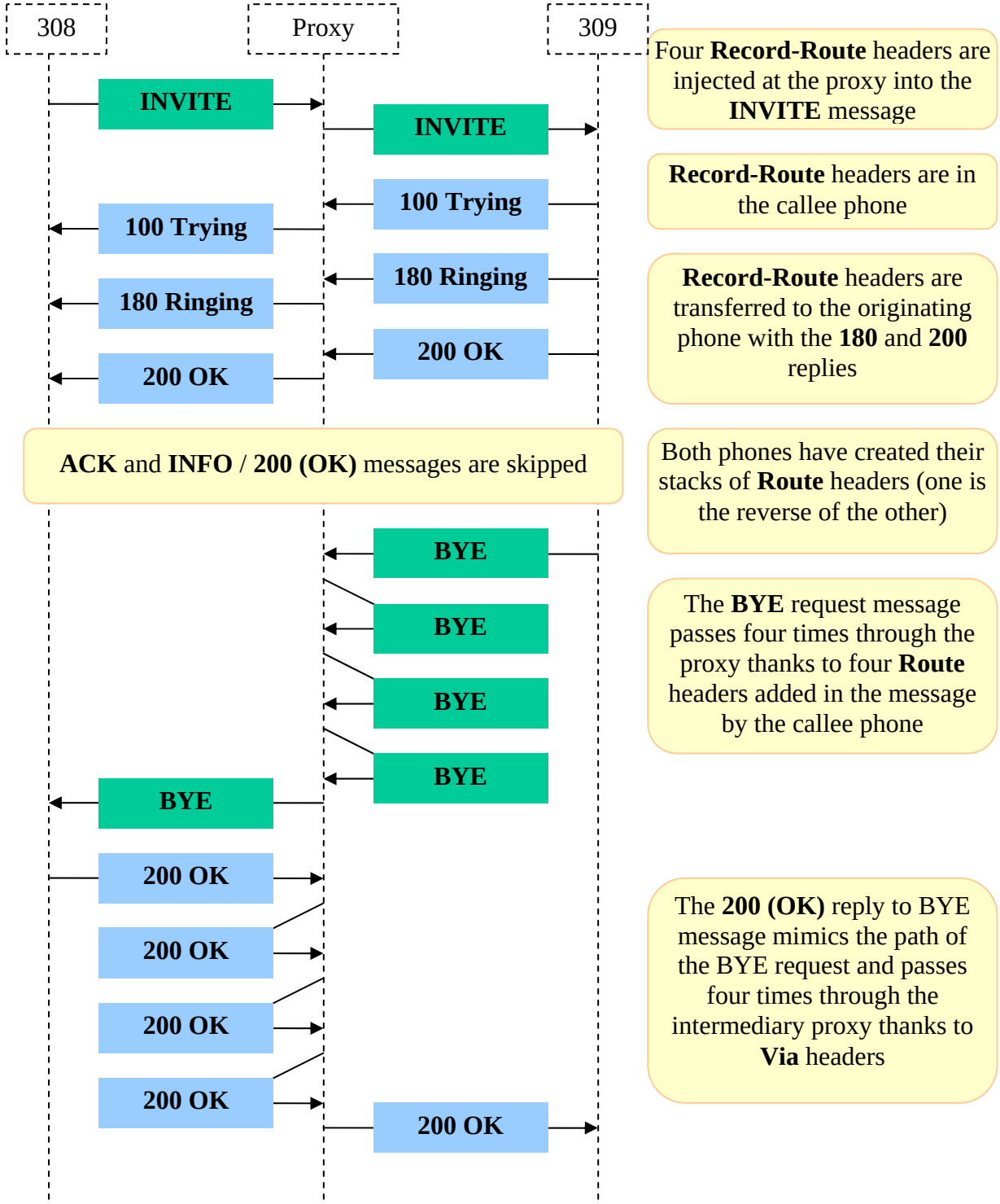
```
[  Method BYE from 192.168.1.11:5060  ]
BYE sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
Route: <sip:192.168.1.15;lr=on;hop=fourth-proxy>
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{skipped}
[  End of Request  ]


[  Method BYE from 192.168.1.15:5060  ]
BYE sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.65256fe6.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
Route: <sip:192.168.1.15;lr=on;hop=third-proxy>
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{skipped}
[  End of Request  ]


[  Method BYE from 192.168.1.15:5060  ]
BYE sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.75256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.65256fe6.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
Route: <sip:192.168.1.15;lr=on;hop=second-proxy>
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{skipped}
[  End of Request  ]


[  Method BYE from 192.168.1.15:5060  ]
BYE sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.85256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.75256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.65256fe6.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
Route: <sip:192.168.1.15;lr=on;hop=first-proxy>
{skipped}
[  End of Request  ]
```

```
[  Reply 200 (OK) from 192.168.1.10:5060 concerning BYE  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.95256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.85256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.75256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.65256fe6.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning BYE  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.85256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.75256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.65256fe6.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning BYE  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.75256fe6.0
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.65256fe6.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
{skipped}
[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.15:5060 concerning BYE  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK12e1.65256fe6.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bK17c2ac598c8921a9
{skipped}
[  End of Reply  ]
```

The full printout is available [htm], [txt], [doc].

4. The diagram of the SIP dialog

A part of the discussed SIP dialog can be summarized by the diagram below. The figure does not show the ACK message and the INFO / 200 (OK) exchanges carried out during the phone call. We show only the INVITE transaction and the call disconnection messages.



5. Other experiments with SIP

Examining the STUN settings of a SIP phone

Creating and sending INVITE and CANCEL SIP text messages

Direct calls between two SIP phones without passing through a SIP proxy

SIP messages, transactions, and dialogs (understanding SIP exchanges by experimenting)

The path of SIP signalling messages (understanding Via, Record-Route, and Route headers)

This document [htm], [pdf], [doc]

6. Related links

OpenSER pseudo-variables (http://openser.org/dokuwiki/doku.php/pseudovariables:1.2.x)

OpenSER transformations of pseudo-variables (http://openser.org/dokuwiki/doku.php/transformations:1.2.x)

OpenSER rr Module (http://www.openser.org/docs/modules/1.2.x/rr.html)

loose_route()

record_route()

OpenSER Packages (http://openser.org/pub/openser/1.2.0/packages/deb-stable/)

openser_1.2.0-0_i386.deb