SIP messages

- Messages
  - Requests - Client → Server direction
  - Responses - Server → Client direction
- No response without request (every response has its meaning only with respect to the request which invoked it)
- Both messages (requests/responses) use the generic-message-format

Generic message format

- Every message consists of:
  - Start line
  - Header fields / Headers (one or more)
  - Empty line (CRLF) – indicates the end of message header fields
  - Message body, if any (optional)

SIP message

```
INVITE sip:411@salzburg.at;user=phone SIP/2.0
Via: SIP/2.0/UDP salzburg.edu.at:5060;branch=z9hG4bK1d32hr4
Max-Forwards:70
To: <sip:411@salzburg.at;user=phone>
From: Christian Doppler <sip:c.doppler@salzburg.edu.at>
;tag=817234
Call-ID: 12-45-A5-46-F5@salzburg.edu.at
CSeq: 1 INVITE
Subject: Train Timetables
Contact: sip:c.doppler@salzburg.edu.at
Content-Type: application/sdp
Content-Length: 151
v=0
o=doppler 2890842326 2890844532 IN IP4 salzburg.edu.at
s=Phone Call
c=IN IP4 50.61.72.83
t=0 0
```

Header 1/3

- A header is a component of a SIP message that conveys information about the message. It is structured as a sequence of header fields.

Header 2/3

- Header Field
  - A header field is a component of the SIP message header.
  - A header field can appear as one or more header field rows.
  - Header field rows consist of a header field name and zero or more header field values.
  - Multiple header field values on a given header field row are separated by commas.

Header 3/3

- Header Field Value
  - A header field value is a single value.
  - A header field consists of zero or more header field values.

Request

- Start line = Request line
  - Request-Line contains a method name, a Request-URI, and the protocol version separated by a single space (SP) character.
  - Request-Line ends with CRLF. No CR or LF are allowed except end-of-line CRLF sequence.
  - No linear white space (LWS) is allowed in any of the elements.
- Request line = Method SP Request-URI SP SIP-Version CRLF
- Example:

INVITE sip:franta.novak@trainingpoint.cz SIP/2.0

Response

- Start line = Status line
  - Consists of the protocol version followed by a numeric Status-Code and its associated textual phrase, with each element separated by a single SP character.
  - No CR or LF is allowed except in the final CRLF sequence.
- Example:

SIP/2.0 200 OK

Header field format 1/2

- Header field consists of a field name followed by a colon (“:”) and the field value.
- Format:
  - Header : parameter; parameter=value; par2=value2...
- Arbitrary white-space is allowed on both sides of colon:
  - Subject : lunch
  - Subject : lunch
  - Subject : lunch
  - Subject : lunch

Header field format 2/2

- Case-insensitive
  - Field names (always)
  - Field values
  - Paramaters names
  - Paramaters values
  - Tokens (always)
- Case-sensitive
  - Values expressed as quoted strings

Header fields

- Relative order of particular header fields is not significant except more header fields with same field names
- It is RECOMMENDED that header fields which are needed for proxy processing (Via, Route,…) appear towards the top of the message.
- Example:

Both valid but not same

```
Route: <sip:alice@atlanta.com>
Route: <sip:carol@chicago.com>
Route: <sip:bob@biloxi.com>
```

```
Route: <sip:bob@biloxi.com>
Route: <sip:alice@atlanta.com>
Route: <sip:carol@chicago.com>
```

Multiple header fields

It MUST be possible to combine multiple header fields:

Route: <sip:alice@atlanta.com>
Route: <bob@biloxi.com>
Route: <sip:carol@chicago.com>

Is the same as:

Route: <sip:alice@atlanta.com>, <bob@biloxi.com>,
<sip:carol@chicago.com>

Compact form of header

- Some often used header fields could be used in abbreviated form:

  From: <sip:user1_public1@home1.net>;tag=171828
  f: <sip:user1_public1@home1.net>;tag=171828

- Notes:
  - Not every header has this form
  - The first character must not be the abbreviation:

    Call-ID: f81d4fae-7dec-11d0-A765-00a0c91e6bf6@biloxi.com
    i: f81d4fae-7dec-11d0-A765-00a0c91e6bf6@biloxi.com

Treatment of Unrecognized Responses

- UAC MUST treat any final XYZ response it does not recognize as equivalent X00 response
- UA MUST be able to process the x00 response code for all classes.
- UAC MUST treat any provisional response different than 100 that it does not recognize as 183 (Session Progress).
- UAC MUST be able to process 100 and 183 responses

IANA

- The Internet Assigned Numbers Authority (IANA) is the entity that oversees global IP address allocation, DNS root zone management, and other Internet protocol assignments.
- Compleete list method names, header field names, status codes, andoption tags used in SIP applications are registered with IANA.
- http://www.iana.org/assignments/sip-parameters

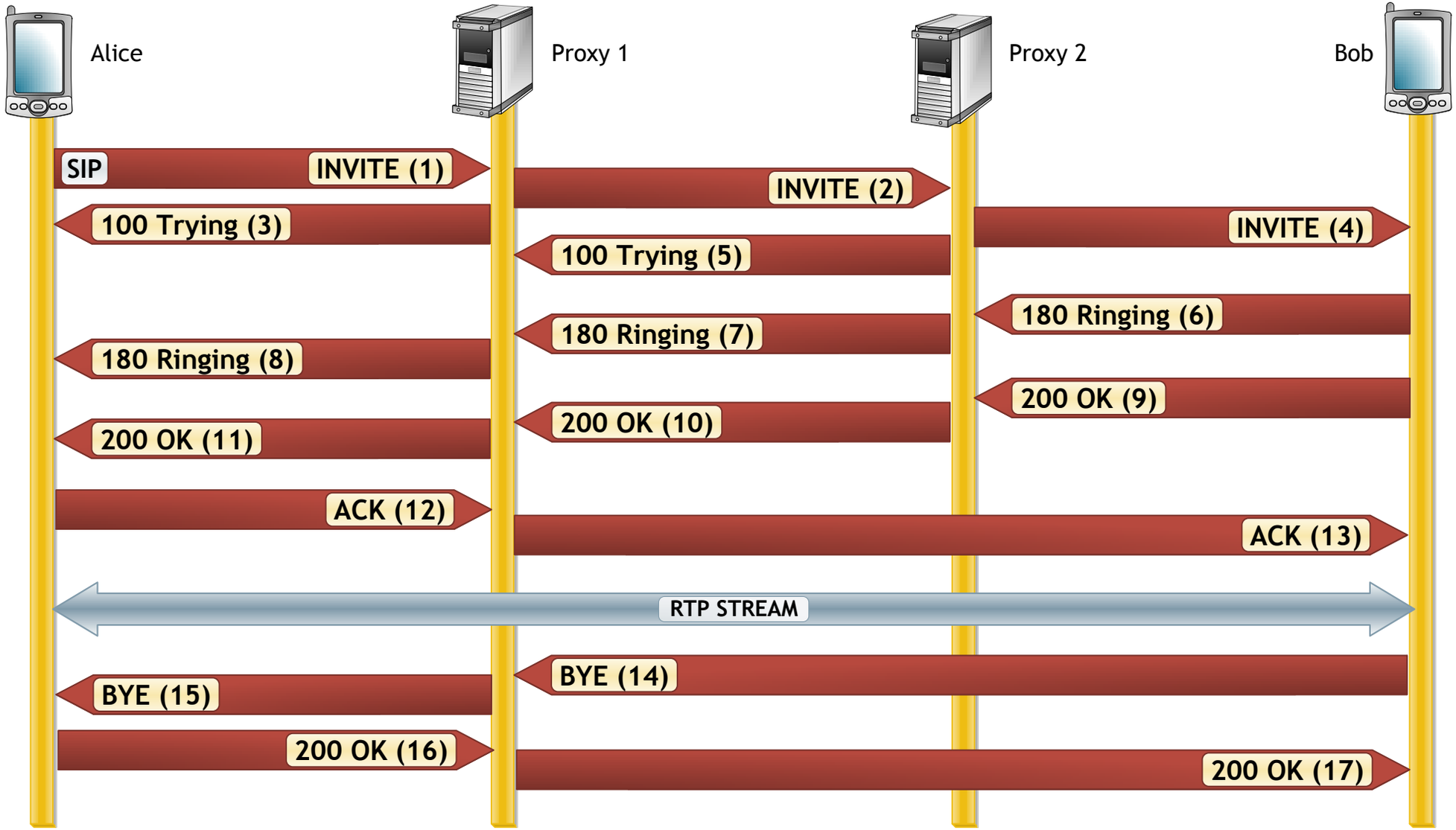
Basic SIP operations

- SIP call
  - Caller hangs up
  - Callee is busy
  - INVITE with no SDP
  - Reinvite
  - UPDATE
  - Early media session
  - Statefull vs. Stateless Proxy

Basic SIP operations

- Redirection / Redirect server
- Registration & Authentication
- Location Service
- Enum

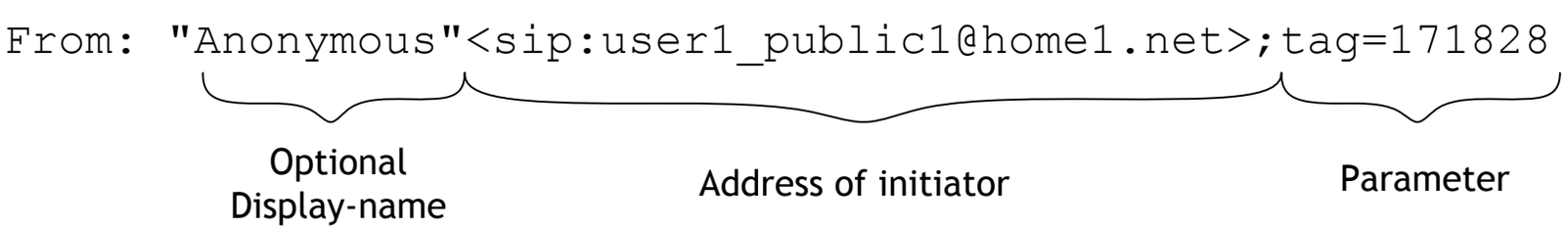
SIP call


Alice
Proxy 1
Proxy 2
Bob
SIP
INVITE (1)
INVITE (2)
100 Trying (3)
INVITE (4)
100 Trying (5)
180 Ringing (6)
180 Ringing (7)
180 Ringing (8)
200 OK (9)
200 OK (10)
200 OK (11)
ACK (12)
ACK (13)
RTP STREAM
BYE (14)
BYE (15)
200 OK (16)
200 OK (17)

From header field

The From header field indicates the initiator of the *request*. This may be different from the initiator of the dialog. Requests sent by the callee to the caller use the callee's address in the From header field.

- Copied from the request to the response
- Compact form is f



To header field

The To header field specifies the logical recipient of the request. The *tag* parameter serves as a general mechanism for dialog identification.

- Copied from the request to the response
- Compact form is t

```
To: "Bob Smith"<sip:user2_public2@home2.net>;tag=171828
```

Optional Display-name

Address of recipient

Parameter

Call-ID header field

The Call-ID header field uniquely identifies a particular invitation or all registrations of a particular client.

- Copied from the request to the response
- Compact form is i
- Case sensitive and simply compared byte-by-byte
- Call-ID together with From tag and To tag uniquely determines the dialog (by reason of forking)

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@biloxi.com
```

Contact header field

The Contact header field provides a URI where the user can be reached directly

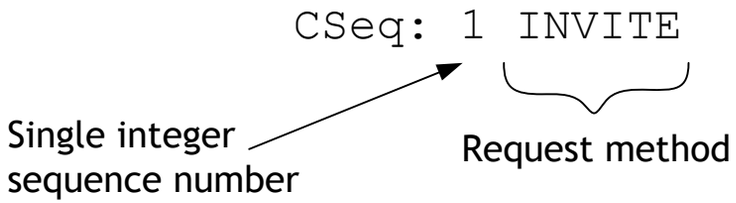
- Compact form is m

```
Contact: "Mr. Watson"<sip:watson@worcester.bell-telephone.com>; expires=3600
```

CSeq header field

The CSeq header field serves to order transactions within dialog, to provide a means to uniquely identify transactions, and to differentiate between new requests and request retransmissions.

- Copied from the request to the response



Allow header field

The Allow header field lists the set of methods supported by the UA generating the message.

- All methods MUST be included in the list
- Used in requests and responses

```
Allow: INVITE, ACK, OPTIONS, CANCEL, BYE
```

Content-Type header field

The Content-Type header field indicates the media type of the message-body sent to the recipient.

- May be present in all requests and responses
- Must be present if the body is not empty
- Compact form is c

```
Content-Type: application/sdp
```

Accept header field

The Accept header field is especially useful for indicating support of various session description formats.

- An empty Accept header field means that no formats are acceptable
- If no Accept is present, the server SHOULD assume a default value of application/sdp.
- Used in requests, 2xx, 415

```
Accept: application/sdp;level=1, application/x-private, text/html
```

Content-Length header field

The Content-Length header field indicates the size of the message-body, in decimal number of octets, sent to the recipient.

- May be present in all requests and responses
- If a stream-based protocol (such as TCP) is used as transport, the header field MUST be used
- If message has no body, the value MUST be set to zero
- Compact form is l

```
Content-Length: 349
```

Content-Encoding header field

The Content-Encoding header field is used as a modifier to the "media-type". When present, its value indicates what additional content coding have been applied to the entity-body.

- Primarily used to allow a body to be compressed without losing the identity of its underlying media type
- May be present in all requests and responses
- All content-coding values are case-insensitive
- Compact form is e

```
Content-Encoding: gzip
```

Content-Language header field

The Content-Language header field describes the natural language (s) of the intended audience for the enclosed entity.

- May be present in all requests and responses

```
Content-Language: fr
```

Max-Forwards header field

The Max-Forward header field must be used to limit the number of proxies or gateways that can forward the request to the next downstream server.

- Inserted by elements that can not guarantee loop detection; count is decremented by each server
- Integer in the range 0-255
  (recommended initial value is 70)
- Used in requests

```
Max-Forwards: 6
```

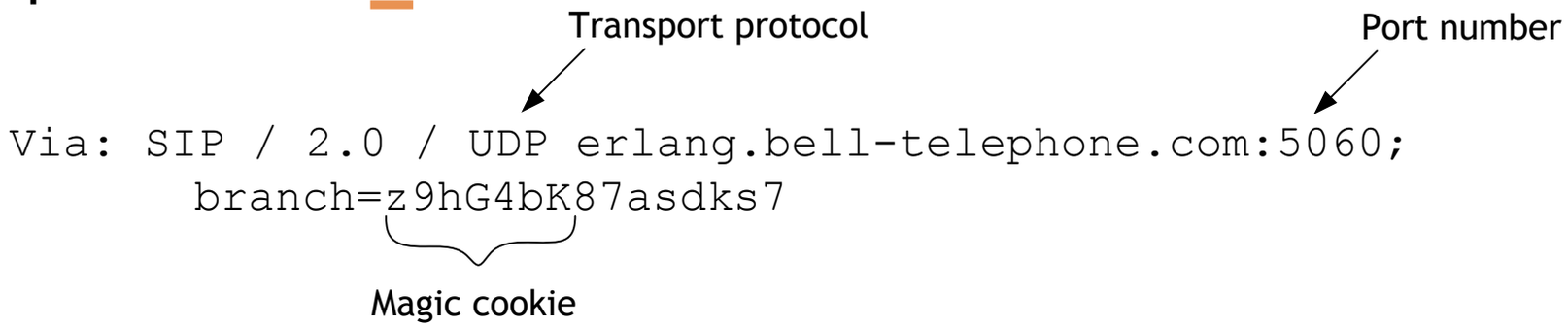
INVITE (1)

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:proxy1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 384827629820188511@atlanta.example.com
CSeq: 31 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Contact: <sip:alice@client.atlanta.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 151
```

Via header field

The Via header field indicates the path taken by the request so far and indicates the path that should be followed in routing responses.

- Used in requests and responses
- Branch ID parameter serves as a transaction identifier
  (used by proxies to detect loops)
- Value of the branch parameter starts with the magic cookie “z9hG4bK”
- Compact form is v



INVITE (2)

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1 -- Via added
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065 -- Parameters received and rport added; see RFC 3581
Max-Forwards: 69
Record-Route: <sip:proxy1.atlanta.example.com;lr> -- Record-Route added
-- Route: <sip:proxy1.atlanta.example.com;lr> -- Route deleted
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Contact: <sip:alice@client.atlanta.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 151
    ...
```

Record-Route header field

The Record-Route header field is inserted by proxies in a request to force future requests in the dialog to be routed through proxy.

- Used in requests, 18x, 2xx, 401, 484

```
Record-Route: <sip:server10.biloxy.com;lr>,
<sip:bigbox3.site3.atlanta.com;lr>
```

100 Trying (3)

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 INVITE
Content-Length: 0
```

INVITE (4)

```
INVITE sip:bob@client.biloxi.example.com SIP/2.0 -- Request-URI changed
Via: SIP/2.0/UDP proxy2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=9127 -- received and rport added
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
Max-Forwards: 68
Record-Route: <sip:proxy1.atlanta.example.com;lr> -- No Record-Route added
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Contact: <sip:alice@client.atlanta.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 151
    …
```

100 Trying (5)

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=9127
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 31 INVITE
Content-Length: 0
```

180 Ringing (6)

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP proxy2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
;received=192.0.2.222;rport=7810 -- received and rport added
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=9127
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
Record-Route: <sip:proxy1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159 -- tag added
Call-ID: 384827629822018851l@atlanta.example.com
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
CSeq: 31 INVITE
Content-Length: 0
```

All *Via* headers copied from INVITE (F4)

180 Ringing (7)

```
SIP/2.0 180 Ringing
-- Via: SIP/2.0/UDP proxy2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
;received=192.0.2.222;rport=7810
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=9127
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
Record-Route: <sip:proxy1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
CSeq: 31 INVITE
Content-Length: 0
```

Via headers deleted

180 Ringing (8)

```
SIP/2.0 180 Ringing
-- Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=9127
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
Record-Route: <sip:proxy1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
CSeq: 31 INVITE
Content-Length: 0
```

Via header deleted

200 OK (9)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
;received=192.0.2.222;rport=7810
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=9127
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
Record-Route: <sip:proxy1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, INFO
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 147
    …
```

All *Via* headers copied from INVITE (F4)

200 OK (10)

```
SIP/2.0 200 OK
-- Via: SIP/2.0/UDP proxy2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
;received=192.0.2.222;rport=7810
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=9127
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
Record-Route: <sip:proxy1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, INFO
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 147
    …
```

Via header deleted

200 OK (11)

```
SIP/2.0 200 OK
-- Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060
;branch=z9hG4bK2d4790.1;received=192.0.2.111;rport=9167
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
;received=192.0.2.101;rport=12065
Record-Route: <sip:proxy1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, INFO
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 147
    …
```

Via header deleted

ACK (12)

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b76
Max-Forwards: 70
Route: <sip:proxy1.atlanta.example.com;lr> -- copied from Record-Route
of 200 OK (F11)
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 ACK
Content-Length: 0
```

Route header field

The Route header field is used to force routing for a request through the listed set of proxies.

- Used in requests

```
Route: <sip:bigbox3.site3.atlanta.com;lr>
```

Route set

A route set is a collection of ordered SIP or SIPS URI which represent a list of proxies that must be traversed when sending a particular request. A route set can be learned, through headers like Record-Route, or it can be configured.

ACK (13)

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1 -- Via added
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b76
;received=192.0.2.101;rport=12065 -- received added
Max-Forwards: 69
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 31 ACK
Content-Length: 0
```

BYE (14)

```
BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP
client.biloxi.example.com:5060;branch=z9hG4bKnashds7;rport
Max-Forwards: 70
Route: <sip:proxy1.atlanta.example.com;lr> -- copied from Record-
Route of INVITE (F4)
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 26 BYE
Content-Length: 0
```

Field values changed

BYE (15)

```
BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1--Via added
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
;received=192.0.2.201;rport=13401 -- received and rport added
Max-Forwards: 69
-- Route: <sip:proxy1.atlanta.example.com;lr> -- Route deleted
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 26 BYE
Content-Length: 0
```

200 OK (16)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
;received=192.0.2.111;rport=16116 -- rport added
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
;received=192.0.2.201;rport=13401
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 26 BYE
Content-Length: 0
```

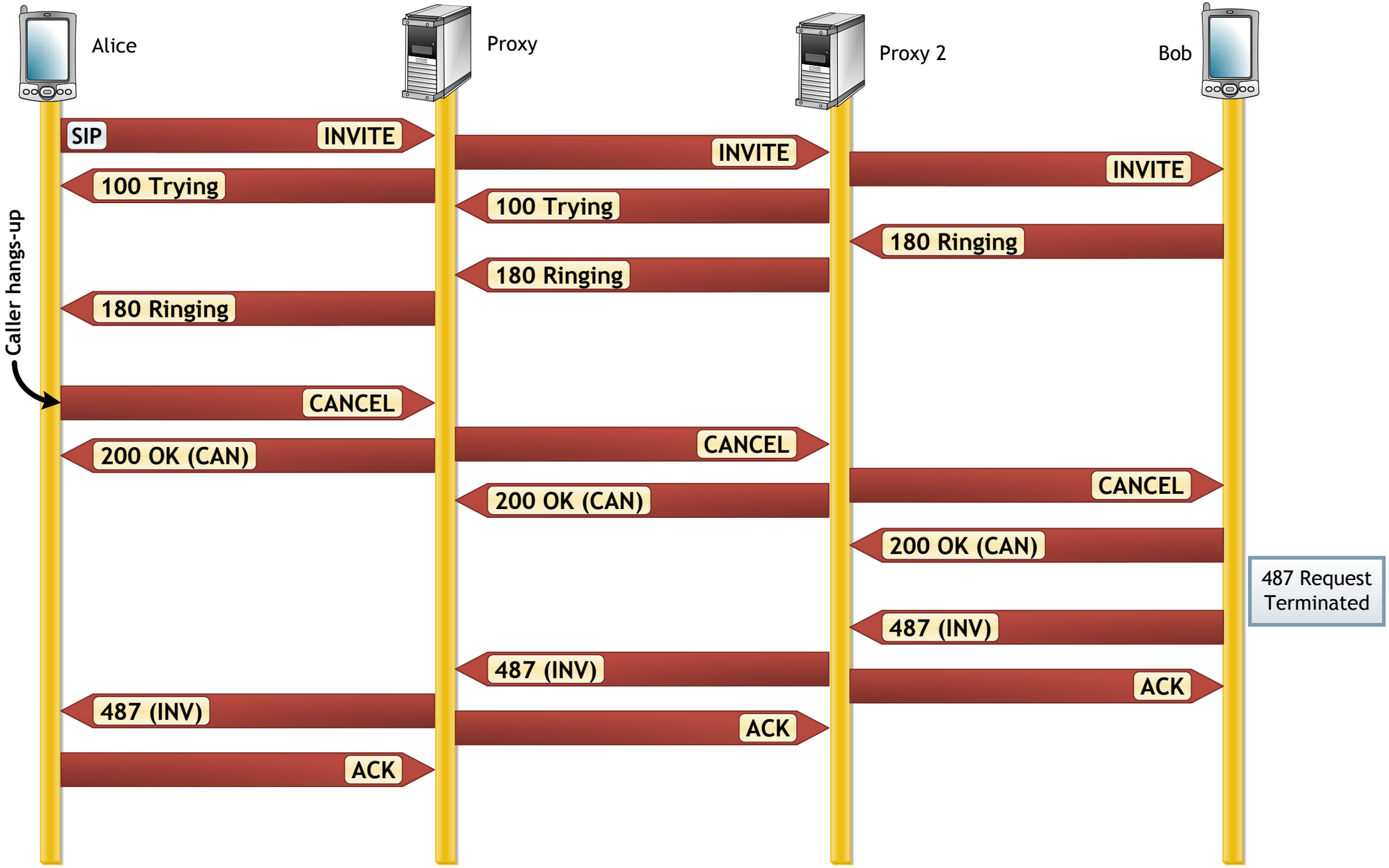
All *Via* headers copied from BYE (F15)

200 OK (17)

```
SIP/2.0 200 OK
-- Via: SIP/2.0/UDP proxy1.atlanta.example.com:5060
;branch=z9hG4bK2d4790.1;received=192.0.2.111;rport=16116
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
;received=192.0.2.201;rport=13401
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 26 BYE
Content-Length: 0
```

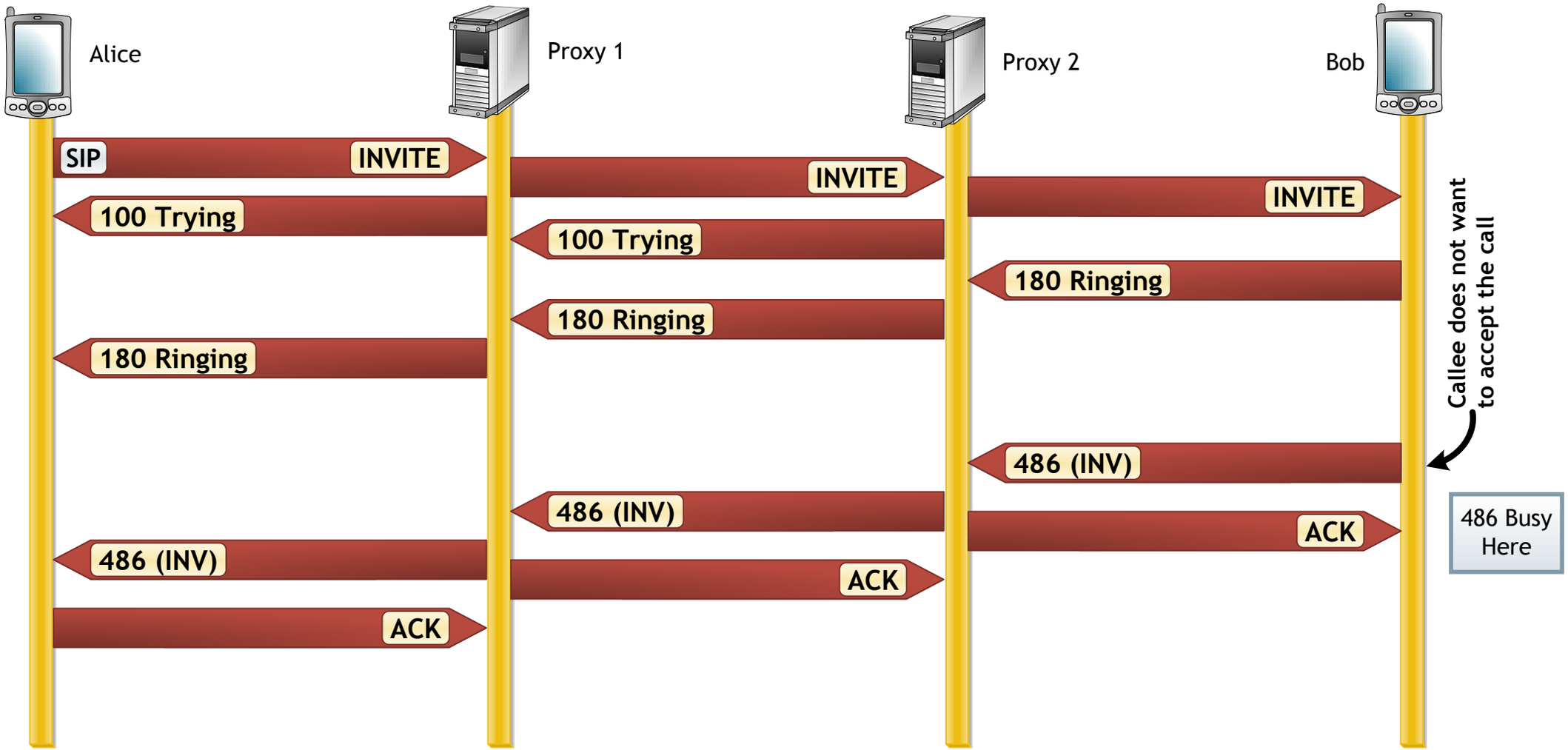
Via header deleted

Caller hangs up

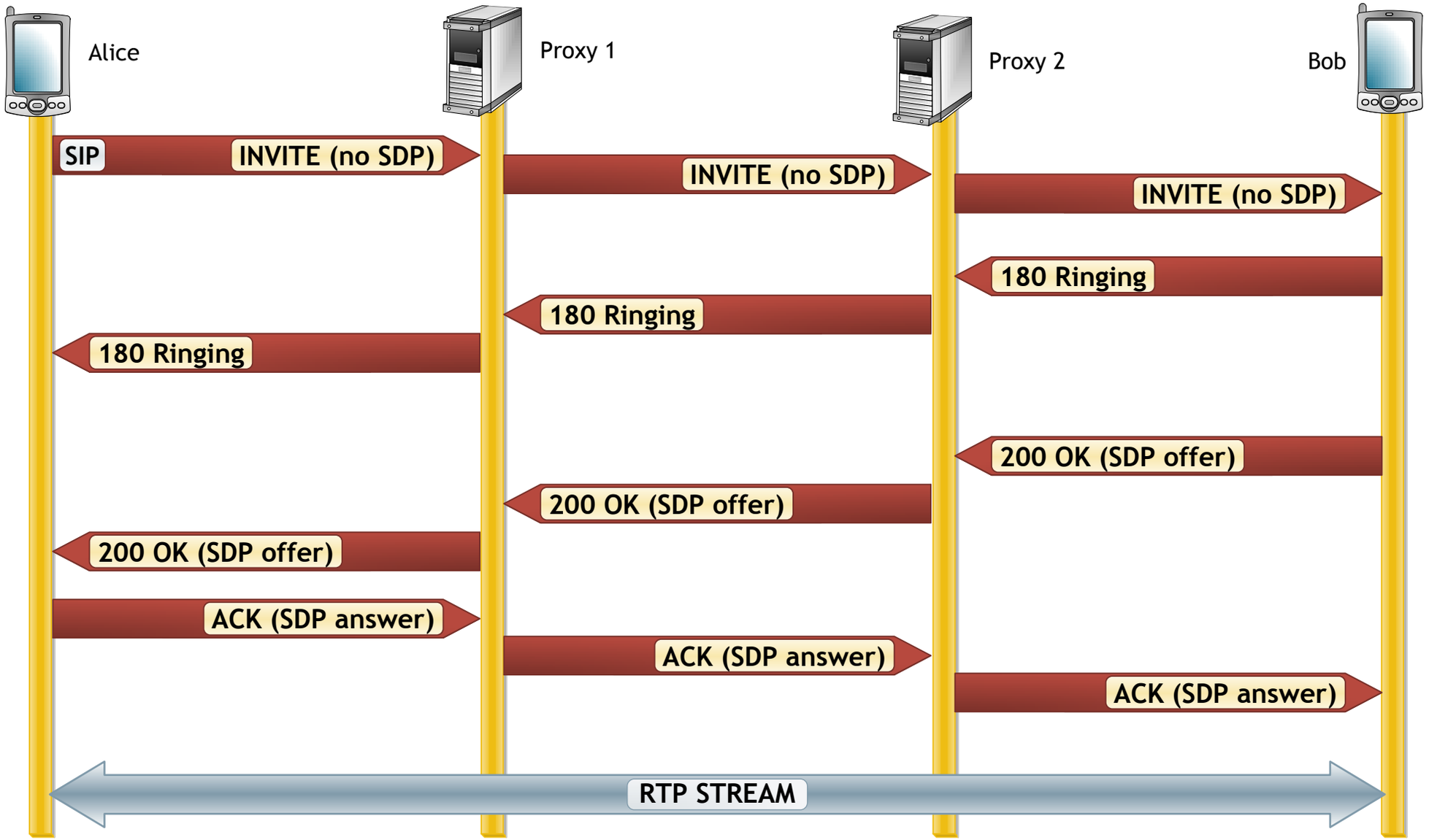


Note: CANCEL must not contain any message body!

Callee is busy



INVITE has no SDP Offer - successful case



Re-INVITE

Re-INVITE is an INVITE request sent within an existing dialog.

It is used for:

- modification of the actual session (e.g. changing addresses or ports, adding or deleting media streams, etc.)
- ensuring about the state of the dialog
- update some timers

Re-INVITEs use the same offer/answer model that applies to session description in INVITEs. (Note: full description of the session is used.)

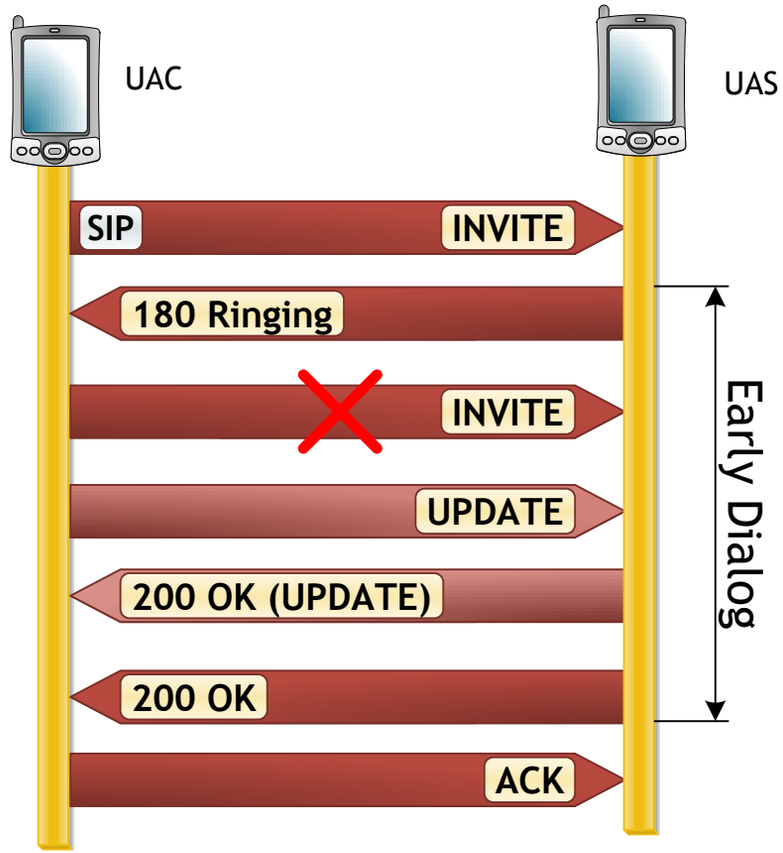
Re-INVITE characteristics

- Either the caller or callee can modify an existing session.
- Re-INVITE will never fork.
- SIP entity MUST able to respond to re-INVITE.
- Conclusion of the re-INVITE procedure have no direct influence to the state of original dialog.
- UAC MUST not initiate a new INVITE transaction within a dialog while another INVITE transaction is in progress in either direction.

UPDATE #1

Update request allows to client to update parameters of a session (such as the media streams and their codecs) but has no impact on the state of a dialog.

- UPDATE = target refresh request
- Unlike a re-INVITE, UPDATE can be sent before the initial INVITE has been completed → EARLY DIALOGS



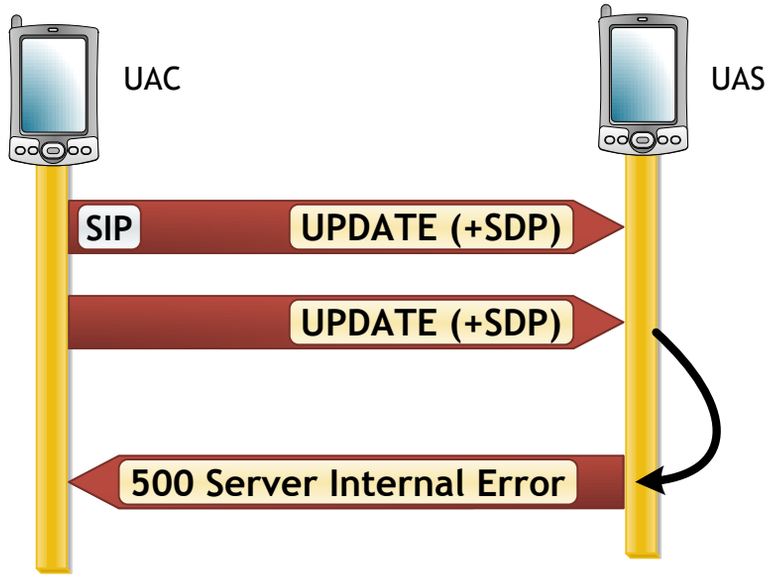
Note: A reliable provisional response will always create an early dialog. It is necessary in order to receive UPDATE from the callee.

UPDATE #2

- Both caller and callee are able to modify the haracteristics of the session
- Allow header field indicates support for UPDATE
- If a UA receives a non-2xx final response to a UPDATE, the session parameters MUST remain unchanged (x 481, 408, …)
- Although UPDATE can be used on confirmed dialogs, it is RECOMMENDED that a re-INVITE be used instead
  - UPDATE needs to be answered immediately, ruling out the possibility of user approval
  - If the UAS cannot change the session parameters without prompting the user, it SHOULD reject the request with 504 (Server Time-out) response.

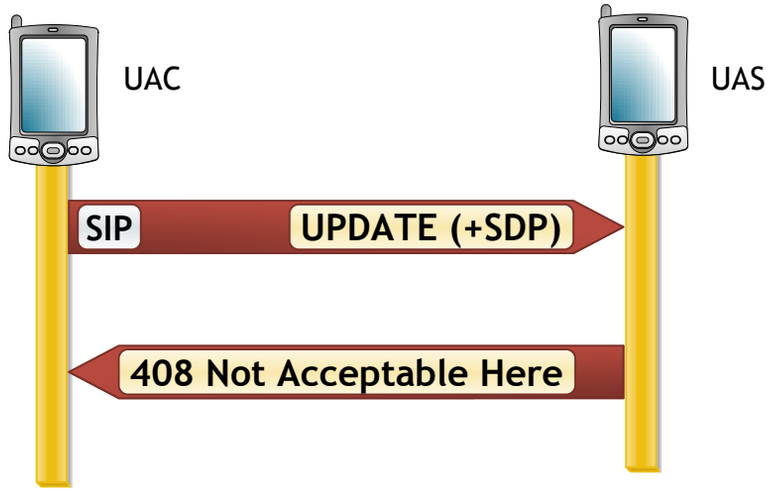
UPDATE #3

- SDP offer in UPDATE request can be sent only when there are not any other unanswered SDP offers.
- A UAS that receives an UPDATE before it has generated a final response to previous UPDATE on the same dialog MUST return a 500 (Server Internal Error) response to the new UPDATE, and MUST include a Retry-After header field with a randomly chosen value between 0 and 10 seconds.

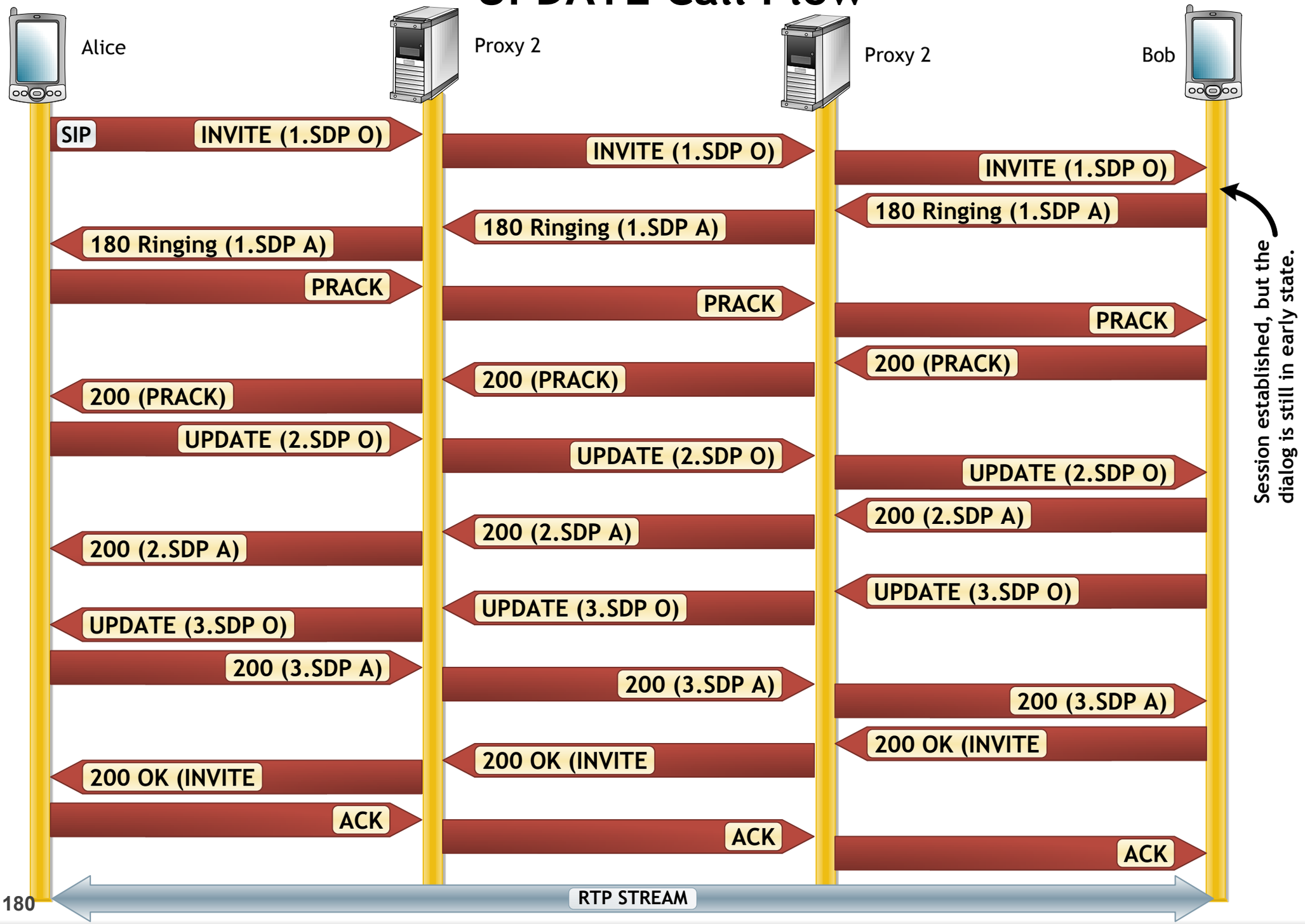


Rejection against an SDP Offer in UPDATE

- When a UA receives an UPDATE request with an SDP offer which it can not accept, it should respond with a 488 (Not Acceptable Here) response preferably with Warning header field indicating the reason of the rejection unless other response code is more appropriate to reject it.



UPDATE Call Flow



Early media session

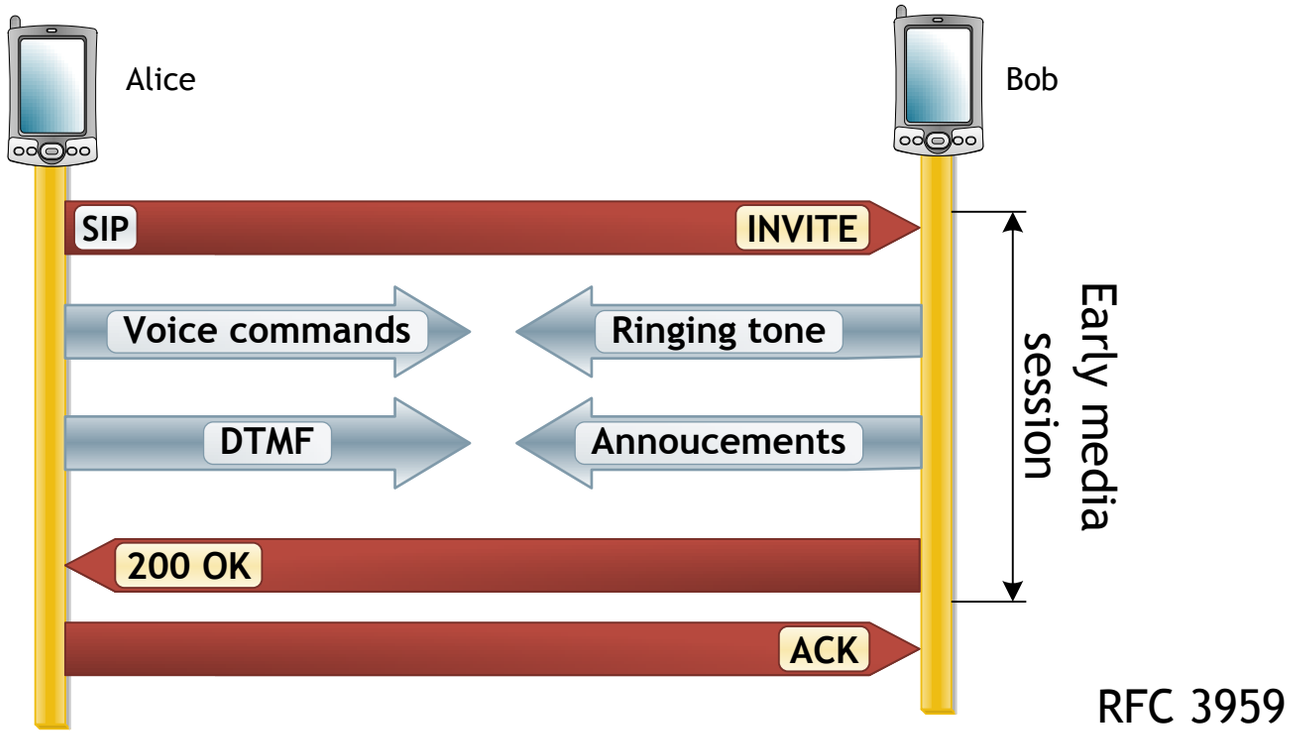
Early media session is a session within an early dialog which have been established in the same way as a regular sessions using an offer/answer model.

- Early media refers to media (e.g. audio and video) that is exchanged before a particular session is accepted by the called user (from sending of initial INVITE until final response generating).
- Can be unidirectional or bidirectional, and can be generated by the caller, the callee or both.
- Typical examples of early media generated by:
  - Caller – voice commands and DTMF to drive IVR
  - Callee – ringing tones and announcements

Early-session

For the purpose of early media session there is a new disposition type and a new option tag defined.

- Content-Disposition: early-session
- Supported: early-session



RFC 3959

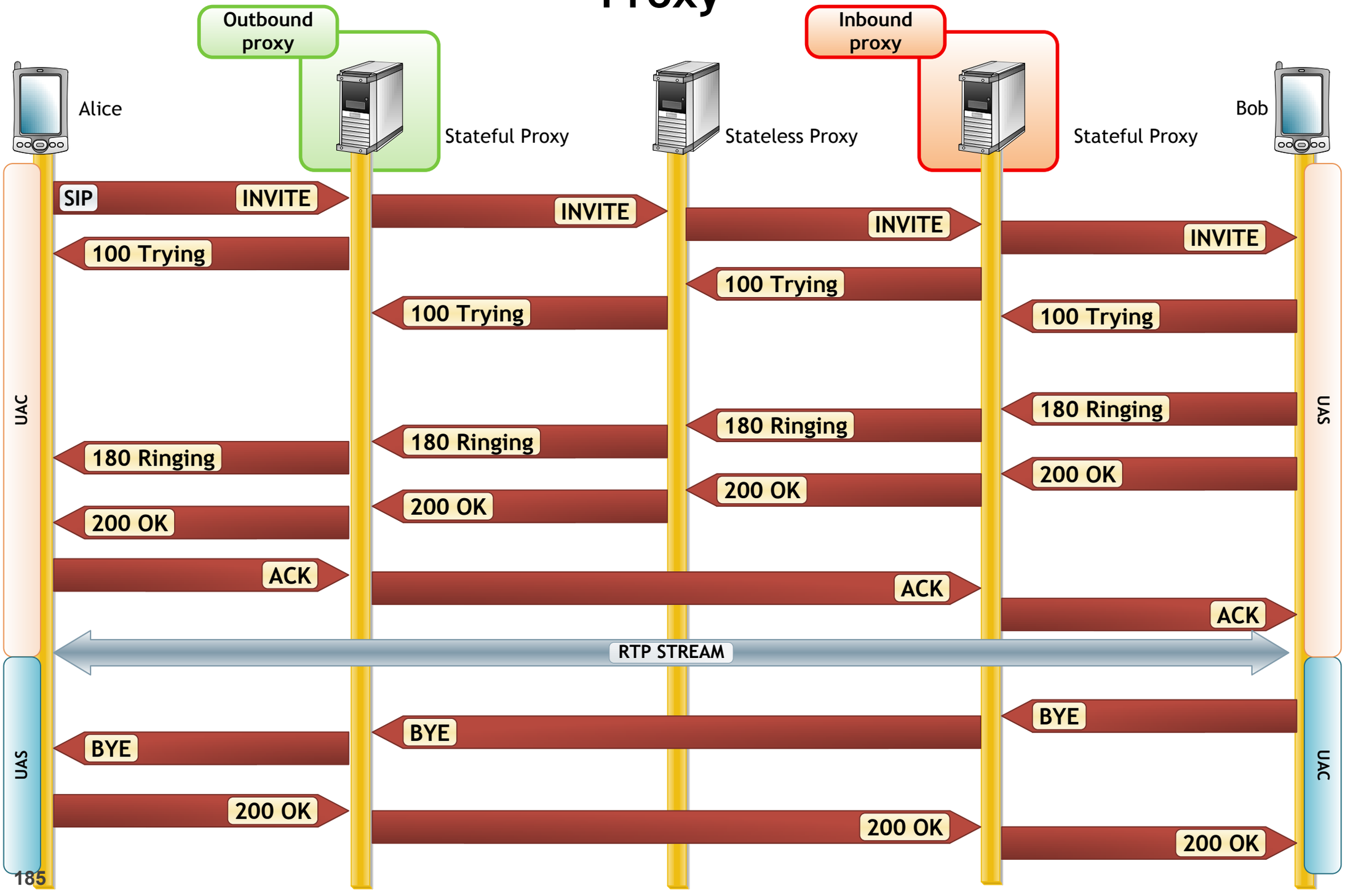
Stateful vs. stateless Proxy

- Stateful Proxy
  - A logical entity
  - Maintains the client and server transaction state machines defined by this specification during the processing of a request,
  - known as a transaction stateful proxy.
- Stateless Proxy
  - A logical entity
  - does not maintain the client or server transaction

Outbound vs. Inbound Proxy

- Outbound Proxy
  - A proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI.
  - Typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols.
- Inbound proxy
  - A proxy that handles Inbound requests for local domain. Could be the same server that acts as local domain Outbound Proxy

Proxy


Outbound proxy
Inbound proxy
Alice
Stateful Proxy
Stateless Proxy
Stateful Proxy
Bob
UAC
UAS
SIP
INVITE
INVITE
INVITE
INVITE
100 Trying
100 Trying
100 Trying
100 Trying
180 Ringing
180 Ringing
180 Ringing
180 Ringing
200 OK
200 OK
200 OK
200 OK
ACK
ACK
ACK
RTP STREAM
UAS
UAC
BYE
BYE
BYE
200 OK
200 OK
200 OK

Definitions

- Call Stateful Proxy

  A proxy is call stateful if it retains state for a dialog from the initiating INVITE to the terminating BYE request. A call stateful proxy is always transaction stateful, but the converse is not necessarily true.

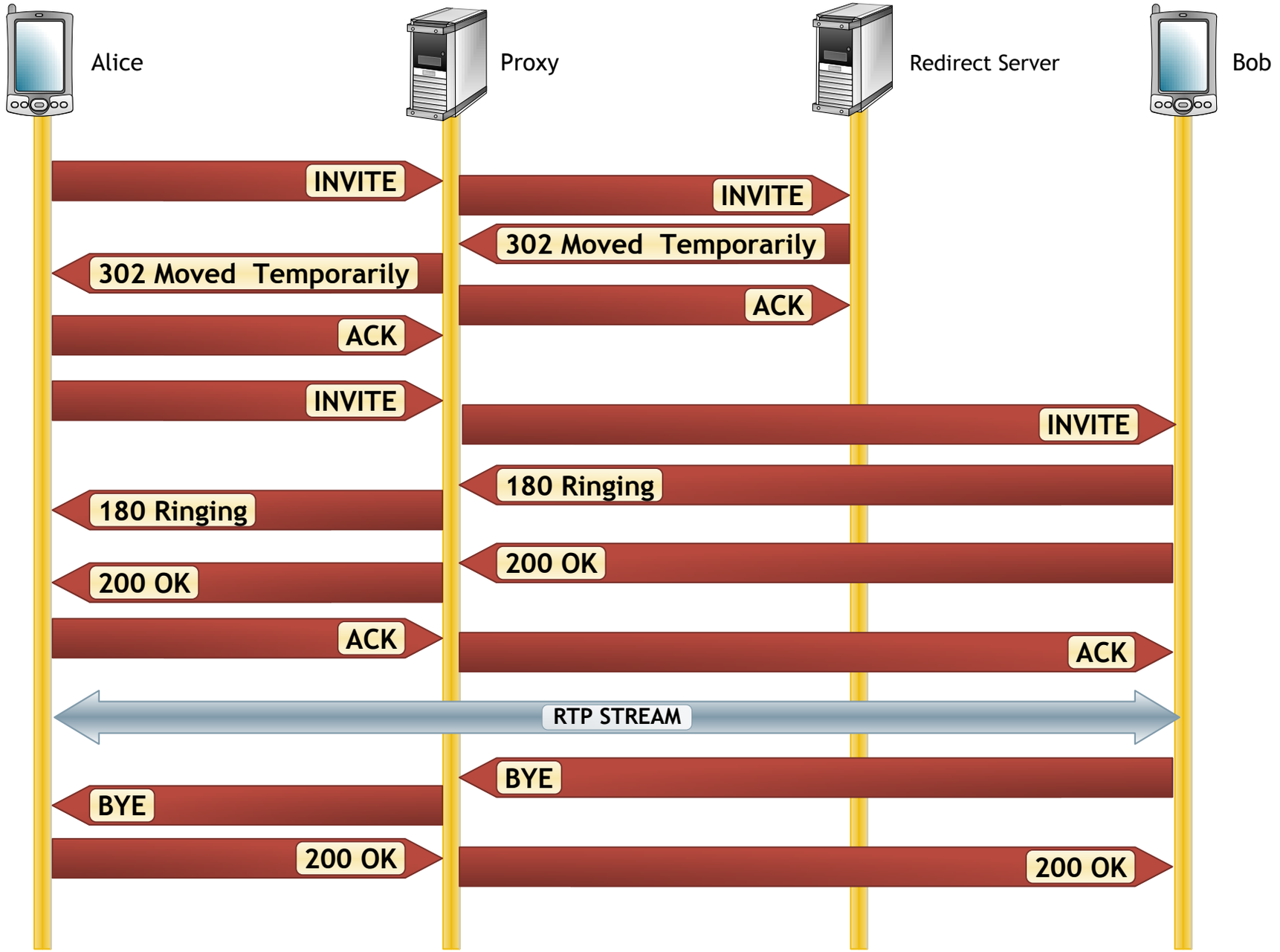
- Transaction Stateful Proxy

  The same as the stateful proxy.

Redirect Server

- A server that accepts a SIP request, maps the address into zero or more new addresses and returns these addresses to the client.
  - Unlike a proxy server, the redirect server does not initiate its own SIP request.
  - Unlike a user agent server, the redirect server does not accept or terminate calls.
- When returns a 3xx response to request, it populates the list of alternative locations into the Contact header field (it has access to the location service).
- Must ignore the features that are not understood.
- Advantage: reduce the processing load on servers.

SIP Call - redirect



Registration

Registration creates bindings in a location service for a particular domain that associates an address-of-record URI with one or more contact addresses.

Registration entails sending a REGISTER request to a special type of UAS known as REGISTRAR.

REGISTER request add, remove, and query bindings, but does not establish a dialog.

UAs MUST NOT send a new registration until they have received a final response from the registrar for the previous one or the previous REGISTER request has timed out.

Registrar

A registrar is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.

Registration

- Address-of-Record

An address-of-record (AOR) is a SIP or SIPS URI that points to a domain with a location service that can map the URI to another URI where the user might be available. Typically, the location service is populated through registrations. An AOR is frequently thought of as the "public address" of the user.

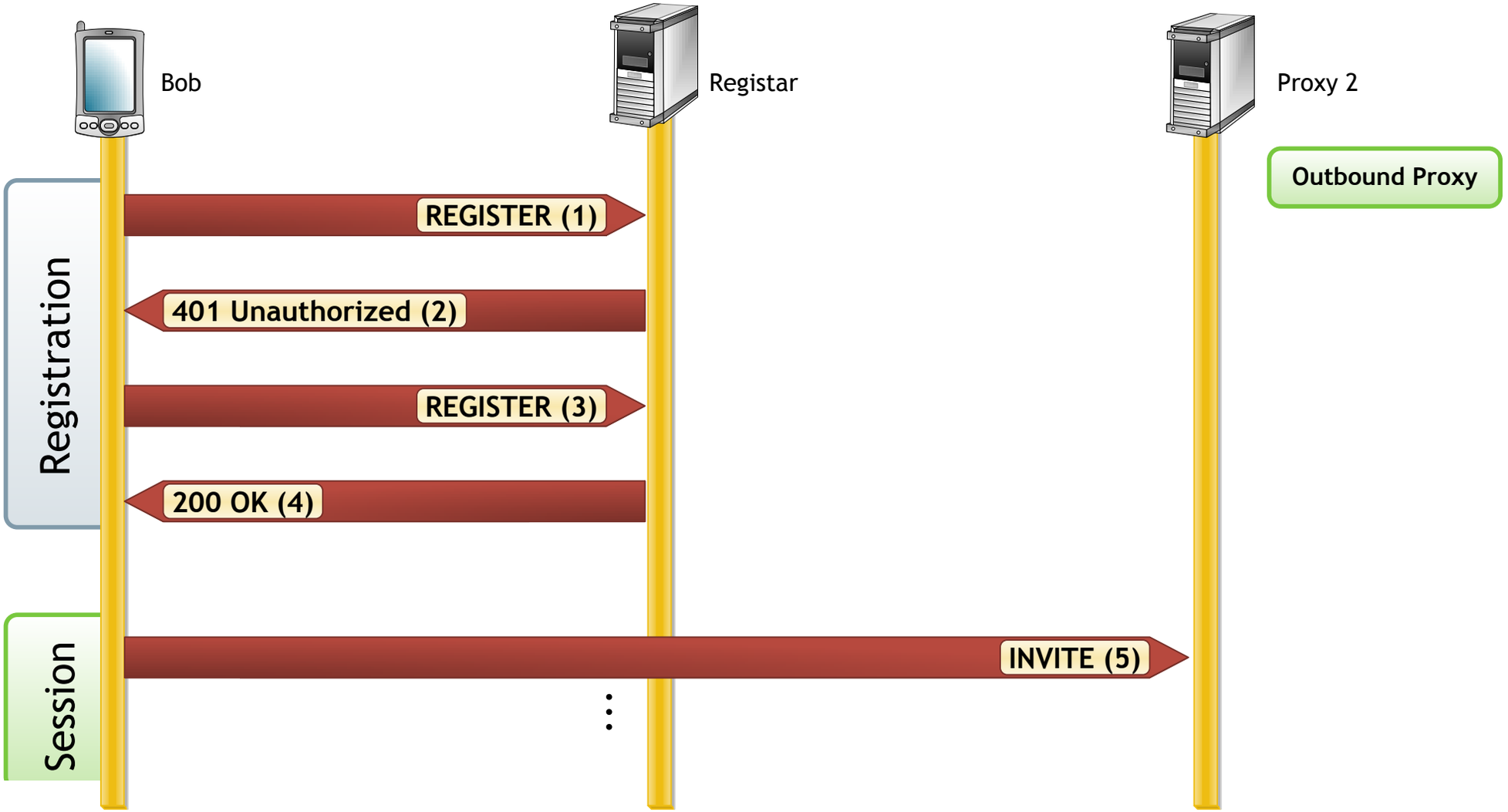
- Home domain

The domain providing service to a SIP user. Typically, this is the domain present in the URI in the address-of-record of a registration.

Registration

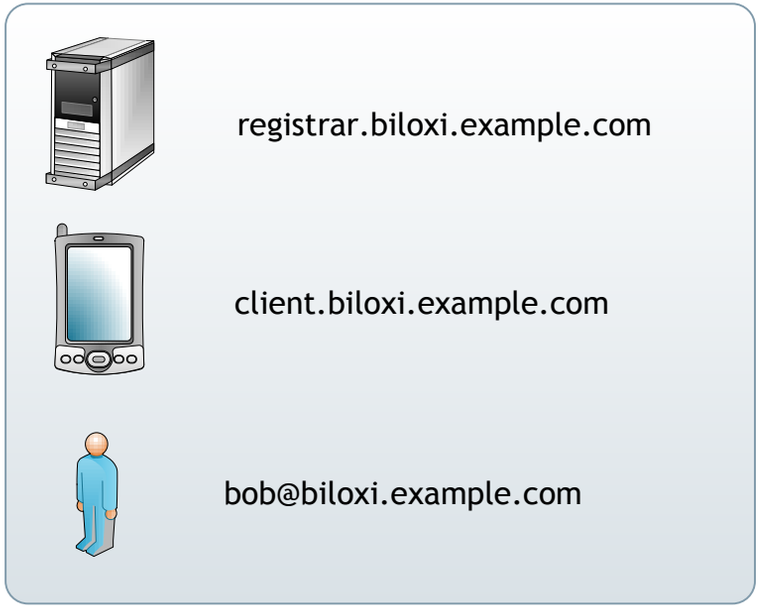
- Procedure of Registration
  - The first request from client to registrar includes the user's contact list (user informs server about her/his addresses; binding of URI to IP)
  - Registrar provides challenge to Bob
  - Bob enters to a terminal his valid user ID and password
  - Bob's SIP client encrypts the user information according to the challenge issued by the registrar
  - Client sends the response to the registrar
  - Registrar validates the user's credentials and registers the user in its contact list
  - Registrar returns the response to Bob's SIP client

Registration



REGISTER (1)

```
REGISTER sip:registrar.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP
client.biloxi.example.com:5060;branch=z9hG4bKnashds7
From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Contact: <sip:bob@client.biloxi.example.com>;expires=250000
Content-Length: 0
```



401 Unauthorized (2)

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP
client.biloxi.example.com:5060;branch=z9hG4bKnashds7
;received=192.0.2.201; rport= 13401 -- parameters received and
rport added
From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sip:bob@biloxi.example.com>;tag=1410948204 -- tag added
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="biloxi.example.com", qop="auth",
nonce=" ea9c8e88df84f1cec4341ae6cbe5a359 ",
opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0


-- WWW-Authenticate header added (carries fields
-- used for encryption)
```

registrar.biloxi.example.com

client.biloxi.example.com

bob@biloxi.example.com

REGISTER (3)

```
REGISTER sip:registrar.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP
client.biloxi.example.com:5060;branch=z9hG4bKnashd92
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=ja743ks76zlflH
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 2 REGISTER -- CSeq increased
Contact: <sip:bob@client.biloxi.example.com>;expires=250000
Authorization: Digest username="bob", realm="biloxi.example.com"
nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="",
uri="sip:registrar.biloxi.example.com", qop="auth",
nc=00000001, cnonce="0a4f113b",
response="dfe56131d1958046689d83306477ecc"
Content-Length: 0


-- Authorization header added
-- parameter response - MD5 hash
```

registrar.biloxi.example.com

client.biloxi.example.com

bob@biloxi.example.com

200 OK (4)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashd92
;received=192.0.2.201; rport=15981 -- received and rport added
From: Bob <sip:bob@biloxi.example.com>;tag=ja743ks76zlflH
To: Bob <sip:bob@biloxi.example.com>;tag=37GkEhwl6 -- tag added
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 2 REGISTER
Contact: <sip:bob@client.biloxi.example.com>;expires=600
-- expires: duration of registration is only 600 s
Service-Route:<sip:proxy2.biloxi.example.com;lr>
-- Service-Route added (address of outbound proxy)
-- used in Route header
Authentication-Info: nextnonce="47364c23432d2a5fb210812c"
-- Authentication-Info header field may be added
-- contains nonce which should be used during the next
authentication procedure
Content-Length: 0
```

registrar.biloxi.example.com

client.biloxi.example.com

bob@biloxi.example.com

INVITE (5)

```
INVITE sip:alice@atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP
client.biloxi.example.com:5060;branch=z9hG4bK74bf9;rport
Max-Forwards: 70
Route: <sip:proxy2.biloxi.example.com;lr>
From: Bob <sip:bob@biloxi.example.com>;tag=2kxc5d76hd
To: Alice <sip:alice@atlanta.example.com>
Call-ID: 384827629822018851l@biloxi.example.com
CSeq: 3 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 151
```

registrar.biloxi.example.com

client.biloxi.example.com

bob@biloxi.example.com

Header Fields

- Call-ID: All registrations from a UAC SHOULD use the same Call-ID header field value for registrations sent to a particular registrar. If the same client were to use different Call-ID values, a registrar could not detect whether a delayed REGISTER request might have arrived out of order.

- CSeq: Guarantees proper ordering of REGISTER requests. A UA MUST increment the CSeq value by one for each REGISTER request with the same Call-ID.

Header Fields

- Contact: REGISTER requests MAY contain a Contact header field with zero or more values containing address bindings and requested expiration time. If no Contact header field is present in a REGISTER request, the list of bindings is left unchanged.

- From: Contains the address-of-record of the person responsible for the registration. The value is the same as the To header field unless the request is a third-party registration.

- To: Contains the address-of-record whose registration is to be created, queried, or modified.

Expires

- Expires #1: The "expires" Contact parameter indicates how long the UA would like the binding to be valid. The value is a number indicating seconds.

If expires parameter is not provided, the value of the Expires header field can be used instead – the values MUST be taken as the requested expiration.

The registrar MAY choose an expiration less than the requested expiration interval.

Setting the registration interval protects the registrar against excessively frequent registration refreshes.

Expires cont.

- Expires header field

The Expires header field gives the relative time after which the message (or content) expires.

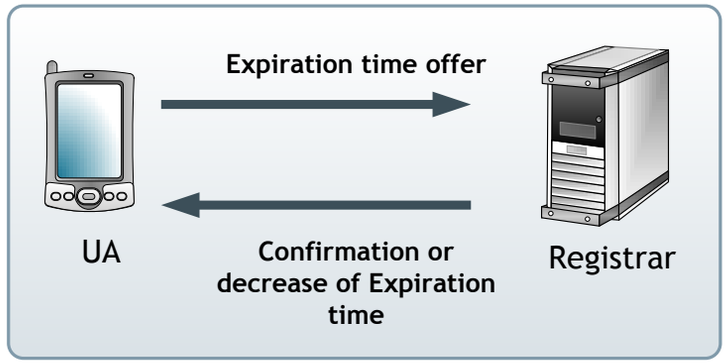
The value of this field is an integral number of seconds (in decimal) between 0 and (2^32)-1, measured from the receipt of the request.

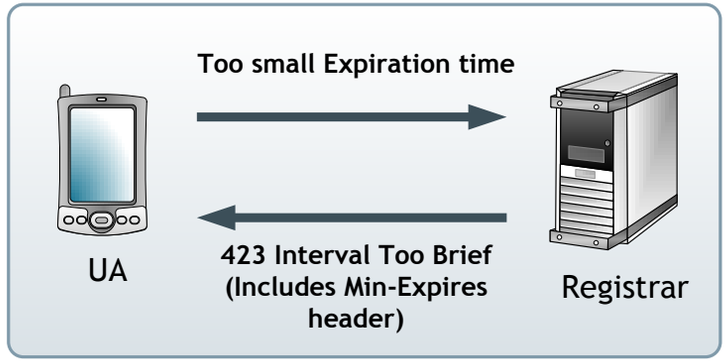
Example: Expires: 5

Expires cont.

- Expires #2: The registrar MAY choose an expiration less than the requested expiration interval (but not more).

If the requested expiration interval is greater than zero and smaller than a registrar-configured minimum, the registrar may reject the registration with a response 423 (Interval Too Brief). Response 423 must contain a Min-Expires header field that states the minimum expiration interval.





Min-Expires

- Min-Expires header field

The Min-Expires header field conveys the minimum refresh interval supported for soft-state elements managed by that server.

Header is used in response 423 (Interval Too Brief) -> server rejects the request because expiration interval in request is too small.

Example: Min-Expires: 60

Cancellation of Registration

- Cancellation of Registration

Registrations are soft state and expire unless refreshed, but can also be explicitly removed.

Registration can be cancelled by:

- Specifying an expiration interval of “0” that contact address

```
Contact: <sip:bob@client.biloxi.example.com>;expires=0
```

- Setting Contact header field value to “*” – applies to all registrations, but it MUST NOT be used unless the Expires header field is present with a value of “0”

```
Contact: * and Expires: 0
```

SIP Authentication

SIP provides challenge-based mechanism that is based on authentication in HTTP.

Methods ACK and CANCEL have special handling for authentication:

- ACK - servers must not attempt to challenge an ACK (is possible to use credentials from INVITE)
- CANCEL- request should be accepted by server if it comes from the same hop that sent the request being canceled

SIP Authentication

Authorization header contains:

- username
- realm
- nonce
- digest URI – URI of Request-Line (duplicated because proxies are allowed to change the Request-Line in transit)
- qop – optional because of compatibility with RFC 2069
- nc (nonce-count) – number of requests, must not be used if no qop is sent
- cnonce – client nonce must not be used if no qop has been received
- opaque
- response – MD5 hash

Digest Access Authentication

- Used for negotiating credentials
- Builds upon the basic authentication scheme, allowing user identity to be established without having to send a password in plaintext over the network
- Based on hash functions like MD5

Response value computing

1) The MD5 hash of the combined username, realm and password is calculated=>

```
HA1 = H ((username) ":" (realm) ":" (password))
```

2) The MD5 hash of the combined method and digest URI is calculated =>

```
HA2 = H ((method) ":" (Request-URI))
```

3) The MD5 hash of the combined A1 and A2 result, nonce, nc (nonce count), cnonce, qop (quality of protection) is calculated =>

```
response = H ((HA1) ":" (nonce) ":" (nc) ":" (cnonce) ":" (qop) ":" (HA2))
```

SIP Authentication

SIP servers must always send a “qop” parameter in WWW-Authenticate and Proxy-Authenticate header field values.

If a client receives a “qop” parameter in a challenge, it must also send the “qop” parameter in any resulting authorization header filed.

SIP Authentication

WWW-Authenticate and Proxy-Authenticate headers contain:

- realm – defines the protection domain
- qop – quality of protection the client has applied to message
- nonce – a server-specified data string which should be uniquely generated each time a 401 response is made
- opaque – a string of data, specified by the server
- stale – a flag indicating that the previous request from the client was rejected
- algorithm – algorithm used to backward digest and a checksum, MD5 used there

Authentication-Info header field

The Authentication-Info header field provides for mutual authentication with HTTP Digest. A UAS MAY include this header field in a 2xx response to a request that was successfully authenticated using based on Authorization header field. Syntax and semantics follow those specified in RFC 2617.

This header contains nonce which should be used during the next authentication procedure.

Example:

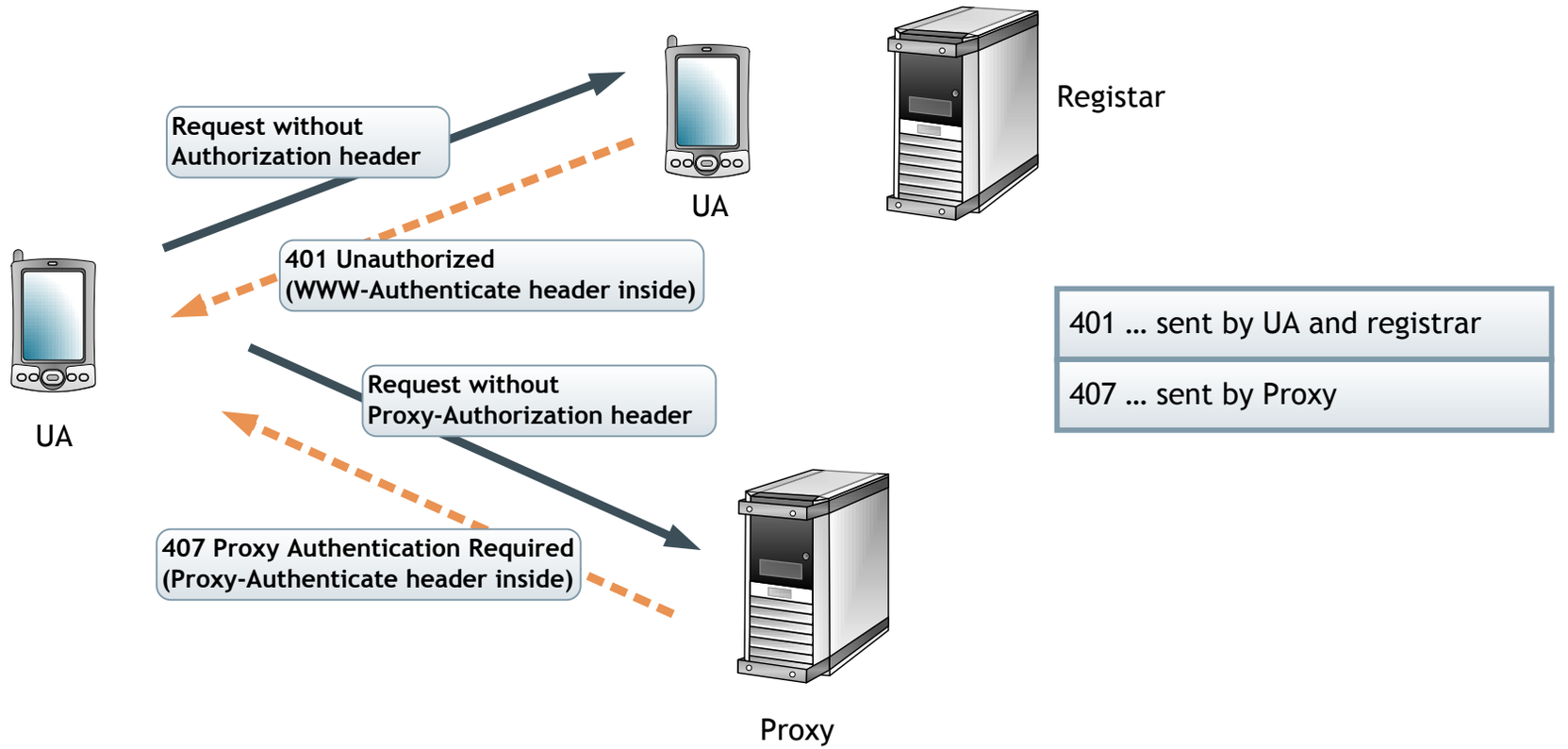
```
Authentication-Info: nextnonce="47364c23432d2a5fb210812c"
```

Location Service

A location service is used by a SIP redirect or proxy server to obtain information about a callee's possible location(s). It contains a list of bindings of address-of-record keys to zero or more contact addresses. The bindings can be created and removed in many ways; this specification defines a REGISTER method that updates the bindings.

SIP Authentication

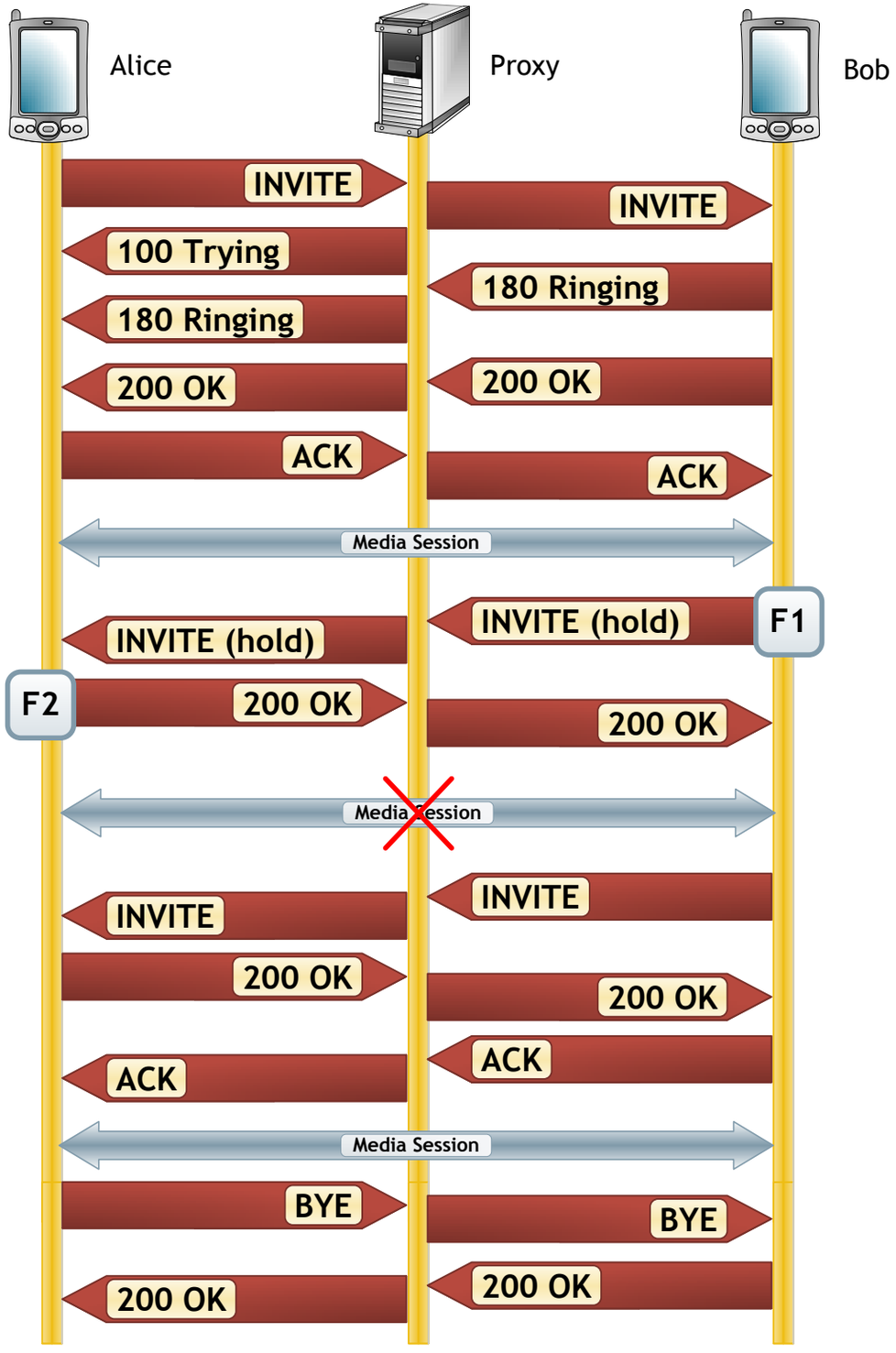
When a server receives a request from a client, the server MAY authenticate the originator before the request is processed. If no credentials are provided in the request, the server can challenge the originator to provide credentials by rejecting the request with a 401/407 status code.



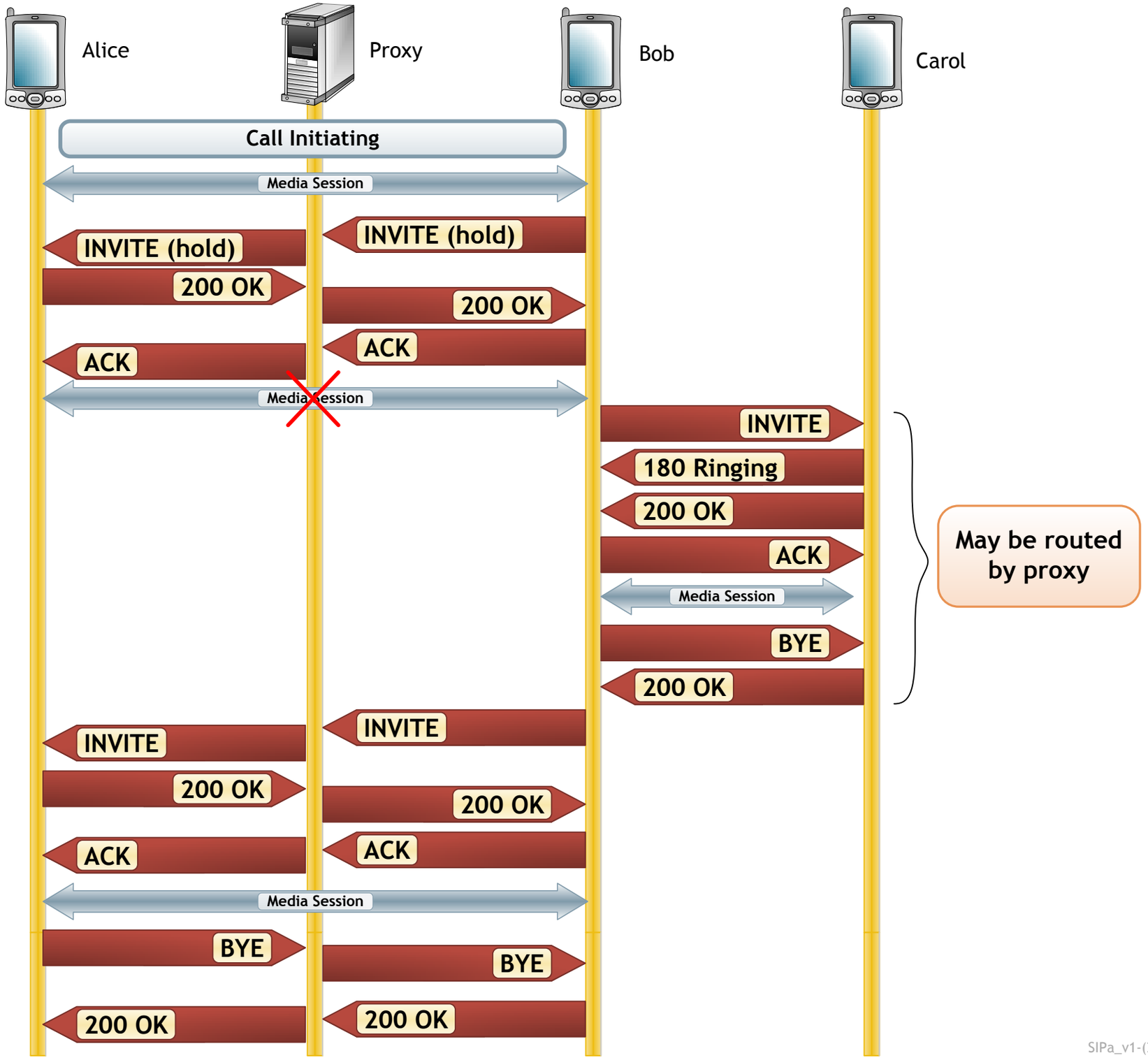
SIP services

- call hold
- call forward
- call transfer
- conference
- CCBS
- messaging
- presence

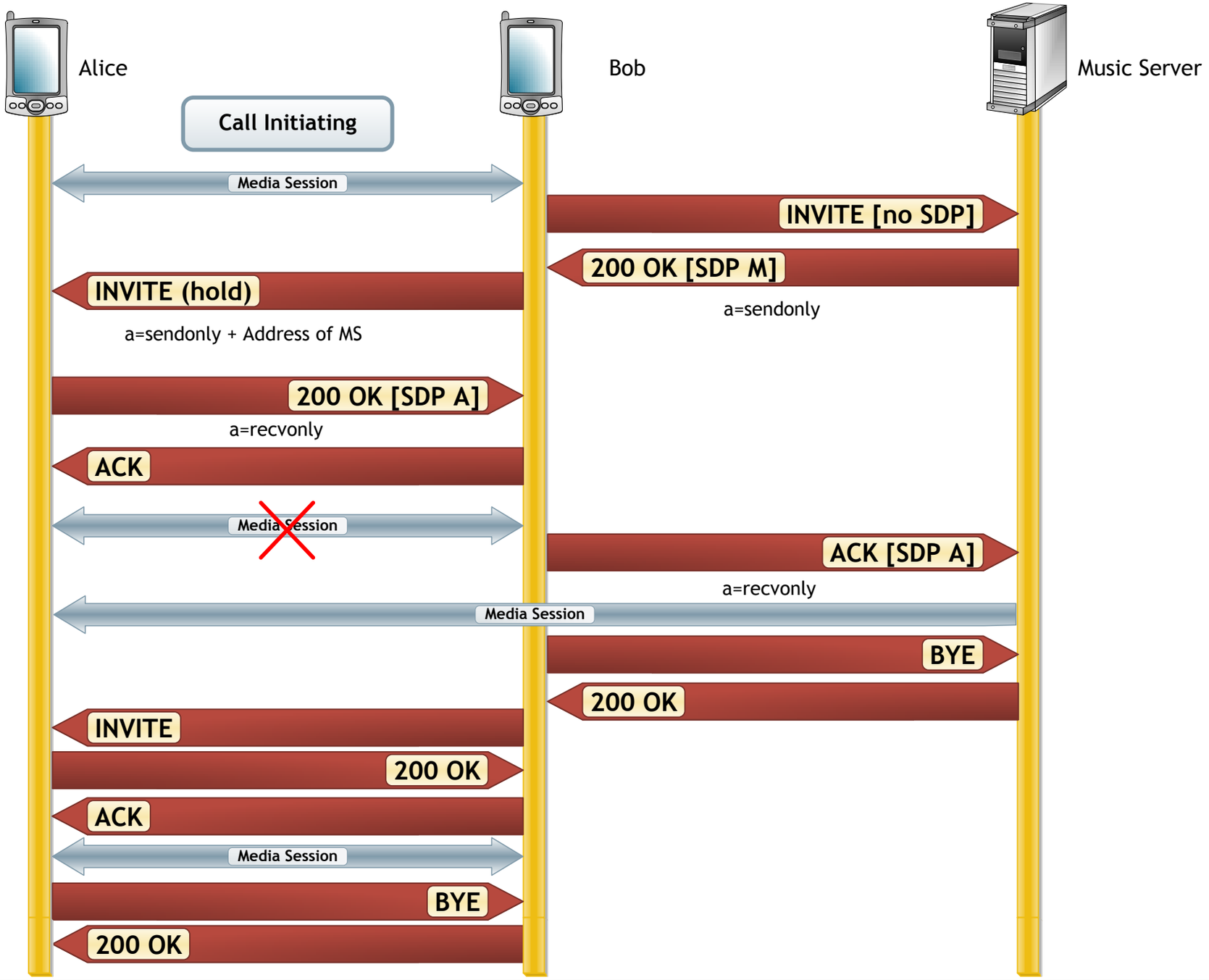
Call Hold


Alice
Proxy
Bob
INVITE
INVITE
100 Trying
180 Ringing
180 Ringing
200 OK
200 OK
ACK
ACK
Media Session
INVITE (hold)
INVITE (hold)
F1
F2
200 OK
200 OK
Media Session
INVITE
INVITE
200 OK
200 OK
ACK
ACK
Media Session
BYE
BYE
200 OK
200 OK

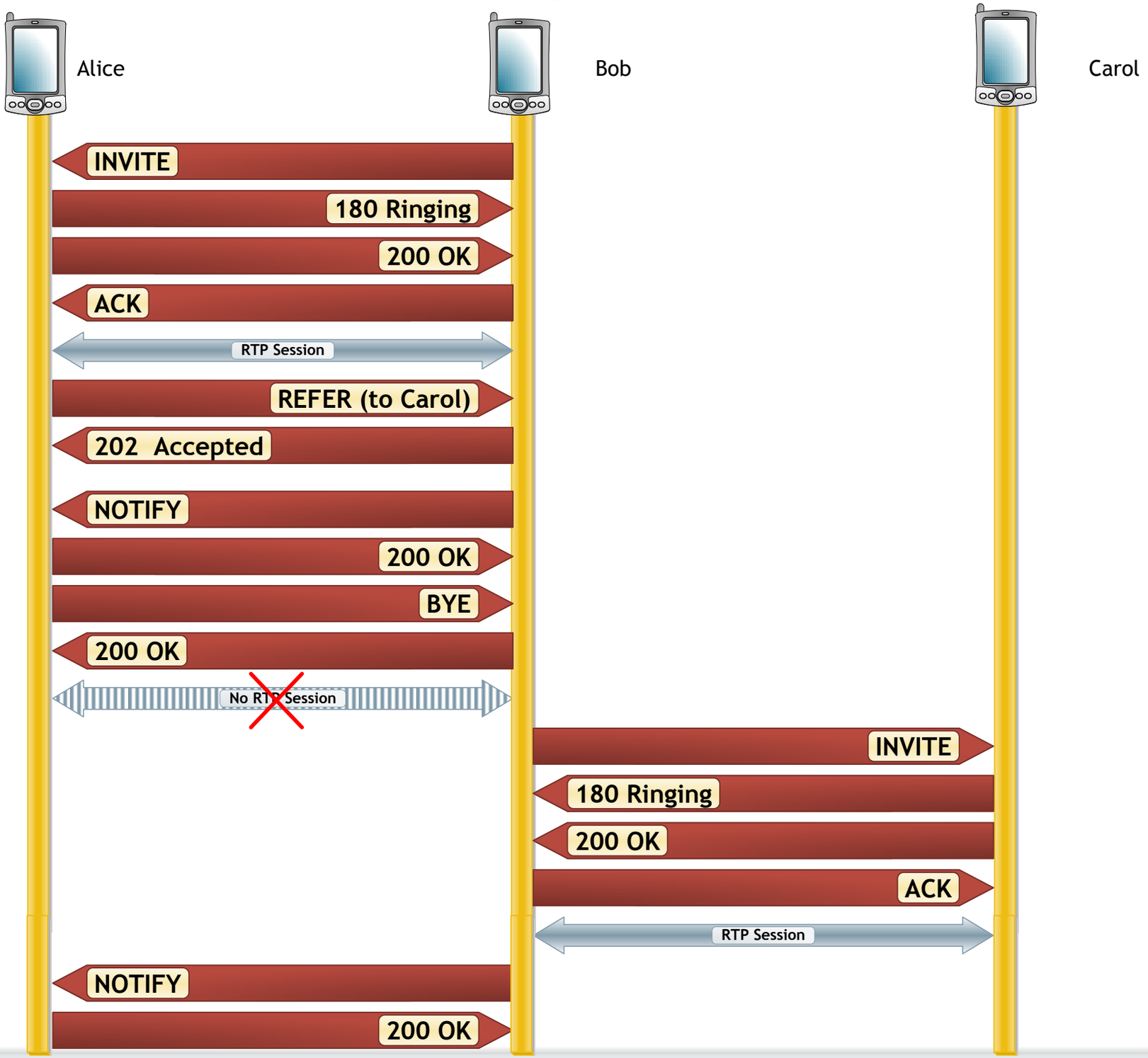
Consultation Hold



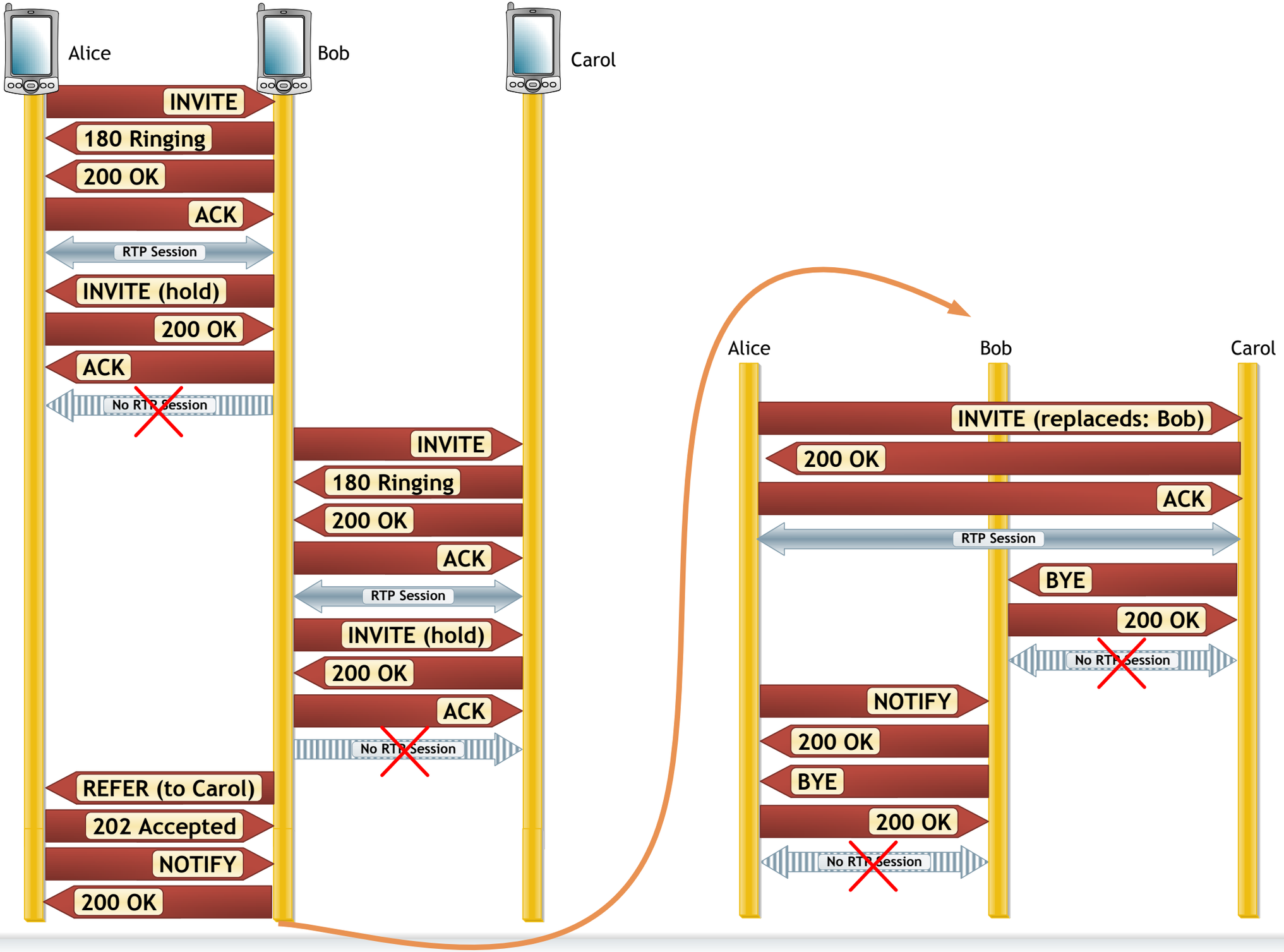
Music on hold



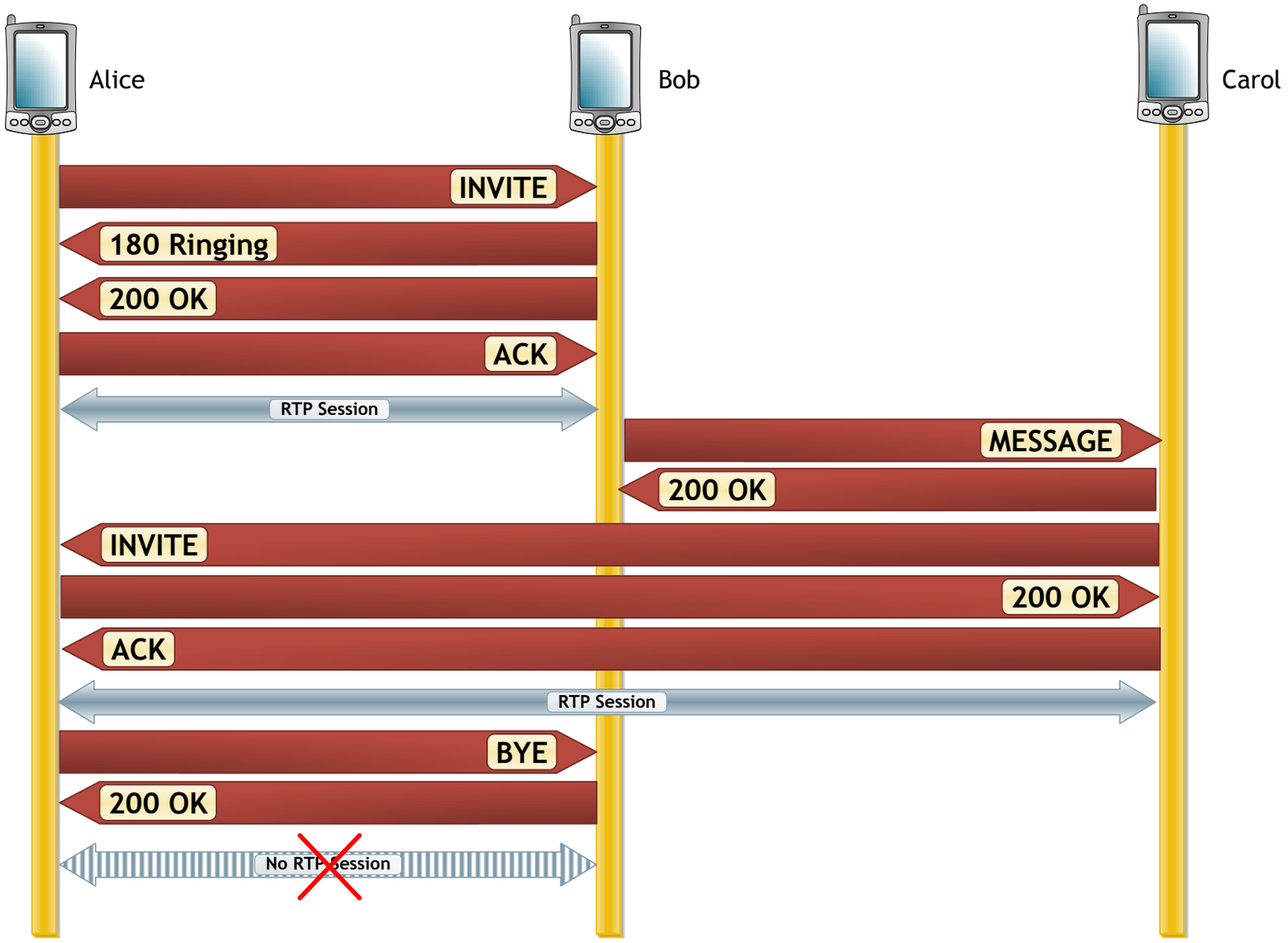
Transfer - Unattended



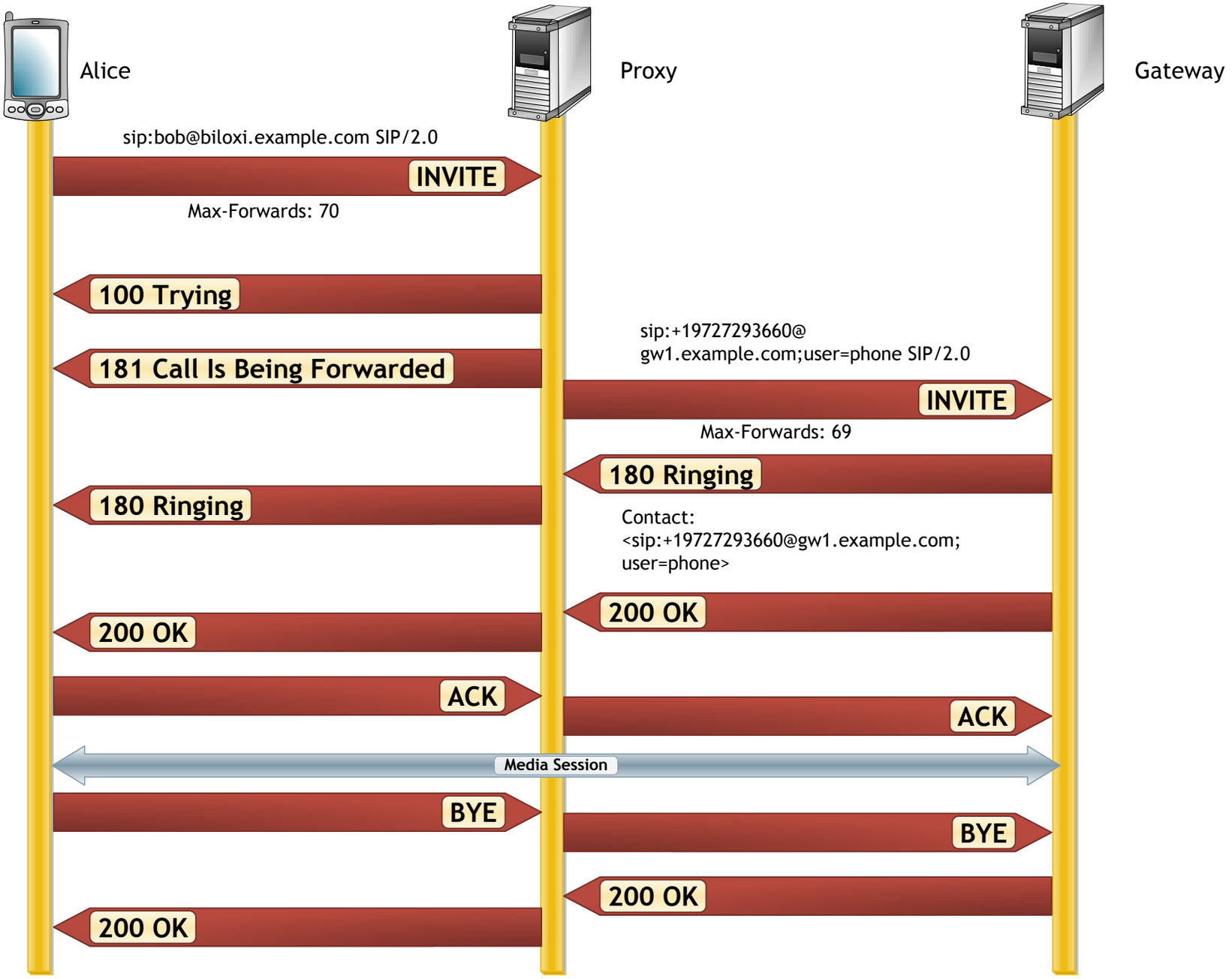
Transfer - Attended



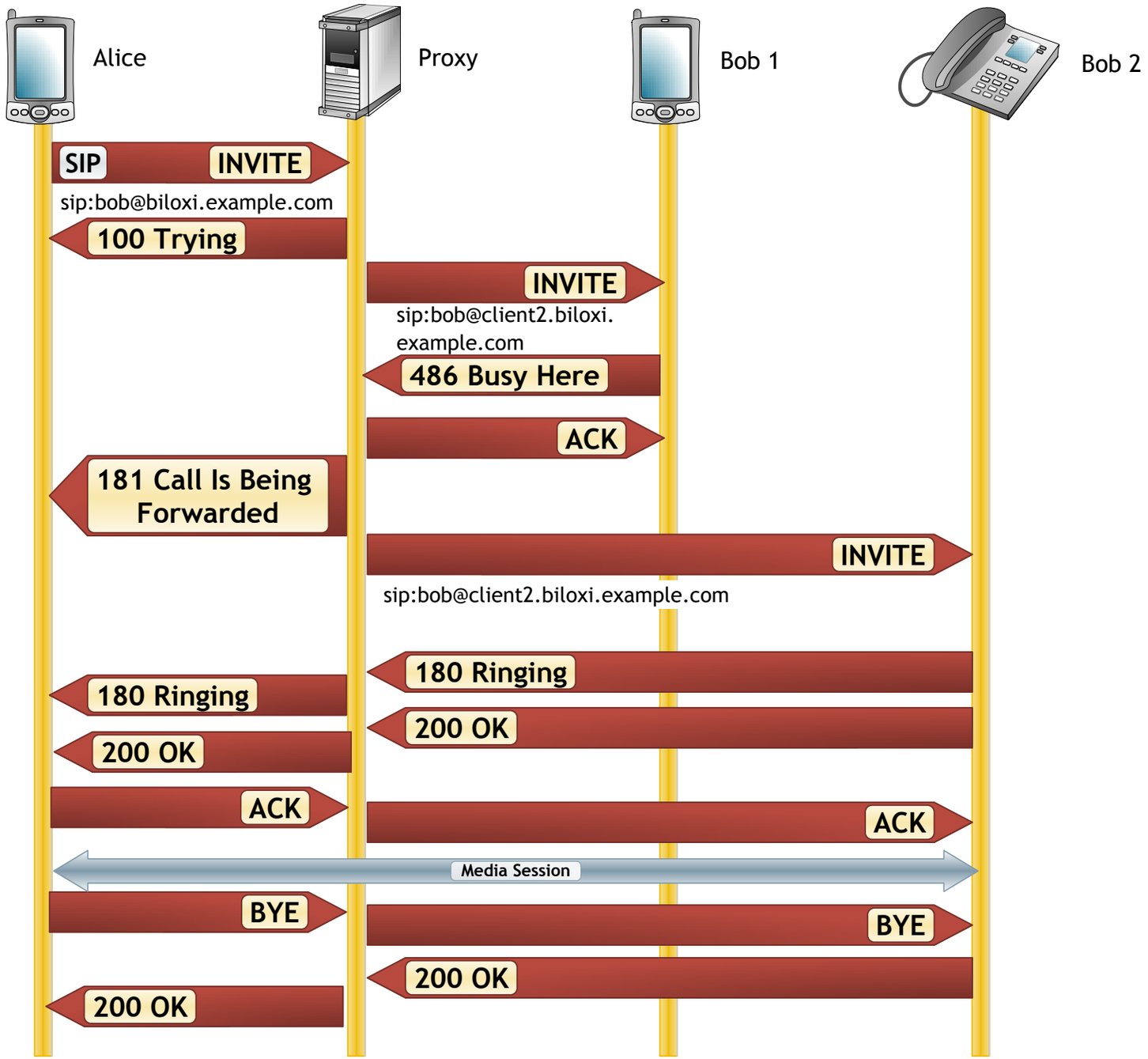
Transfer: Instant Messaging



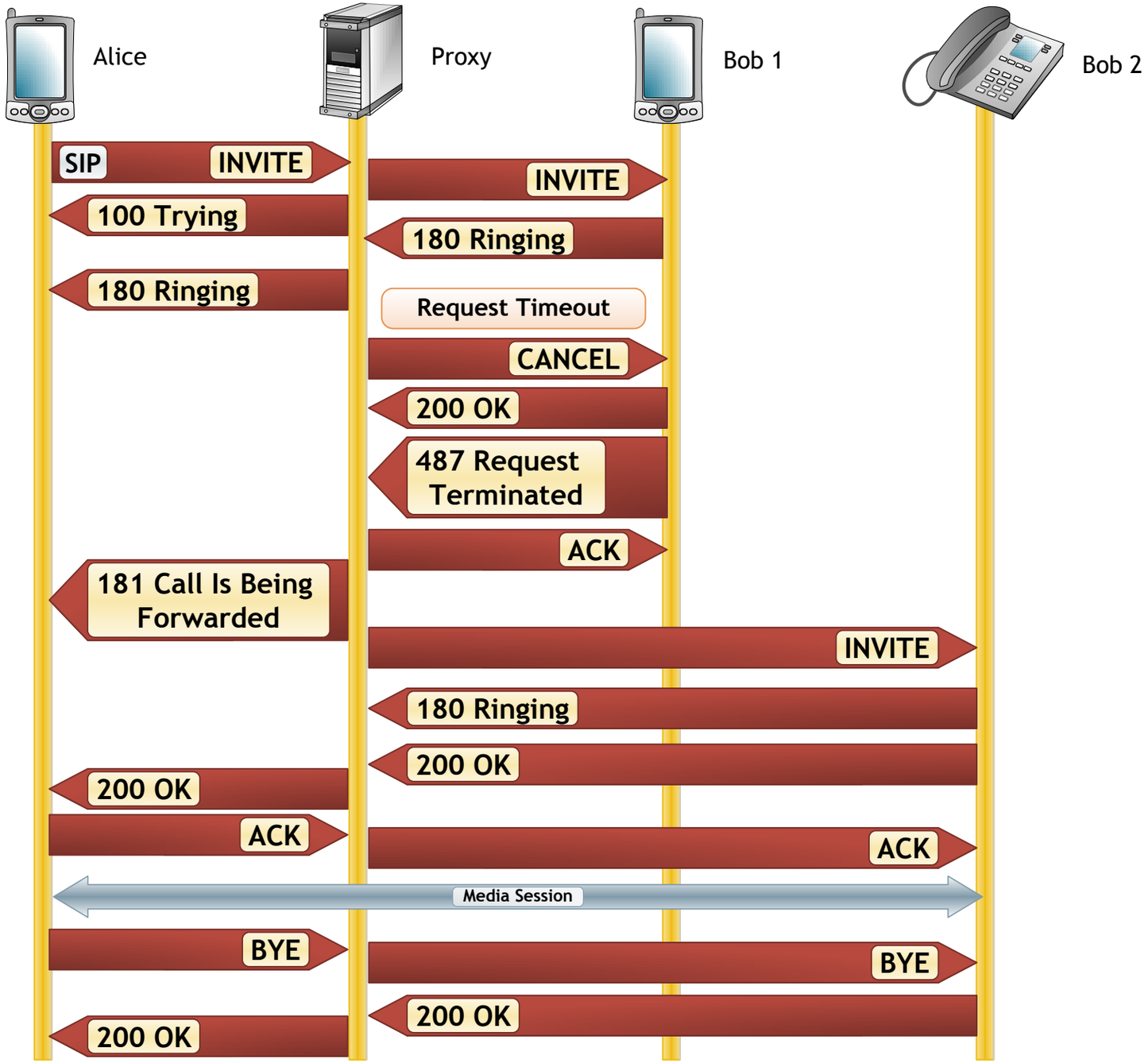
Call Forwarding Unconditional



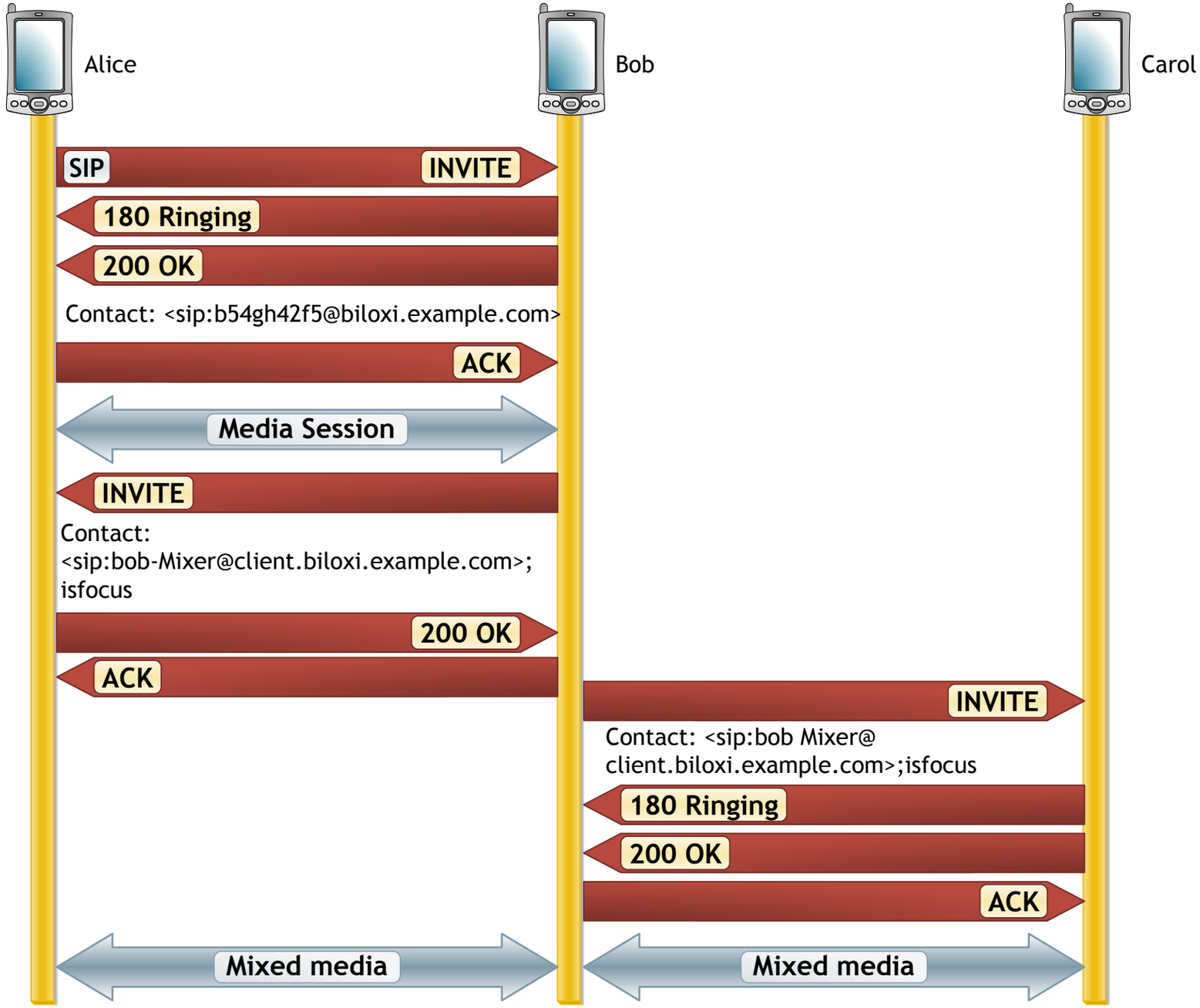
Call Forwarding Busy



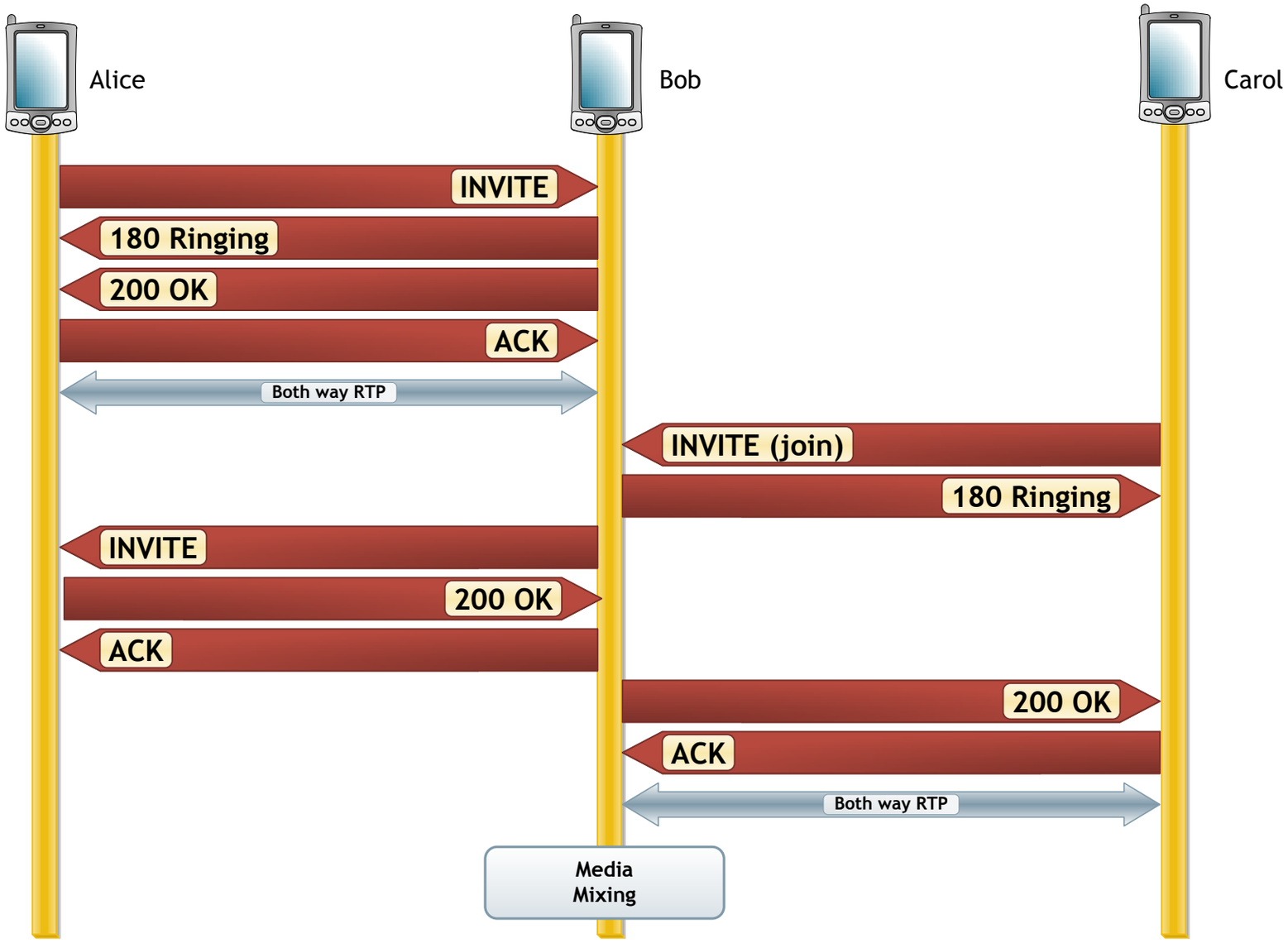
Call Forwarding - No Answer



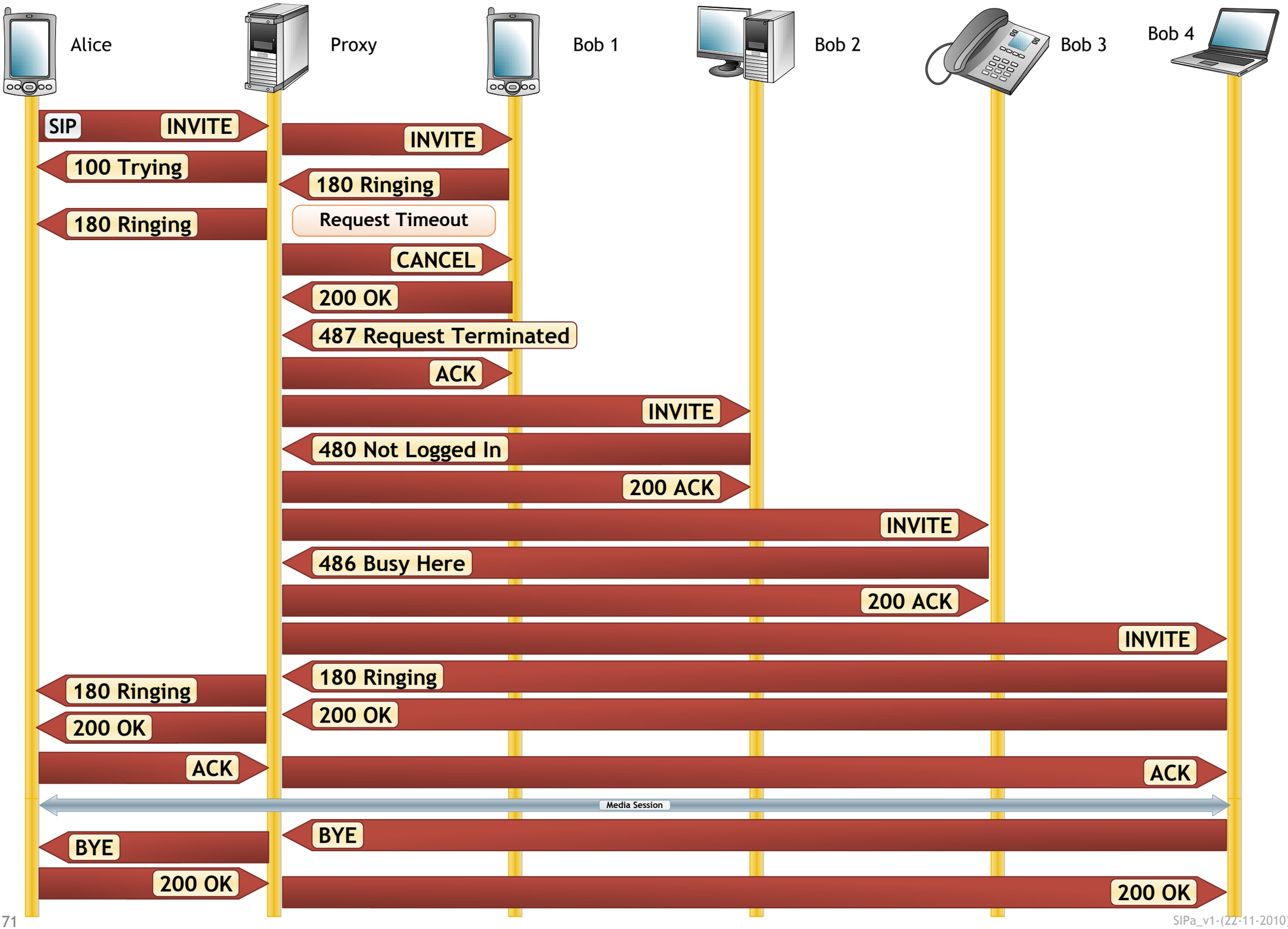
3way Conference - Third Party is Added



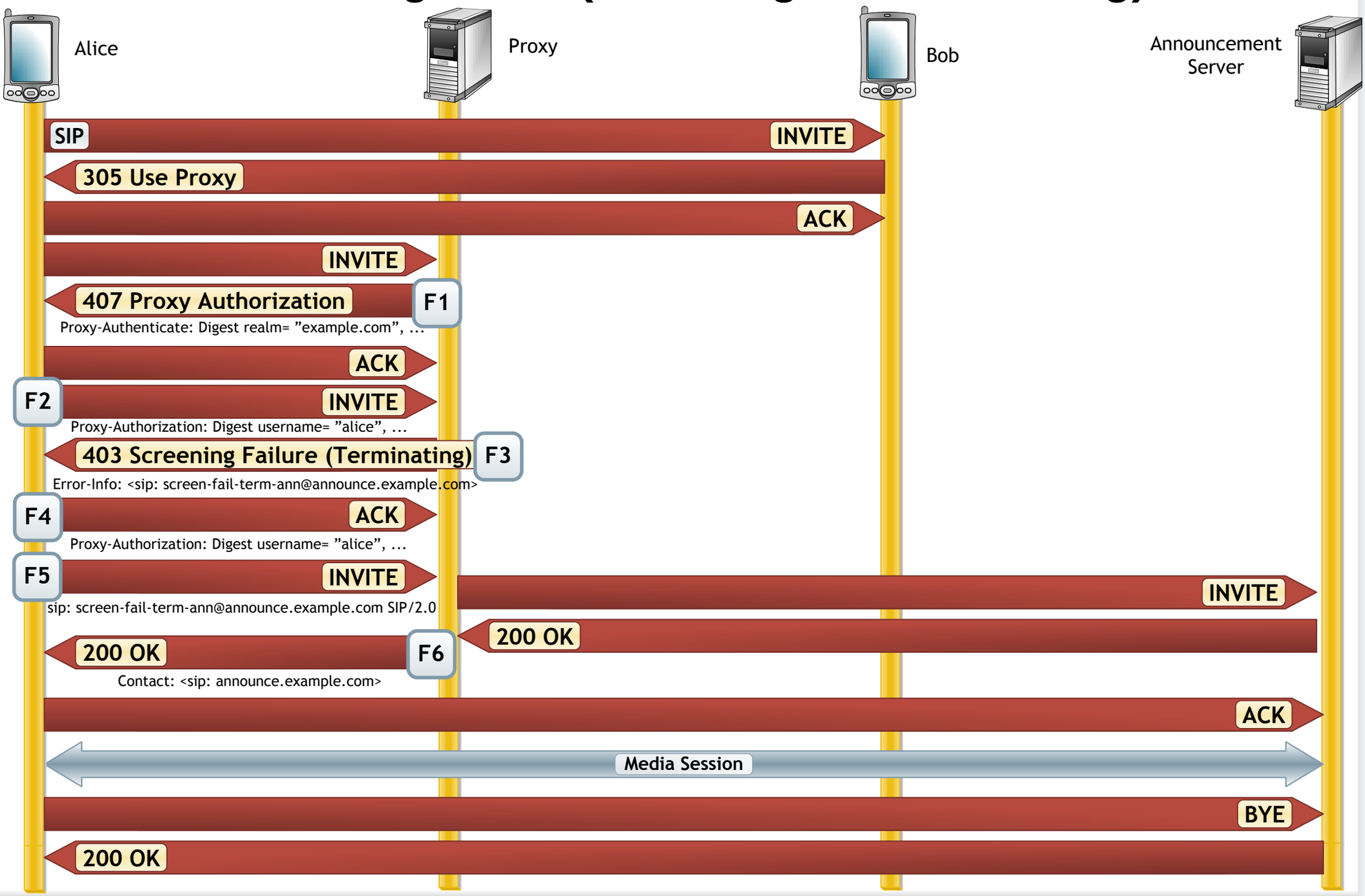
Three-Way Conference (Third Party Joins)



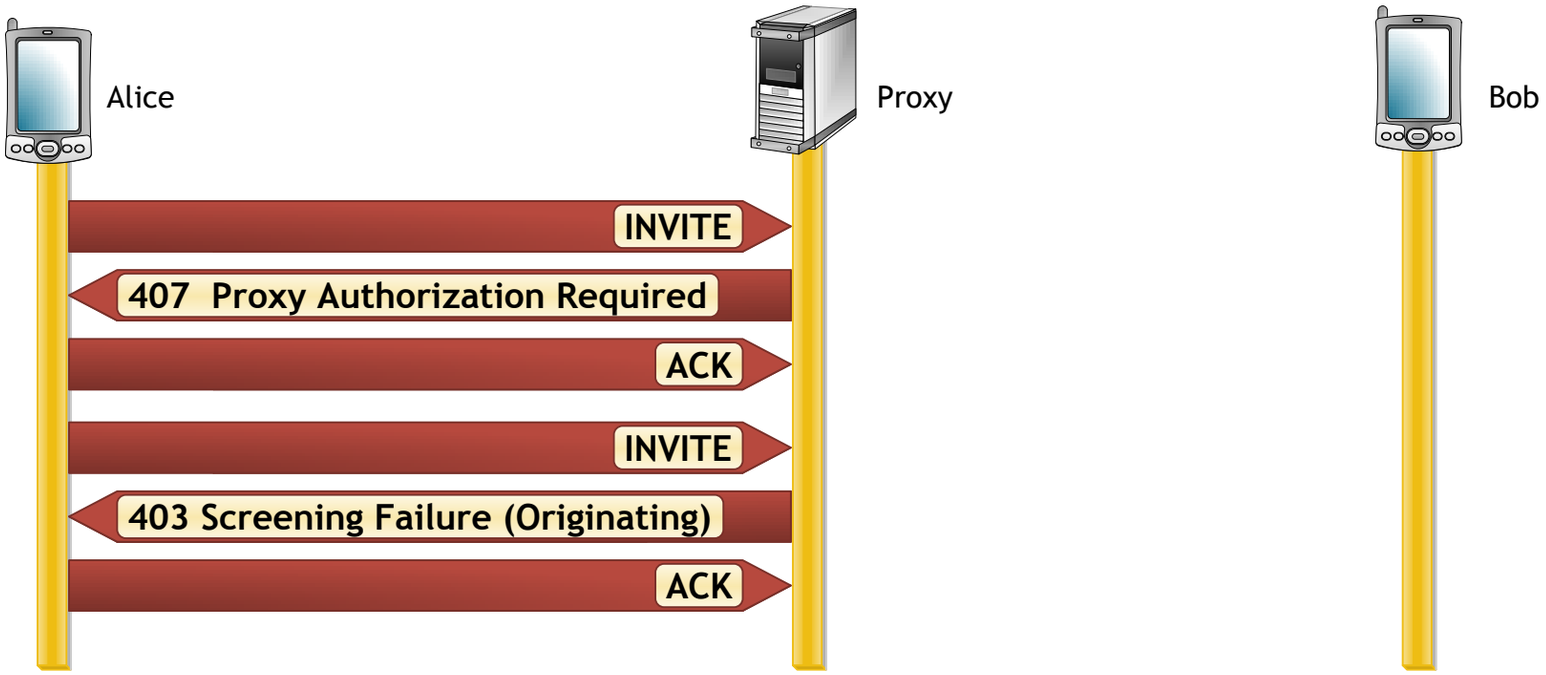
FindMe


Alice
Proxy
Bob 1
Bob 2
Bob 3
Bob 4
SIP INVITE
INVITE
100 Trying
180 Ringing
180 Ringing
Request Timeout
CANCEL
200 OK
487 Request Terminated
ACK
INVITE
480 Not Logged In
200 ACK
INVITE
486 Busy Here
200 ACK
INVITE
180 Ringing
180 Ringing
200 OK
200 OK
ACK
ACK
Media Session
BYE
BYE
200 OK
200 OK

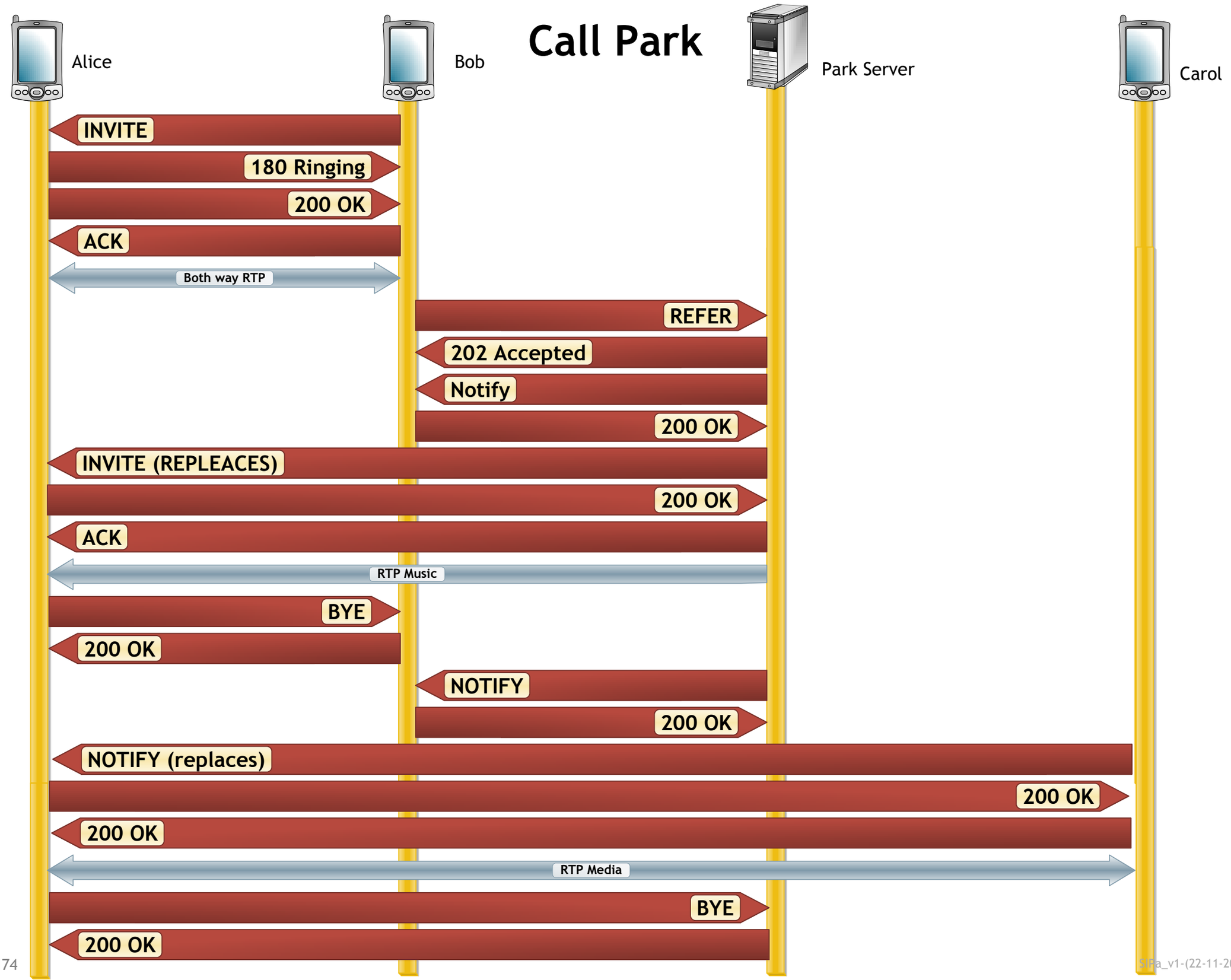
Call Management (Incoming Call Screening)



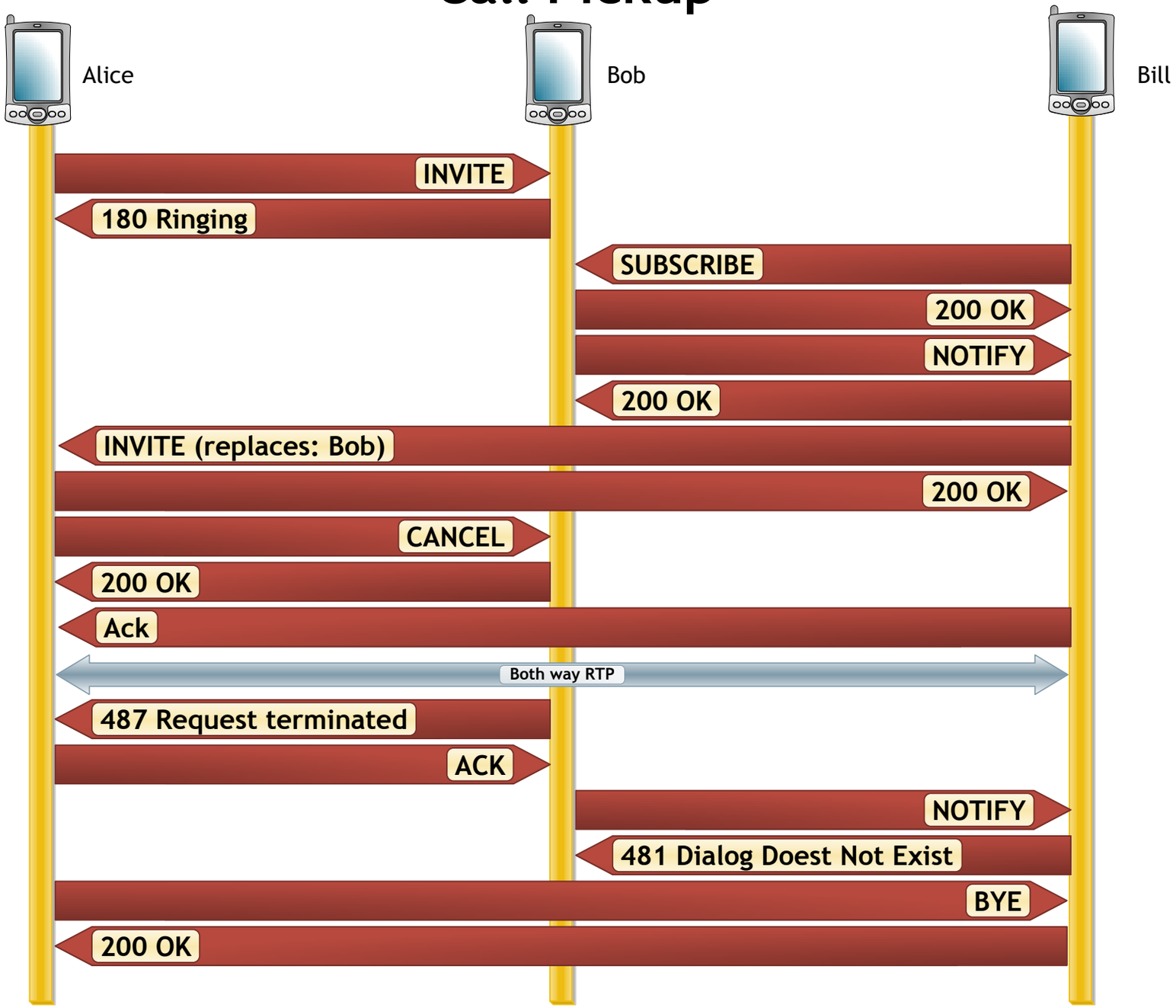
Call Management (Outgoing Call Screening)



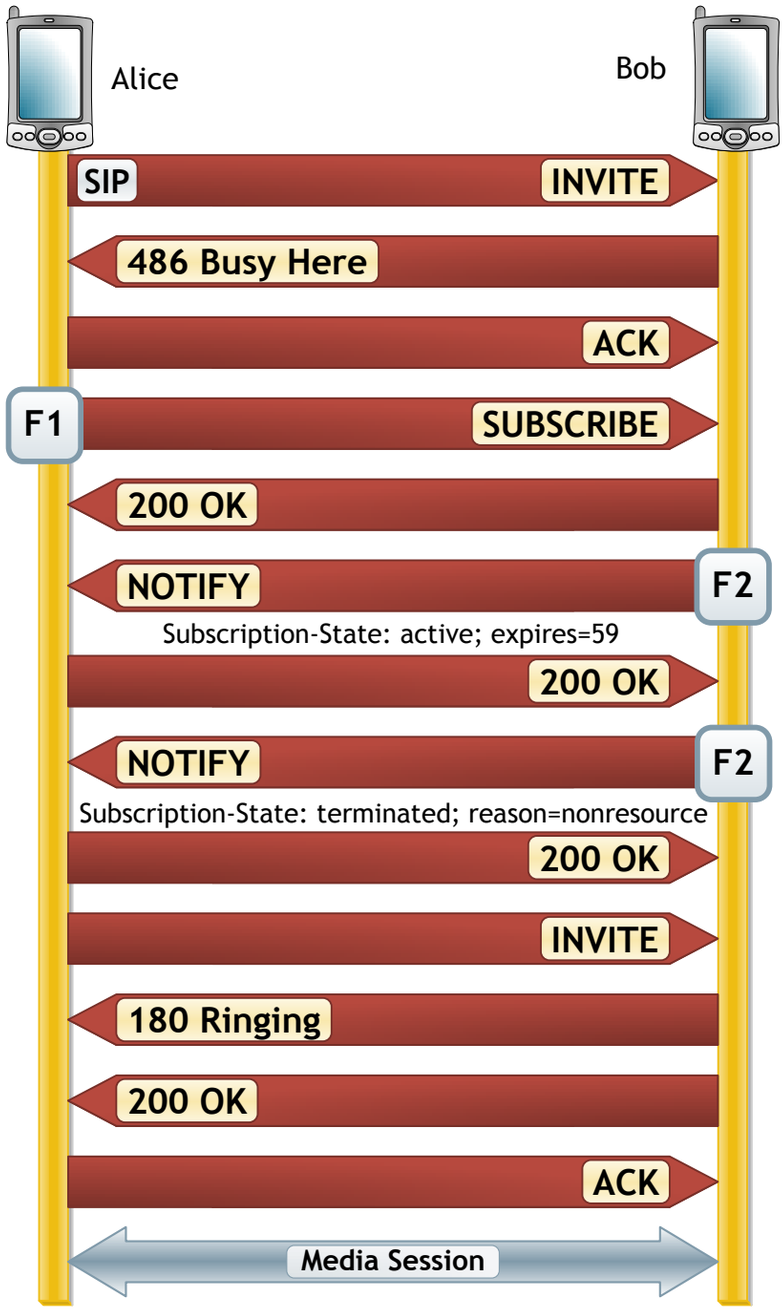
Call Park


Alice
Bob
Park Server
Carol
INVITE
180 Ringing
200 OK
ACK
Both way RTP
REFER
202 Accepted
Notify
200 OK
INVITE (REPLEACES)
200 OK
ACK
RTP Music
BYE
200 OK
NOTIFY
200 OK
NOTIFY (replaces)
200 OK
200 OK
RTP Media
BYE
200 OK

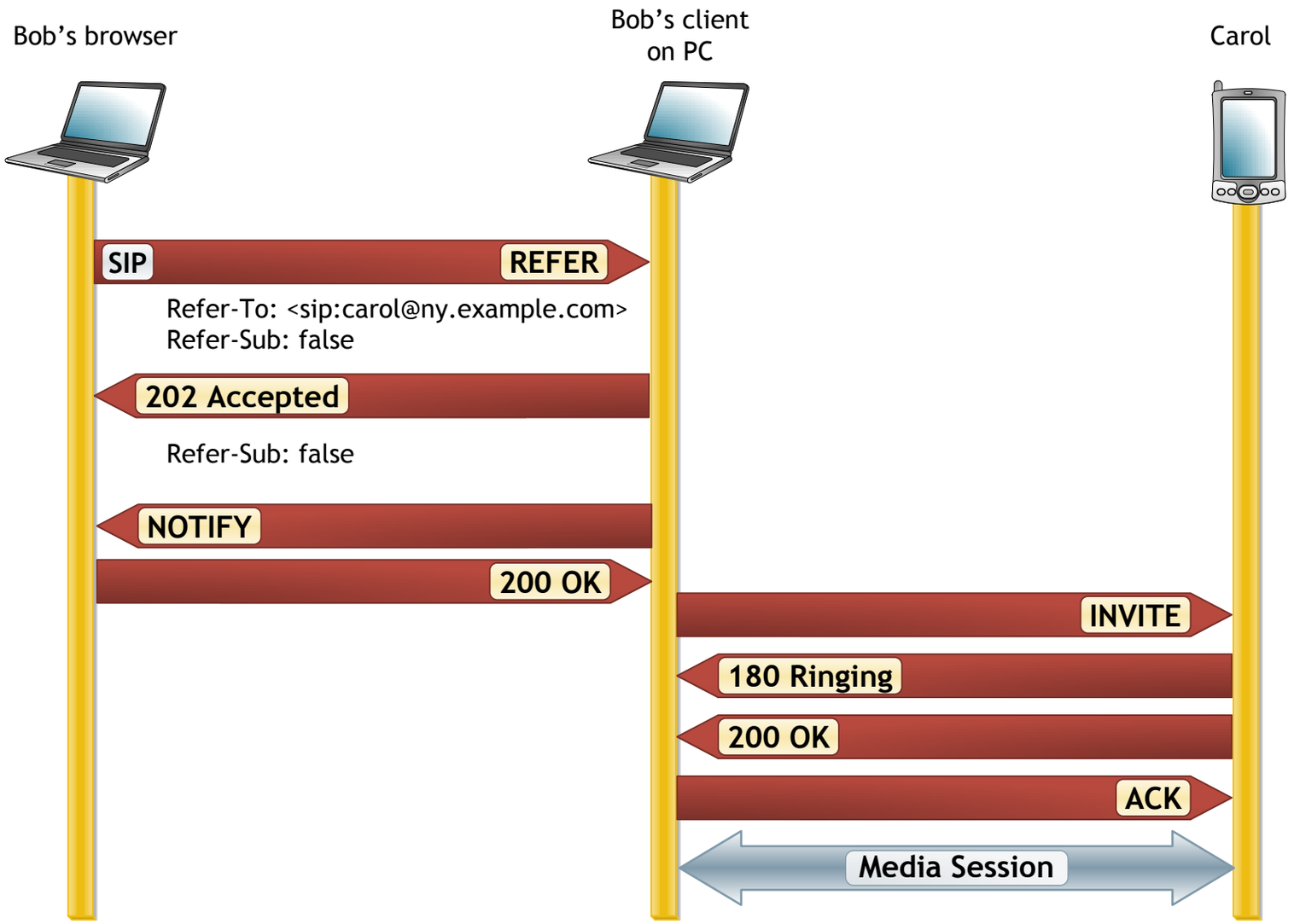
Call Pickup


Alice
Bob
Bill
INVITE
180 Ringing
SUBSCRIBE
200 OK
NOTIFY
200 OK
INVITE (replaces: Bob)
200 OK
CANCEL
200 OK
Ack
Both way RTP
487 Request terminated
ACK
NOTIFY
481 Dialog Doest Not Exist
BYE
200 OK

Automatic Redial



Click to Dial



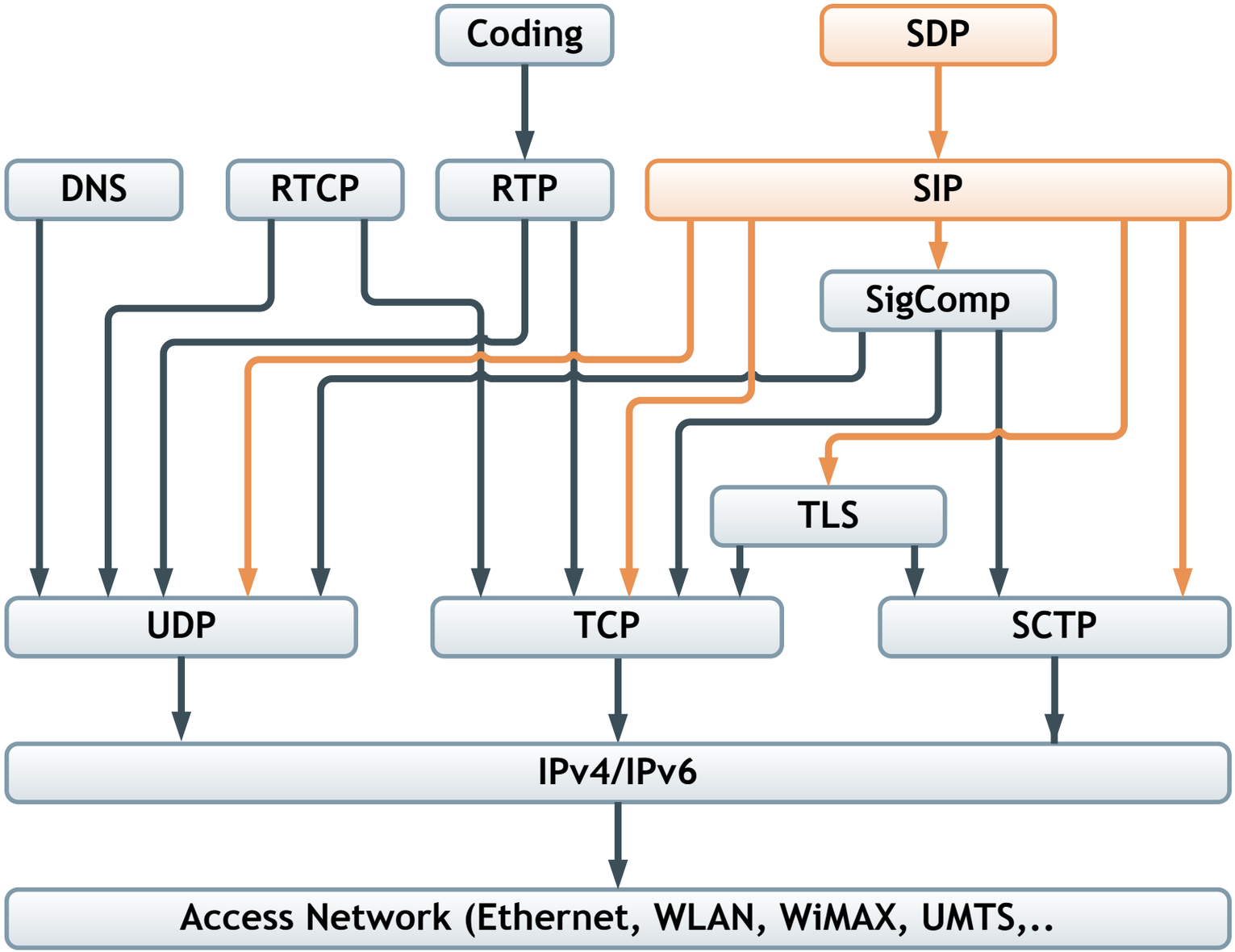
SIP and media transfer

- SDP and RTP are not directly related to SIP
- in fact SIP is pure control plane protocol and does not care about media much
- SDP content is transferred in SIP messages bodies

SDP

- Is used for description of the session to be established
- It contains enough information for the remote user to join the session
- In multimedia sessions this information includes the IP address and pot number wher the media needs to be sent and the codecs used to encode the media stream

Protocol Stack



SDP

- Contains the following informationabout the media session:
  - IP Address (IPv4 address or host name);
  - Port number (used by UDP or TCP for transport);
  - Media type (audio, video, interactive whiteboard, and so forth);
  - Media encoding scheme (PCM A-Law, MPEG II video, and so forth).
- Additional information
  - Subject of the session;
  - Start and stop times;
  - Contact information about the session.

SDP

- SDP uses text coding.
- An SDP message is composed of a series of lines, called fields, whose names are abbreviated by a single lower-case letter, and are in a required order to simplify parsing.
- SDP was not designed to be easily extensible.
- Parsing rules are strict.

SDP message format

- The general form of a SDP message:

  x = parameter1 parameter2 ... parameterN

- Example SDP message

```
v=0
o=johnston 2890844526 2890844526 IN
IP4 43.32.1.5
s=SIP Tutorial
i=This broadcast will cover this new
IETF protocol
u=http://www.digitalari.com/sip
e=Alan Johnston alan@mci.com
p=+1-314-555-3333 (Daytime Only)
c=IN IP4 225.45.3.56/236
b=CT:144
t=2877631875 2879633673
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 23422 RTP/AVP 31
```

SDP fields

Field	Name	Mandatory/Optional
v=	Protocol version number	m
o=	Owner/creator and session identifier	m
s=	Session name	m
i=	Session information	o
u=	Uniform Resource Identifer	o
e=	Email address	o
p=	Phone number	o
c=	Connection information	m
b=	Bandwidth information	o
t=	Time session starts and stops	m
r=	Repeat times	o
z=	Time zone corrections	o
k=	Encryption key	o
a=	Attribute lines	o
m=	Media information	o
a=	Media attributes	o

SDP used in SIP

- The calling party lists the media capabilities that they are willing to receive in SDP in either an INVITE or in an ACK
- The called party lists their media capabilities in the 200OK response to the INVITE
- More generally, offers or answers may be in INVITEs, PRACKs, UPDATEs or in reliably sent 18x or 200 responses to these methods
- A typical SIP use of SDP includes:
  - Version,
  - Origin,
  - Subject,
  - Time,
  - Connection,
  - One or more media and the same order

SDP used in SIP

- Origin, subject, and time fields are not used by SIP but are included for compatibility.
- SIP uses the connection, media, and attribute fields to set up sessions between user agents. Because the type of media session and codec to be used are part of the connection negotiation, SIP can use SDP to specify multiple alternative media types and to selectively accept or decline those media types.
- When multiple media codecs are listed, the caller and called party's media fields must be aligned that is, there must be the same number, and they must be listed in same order.

SDP Field

- v= field contains the SDP version number. Because the current version of SDP is 0, a valid SDP message will always begin with v=0.
- o= field contains information about the originator of the session and session identifiers. This field is used to uniquely identify the session.

```
o=username session-id version network-type address-type address
```

  - The username parameter contains the originator's login or host or if none.
  - The session-id parameter is a Network Time

SDP Field

- s= field contains a name for the session. It can contain any non-zero number of characters. The optional i= field contains information about the session. It can contain any number of characters.
- u= field contains a uniform resource indicator (URI) with more information about the session.

SDP Field

- e= field contains an e-mail address of the host of the session. If a display name is used, the e-mail address is enclosed in <>.
- The optional p= field contains a phone number. The phone number should be given in globalized format.

SDP Field

- c= field contains information about the media connection.

```
c=network-type address-type connection-address
```

- network-type parameter is defined as IN for the Internet.
- address type is defined as IP4 for IPv4 addresses, IP6 for IPv6 addresses.
- connection-address is the IP address that will be sending the media packets, which could be either multicast or unicast.
- If multicast, the connection-address field

- b= field contains information about the bandwidth required.

```
b=modifier:bandwidth-value
```

- The modifier is either CT for conference total or AS for application specific.
  - CT is used for multicast session to specify the total bandwidth that can be used by all participants in the session.
  - AS is used to specify the bandwidth of a single site. The bandwidth-value parameter is the specified number of kilobytes per second.

Time, Repeat Times, and Time Zones

- t= field contains the start time and stop time of the session.

```
t=start-time stop-time
```

- The times are specified using NTP timestamps. For a scheduled session, a stop-time of zero indicates that the session goes on indefinitely. A start-time and stop-time of zero for a scheduled session indicates that it is permanent.
- optional r= field contains information about the repeat times that can be specified in either in NTP or in days (d), hours (h), or minutes (m).
- optional z= field contains information about the time zone offsets. This field is used if a reoccurring session spans a change from daylight-savings to standard time, or vice versa.

Encryption Keys

- optional k= field contains the encryption key to be used for the media session.

```
k=method:encryption-key
```

- The method parameter can be clear, base64, uri, or prompt. If the method is prompt, the key will not be carried in SDP; instead, the user will be prompted as they join the encrypted session. Otherwise, the key is sent in the encryption-key parameter.

Media Announcements

- The optional m= field contains information about the type of media session.

```
m=media port transport format-list
```

- The media parameter Is
  - audio, video, application, data, telephone-event, or control.
- The port parameter contains the port number.
- The transport parameter contains the transport protocol, which is either RTP/AVP or UDP. (RTP/AVP stands for Real-time Transport Protocol)
- The format-list contains more information about the media. Usually, it contains media payload types defined in RTP audio video profiles. More than one media payload type can be listed, allowing multiple alternative codecs for the media session. For example, the following media field lists three codecs:

```
m=audio 49430 RTP/AVP 0 6 8 99
```

Attributes

- optional a= field contains attributes of the preceding media session.
- can be used to extend SDP to provide more information about the media.
- If not fully understood by a SDP user, the attribute field can be ignored.
- There can be one or more attribute fields for each media payload type listed in the media field.
  RTP/AVP example attribute fields :
  a=rtpmap:0 PCMU/8000
  a=rtpmap:6 DVI4/16000
  a=rtpmap:8 PCMA/8000
  a=rtpmap:99 iLBC

SDP Attribute Values

Attribute	Name
a=rtpmap:	RTP/AVP list
a=cat:	Category of the session
a=keywds:	Keywords of session
a=tool:	Name of tool used to create SDP
a=ptime:	Length of time in milliseconds for each packet
a=recvonly	Receive only mode
a=sendrecv	Send and receive mode
a=sendonly	Send only mode
a=orient:	Orientation for whiteboard sessions
a=type:	Type of conference
a=charset:	Character set used for subject and information fields
a=sdplang:	Language for the session description
a=lang:	Default language for the session
a=framerate:	Maximum video frame rate in frames per second
a=quality:	Suggests quality of encoding
a=fmtp:	Format transport
a=mid:	Media identification grouping
a=direction:	Direction for symmetric media
a=rtcp:	Explicit RTCP port (and address)
a=inactive	Inactive mode

RTP

- Real-time Transfer Protocol (RTP) provides end-to-end delivery services for data (such as interactive audio and video) with real-time characteristics.
- It was primarily designed to support multiparty multimedia conferences. However it is used for different types of applications.
- RTP is a standard specified in RFC 1889. More recent versions are RFC 3550 and RFC 3551. For an introduction like this we will stick to RFC1889
- RTP used in CS User plane

RTP gives no guarantee for timely delivery

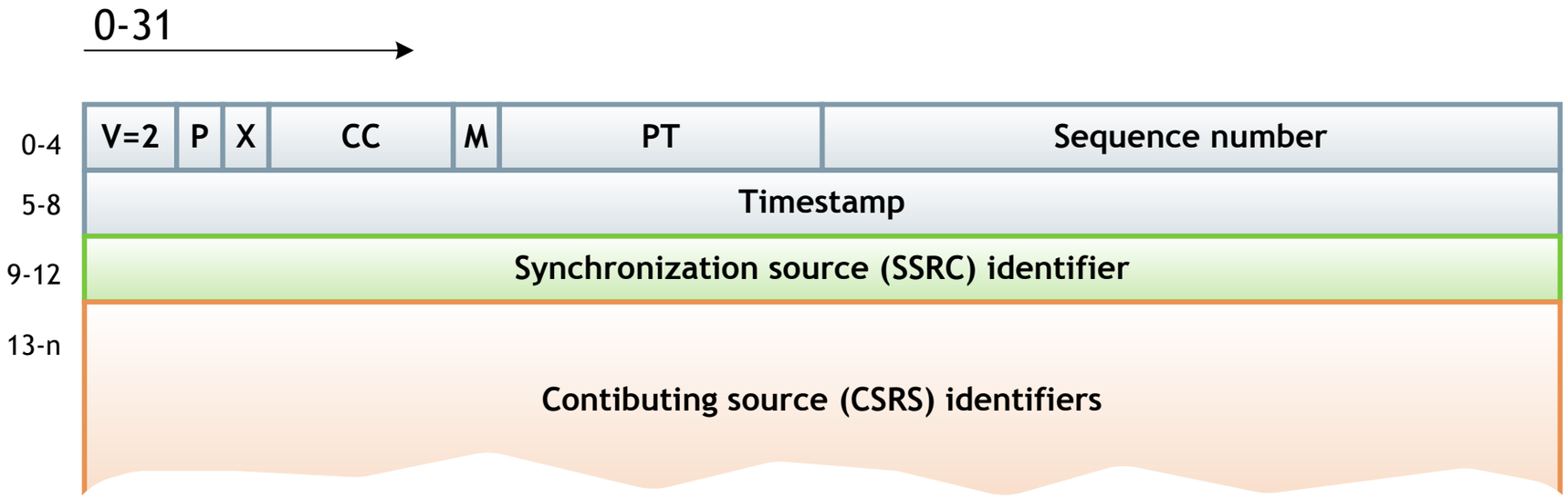
- RTP itself does not provide any mechanism to ensure timely delivery or any other QoS guarantees.
- It relies on lower-layer services (e.g. UDP, TCP).
- RTP provides suitable functionality for carrying real-time contents.

RTP

- RTP allows detection of some of the impairments introduced by an IP network:
  - Packet loss;
  - Variable transport delay;
  - Out of sequence packet arrival;
  - Asymmetric routing.

Packet structure of RTP

Header structure of RTP



RTP header fields 1/2

- version (V): 2 bits - The version is 2 upto RFC 1889.
- padding (P): 1 bit - If set, the packet contains one or more additional padding octets at the end which are not part of the payload.
- extension (X): 1 bit-If the extension bit is set, the fixed header is followed by exactly one header extension.
- CSRC count (CC): 4 bits - contains the number of CSRC identifiers that follow the fixed header.
- marker (M): 1 bit - is used by specific applications to serve a purpose of its own. We will discuss this in more detail when we study Application Level Framing.
- payload type (PT): 7 bits - identifies the format (e.g. encoding) of the RTP payload and determines its interpretation by the application. This field is not intended for multiplexing separate media.

RTP header fields 2/2

- sequence number: 16 bits - increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. The initial value of the sequence number is random (unpredictable).
- timestamp: 32 bits - reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations.
- SSRC: 32 bits - The SSRC field identifies the synchronization source. This identifier is chosen randomly, with the intent that no two synchronization sources within the same RTP session will have the same SSRC identifier.
- CSRC list: 0 to 15 items, 32 bits each - The CSRC list identifies the contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field. If there are more than 15 contributing sources, only 15 may be identified. CSRC identifiers are inserted by mixers, using the SSRC identifiers of contributing sources.

RTCP

- allows participants in an RTP session to send each other quality reports and statistics, and exchange some basic identity information.
- RTCP packets are to be sent about every 5 seconds (2 participants)
- 4 participants, RTCP packets can be sent every 10 seconds etc.
- Sender reports (SR) or receiver reports (RR) packets are sent the most frequently, with the other packet types being sent less frequently.

The use of reports allows feedback on the quality

- Number of packets sent and received
- Number of packets lost
- Packet jitter.

RTCP Packet Types

Packet Type	Name	Description
SR	Sender report	Sent by a participant that both sends and receives RTP packets
RR	Receiver report	Sent by a participant that only receives RTP packets
SDES	Source description	Contain information about the participant in the session including e-mail address, phone number, and host
BYE	Bye	Sent to terminate the RTP session
APP	Application specific	Defined by a particular profile
XR	Extended report	Extended report and summaries

RTP Audio Video Profiles

- The use of profiles enables RTP to be an extremely general media transport protocol.
- The profile document makes the following specifications for RTP:
  - UDP is used for underlying transport;
  - RTP port numbers are always even, the corresponding RTCP port number is the next highest port, always an odd number;
  - No header extensions are used.

RTP/AVP Audio and Video Payload Types

Payload	Codec	Clock	Description
0	PCMU	8000	ITU G.711 PCM μ-Law Audio 64 Kbps
1	1016	8000	CELP Audio 4.8 Kbps
2	G721	8000	ITU G721 ADPCM Audio 32 Kbps
3	GSM	8000	European GSM Audio 13 Kbps
5	DVI4	8000	DVI ADPCM Audio 32 Kbps
6	DVI4	16000	DVI ADPCM 64 Kbps
7	LPC	8000	Experimental LPC Audio
8	PCMA	8000	ITU G.711 PCM A-Law Audio 64 Kbps
9	G722	8000	ITU G.722 Audio
10	L16	44100	Linear 16-bit Audio 705.6 Kbps
11	L16	44100	Linear 16-bit Stereo Audio 1411.2 Kbps
14	MPA	90000	MPEG-I or MPEG-II Audio Only
15	G728	8000	ITU G.728 Audio 16 Kbps
25	CELB	90000	CelB Video
26	JBEG	90000	JBEG Video
28	NV	90000	nv Video
31	H261	90000	ITU H.261 Video
32	MPV	90000	MPEG-I and MPEG-II Video
33	MP2T	90000	MPEG-II transport stream Video
dynamic	iLBC	--	Internet low bit rate 15 Kbps [6]
dynamic	AMR	--	Adaptive Multirate Codec [12]

SIP security

- Considering the security aspects of an identity we distinguish three issues:
  - Identity theft
  - Subscription theft
  - Privacy

Identity theft

- By stealing the identity of another user, an attacker can pretend to be that user and benefit from any privileges this user might have. This might include getting access to certain services or misusing the trust that other users might have placed in the user.

Subscription theft

- Subscription theft is a severe case of identity theft in which the attacker can not only pretend to be another user and charge the costs to that user but receive calls destined to that user as well.

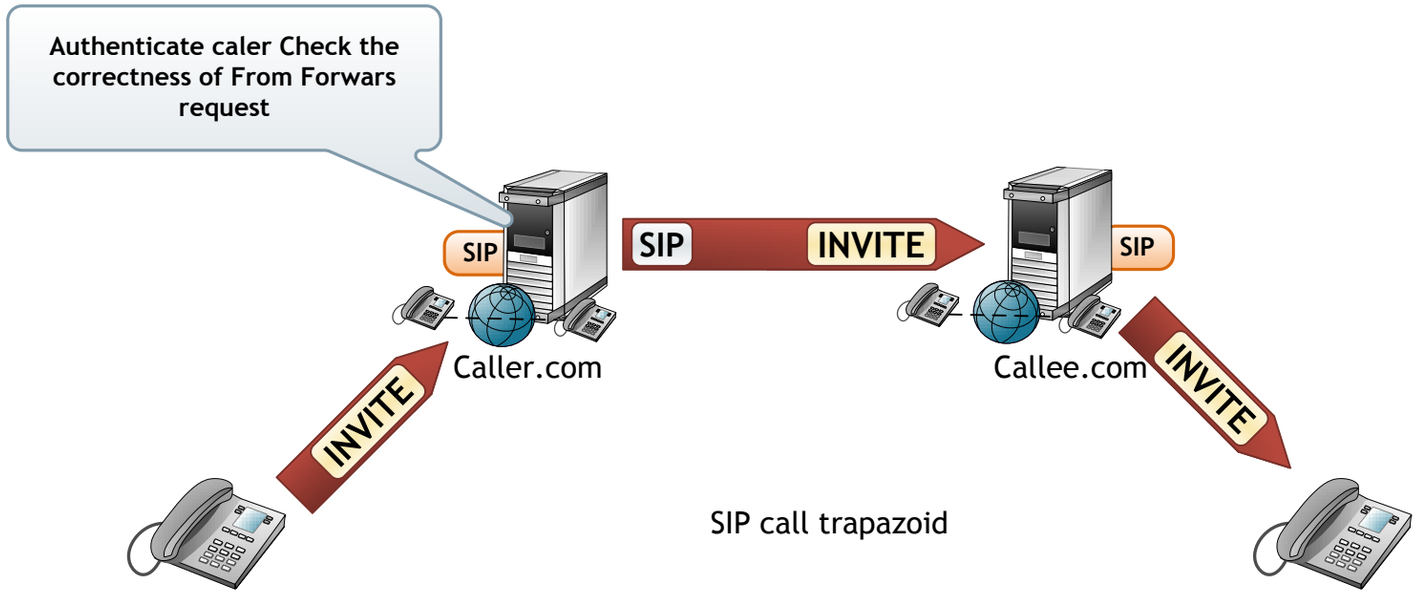
Privacy

- While a callee would want to have some assurance that the identity of a caller is the true one, a caller might often want to hide his identity from the callee and untrusted service providers. To ensure the privacy of the caller, his identity information has to be concealed in such a manner so as to hide the information from the callee or untrusted service providers but still enable the service provider of the user to correctly route and bill the user's calls.

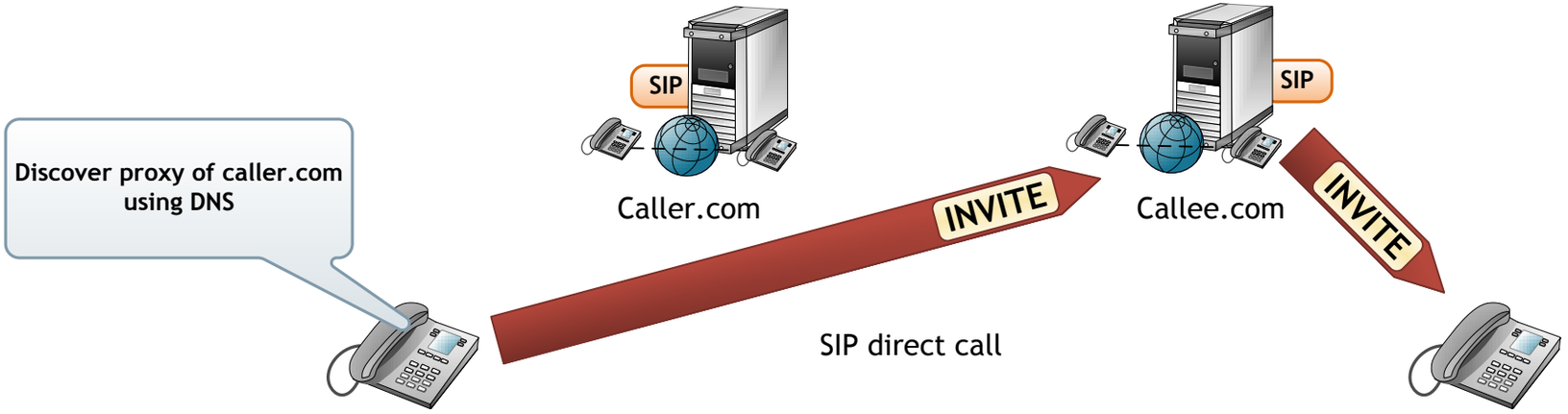
Identity Theft

- Normally INVITE requests are authorized by provider but there is no obligation to do so.
- Simplest way to stole the identity is to try sending fake INVITE with some-ones else identity.
- Attacker can even try to evade callers provider security and try to contact callee or his provider directly.

Identity Theft



SIP call trapazoid



SIP direct call

Identity Authentication using S/MIME

- S/MIME provides the necessary mechanisms for ensuring the integrity and confidentiality of application level information such as email or SIP messages.
- Using his own private key the user can sign his transmitted SIP messages.
- For encrypting a message the sender uses the public key of the receiver.

S/MIME

- When S/MIME is used, the body of a SIP message must be structured as MIME bodies.
- In order to protect the SIP headers, the SIP message is encapsulated in a message/sip MIME body.
- If confidentiality is desired, then the message/sip MIME body is carried in encrypted form inside a application/pkcs7-mime MIME body of type enveloped-data
- The signature that provides integrity protection and data origin authentication is transported in an application/pkcs7-signature MIME body
- The clear text or encrypted SIP message and the signature are encapsulated in the SIP message, the content type of which shall be multipart/signed.
- If confidentiality is provided to the SIP message, sensitive information may be removed from the headers of the SIP message that carries the encrypted original SIP message.

Encryption with S/MIME

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Anonymous <sip:anonymous@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:pc33.atlanta.com>
Content-Type: application/pkcs7-mime; smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m; handling=required
Content-Length: 231
```

```
Content-Type: message/sip
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <bob@biloxi.com>
From: Alice <alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
v=0
o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com
s=Session SDP
t=0 0
c=IN IP4 pc33.atlanta.com
m=audio 3456 RTP/AVP 0 1 3 99
a=rtpmap:0 PCMU/8000
```

Encrypted part

S/MIME

- Sending the SDP information encrypted improves the confidentiality
- Various components in a SIP network such as application layer gateways require this information for enabling the establishment of SIP sessions across NATs.
- Session border controllers might want to control the type and amount of media sent by the user.
- In such cases, encrypting the SDP bodies either requires these components to have access to the private keys of the receivers or the session establishment would fail.
- Saving a user's private key on a session border controller, NAT or SIP proxy is, however, generally not acceptable.

Using S/MIME for signing SIP messages

```
INVITE sip:bob@biloxi.com SIP/2.0
       Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
       To: Bob <sip:bob@biloxi.com>
       From: Alice <sip:alice@atlanta.com>;tag=1928301774
       Call-ID: a84b4c76e66710
       CSeq: 314159 INVITE
       Max-Forwards: 70
       Date: Thu, 21 Feb 2002 13:02:03 GMT
       Contact: <sip:alice@pc33.atlanta.com>
       Content-Type: multipart/signed;
         protocol="application/pkcs7-signature";
         micalg=sha1; boundary=boundary42
       Content-Length: 568

--boundary42
       Content-Type: message/sip

       INVITE sip:bob@biloxi.com SIP/2.0
       Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
       To: Bob <bob@biloxi.com>
       From: Alice <alice@atlanta.com>;tag=1928301774
       Call-ID: a84b4c76e66710
       CSeq: 314159 INVITE
       Max-Forwards: 70
       Date: Thu, 21 Feb 2002 13:02:03 GMT
       Contact: <sip:alice@pc33.atlanta.com>
       Content-Type: application/sdp
       Content-Length: 147
       v=0
       o=UserA 2890844526 2890844526 IN IP4 here.com
       s=Session SDP
       c=IN IP4 pc33.atlanta.com
       t=0 0
       m=audio 49172 RTP/AVP 0
       a=rtpmap:0 PCMU/8000
```

Signed part

```
--boundary42
       Content-Type: application/pkcs7-signature; name=smime.p7s
       Content-Transfer-Encoding: base64
       Content-Disposition: attachment; filename=smime.p7s;
         handling=required
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT64VQp
fyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnjn8HHGTrfv
hJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF47GhIGfHfYT64V
Qbnj756
--boundary42-
```

Signature

Identity Authentication in Trusted Environments

- The operator that is asserting an identity, i.e., adding the P-Asserted-Identity to a request, has authenticated the user first.
- The operator that is forwarding a request including a P-Asserted-Identity header, has received the request from a trusted server.
- The request was transported in a manner that prevents manipulation by other parties.
- The provider receiving a request including a P-Asserted-Identity header will not misuse the asserted information and will respect the user's privacy preferences.

User Privacy and Anonymity

- SIP distinguish various information that a user might want to keep private
  - Personal information: Personal information include the user's identity, equipment and work place. Caried in the SIP messages in headers such as To, From, Call-Id, Contact, Organization, Server or User-Agent.
  - Location information: Location information indicate the IP address, used hosts, traversed networks and service provider of the user. This information is included in the Via, Contact, Route and Record-Route headers as well as the SDP part.
  - Call information: Call information indicate that a user is making a call and with whom he is communicating

Theft of SIP Services

- Password guessing
- Credential emulation
- Route Misuse

Securing media

- The media exchanged during sessions established using SIP include audio, video, text and application data (e.g. fax, whiteboard).
- This data is exchanged by means of the Real-time Transport Protocol (RTP).
- For protecting the RTP datagrams, we may consider IPsec as well as TLS or DTLS as possible candidates.
- However, the exchange of real time data introduces some requirements, the fulfilling of which have a great impact on the performance of transporting and processing the media.

Requirements

- There must be no or a minimal increase in the payload size.
  - The use of additional headers and payload fields to carry per-packet information such as initialization vectors, sequence numbers, security association identifiers and padding mustbe avoided.
  - Instead, fields that already exist in the RTP payload must be used to derive these information.

Requirements

- Header compression mechanisms must not be hampered.
  - These compression mechanisms are based on the idea that the values of many of the fields contained in the IP, UDP and RTP headers remain unchanged across the datagrams that make up a given data flow.
  - Also, some other fields change in a predictable way. As a result, the values of all those fields can be conveyed only once when the data flow is established and need not be transmitted in the subsequent packets.
  - At the other end, a packet is identified by a ”connection ID” and the headers are restored.
  - On wireless and slow point-to-point links, these techniques reduce the size of the headers from 40 bytes (on IPv4) to just a few bytes.
  - As a result, the headers up to the RTP header must be left unencrypted.

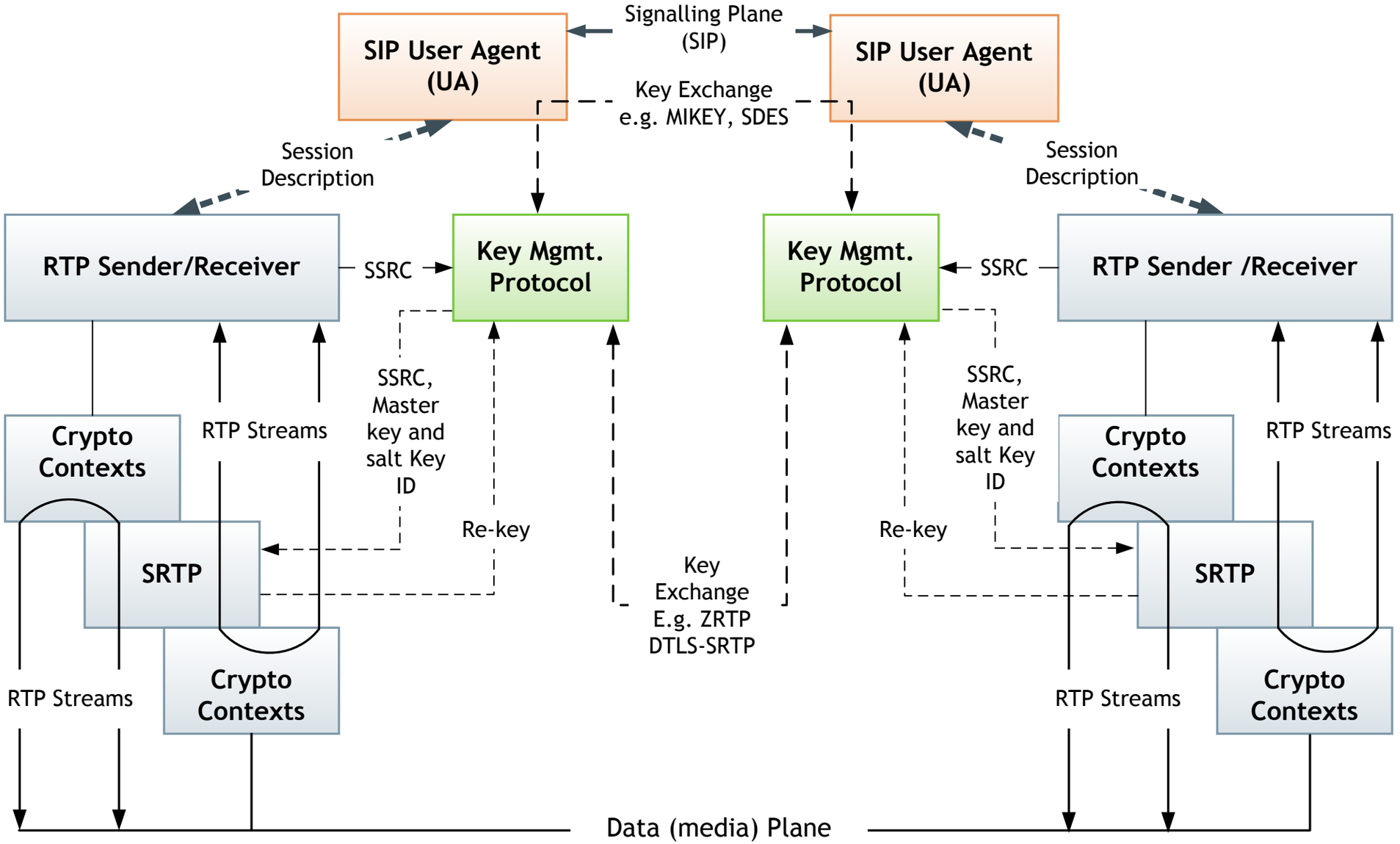
Requirements

- The decryption of the encrypted payloads must be possible despite datagrams being reordered or lost.
- The decryption process must be able to start before the entire data is received. Also, pipelining and parallelization of the cryptographic operations must be enabled to the largest possible extent, in order to achieve real-time performance.

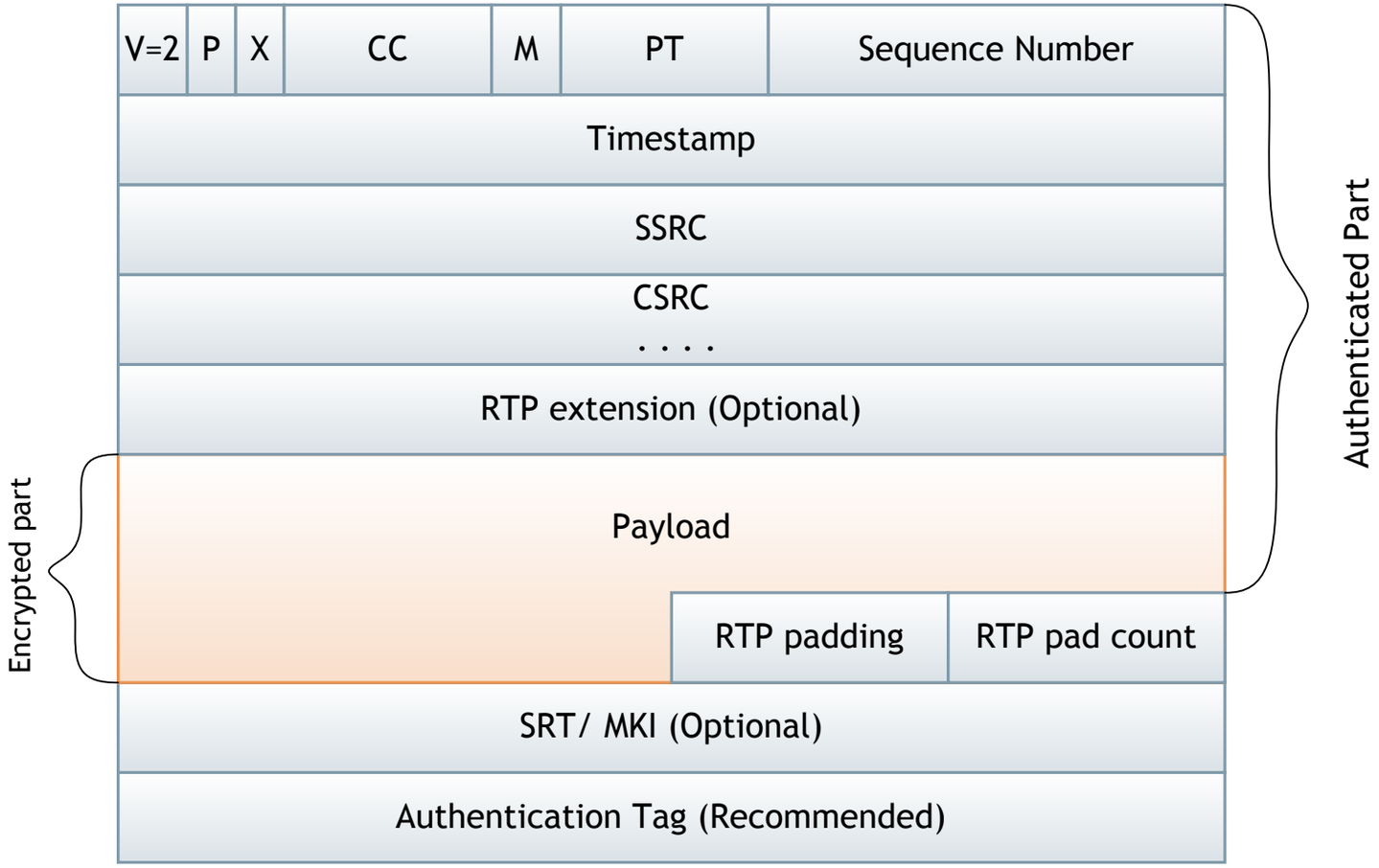
Requirements

- The hardware (in terms of number of gates) required by the cryptographic algorithms must be minimized because it will be typically be used on low power mobile terminals.
  - This has to do not only with the design of the cryptographic algorithm (which must be fast both in hardware and software) but also with the mode in which the algorithm is used: for instance the stream ciphers use only the encryption operation of the algorithm for both the enciphering and deciphering, which results in reducing the silicon footprint by half.

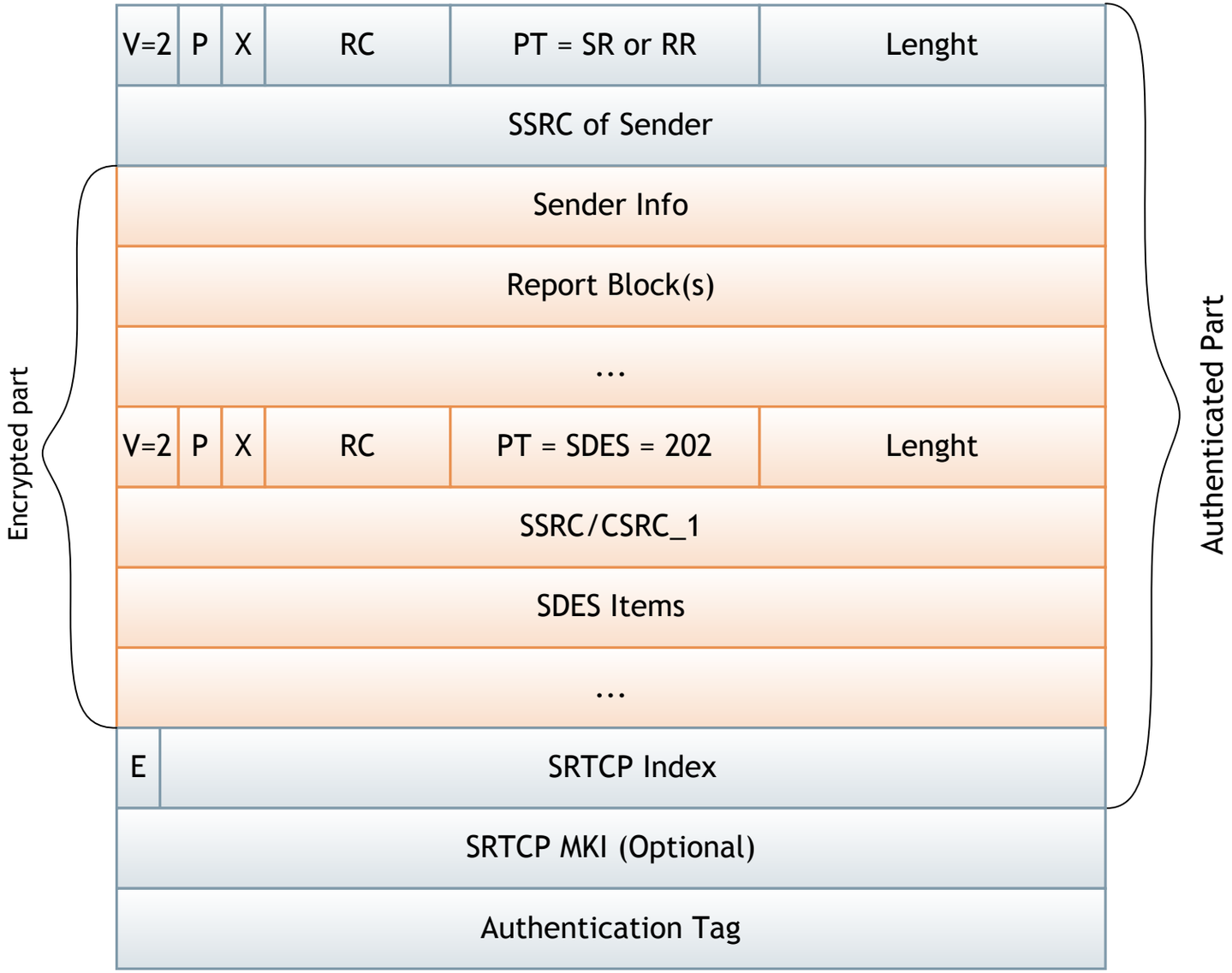
The main elements of the media plane security framework



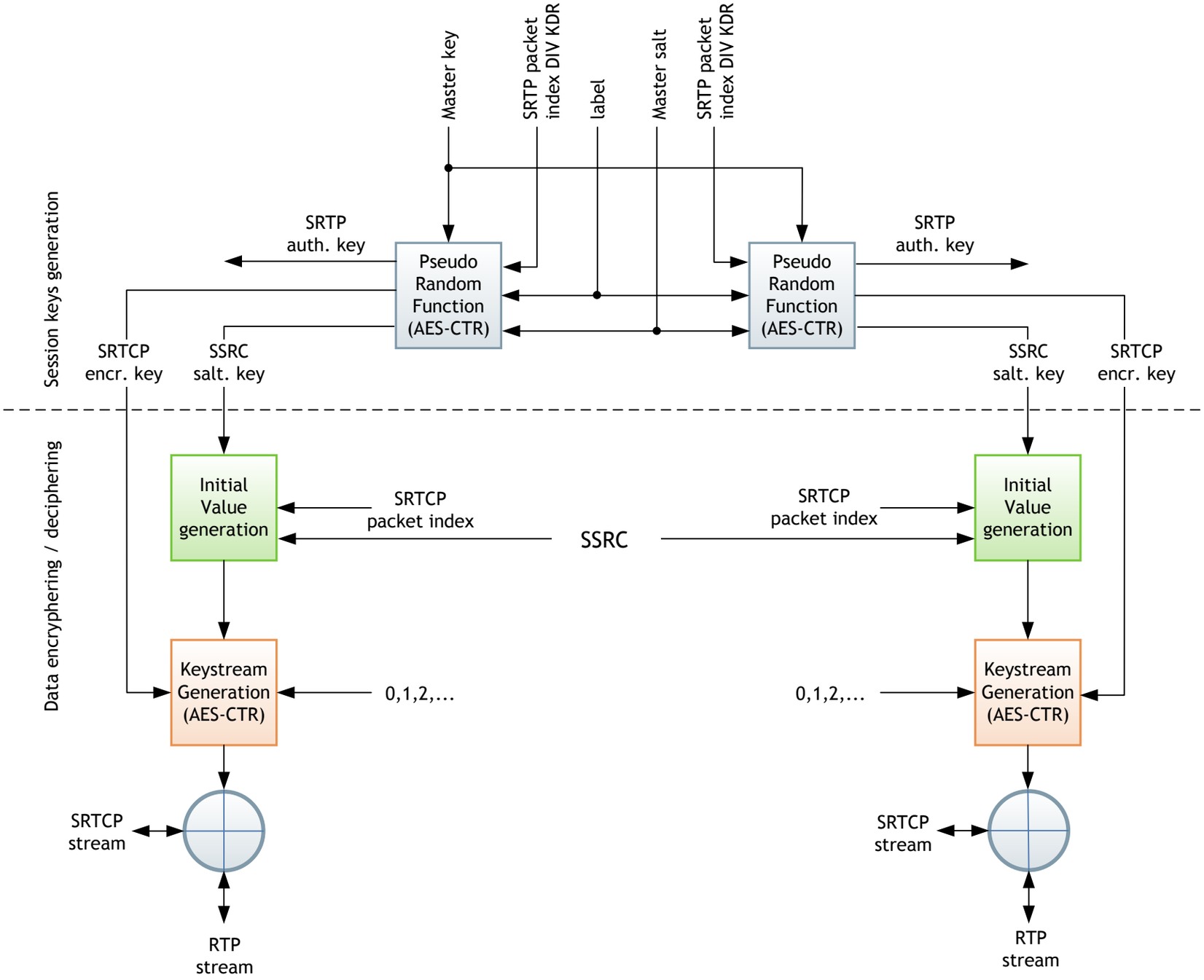
The structure of an SRTP packet



The structure of an SRTCP packet



The SRTP key derivation procedure



Key Exchange

- SDP Security Descriptions for Media Streams (SDES);
- Multimedia Key Exchange (MIKEY) and the Key Management Extensions for SDP and RTSP (key-mgmt);
- ZRTP;
- DTLS-SRTP.

SIP signalling protection

- S/MIME - can provide end-to-end confidentiality for the body of a SIP message and partial confidentiality for the
  - SIP headers,
  - integrity protection for the body and the immutable headers of the SIP message
  - identity authentication for the sender of the SIP message.
- The main drawback of S/MIME is that encrypting theSDP payloads end-to-end hinders intermediate signalling entities that need to access and/or update SDP attributes.

SIP signalling protection

- TLS - does not offer end-to-end protection to the SIP messages but only hop-by-hop confidentiality and integrity protection.
- This is an acceptable security model for those scenarios where a trust relationship has been established between the terminals and the SIP proxies, as well as between the SIP proxies operating in different administrative domains.
- In such a scenario it is assumed that the SIP proxies do not act maliciously and that the outbound SIP proxies grant access to the SIP infrastructure only to authenticated users
- nowadays most used variant

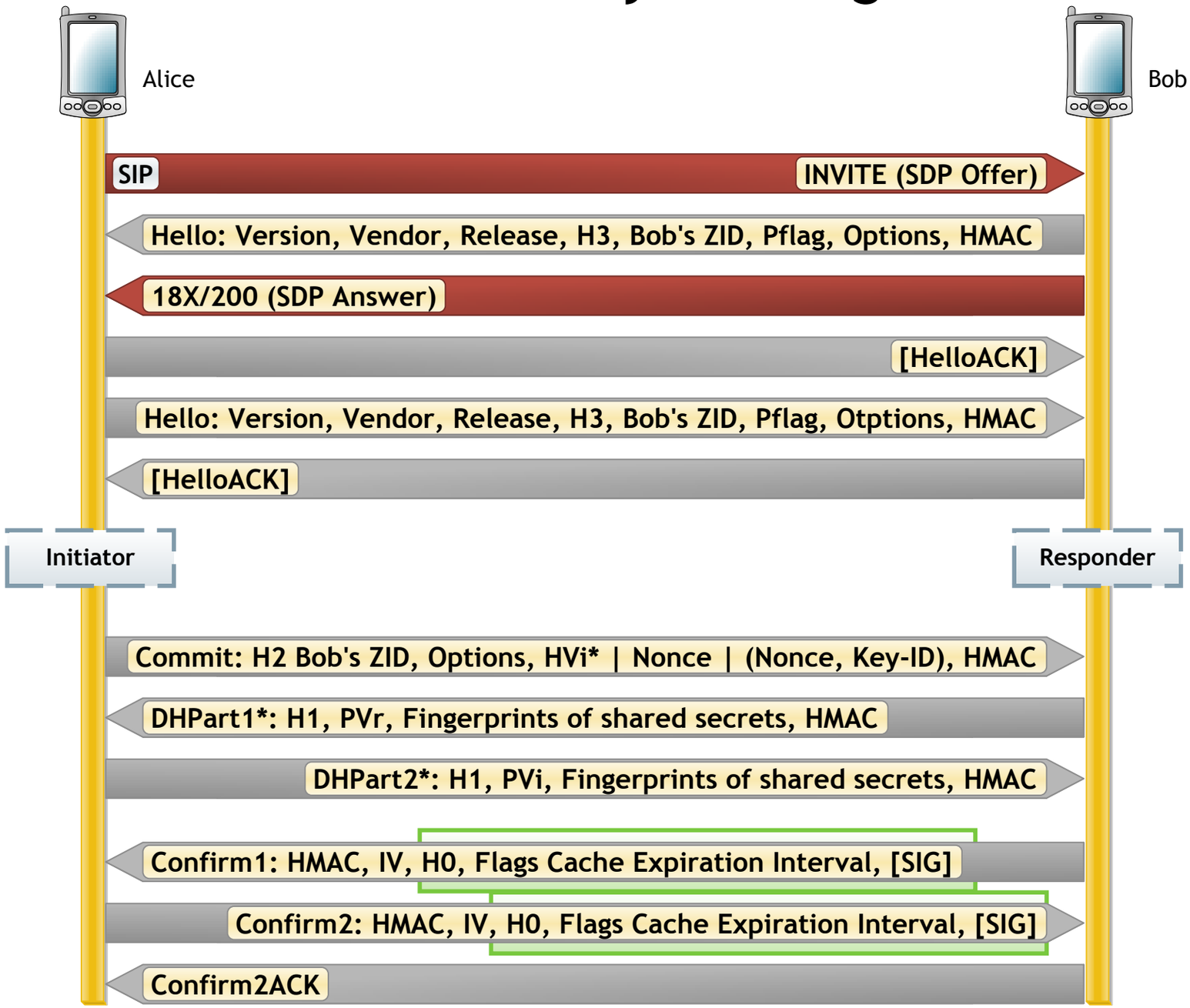
Example of an SDP offer containing a key-mgmt attribute

```
V=0
o=alice 2891092738 2891092738 IN IP4 host.example.com
s=Secured session
t=0 0
c=IN IP4 host.example.com
a=key-mgnt:mikey HRoYXQgYnkgYSBw...
a=key-mgnt:hypokmp mR1ZmF0aWdhYmxl...
m=audio 39000 RTP/SAVP 98
a=rtpmap:98 AMR/8000
m=video 42000RTP/SAVP 31
a=rtpmap:31 H261/90000
```

ZRTP

- ZRTP is a key agreement protocol that enables the participants in an one-to-one multimedia session to establish a shared secret and agree on the security parameters for setting up SRTP sessions. ZRTP is essentially an authenticated ephemeral Diffie-Hellman exchange, with the mutual authentication being performed by means of a Short Authentication String (SAS). In this way ZRTP can operate without support from a PKI. In addition to the SAS validation, ZRTP provides alternative mechanisms that can provide mutual authentication in case the signalling is integrity protected end-to-end or a PKIthat extends to the ZRTP endpoints is however available

The ZRTP key exchange



Legend

Payloads

P flag:	Flag to indicate the passive role in the exchange
H3, H2, H1, H0:	Hash images
ZID:	An unique ZRTP endpoint identifier
Options:	List of supported/selected hash, encryption and HMAC algorithms, key agreement types, SAS format
HVi:	A hash over the initiaator's DHPart2 and the responder's Options
PV:	Ephemeral Diffie-Hellman keys
SIG:	Digital signature over the exchanged SAS
IV:	Initial Value

Notations

[]	Optional payload or message
*	Message or payload is only used in "Diffie-Hellman mode" and they are missing otherwise
|	Logical OR
i suffix	Denotes initiaror
r suffix	Denotes responder
	Encrypted Payload

DTLS-SRTP

- DTLS-SRTP is an extension of the DTLS protocol that enables the participants in a point-to-point media session to agree on the cryptographic parameters and derive keying material necessary to establish SRTP sessions.
- The DTLS-SRTP handshake takes place in the media plane and is multiplexed on the same UDP ports as the media itself