[English], [Français]

Understanding SIP exchanges by experimentation

Emin Gabrielyan
2007-04-10
Switzernet Sàrl

We analyze a few simple scenarios of SIP message exchanges for a call setup between two SIP phones. We use an SIP Express Router (SER) for proxying and monitoring the SIP messages. We show and analyze INVITE, CANCEL, ACK, and BYE requests and their responses. We discuss SIP transactions and SIP dialogs. We experiment with the case in which the intermediary proxy server participates only in call setup and with the case in which the intermediary proxy modifies the path of SIP signalling messages in order to participate in the signalling of the entire SIP session. We do not discuss the media flow. In our experiments the RTP media streaming packets are transmitted between two SIP phones directly, without intermediaries.

1. The test-bed configuration

In our experiments we have two SIP phones (Grandstream Budge Tone-100) [308, bmp, htm], [309, bmp, htm], and one SIP proxy server running OpenSER, a spin-off of SIP Express Router (SER). The SIP phones are located at IP addresses 192.168.1.10 and 192.168.1.11 respectively. The proxy server is located at IP address 192.168.1.15.

2. SIP proxy handles call setup only and SIP phones transmit all other signalling directly

2.1. Overview of the scenario

The scenario where the SIP proxy handles only the call setup signalling and the rest of the signalling is carried out directly between the SIP phones (bypassing the SIP proxy) is the first example discussed in RFC3261 [txt], [htm]. See the diagram below [rfc3261, p.10-11].

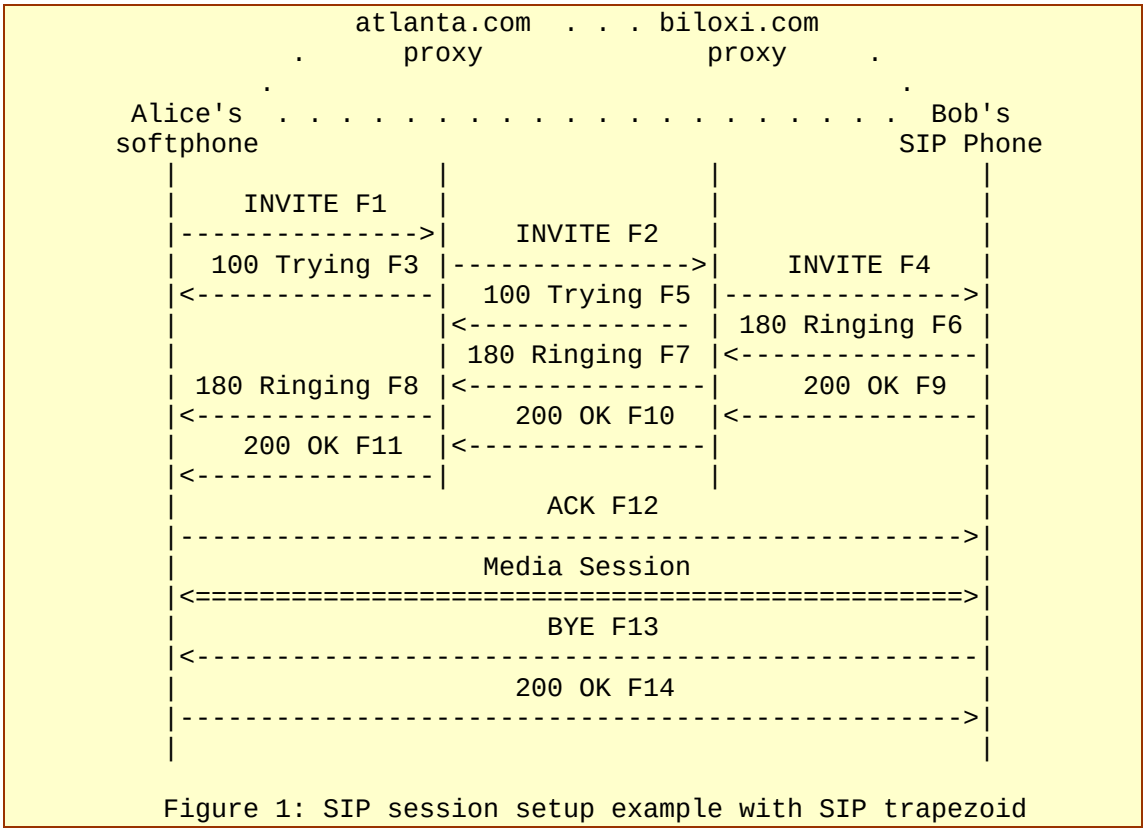


Figure 1: SIP session setup example with SIP trapezoid

In the example of RFC3261, there are two intermediary proxies. The SIP signalling messages for call setup are exchanged between two SIP phones via the two intermediary proxies. The call setup signalling messages are INVITE, 100 (Trying), 180 (Ringing), and 200 (OK). The exchange of these four messages constitutes one **SIP transaction**. A transaction is a short sequence of SIP message exchanges. In the scope of one SIP transaction the signalling messages follow the same path, meaning that if the initial INVITE message of this example was communicated through two intermediary proxies, then all other messages of this transaction will also pass through the same proxies [rfc3261, p.13-14]. Within the scope of one transaction, the path is maintained by a stack of Via fields added to the headers of the SIP messages [rfc3261, p.12].

Further, in the same example, we see that once the call setup transaction, initiated by an INVITE message, is accomplished, the rest of the SIP signalling messages are

exchanged directly between the two SIP phones. The ACK individual message (denoting the beginning of the media session) is transmitted directly from the calling phone to the called phone. The call disconnection transaction, consisting of the BYE request, is also carried out directly, without involving the two proxy servers.

In this document we analyze SIP transactions by experimenting with a simpler model comprising only one intermediary proxy.

The control of the path of media packets is not discussed in this document. In all examples that will be discussed, the media streaming is carried out directly between two SIP phones regardless of whether all SIP signalling messages are passing through a proxy server or not.

2.2. What do the SIP messages look like?

SIP messages are text-encoded lines. They can be printed on paper. An SIP message is usually carried by a UDP packet. We documented a short experiment with text messages and a UDP packet sender [htm]. One can repeat this experiment and show that the SIP messages are human readable, can be stored into text files and edited. SIP is based on an HTTP-like request/response transaction model. Each transaction consists of a request and at least one response. The example of the INVITE message corresponding to the first figure of RFC3261 is shown below [rfc3261, p.10-11].

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

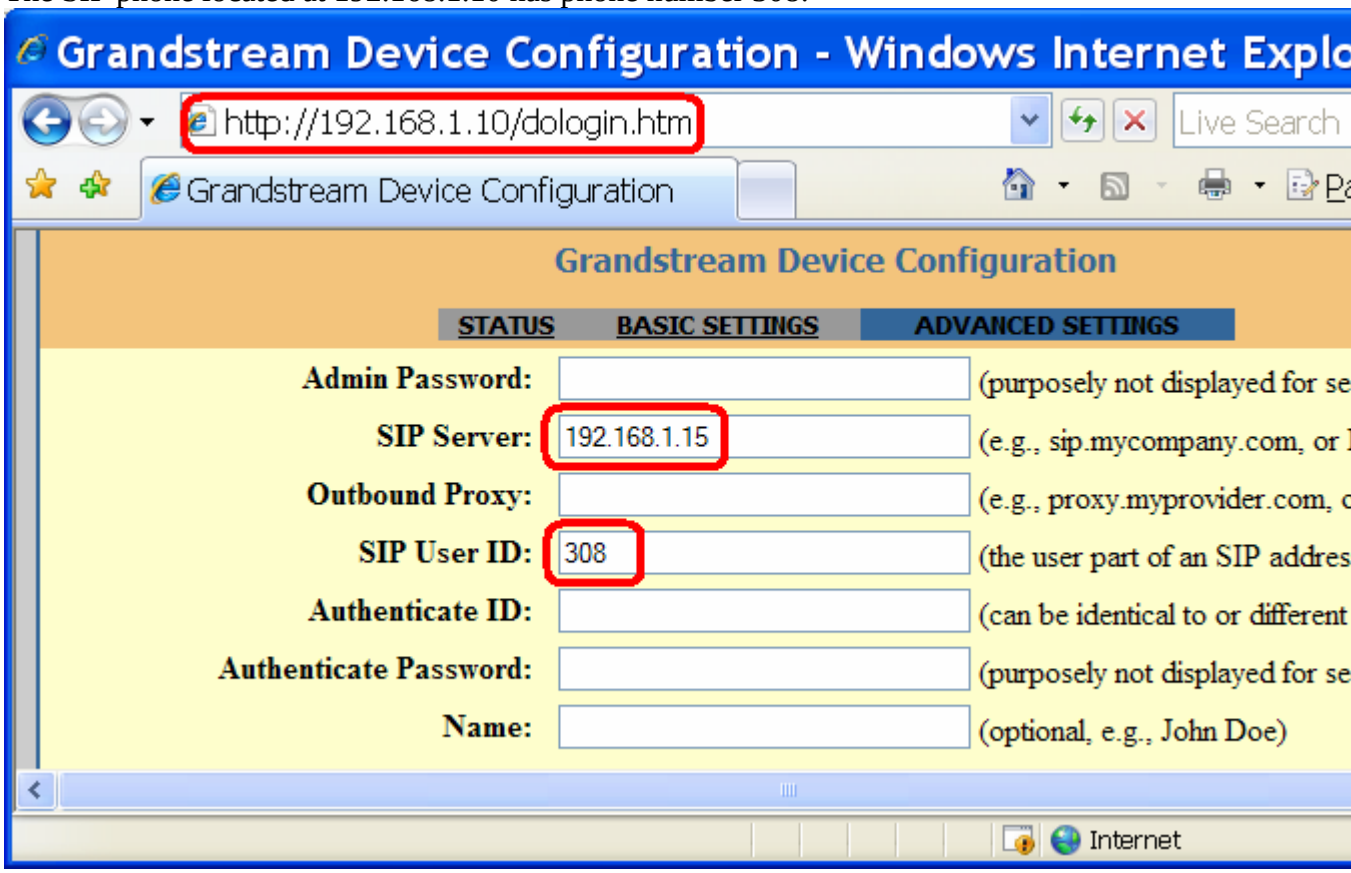
The first line of the message contains the method name (INVITE). The lines that follow are a list of header fields [rfc3261, p.10-11]. Many SIP messages are limited to the first line and the following header fields. In the example the message also contains a message body (which is not shown). The body is carried by an SIP message in a way that is analogous to a document attachment being carried by an email message, or a web page being carried in an HTTP message [rfc3261, p.12-13].

The body of an INVITE message contains the details of the session, such as the codec information or the media streaming IP and port information of the SIP phones. These details are not described using SIP format. The body of the INVITE message uses another protocol format, namely the Session Description Protocol (SDP) described in RFC 2327 [txt], [htm].

2.3. Configuration of SIP phones

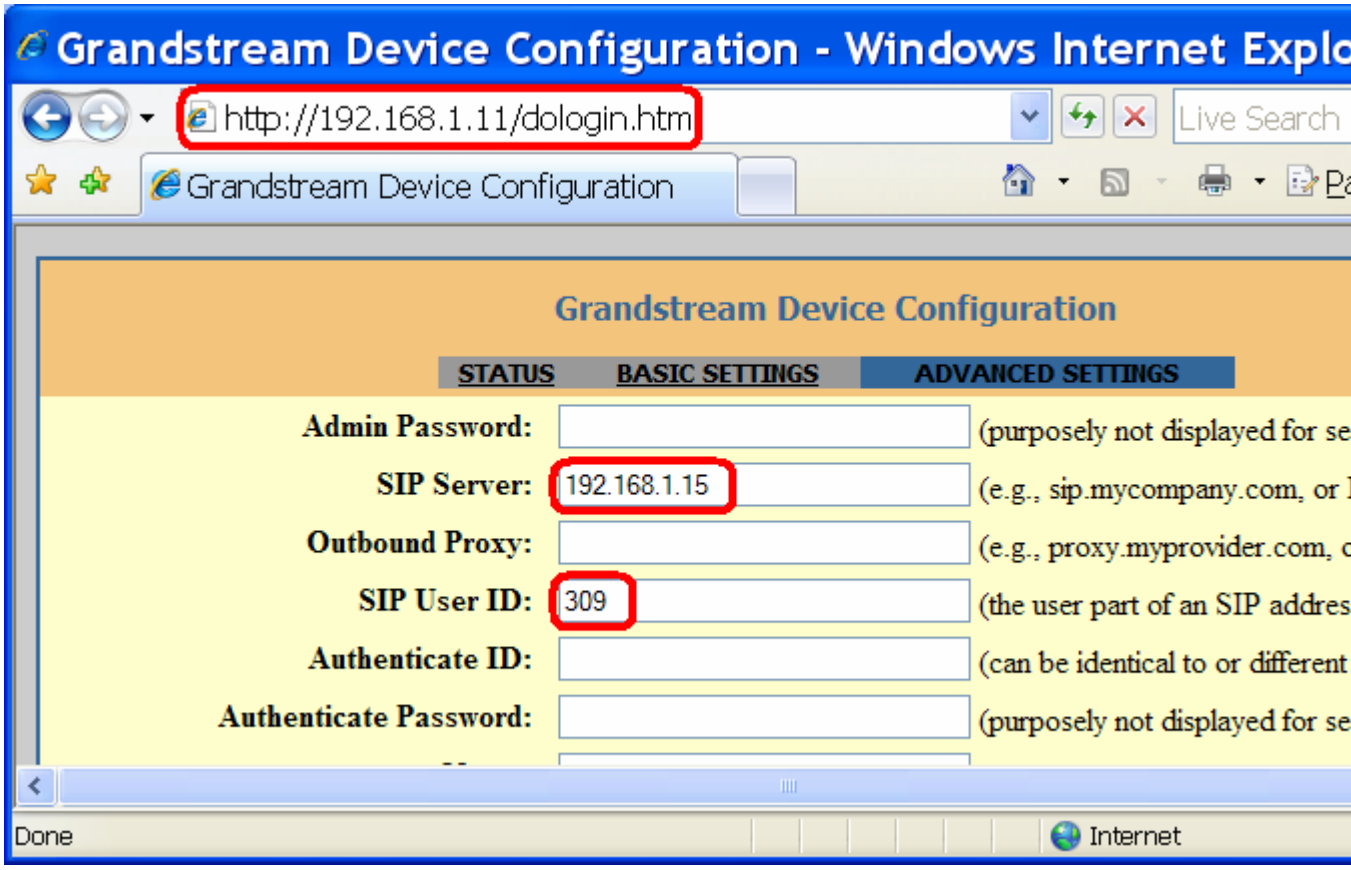
In our experiment we use two Grandstream Budge Tone-100 SIP phones. They are configured with static IP addresses: 192.168.1.10 and 192.168.1.11. Static IP addresses are used only for clarity of explanations. The experiment can also be successfully accomplished with dynamic IP addresses distributed by a DHCP server. Both SIP phones are configured to use IP 192.168.1.15 as their SIP server.

The SIP phone located at 192.168.1.10 has phone number 308:



[htm]

The SIP phone located at 192.168.1.11 has phone number 309:



[htm]

2.4. Configuration of the SIP proxy server

We are running an OpenSER proxy at the IP address 192.168.1.15. OpenSER is a spin-off of SIP Express Router (SER). SER proxy server configuration relies on a single text file. The simplest form of SER configuration provides registrations without an authentication and permits calls between SIP phones. Such a configuration file is only 100 to 200 lines long [ser.cfg], [openser.cfg]. Configuration uses a C-like script. The configuration file basically defines the operations to be carried out upon reception of individual SIP requests. Replies can also be handled with stateful processing (discussed later in sections 2.7 and 3). The scripts of SER and OpenSER are nearly compatible. With minimal modifications a script of SER can be used in OpenSER and vice-versa.

We are running the OpenSER server from the terminal in debug mode. Upon the reception of an SIP request we print the entire content of the message and we wait for all replies to this message [cfg]. Processing of the SIP message is carried out according to the minimum required script. The details of the configuration file are beyond the scope of this document. The script language is discussed in "SER Getting Started".

The entire message buffer is printed using the following command:

```
  xlog("L_INFO","\n\n$Cbg[  Method $rm from $si  ]$Cxx\n$mb$Cbg[  End
of Request  ]$Cxx\n");
```

Here $mb specifies the entire SIP message buffer including the attached message body. Just the attached body of a request or reply can be printed with the $rb pseudo variable. The xlog logging function is defined in the xlog.so module [xlog ser], [xlog openser], [readme] (the difference between the xlog function of SER and that of OpenSER is that one uses the '%' sign for pseudo variables and the other uses the '$' sign). The pseudo variable $si specifies the IP source address of the sender of the message and the pseudo variable $rm specifies the requested method.

Replies to SIP requests can be handled within the SER script. In the main routing logic we must specify the function for handling the replies:

```
  t_on_reply("1");
```

We must also provide the body of the function for handling the replies:

```
onreply_route[1]
{
  xlog("L_INFO","\n\n$Cbc[  Reply $rs ($rr) from $si concerning $rm
]$Cxx\n$mb$Cbc[  End of Reply  ]$Cxx\n");
}
```

The replies will be handled only if the requests were previously relayed with a stateful function (i.e. with the t_relay() function instead of forward() function):

```
route[1]
{
  if(!t_relay())
    sl_reply_error();
  xlog("L_INFO","$CbxMessage is relayed; now exiting$Cxx\n");
  exit;
}
```

In the body of onreply_route[1] we only print the content of the reply message ($mb), the reply status ($rs), the reply reason ($rr), the IP address of the sender ($si), and the initial request method with which the present reply is associated ($rm). This association is made due to the stateful processing of transactions by SER server. The SIP replies are correlated with initial requests thanks to identification parameters provided within Via header fields of SIP messages [rfc3261, p.13].

2.5. Call setup messages

Here we discuss an example where we make a call from SIP phone 308 (located at IP address 192.168.1.10) to SIP phone 309 (located at IP address 192.168.1.11). In this example the destination SIP phone (309) answers the call, and the call is established. The OpenSER server, which is the proxy for both SIP phones, logs the received SIP messages on the screen (according to the discussed configuration file [cfg]). Below is the printout of the server corresponding to our call from 308 to 309. The message bounded by green horizontal bars represents the INVITE requests; messages bounded with blue bars represent the replies. The INVITE request is sent from the calling phone located at 192.168.1.10. Then we see 100 (Trying) sent back from the called phone 192.168.1.11. This reply is immediately followed by 180 (Ringing), also sent by the called phone. When the originating phone receives 180 (Ringing), it generates the ring-back tone for the user. The answer to the call is confirmed by a 200 (OK) reply sent by the called phone. These three messages constitute one **SIP transaction** which groups one INVITE request with its three responses 100 (Trying) [rfc3261, p.13], 180 (Ringing), and 200 (OK).

```
[  Method INVITE from 192.168.1.10  ]
INVITE sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK533110eb792e9593
From: <sip:308@192.168.1.15>;tag=839d16b92cebf0ae
To: <sip:309@192.168.1.15>
Contact: <sip:308@192.168.1.10>
Supported: replaces
Call-ID: 278956deb55db668@192.168.1.10
CSeq: 23290 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/sdp
Content-Length: 225

v=0
o=308 8000 8000 IN IP4 192.168.1.10
s=SIP Call
c=IN IP4 192.168.1.10
t=0 0
m=audio 5004 RTP/AVP 4 18 97
a=sendrecv
a=rtpmap:4 G723/8000
a=rtpmap:18 G729/8000
a=rtpmap:97 iLBC/8000
a=fmtp:97 mode=20
a=ptime:60
[  End of Request  ]


[  Reply 100 (Trying) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK4599.12d426e3.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK533110eb792e9593
From: <sip:308@192.168.1.15>;tag=839d16b92cebf0ae
To: <sip:309@192.168.1.15>
Call-ID: 278956deb55db668@192.168.1.10
CSeq: 23290 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Content-Length: 0

[  End of Reply  ]


[  Reply 180 (Ringing) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK4599.12d426e3.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK533110eb792e9593
From: <sip:308@192.168.1.15>;tag=839d16b92cebf0ae
To: <sip:309@192.168.1.15>;tag=e821cb882a12d201
Call-ID: 278956deb55db668@192.168.1.10
CSeq: 23290 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Content-Length: 0

[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 200 OK
```

```
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK4599.12d426e3.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK533110eb792e9593
From: <sip:308@192.168.1.15>;tag=839d16b92cebf0ae
To: <sip:309@192.168.1.15>;tag=e821cb882a12d201
Call-ID: 278956deb55db668@192.168.1.10
CSeq: 23290 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Contact: <sip:309@192.168.1.11>
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/sdp
Supported: replaces
Content-Length: 154

v=0
o=309 8000 8000 IN IP4 192.168.1.11
s=SIP Call
c=IN IP4 192.168.1.11
t=0 0
m=audio 5004 RTP/AVP 4
a=sendrecv
a=rtpmap:4 G723/8000
a=ptime:60
[  End of Reply  ]
```

[htm], [doc], [txt]

All messages of the transaction are communicated via the proxy server and as a result we can see them all. Within the scope of one SIP transaction the messages always follow the same path. **Via** header filed in SIP messages is responsible for identification of transactions and for maintaining the path of messages belonging to the one transaction. The transaction is identified by a **branch parameter** incorporated in the Via field [rfc3261, p.12]. The responses are correlated with requests thanks to the branch parameter [rfc3261, p.13]. In the above printout the branch parameters are marked bold. We see that all branch parameters are the same for all three messages.

The path of messages is maintained thanks to the stack of Via fields added by intermediary proxies to the header of request messages as they traverse the chain of proxies (we have only one proxy in our example). The SIP phone makes sure that the SIP responses to an SIP request contain the same stack of Via fields. When the responses are transferred back, each intermediary proxy removes its address from the top of the stack and uses the next Via header field to determine where to send the response. As a result each proxy that sees a request will also see all responses to that request [rfc3261, p.13-14].

The transaction shown in the above printout initiates a phone call. Several other transactions can take place during and at the end of the phone conversation. Although (as we discussed) within the scope of one transaction all SIP messages always follow the same path, other SIP transactions related to the same phone call do not necessarily follow the path of the call setup transaction. Instead of passing via the intermediary proxies the successive transactions are often carried out directly between the SIP phones (see the first example of RFC3261 [rfc3261, p.15]). This occurs because during the first INVITE/200 (OK) transaction the endpoints learn each other's addresses from the Contact header fields [rfc3261, p.12].

The printout of this section also corresponds to a scenario similar to the first example of RFC3261 [rfc3261, p.10-11]. As soon as the 100 (OK) for the initial INVITE is transmitted (i.e. the call setup transaction is accomplished), no more signalling messages will pass through the proxy [rfc3261, p.15]. All successive SIP transactions are carried out directly between two SIP phones, bypassing the proxy server. Therefore we do not see (and we cannot log) the other messages of the call (which are, at minimum the ACK, BYE, and the 200 (OK) reply to the BYE).

In the INVITE request and the 200 (OK) reply of the printout we also see the bodies of attached SDP messages [rfc3261, p.12-13]. By the exchange of these two SDP messages the SIP phones negotiate the parameters of the future media session. In particular, the SIP phones inform each other about the IP addresses and ports for media streaming (i.e. 192.168.1.10:5004 and 192.168.1.11:5004), and they negotiate media codec G723.

2.6. Cancelled call setup

The calling party can hang up before the called party answers the call. This adds additional message exchanges into the transaction of the call setup. The printout below corresponds to such scenario. First, phone 308 (at 192.168.1.10) dials the number 309, and then it hangs up without waiting for the call to be answered by 309.

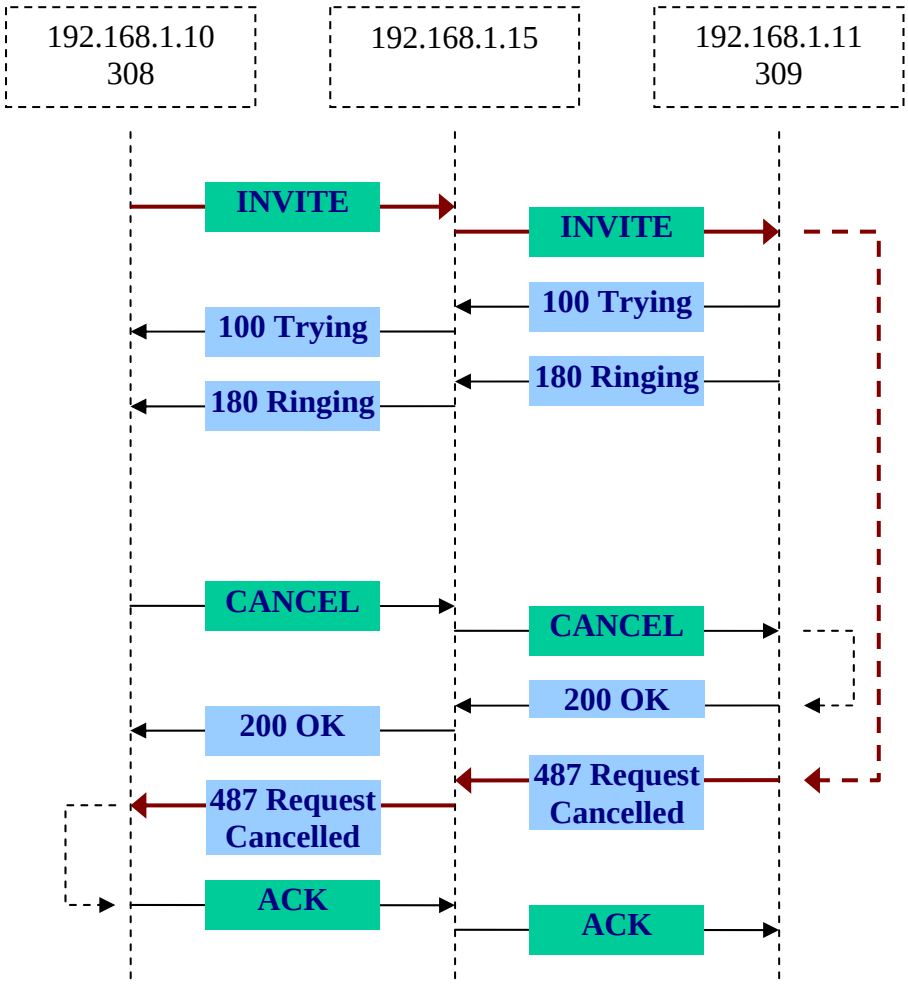
```
[  Method INVITE from 192.168.1.10  ]
INVITE sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKdfda7b9079412bd5
From: <sip:308@192.168.1.15>;tag=8b2723dc35649705
To: <sip:309@192.168.1.15>
Contact: <sip:308@192.168.1.10>
Supported: replaces
Call-ID: 64f567bcc5d8f80c@192.168.1.10
CSeq: 14107 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/sdp
Content-Length: 225

v=0
o=308 8000 8000 IN IP4 192.168.1.10
s=SIP Call
c=IN IP4 192.168.1.10
t=0 0
m=audio 5004 RTP/AVP 4 18 97
a=sendrecv
a=rtpmap:4 G723/8000
a=rtpmap:18 G729/8000
a=rtpmap:97 iLBC/8000
a=fmtp:97 mode=20
a=ptime:60
[  End of Request  ]


[  Reply 100 (Trying) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 100 Trying
```

```
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK5647.03eb25b7.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKdfda7b9079412bd5
From: <sip:308@192.168.1.15>;tag=8b2723dc35649705
To: <sip:309@192.168.1.15>
Call-ID: 64f567bcc5d8f80c@192.168.1.10
CSeq: 14107 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Content-Length: 0

[  End of Reply  ]


[  Reply 180 (Ringing) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK5647.03eb25b7.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKdfda7b9079412bd5
From: <sip:308@192.168.1.15>;tag=8b2723dc35649705
To: <sip:309@192.168.1.15>;tag=76adf65f887d5f3f
Call-ID: 64f567bcc5d8f80c@192.168.1.10
CSeq: 14107 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Content-Length: 0

[  End of Reply  ]


[  Method CANCEL from 192.168.1.10  ]
CANCEL sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKdfda7b9079412bd5
From: <sip:308@192.168.1.15>;tag=8b2723dc35649705
To: <sip:309@192.168.1.15>
Supported: replaces
Call-ID: 64f567bcc5d8f80c@192.168.1.10
CSeq: 14107 CANCEL
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Length: 0

[  End of Request  ]


[  Reply 200 (OK) from 192.168.1.11 concerning CANCEL  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK5647.03eb25b7.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKdfda7b9079412bd5
From: <sip:308@192.168.1.15>;tag=8b2723dc35649705
To: <sip:309@192.168.1.15>;tag=76adf65f887d5f3f
Call-ID: 64f567bcc5d8f80c@192.168.1.10
CSeq: 14107 CANCEL
User-Agent: Grandstream BT110 1.0.8.33
Contact: <sip:309@192.168.1.11>
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Supported: replaces
Content-Length: 0

[  End of Reply  ]
 0(22535)

[  Reply 487 (Request Cancelled) from 192.168.1.11 concerning INVITE
]
```

```
SIP/2.0 487 Request Cancelled
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK5647.03eb25b7.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKdfda7b9079412bd5
From: <sip:308@192.168.1.15>;tag=8b2723dc35649705
To: <sip:309@192.168.1.15>;tag=76adf65f887d5f3f
Call-ID: 64f567bcc5d8f80c@192.168.1.10
CSeq: 14107 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Content-Length: 0

[  End of Reply  ]
 0(22535)

[  Method ACK from 192.168.1.10  ]
ACK sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKdfda7b9079412bd5
From: <sip:308@192.168.1.15>;tag=8b2723dc35649705
To: <sip:309@192.168.1.15>;tag=76adf65f887d5f3f
Contact: <sip:308@192.168.1.10>
Call-ID: 64f567bcc5d8f80c@192.168.1.10
CSeq: 14107 ACK
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Length: 0

[  End of Request  ]
```

[htm], [doc], [txt]

All seven messages belong to one transaction. This transaction attempts to setup a call but then cancels this attempt. The message exchange diagram is shown below.



The message headers shown in the printout contain two branch parameters. One branch parameter (branch=z9hG4bKdfda7b9079412bd5) is in the Via field added by the calling SIP phone 192.168.1.10. The second branch parameter (branch=z9hG4bK5647.03eb25b7.0) is in the Via field added by the SIP proxy. The transaction can be identified by either of these two branch parameters. Note that in this scenario, when we cancel the call before it is answered, the ACK message is transmitted in the scope of the same transaction (it has the same branch parameter), while in the case where the call is answered (see the printout of the previous section 2.5), the ACK message is sent with a different branch parameter and therefore does not belong to the transaction of the INVITE request.

When the SIP proxy 192.168.1.15 receives the INVITE request and sees branch=z9hG4bKdfda7b9079412bd5 the first time (the branch parameter having been added by the originating SIP phone located at 192.168.1.10), the proxy creates an internal transaction number z9hG4bK5647.03eb25b7.0 to be associated with all subsequently expected messages of this transaction. Both the internal transaction number created by the proxy and the branch parameter of the originating SIP phone are synonymous identifying the same transaction. When the SIP proxy receives other SIP requests from 192.168.1.10 with the same branch parameter (provided by the originating phone), the proxy can identify the on-going transaction and will associates the already created internal transaction number with such messages. Further, the proxy includes its own internal transaction number in the Via header field added to the message.

The printout of this section shows the SIP message contents as they arrived, i.e. the Via header field of the proxy is not yet included in the requests. The SIP responses are transferred back from the 192.168.1.11 SIP phone to the proxy 192.168.1.15 and they contain the Via headers of the originating phone and of the proxy itself (inserted into the SIP request at the time when it was travelling toward the called phone). The stateful proxy associates the response message with the on-going transaction using either its own branch parameter or that of the calling phone. For example, it is possible to access the request's method ($rm) pseudo variable while handling the response messages thanks to the proper association of the response with the on-going transaction. Before relaying the response back to the originating phone, the proxy server removes its own Via header field from the response [rfc3261, p.13-14]. The printout shows the response messages upon their arrival to the proxy, so the Via header field of the proxy is not yet removed.

The originating phone learns the direct contact information of the destination SIP phone from the Contact header of the 200 (OK) reply to the CANCEL message (marked bold in the printout). However, the ACK message is not sent directly. The ACK message contains the same branch number, and is therefore in the scope of the same transaction. Thus the ACK follows the same path, passing via the proxy server.

Since the call attempt is cancelled, in the presented transaction only the INVITE message has an SDP attachment. There is no need for an SDP attachment in the 487 (Request Cancelled) reply.

2.7. Stateless configuration of SER server

When using the stateless functions of SER server, the proxy simply forwards the messages without keeping the transaction information in memory. The forward() function is used for stateless processing and t_relay() for stateful:

```
route[1]
{
  if(!forward())
    sl_reply_error();
  xlog("L_INFO","$CbxMessage is relayed; now exiting$Cxx\n");
  exit;
}
```

With stateless processing the messages are handled with no context and the replies cannot be associated with the requests [SER Getting Started, p.9]. It means that before relaying an SIP request the SIP proxy does not store in memory the branch parameters of this SIP request in order to handle SIP replies with the same branch parameters.

With stateless message processing [cfg] during the cancelled call setup attempt, only INVITE and CANCEL messages will be passed and logged by the SER script logic (see the printout below). All other messages associated with the transaction will not be handled:

```
 [  Method INVITE from 192.168.1.10  ]
INVITE sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKef1e1f8da5c0298d
From: <sip:308@192.168.1.15>;tag=0434481ac70e589b
```

```
To: <sip:309@192.168.1.15>
Contact: <sip:308@192.168.1.10>
Supported: replaces
Call-ID: f146448b7830985b@192.168.1.10
CSeq: 24573 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/sdp
Content-Length: 225

v=0
o=308 8000 8000 IN IP4 192.168.1.10
s=SIP Call
c=IN IP4 192.168.1.10
t=0 0
m=audio 5004 RTP/AVP 4 18 97
a=sendrecv
a=rtpmap:4 G723/8000
a=rtpmap:18 G729/8000
a=rtpmap:97 iLBC/8000
a=fmtp:97 mode=20
a=ptime:60
[  End of Request  ]


[  Method CANCEL from 192.168.1.10  ]
CANCEL sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKef1e1f8da5c0298d
From: <sip:308@192.168.1.15>;tag=0434481ac70e589b
To: <sip:309@192.168.1.15>
Supported: replaces
Call-ID: f146448b7830985b@192.168.1.10
CSeq: 24573 CANCEL
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Length: 0

[  End of Request  ]
```

[htm], [doc], [txt]

3. SIP proxy handles the call setup and all subsequent signalling messages

In section 2.5 we presented call establishment between two SIP phones. In the example of section 2.5, the SIP proxy participates in the call setup but does not see any SIP messages beyond the call setup transaction. SIP headers provide to intermediary proxies the possibility to remain in the signalling path after call setup. The signalling messaging between the endpoints can be heard for the entire duration of an SIP session. If the proxy server wishes to remain in the SIP messaging path after the call setup transaction (i.e. the INVITE with all associated messages), it should add to the INVITE a required routing header field known as Record-Route that

contains the hostname or IP address of the proxy. This information is received by the called SIP phone and by the calling SIP phone due to the Record-Route header field being passed back in the 180 (Ringing) and 200 (OK). These Record-Route headers are marked in bold in the 180 and 200 replies of the following printout. The Record-Route information is stored for the duration of the dialog. The proxy server will then also receive and proxy the ACK and BYE / 200 (OK). Each proxy can independently decide to receive subsequent messages, and those messages will pass through all proxies that elect to receive them [rfc3261, p.16].

The Record-Route header field is added to the message by the following script in the configuration file [cfg]:

```
  if(method!="REGISTER") {
    xlog("L_INFO","$CbxAdding the Route header$Cxx\n");
    record_route();
  }

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

The long printout below shows that our proxy server received all signalling messages starting from the call setup until the end of the conversation.

When endpoint SIP phones receive the stack of **Record-Route** headers (subsequently added by each intermediary proxy wishing to stay in the dialog) they copy this stack into a stack of **Route** headers. The Route headers dictate the transmission path of messages sent out by the SIP phone. The SIP phone will add the Route headers to all requests transmitted within the scope of the current SIP session (Route headers are marked in bold in the following printout).

When the called SIP phone receives an SIP request with a stack of Record-Route headers, it will copy the same Record-Route headers into the reply message. In this way the originating endpoint SIP phone will also receive the stack of Record-Route headers. The originating SIP phone also creates a stack of Route headers, but this stack will be inverted in order (since the sequence of proxies to be traversed from this side is reversed).

The SER server does not maintain the state of the SIP dialog (i.e. it is unaware of the on-going phone call). It is the responsibility of the SIP phone to make sure that within the scope of a phone call the stack of Route headers is added into each transmitted message (thereby making sure that the elected intermediary SIP servers receive all messages related to the current phone call).

```
[  Method INVITE from 192.168.1.10  ]
INVITE sip:309@192.168.1.15 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKf3c60221ce03445c
From: <sip:308@192.168.1.15>;tag=fce371520693b722
To: <sip:309@192.168.1.15>
Contact: <sip:308@192.168.1.10>
Supported: replaces
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 28826 INVITE
```

```
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/sdp
Content-Length: 225

v=0
o=308 8000 8000 IN IP4 192.168.1.10
s=SIP Call
c=IN IP4 192.168.1.10
t=0 0
m=audio 5004 RTP/AVP 4 18 97
a=sendrecv
a=rtpmap:4 G723/8000
a=rtpmap:18 G729/8000
a=rtpmap:97 iLBC/8000
a=fmtp:97 mode=20
a=ptime:60
{end}
[  End of Request  ]


[  Reply 100 (Trying) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKb992.ec6bce33.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKf3c60221ce03445c
From: <sip:308@192.168.1.15>;tag=fce371520693b722
To: <sip:309@192.168.1.15>
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 28826 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Content-Length: 0

[  End of Reply  ]


[  Reply 180 (Ringing) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKb992.ec6bce33.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKf3c60221ce03445c
Record-Route: <sip:192.168.1.15;lr=on;ftag=fce371520693b722>
From: <sip:308@192.168.1.15>;tag=fce371520693b722
To: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 28826 INVITE
User-Agent: Grandstream BT110 1.0.8.33
Content-Length: 0

[  End of Reply  ]


[  Reply 200 (OK) from 192.168.1.11 concerning INVITE  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKb992.ec6bce33.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bKf3c60221ce03445c
Record-Route: <sip:192.168.1.15;lr=on;ftag=fce371520693b722>
From: <sip:308@192.168.1.15>;tag=fce371520693b722
To: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 28826 INVITE
User-Agent: Grandstream BT110 1.0.8.33
```

```
Contact: <sip:309@192.168.1.11>
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/sdp
Supported: replaces
Content-Length: 154

v=0
o=309 8000 8000 IN IP4 192.168.1.11
s=SIP Call
c=IN IP4 192.168.1.11
t=0 0
m=audio 5004 RTP/AVP 4
a=sendrecv
a=rtpmap:4 G723/8000
a=ptime:60
{end}
[  End of Reply  ]


[  Method ACK from 192.168.1.10  ]
ACK sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK9678e44f4ea235df
Route: <sip:192.168.1.15;lr=on;ftag=fce371520693b722>
From: <sip:308@192.168.1.15>;tag=fce371520693b722
To: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
Contact: <sip:308@192.168.1.10>
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 28826 ACK
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Length: 0

[  End of Request  ]


[  Method INFO from 192.168.1.10  ]
INFO sip:309@192.168.1.11 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK93bc61c18d96eb8b
Route: <sip:192.168.1.15;lr=on;ftag=fce371520693b722>
From: <sip:308@192.168.1.15>;tag=fce371520693b722
To: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
Contact: <sip:308@192.168.1.10>
Supported: replaces
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 28827 INFO
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/dtmf-relay
Content-Length: 23

Signal=2
Duration=4800{end}
[  End of Request  ]


[  Reply 200 (OK) from 192.168.1.11 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKc992.e100efc6.0
Via: SIP/2.0/UDP 192.168.1.10;branch=z9hG4bK93bc61c18d96eb8b
```

```
Record-Route: <sip:192.168.1.15;lr=on;ftag=fce371520693b722>
From: <sip:308@192.168.1.15>;tag=fce371520693b722
To: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 28827 INFO
User-Agent: Grandstream BT110 1.0.8.33
Contact: <sip:309@192.168.1.11>
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Supported: replaces
Content-Length: 0

[  End of Reply  ]


[  Method INFO from 192.168.1.11  ]
INFO sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bKd51a30c261b5691e
Route: <sip:192.168.1.15;lr=on;ftag=fce371520693b722>
From: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
To: <sip:308@192.168.1.15>;tag=fce371520693b722
Contact: <sip:309@192.168.1.11>
Supported: replaces
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 52519 INFO
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Type: application/dtmf-relay
Content-Length: 22

Signal=3
Duration=480{end}
[  End of Request  ]


[  Reply 200 (OK) from 192.168.1.10 concerning INFO  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bKb593.5d620aa5.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bKd51a30c261b5691e
Record-Route: <sip:192.168.1.15;lr=on;ftag=352ec99f5ffd23cf>
From: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
To: <sip:308@192.168.1.15>;tag=fce371520693b722
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 52519 INFO
User-Agent: Grandstream BT110 1.0.8.33
Contact: <sip:308@192.168.1.10>
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Supported: replaces
Content-Length: 0

[  End of Reply  ]


[  Method BYE from 192.168.1.11  ]
BYE sip:308@192.168.1.10 SIP/2.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bKd0b04fbb081eb9ab
Route: <sip:192.168.1.15;lr=on;ftag=352ec99f5ffd23cf>
From: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
To: <sip:308@192.168.1.15>;tag=fce371520693b722
Supported: replaces
Call-ID: 8c80d06175b1eb80@192.168.1.10
```

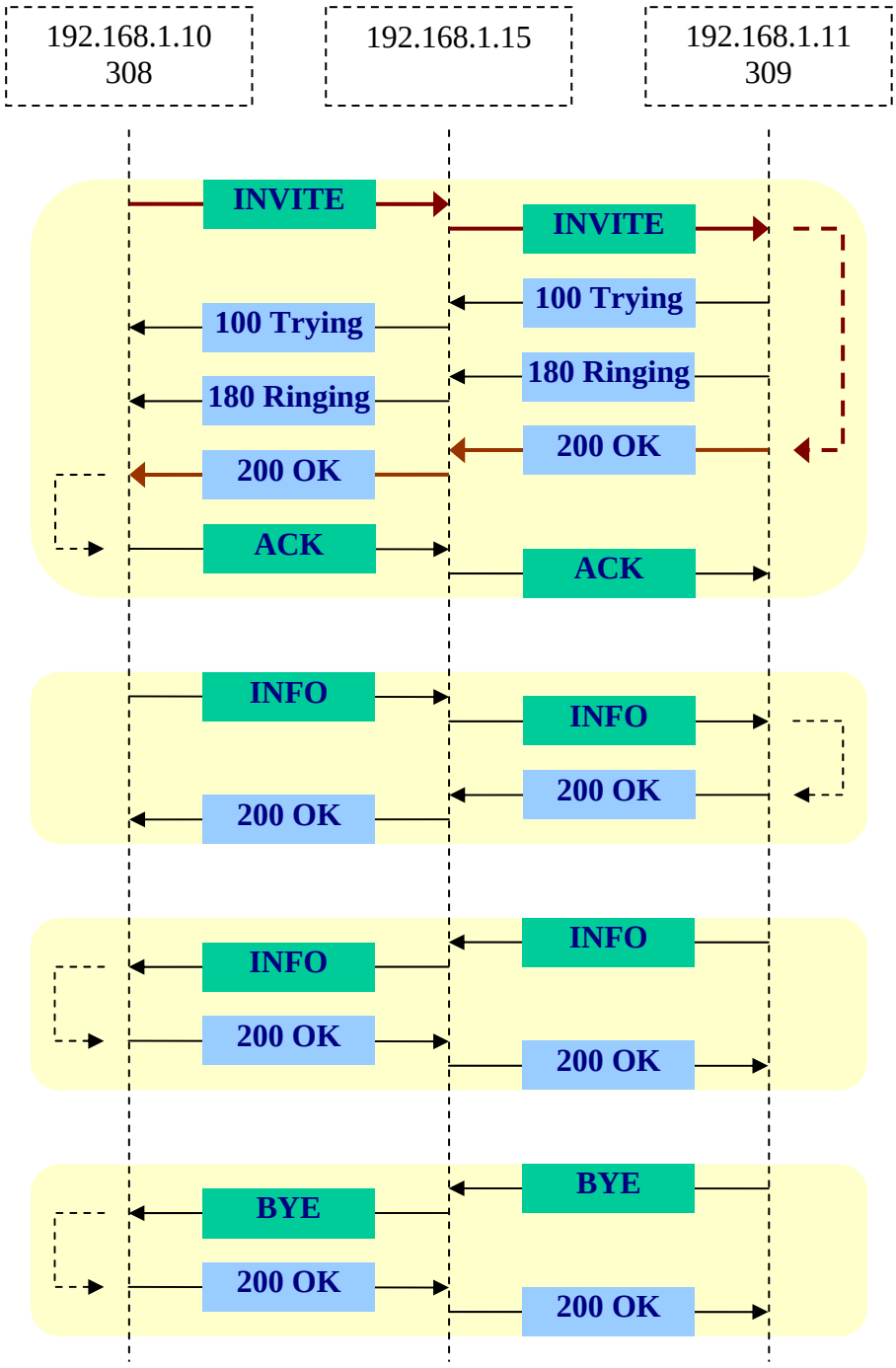
```
CSeq: 52520 BYE
User-Agent: Grandstream BT110 1.0.8.33
Max-Forwards: 70
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Content-Length: 0

[  End of Request  ]


[  Reply 200 (OK) from 192.168.1.10 concerning BYE  ]
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.1.15;branch=z9hG4bK0593.583e3506.0
Via: SIP/2.0/UDP 192.168.1.11;branch=z9hG4bKd0b04fbb081eb9ab
Record-Route: <sip:192.168.1.15;lr=on;ftag=352ec99f5ffd23cf>
From: <sip:309@192.168.1.15>;tag=352ec99f5ffd23cf
To: <sip:308@192.168.1.15>;tag=fce371520693b722
Call-ID: 8c80d06175b1eb80@192.168.1.10
CSeq: 52520 BYE
User-Agent: Grandstream BT110 1.0.8.33
Contact: <sip:308@192.168.1.10>
Allow: INVITE,ACK,CANCEL,BYE,NOTIFY,REFER,OPTIONS,INFO,SUBSCRIBE
Supported: replaces
Content-Length: 0

[  End of Reply  ]
```

[htm], [doc], [txt]

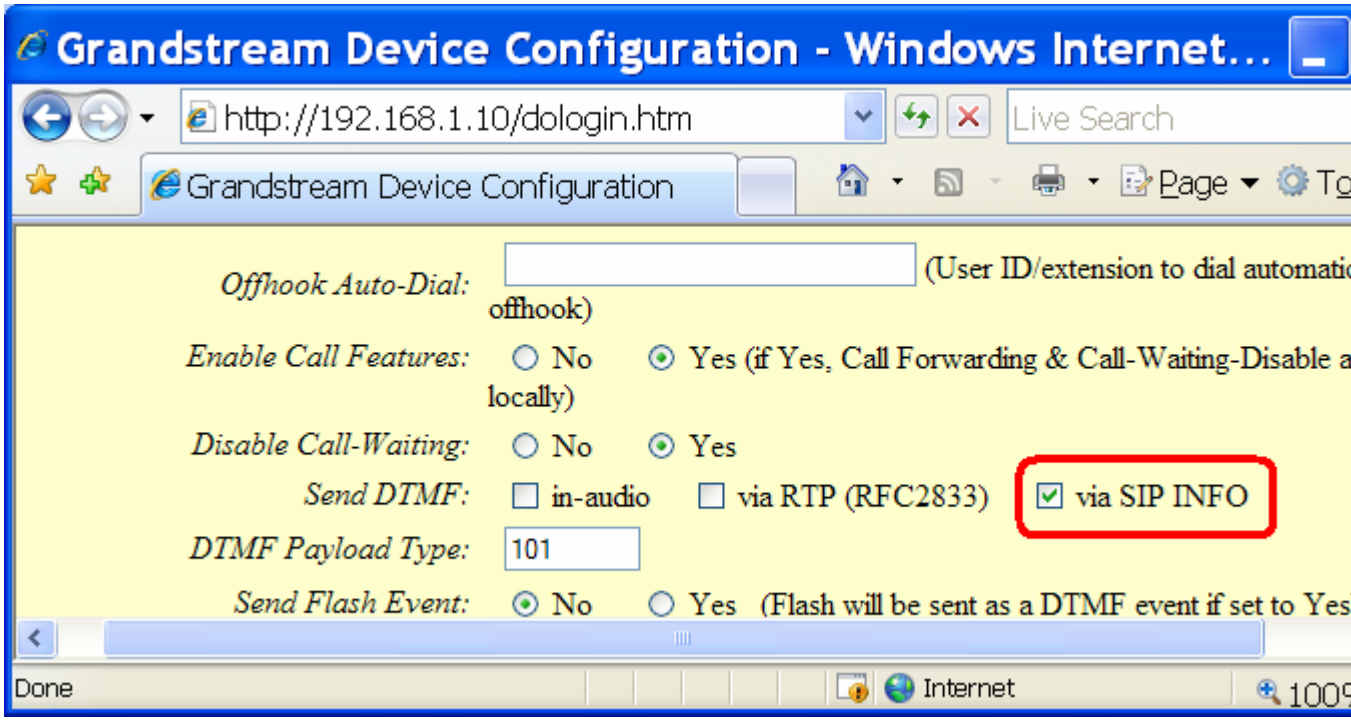
Although the SIP server does not identify phone conversations, it can record all received messages in a database. An external application can then match the messages related to a particular phone conversation. The phone conversation can be identified by the tag of the originating SIP phone (tag=fce371520693b722) [rfc3261, p.12], by the tag of the callee SIP phone (tag=352ec99f5ffd23cf) [rfc3261, p.14-15], and by the Call-Id field (8c80d06175b1eb80@192.168.1.10). The tags and the Call-Id field are generated by endpoint SIP phones and are maintained by them constantly during the phone conversation. Tag parameters are random strings, while Call-Id is generated by a combination of a random string and the originating phone's host name or IP address. The combination of the To tag, From tag, and Call-ID completely defines a peer-to-peer SIP relationship between two SIP phones and is referred to as a dialog [rfc3261, p.12].

Since a phone call (i.e. an SIP dialog) consists of several transactions, and SER does not keep information about transactions throughout a particular phone call, the consequence is that SER does not know that a call is on-going, nor can SER calculate the length of an ended or active call. Neither can SER terminate a phone call. However, SER can store the times at which an INVITE (or ACK) and a BYE message are received and record this info together with the Call-Id. A billing application can then match the INVITE with the BYE and calculate the lengths of calls [SER Getting Started, p.9].

The dialog of the printout is demonstrated by the following diagram:



During the phone conversation there are two transactions with INFO messages. These transactions correspond to transmissions of DTMF signals via SIP messages. The first INFO message tells that the digit “2” is pressed on the calling phone for 4800 milliseconds. The second INFO message tells that the digit “3” is pressed on the called phone for 480 milliseconds. To make sure that SIP INFO messages are transmitted, we must specify the transmission of DTMF signals with SIP messages to SIP phones:



[bmp]

From the printout we deduce that when the call is successfully answered, the ACK message (branch=z9hG4bK9678e44f4ea235df) no longer belongs to the transaction of the INVITE message (branch=z9hG4bKf3c60221ce03445c). This is the reason why, in section 2.5 where the Record-Route header was not used, the ACK message bypassed the proxy server.

It is important to remember that the Record-Route header only gives instructions to endpoint SIP phones to add the corresponding Route headers to all SIP requests during the entire duration of the phone call. Therefore, it is the two endpoint SIP phones that are responsible for ensuring that the Route headers are added to all messages associated with the active call. The endpoint SIP phones are obviously aware of the active phone call, but not the SIP proxy.

4. Other experiments for understanding SIP

Examining the STUN settings of an SIP phone

Creating and sending INVITE and CANCEL SIP text messages

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

Understanding SIP exchanges by experimentation

This document [htm], [pdf], [doc], the whole web page [zip]

5. Glossary

FQDN	Fully Qualified Domain Name
SDP	Session Description Protocol
HTTP	Hyper Text Transfer Protocol
RFC	Request for Comments
SIP	Session Initiation Protocol
SER	SIP Express Router
RTP	Real-time Transport Protocol
UA	User Agent
ACK	Acknowledgement
DTMF	Dual Tone Multi Frequency
URI	Universal Resource Identifier
DHCP	Dynamic Host Configuration Protocol

6. Related links

http://openser.org/docs/pseudo-variables-1.1.x.html, pseudo variables (openser)
http://www.iptel.org/ser/doc/modules/xlog, pseudo variables for xlog (ser)
http://www.iptel.org/ser/doc/gettingstarted, getting started documentation
ftp://siprouter.onsip.org/pub/gettingstarted/configs/, simple configuration files
http://www.faqs.org/rfcs/rfc3261.html, SIP: Session Initiation Protocol
http://www.faqs.org/rfcs/rfc2327.html, Session Description Protocol
http://www.iptel.org/ser/, home page of Sip Express Router
http://www.openser.org/, home page of OpenSER
http://www.openser.org/dokuwiki/doku.php/core-cookbook:1.1.x, core keywords, core values, and core parameters of SER configuration file
http://www.openser.org/docs/modules/1.2.x/textops.html, e.g. searching a regular expression in the body of a SIP message
http://mit.edu/sip/sip.edu/ser.shtml, some SER examples
http://www.iptel.org/how_can_i_dynamically_change_the_invite_timer_fr_invite, configuring timeout