Internet Engineering Task Force
Internet-Draft
Expiration: Apl 23th, 2007

M. HASEBE
J. KOSHIKO
Y. SUZUKI
T. YOSHIKAWA
NTT-East
P. Kyzivat
Cisco Systems, Inc.
Oct 23th, 2006

Examples call flow in race condition on Session Initiation Protocol
draft-hasebe-sipping-race-examples-02.txt

Status of this Memo

Copyright Notice

Abstract

This document gives examples of Session Initiation Protocol (SIP) call flows in race condition. Call flows in race conditions are confusing and this document shows the best practices to handle them. The elements in these call flows include SIP User Agents and SIP Proxies. Call flow diagrams and message details are shown.

Table of Contents

1. Overview

The call flows shown in this document were developed in the design of a SIP IP communications network. These examples are of race condition, which stems from the state transition of the dialog mainly established by INVITE.

When implementing SIP, various complex situations may arise. Therefore, it will be helpful to provide implementors of the protocol with examples of recommended terminal and server behavior.

This document clarifies SIP UA behaviors when messages cross each

other as race conditions. By clarifying operation under race conditions, different interpretations between implementations are avoided and interoperability is expected to be promoted.

It is the hope of the authors that this document will be useful for SIP implementors, designers, and protocol researchers and will help them achieve the goal of a standard implementation of RFC 3261 [1].

These call flows are based on the current version 2.0 of SIP in RFC 3261 [1] with SDP usage described in RFC 3264 [2].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [4].

1.1 General Assumptions

A number of architecture, network, and protocol assumptions underlie the call flows in this document. Note that these assumptions are not requirements. They are outlined in this section so that they may be taken into consideration and help in understanding of the call flow examples.

These flows do not assume specific underlying transport protocols such as TCP, TLS, and UDP. See the discussion in RFC 3261 [1] for details on the transport issues for SIP.

1.2 Legend for Message Flows

Dashed lines (---) and slash lines (/,\) represent signaling messages that are mandatory to the call scenario.(X) represents crossover of signaling messages. Arrow indicate the direction of message flow. Double dashed lines (===) represent media paths between network elements.

Messages with parentheses around their name represent optional messages.

Messages are identified in the Figures as F1, F2, etc. These numbers are used for references to the message details that follow the Figure. Comments in the message details are shown in the following form:

```
/* Comments. */
```

1.3 SIP Protocol Assumptions

This document does not prescribe the flows precisely as they are shown, but rather illustrates the principles for best practice. They are best practice usages (orderings, syntax, selection of

features for the purpose, or handling of error) of SIP methods,

headers and parameters. NOTE: The flows in this document must not be copied as they are by implementors because additional characteristics were incorporated into the document for ease of explanation. To sum up, the procedures described in this document represent well-reviewed examples of SIP usage, which are best common practice according to IETF consensus.

For simplicity in reading and editing the document, there are a number of differences between some of the examples and actual SIP messages. Examples are: Call-IDs are often repeated; CSeq often begins, at 1; header fields are usually shown in the same order; usually only the minimum required header field set is shown; and and Accept, Allow, etc are not shown.

Actors:

Element	Display Name	URI	IP Address
User Agent	Alice	sip:alice@atlanta.example.com	192.0.2.101
User Agent	Bob	sip:bob@biloxi.example.com	192.0.2.201
User Agent	Carol	sip:carol@chicago.example.com	192.0.2.202
Proxy Server		ss.atlanta.example.com	192.0.2.111

2. The Dialog State Machine for INVITE dialog usage

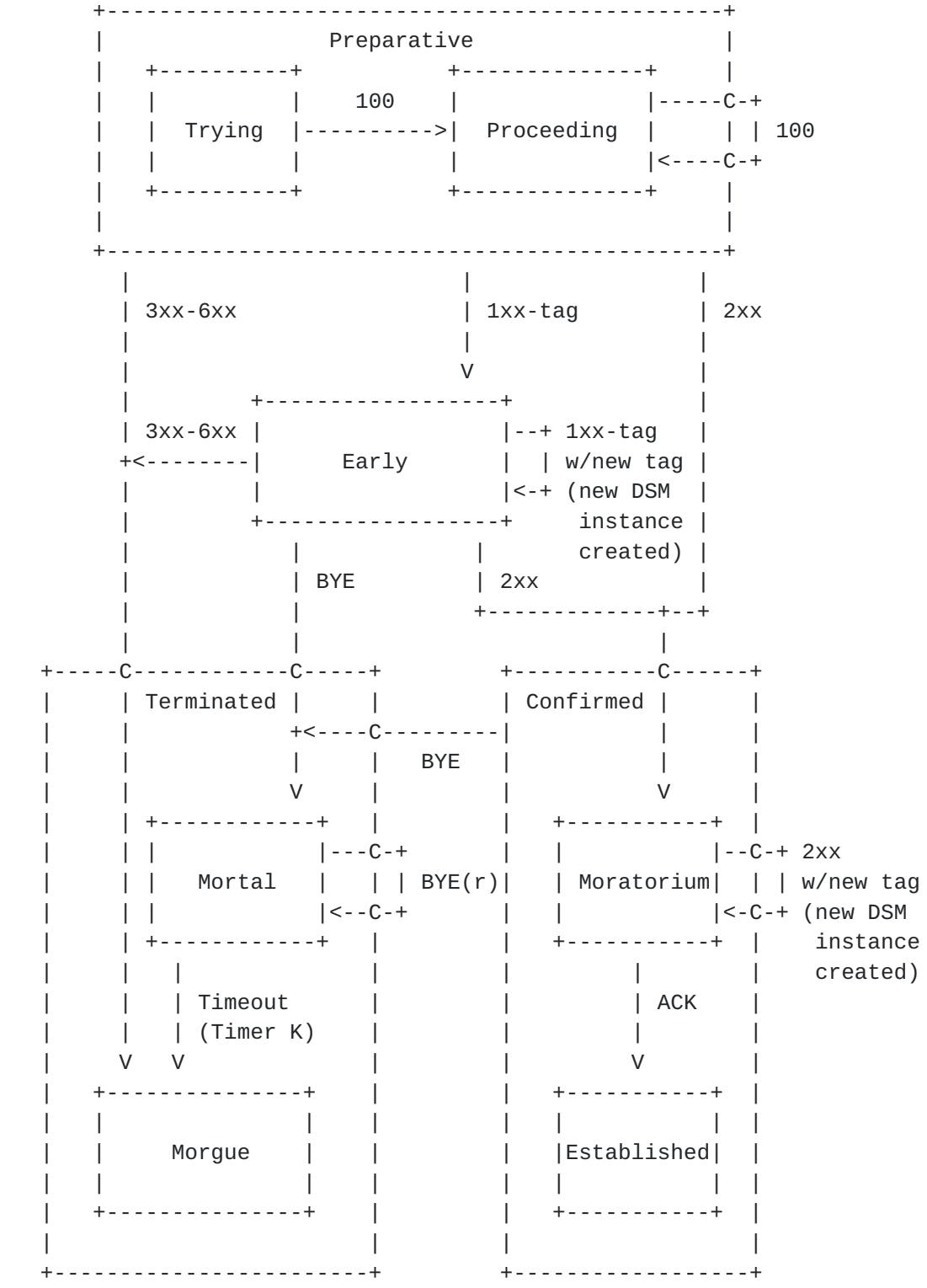
Race conditions are generated when the dialog state of the receiving side differs from the dialog state of the sending side.

For instance, a race condition occurs when UAC (User Agent Client) sends a CANCEL on Early state while UAS (User Agent Server) is transitting from Early state to Confirmed state by sending a 200 OK to ini-INVITE.

The dialog state machine (DSM) for INVITE dialog usage is represented as follows to help the understanding of UA's behavior in such race conditions.

The DSM clarifies UA's behavior by subdividing some internal states showed on FSM (Finite State Machine) for dialog state of the dialog-package [7], without changing the states of the dialog, "early", "confirmed", and "terminated" shown in RFC3261 [1]. Preparative state is put before the Ealy state, which includes Trying and Proceeding. Confirmed state is devided into two sub-states, Moratorium and Established and Terminated state is subdivided into two states, Mortal and Morgue.

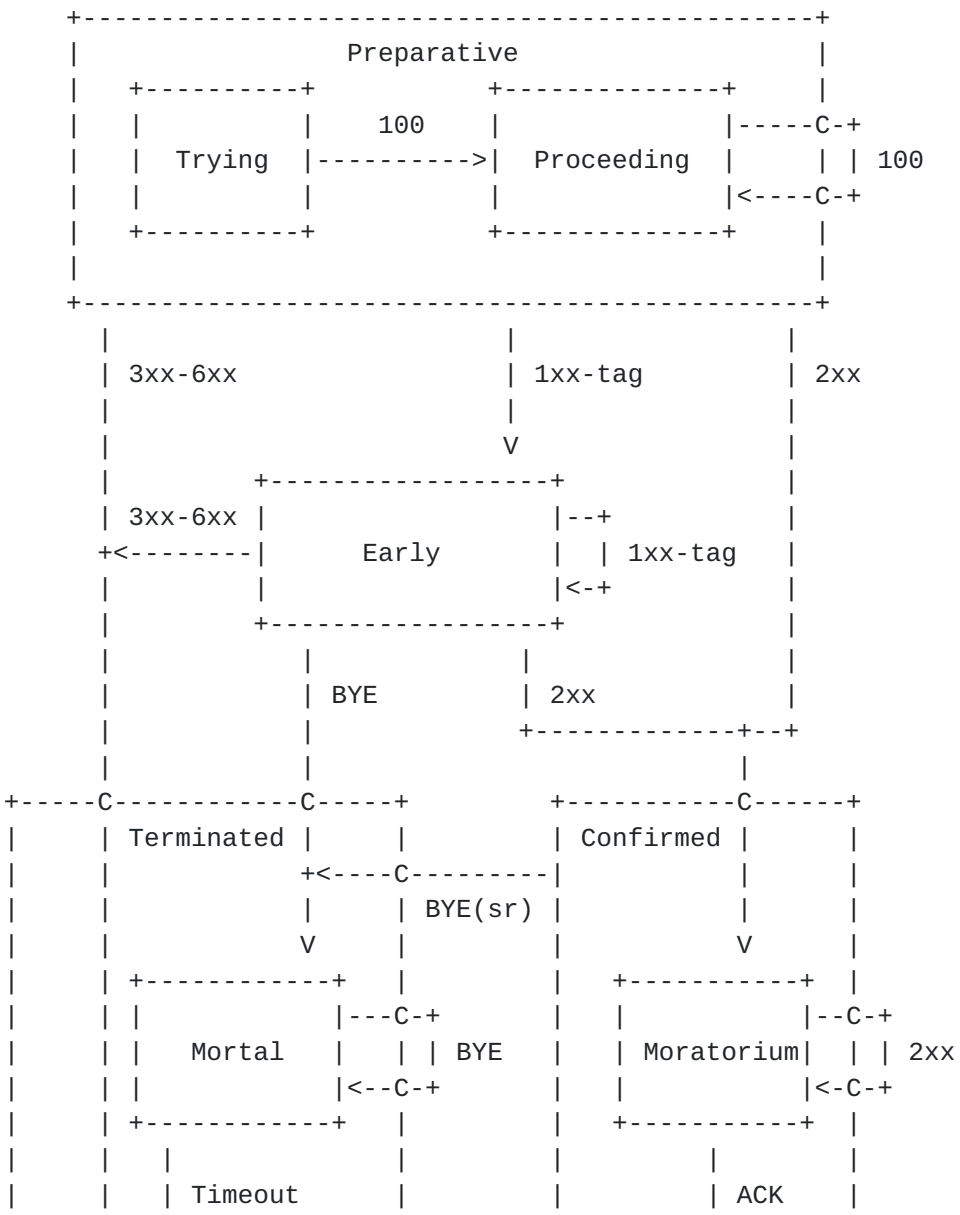
Below represent the DSM for UAC and UAS respectively.



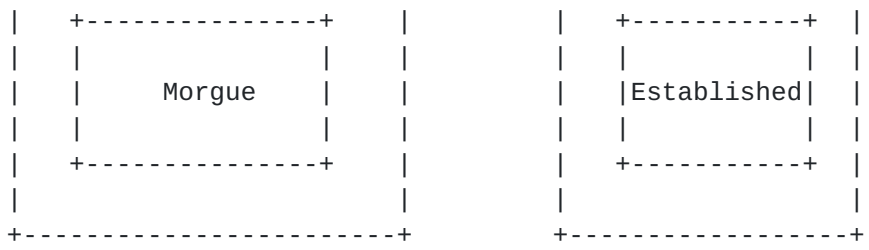
(r): indicates only reception is allowed.
Where (r) is not indicated, a response means receive, a request means send.

figure 1. DSM for INVITE dialog usage (UAC)

Figure 1 shows a DSM for UAC.
UAC MAY send a BYE in Early state. However, this behavior is NOT RECOMMENDED. The dialog which is to be terminated by BYE in Early state. Early state is the one that exists between the UAC and the UAS that constitutes the early dialog with each other. In Early state, it is possible that UAC receives responses from other UASs in forking. Therefore, until the UAC receives the final response and terminates the INVITE transaction, UAC MUST be prepared to establish a dialog by receiving a new response even though it had sent a BYE and terminated the dialog (see Appendix A).




| | | (Timer J) | | | |
| V V | | V |



(sr): indicates that both sending and reception are allowed.
Where (sr) is not indicated, a response means send,
a request means receive.

figure 2. DSM for INVITE dialog usage (UAS)

Figure 2 shows a DSM for UAS.
The figure 2 includes state transition caused by BYE.
Originally, the correct description is that a CANCEL request does not cause a dialog state transition, but the UAS terminates the dialog and triggers the dialog transition by sending 487 immediately after the reception of the CANCEL. The behavior upon the reception of the CANCEL request is further explained in the section 2.1.

The following is UA's behaviors in each state.

Preparative (Pre): Preparative is a state until the Early dialog is established by sending and receiving a provisional response with To-tag after an ini-INVITE is sent and received. The dialog has not existed yet in Preparative state. The dialog state transit from the Preparative to the Early by sending or receiving a provisional response with To-tag. Moreover, the dialog state transit to Moratorium which is a substate of Confirmed state, if UA sends or receives a 2xx response. In addition, the dialog state transit to Morgue state which is a substate of Terminated state, if UA sends or receives a 3xx-6xx response. Sending an ACK to a 3xx-6xx response and retransmissions of 3xx-6xx are not expressed on this DSM because they are sent by INVITE transactions.

Trying (Try): Trying is substate of Preparative and inherits the behavior of Preparative. Trying is started by sending and receiving an ini-INVITE. It transits to Proceeding by sending or receiving a 1xx (usually 100 trying) without To-tag. UAC may retransmit an INVITE on transaction layer and UAC must not send a CANCEL request. UAS may send a 1xx-6xx response.

Proceeding (Pro): Proceeding is substate of Preparative and inherits the behavior of Preparative. Dialog becomes

Proceeding state if dialogs in Trying state send or receive a

1xx without To-tag (usually 100 trying). UAC may send a CANCEL, and UAS may send a 1xx-6xx response in Proceeding state.

Early (Ear): The early dialog is established by sending or receiving a provisional response with To-tag. The early dialog exists though the dialog has not existed in this state yet. The dialog state transits from Early to Moratorium, substate of Confirmed by sending or receiving a 2xx response. In addition, the dialog state transits to the Morgue subdivided internally in the Terminated by sending and receiving a 3xx-6xx response. Sending an ACK to a 3xx-6xx response and retransmissions of 3xx-6xx are not expressed on this DSM because they are automatically processed on transaction layer and don't influence the dialog state. UAC may send CANCEL in Early state. UAC may send BYE (although it is not recommended). UAS may send a 1xx-6xx response. Sending or reception of a CANCEL request does not have direct influences on dialog state. The UA's behavior upon the reception of the CANCEL request is further explaind in the section 2.1 below.

Confirmed (Con): Sending or receiving 2xx final response establishes a dialog. Dialog exists in this state. BYE message changes state from Confirmed to Mortal, substate of Terminated. Confirmed has two substates, Moratorium and Established, they are different in messages UA are allowed to send.

Moratorium (Mora): Moratorium is a substate of Confirmed and inherits the behavior of Confirmed. Moratorium transits to Established by sending or receiving an ACK request. UAC may send an ACK and UAS may send a 2xx final response.

Established (Est): Established is a substate of Confirmed and inherits the behavior of Confirmed. Both caller and callee may send various messages which influences a dialog. Caller supports the transmission of ACK to a retransmission of a 2xx response to an ini-INVITE.

Terminated (Ter): Terminated state is devided into two substates, Mortal and Morgue, to consider a behavior when a dialog is being terminated. In this state, UAs hold information about the dialog which is being terminated. Confirmed transits to Mortal, a substate of Terminated, by sending or receiving a BYE request.

Mortal (Mort): Caller and callee becomes Mortal state by sending or receiving a BYE. UA MUST NOT send any new requests since

there is no dialog. (Here the new requests do not include ACK
for 2xx and BYE for 401 or 407. The further explaind is in

the section 2.2 below.)
In this state, only a BYE or its response can be handled, and no other messages can be received. This is because the use case is taken into consideration that a BYE message are sent by both a caller and a callee to exchange reports about the session when it is being terminated. Therefore, UA possesses dialog information for internal process but dialog shouldn't exist outwardly. UA stops managing dialog state and changes it to Morgue state, when the BYE transaction is done by timer (Timer F or Timer K for UAC. Timer J for UAS).

Morgue (Morg): Dialog doesn't exist any more in this state.
Sending or receiving a signal which influences a dialog is not performed. (It is literally terminated.)

3. Race condition

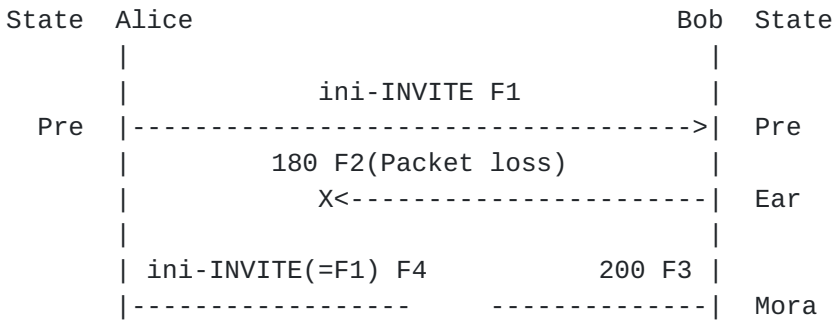
This section details race condition between two SIP User Agents (UAs): Alice and Bob. Alice (sip:alice@atlanta.example.com) and Bob (sip:bob@biloxi.example.com) are assumed to be SIP phones or SIP-enabled devices.
Only significant signals are illustrated. Dialog state transitions caused by sending and reception of SIP messages as well as '*race*', which indicates race condition, are shown. (For abbreviations for the dialog state transitions, refer to Chapter 2)
'*race*' indicates the moment when a race condition occurs.

Examples of such race conditions are shown below.

3.1 Receiving message in the Moratorium State

This section shows some examples of call flow in race condition when receiving the message from other states in the Moratorium state.

3.1.1 Receiving Initial INVITE retransmission (Trying state) in Moratorium state



This scenario illustrates the race condition which occurs when UAS receives a Preparative message in Moratorium state. All provisional responses to the initial INVITE (ini-INVITE F1) are lost, and UAC retransmits an ini-INVITE (F4). At the same time as retransmission, UAS generates a 200 OK (F3) to the ini-INVITE and it terminate an INVITE server transaction. (RFC3261, 13.3.1.4 [1])
However, it is reported that terminating an INVITE server transaction by 200 OK is a SIP bug. (http://bugs.sipit.net/, #769)
Therefore, the INVITE server transaction is not terminated at F3, and the F4 MUST be properly handled as a retransmission.
(UAs that do not deal with this bug still need to recognize the retransmission relying on its From-tag and Call-ID, even though it does not match the transaction.)
In RFC3261 [1], it is not specified whether a UAS retransmits 200 to the retransmission of ini-INVITE. Considering the retransmission of 200 is triggered by timer (TU keeps retransmitting 200 based on T1 and T2 until it receives an ACK) according to Section 13.3.1.4 of RFC3261 [1], it seems unnecessary to retransmit 200 when the UAS receives the retransmission of ini-INVITE. (For implementation, it does not matter if UAS sends the retransmission of 200, since the 200 does not cause any problem.)

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice
/* A 180 response is lost and does not reach Alice. */

F3 200 OK Bob -> Alice
/* According to 13.3.1.4 of RFC3261, an INVITE server transaction is terminated at this point. However, this has been reported as a SIP bug, and UAS MUST correctly recognize the ini-INVITE (F4) as a retransmission. */

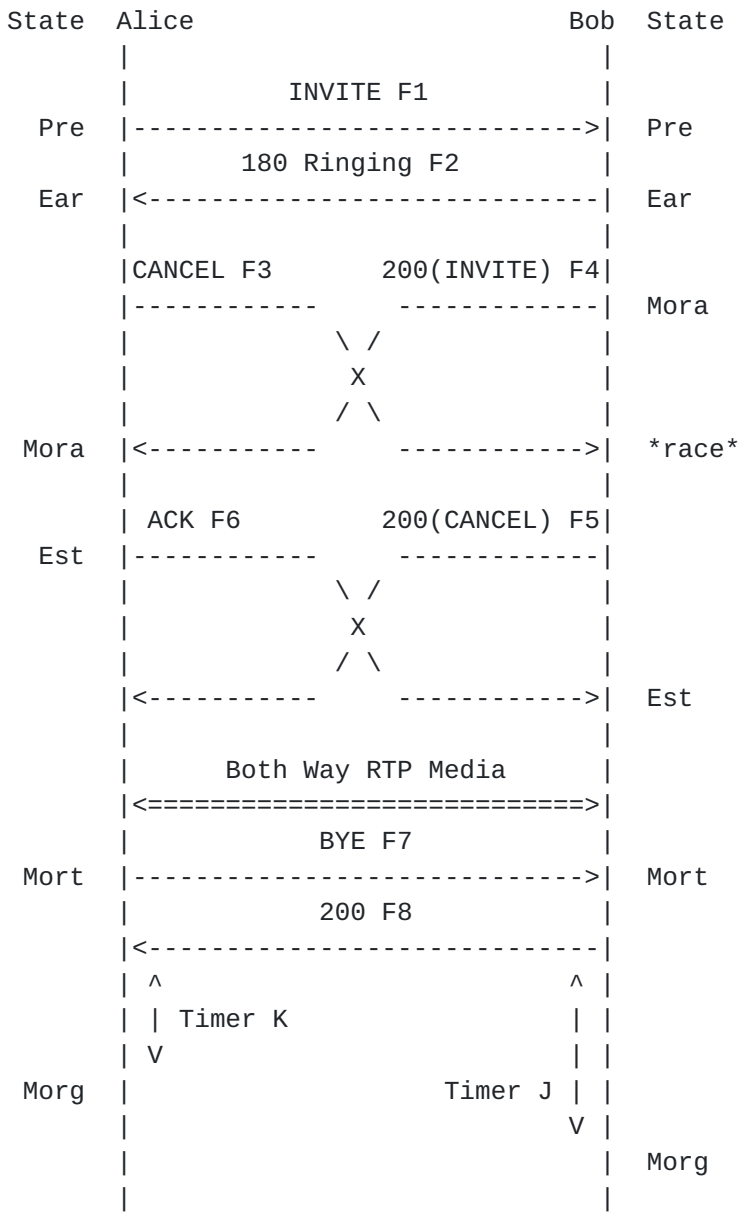
F4 INVITE (retransmission) Alice -> Bob
/* F4 is a retransmission of F1. They are exactly the same INVITE request. When UAs do not deal with the bug reported in #769 (an INVITE server transaction is terminated by 200 to INVITE), this

request does not match the transaction as well as the dialog

since it does not have a To-tag.
However, Bob have to recognize the retransmitted INVITE correctly,
without treating it as the new INVITE. */

F5 ACK Alice -> Bob

3.1.2 Receiving CANCEL (Proceeding or Early state) in Moratorium state



This scenario illustrates the race condition which occurs when UAS receives an Early message (CANCEL) in Moratorium state. Alice sends

a CANCEL and Bob sends a 200 OK response to the initial INVITE message at the same time. As described in the previous section,

according to RFC3261 an INVITE server transaction is terminated by a 200 response, but this has been reported as a bug in #769.
This section describes a case in which an INVITE server transaction is not terminated by a BYE response to the request. In this case, there is an INVITE transaction which matches a CANCEL request, so a 200 response is sent for the request. This 200 response simply means that the next hop received the CANCEL request.
(Successful CANCEL (200) does not mean an INVITE failure)
When UAS does not deal with #769, UAC MAY receive a 481 response for CANCEL since there is no transaction which matches the CANCEL request. This 481 simply means that there is no matching INVITE server transaction and CANCEL is not sent to the next hop.
Regardless of the success/failure of the CANCEL, Alice checks the final response to INVITE, and if she receives 200 to the INVITE request she immediately sends a BYE and terminates a dialog.
(RFC3261, 15)

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 CANCEL Alice -> Bob
/* Alice sends a CANCEL on the Early state. */

F4 200 OK (INVITE) Bob -> Alice
/* Alice receives a 200 to INVITE (F1) on the Moratorium state. */

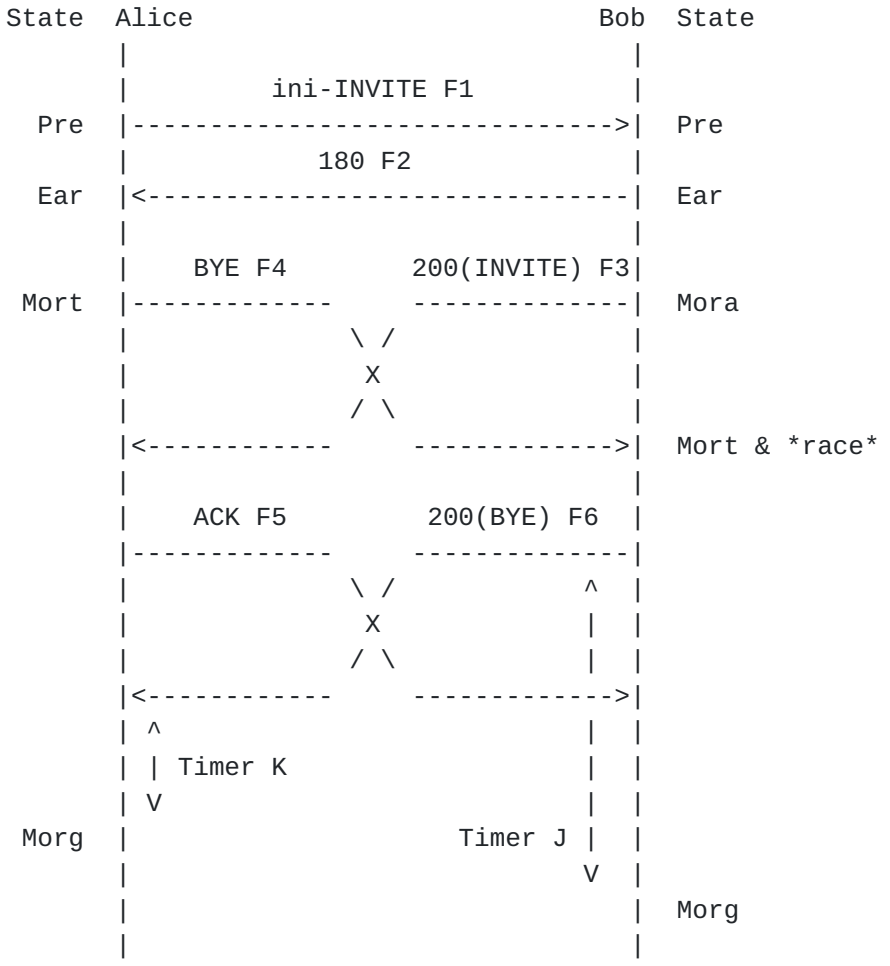
F5 200 OK (CANCEL) Bob -> Alice
/* A 200 to CANCEL simply means that the CANCEL was received.
The 200 response is sent, since this document deals with the bug reported in #769. When an INVITE server transaction is terminated as the procedure stated in RFC3261, UAC MAY receive a 481 response instead of a 200. */

F6 ACK Alice -> Bob
/* INVITE is successful, and a CANCEL becomes invalid. RTP streams are established. However, the next BYE request immediately cleans up the dialog just established. */

F7 BYE Alice -> Bob

F8 200 OK Bob -> Alice

3.1.3 Receiving BYE (Early state) in Moratorium state



This scenario illustrates the race condition which occurs when UAS receives an Early message (BYE) in Moratorium state. Alice sends a BYE on the early dialog and Bob sends a 200 OK response to the initial INVITE message at the same time. Bob receives a BYE on the Confirmed dialog though Alice sended a BYE on the Early dialog. A BYE functions normally even if it is received after the INVITE transaction terminates because a BYE differs from a CANCEL, and is sent to not the request but the dialog. Alice gets into a Mortal state on receiving the BYE response, and remains Mortal until the Timer K timeout occurs. In Mortal state, UAC does not establish a session, even though it receives a 200 response for INVITE. Even so, it sends an ACK to 200 for completion of INVITE transaction.

Editor's Note:
ACK was not sent in the previous version of this draft. Since both dialog usage and session were finished, it was thought that it is not

necessary to send ACK. In this version ACK is sent, according to RFC3261 which states that the INVITE transaction consists of the

three-way handshake of INVITE/200/ACK.

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK (ini-INVITE) Bob -> Alice

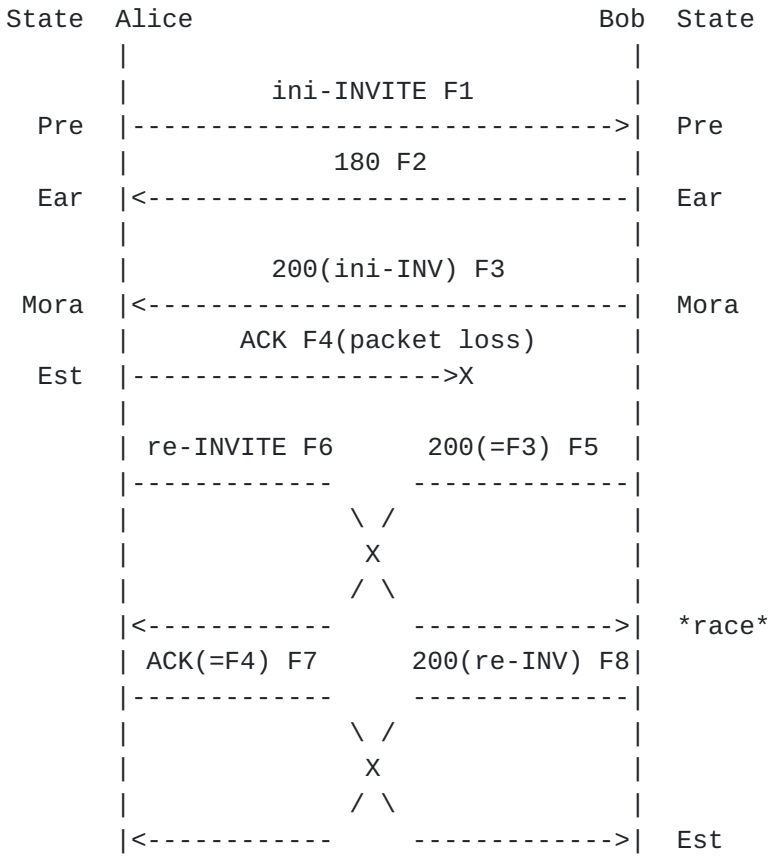
F4 BYE Alice -> Bob

/*Alice transits to the Mortal state upon sending BYE. Therefore, after this, she does not begin a session even though she receives a 200 response with an answer./

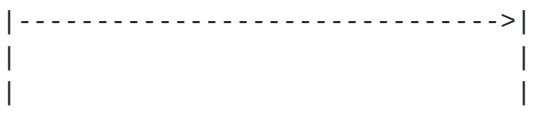
F5 ACK Alice -> Bob

F6 200 OK (BYE) Bob -> Alice

3.1.4 Receiving re-INVITE (Established state) in Moratorium state (case 1)




ACK F9



This scenario illustrates the race condition which occurs when UAS receives a message (re-INVITE) sent on Established state in Moratorium state.
UAS receives a re-INVITE before receiving an ACK to ini-INVITE. UAS sends a 200 OK to the re-INVITE (F8) because it has sent a 200 OK to the ini-INVITE (F3, F5) and the dialog has already been confirmed. (Because F5 is a retransmission of F3, SDP negotiation is not performed here.) If a 200 OK to the ini-INVITE has an offer and the answer would be in the ACK, UA should return by a 491 to the re-INVITE (refer to 3.1.5). As it can be seen in the section 3.3.2 below, the 491 response seems to be closely related to session establishment, even in cases other than INVITE cross-over. This example recommends 200 be sent instead of 491 because it does not influence session. However, a 491 response can lead to the same outcome, so the either response can be used.
Moreover, if UAS doesn't receive an ACK for a long time, it should send a BYE and terminate the dialog.

Message Details

F1 INVITE Alice -> Bob

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 137

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* ini-INVITE contains an offer.  */
```

F2 180 Ringing Bob -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356

Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Length: 0
```

F3 200 OK Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F4 ACK Alice -> Bob

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0

/* A ACK request is lost.  */
```

F5 200 OK (=F3) Bob -> Alice (retransmission)
/* UAS retransmits a 200 OK to an ini-INVITE since it didn't receive
a ACK. */

F6 re-INVITE Alice -> Bob

```
INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly
```

F7 ACK (=F4) Alice -> Bob (retransmission)

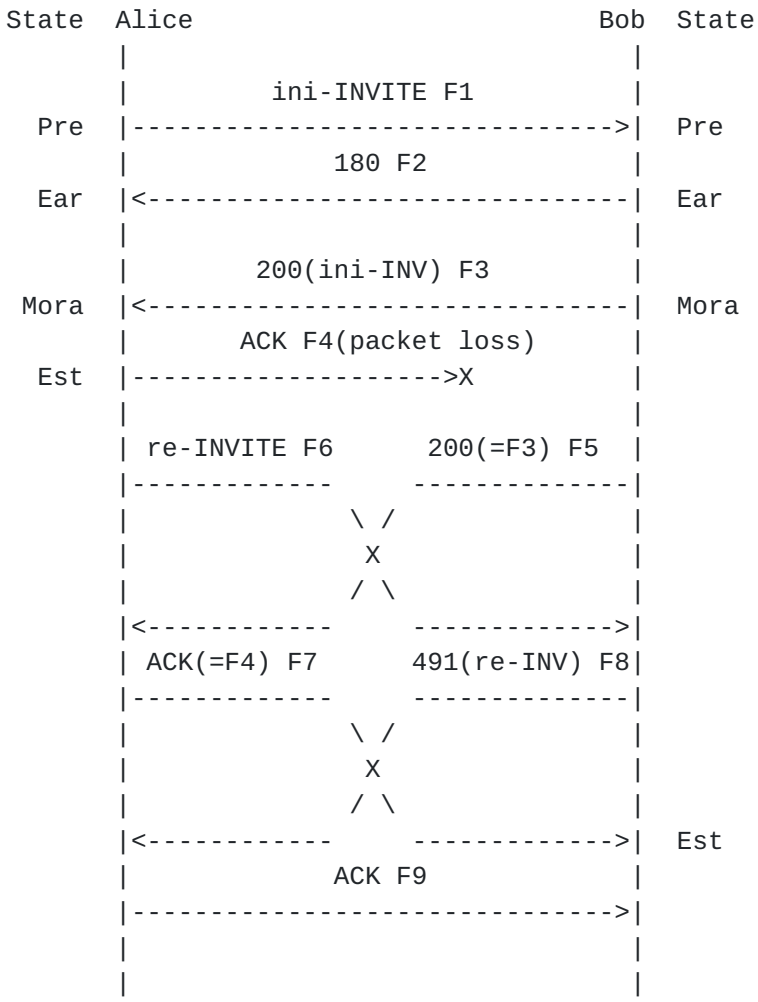
F8 200 OK (re-INVITE) Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 143

v=0
o=bob 2890844527 2890844528 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=recvonly
```

F9 ACK Alice -> Bob

```
ACK sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK230f2.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 ACK
Content-Length: 0
```

3.1.5 Receiving re-INVITE (Established state) in Moratorium state (case 2)



This scenario is basically the same with 3.1.4, but differs in sending an offer in 200 and an answer in ACK. Different to the previous case, the offer in the 200 (F3) and the offer in the re-INVITE (F6) collide with each other.

Bob sends 491 to re-INVITE since he is not able to properly handle a new request until he receives an answer.

Message Details

F1 INVITE Alice -> Bob

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=udp>
Content-Length: 0

/* The request does not contain an offer.  */
```

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* An offer is made in 200 */
```

F4 ACK Alice -> Bob

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
```

```
   Max-Forwards: 70
```

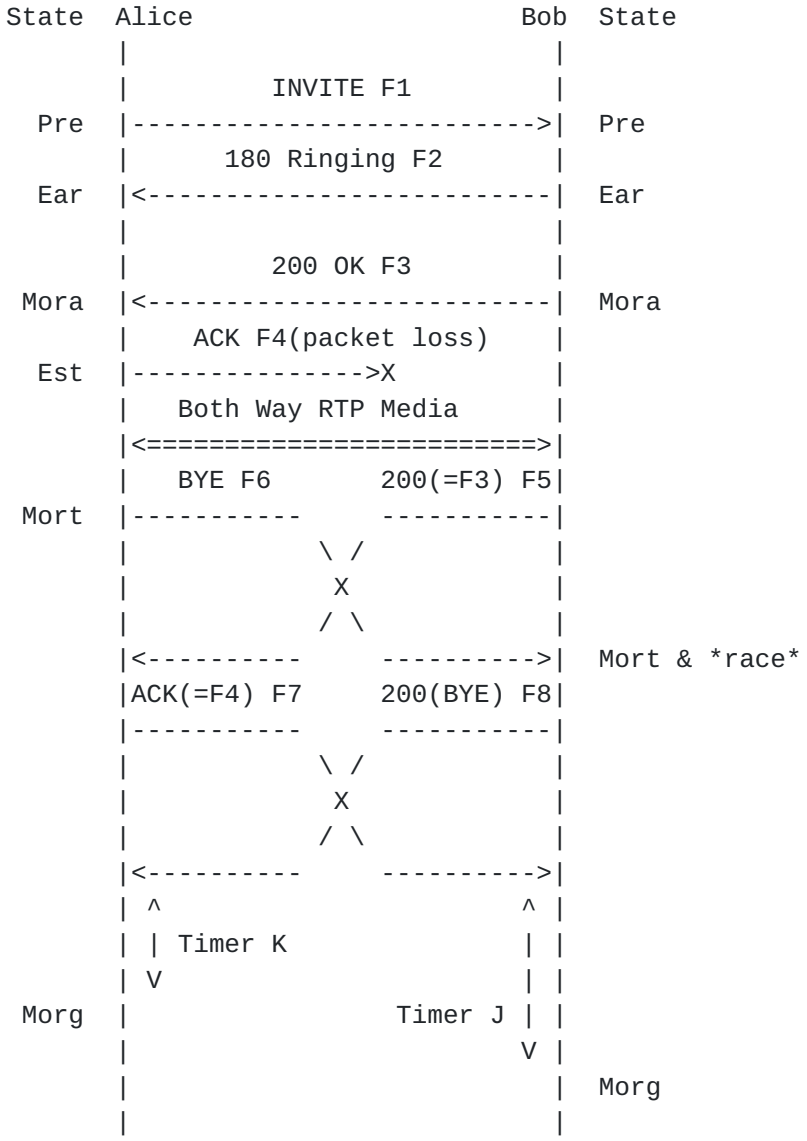
```
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 ACK
   Content-Type: application/sdp
   Content-Length: 137

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* The request contains an answer.  ACK request is lost.  */


   F5 200 OK (=F3) Bob -> Alice (retransmission)
   /* UAS retransmits a 200 OK to an ini-INVITE since it didn't receive
      an ACK.  */


   F6 re-INVITE Alice -> Bob

   INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0

   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9.1
   Max-Forwards: 70
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 2 INVITE
   Content-Length: 147

   v=0
   o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   a=sendonly

   /* The request contains an offer.  */


   F7 ACK (=F4) Alice -> Bob (retransmission)
   /* A retransmission triggered by the reception of a retransmitted
```

```
   200.  */
```

F8 491 (re-INVITE) Bob -> Alice
/* Bob sends 491 (Request Pending), since Bob has a pending offer. */

F9 ACK Alice -> Bob

3.1.6 Receiving BYE (Established state) in Moratorium state



This scenario illustrates the race condition which occurs when UAS receives an Established message (BYE) in Moratorium state. An ACK request to a 200 OK response is lost (or delay),

immediately after Bob sends the retransmitted 200 OK to

ini-INVITE and Alice sends a BYE at the same time.
Depending on the implement of a SIP user agent, Alice may start a session again by reception of the retransmitted 200 OK with SDP since she has already terminated a session by sending a BYE. In that case, if UAC receives a retransmitted 200 OK after sending a BYE, you should not start a session again since the session which is not associated with dialog remains. Moreover, in the case where UAS sends an offer with a 200 OK, if UAS receives a retransmitted ACK after receiving a BYE, UAS should not start a session again for the same reason.

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

/* An ACK request is lost. */

F5 200 OK (retransmission) Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
```

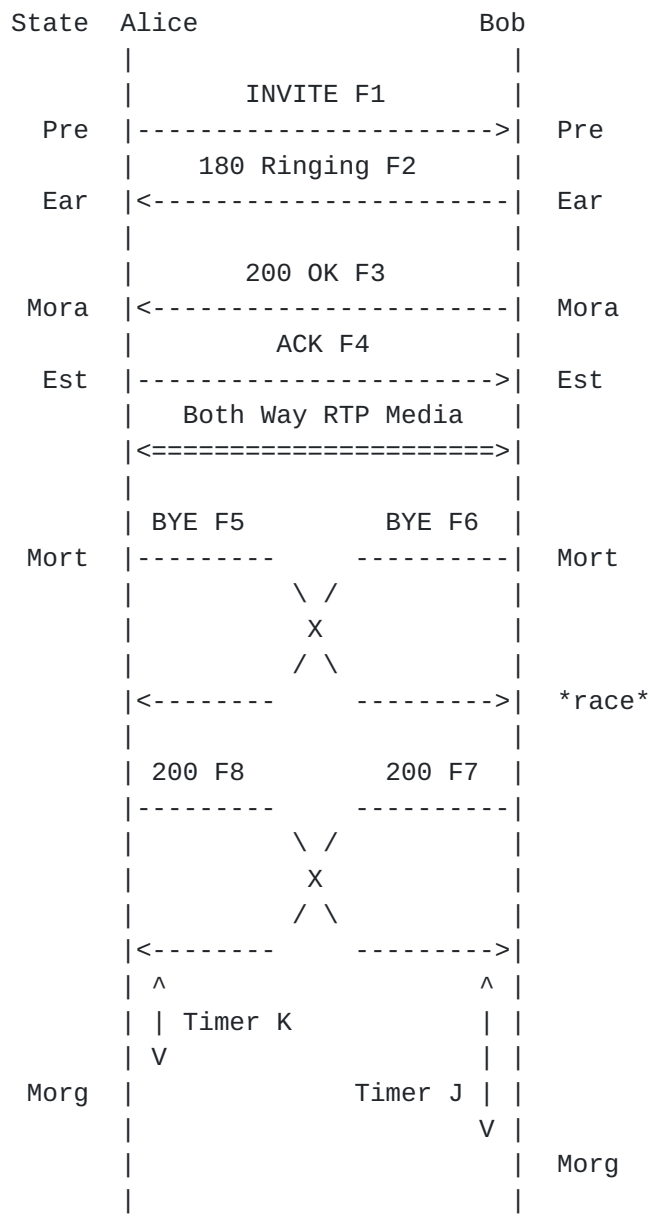
```
s=-
```

```
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* UAS retransmits a 200 OK to an ini-INVITE since it didn't receive
   an ACK.  */


F6 BYE Alice -> Bob

BYE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0

/* Bob retransmits a 200 OK and Alice sends a BYE at the same time.
   Alice has transited to the Mortal state, so she does not begin a
   session after this even though she receives a 200 response to
   the re-INVITE.  */

F7 ACK(=F4) Alice -> Bob

F8 200 OK (BYE) Bob -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0

/* Bob sends a 200 OK to a BYE.  */
```

3.2 Receiving message in the Mortal State

This section shows some examples of call flow in race condition when receiving the message from other states in the Mortal state.

3.2.1 Receiving BYE (Establish state) in Mortal state



This scenario illustrates the race condition which occurs when UAS receives an Established message (BYE) in Mortal state. Alice and Bob send a BYE at the same time. A dialog and session is ended shortly after a BYE request is passed to a client transaction. As shown in section 2, UA remains in Mortal state. UAs in Mortal state return error responses to the requests that operate dialog or session, such as re-INVITE, UPDATE, or REFER. However, UA shall return 200 OK to the BYE because the use case is taken into consideration that a BYE message are sent by both a caller and a callee to exchange reports about the session when it is being terminated.
(Since the dialogue and the session both terminate when a BYE

is sent, the choice of sending 200 or error response upon receiving BYE in Mortal state does not affect the resulting

termination. Therefore, even though this example uses a 200
response, other responses can be used.)

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 BYE Alice -> Bob

```
BYE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

```
/* The session is terminated at the moment Alice sends a BYE.
   The dialog still exists then, but it is certain to be
   terminated in a short period of time. The dialog is
   completely terminated when the timeout of the BYE request
   occurs. */
```

F6 BYE Bob -> Alice

```
BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

```
/* Bob has also transmitted a BYE simultaneously with Alice.
   Bob terminates a session and a dialog. */
```

F7 200 OK Bob -> Alice

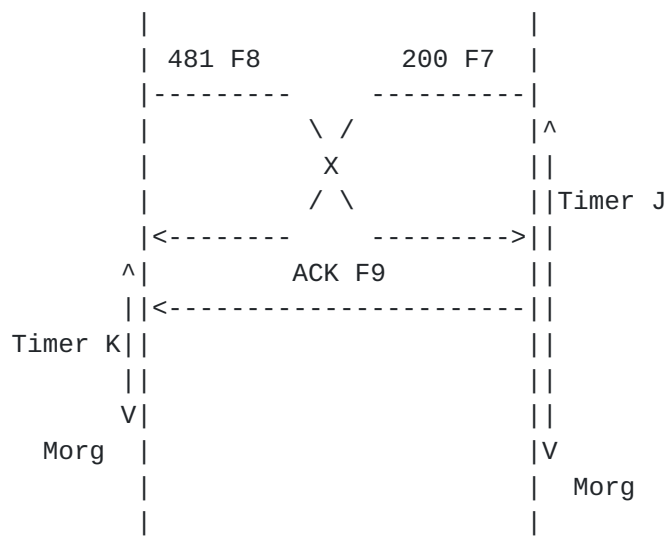
```
   SIP/2.0 200 OK
```

```
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
    ;received=192.0.2.201
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 2 BYE
   Content-Length: 0

   /* Since the dialog is Moratorium state, Bob responds with
      a 200 to the BYE.  */


   F8 200 OK Alice -> Bob

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
    ;received=192.0.2.201
   From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 BYE
   Content-Length: 0

   /* Since Alice has transited from the established state to Mortal
      state by sending a BYE, Alice responds with a 200 to a BYE.  */
```

3.2.2 Receiving re-INVITE (Establish state) in Mortal state

```
  State  Alice                  Bob
         |                       |
         |       INVITE F1       |
    Pre  |---------------------->|  Pre
         |    180 Ringing F2     |
    Ear  |<----------------------|  Ear
         |                       |
         |       200 OK F3       |
   Mora  |<----------------------|  Mora
         |         ACK F4        |
    Est  |---------------------->|  Est
         |   Both Way RTP Media  |
         |<=====================>|
         |                       |
         | BYE F5     re-INVITE F6|
   Mort  |---------     ----------|
         |          \ /          |
         |           X           |
         |          / \          |
```

```
 *race*  |<--------     --------->|  Mort
```



This scenario illustrates the race condition which occurs when UAS receives an Established message (re-INVITE) in Mortal state. Bob sends a re-INVITE, and Alice sends a BYE at the same time. The re-INVITE of Bob is returned by a 481, since TU of Alice has transited from Established state to Mortal state by sending a BYE. Bob sends ACK to the 481 response, because ACK for error responses is handled by the transaction layer, and at the point of receiving the 481 the INVITE client transaction still remains (even though the dialog has been terminated).

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 BYE Alice -> Bob

```
BYE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

```
   /* Alice sends a BYE and terminates a session, and transits from
```

Established state to Mortal state. */

F6 re-INVITE Bob -> Alice

```
INVITE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

```
/* Alice sends a BYE, and Bob sends a re-INVITE at the same time.
   The state of dialog transits to Mortal state at the moment
   Alice sends a BYE, but Bob doesn't know it until he receives
   the BYE. Therefore, the dialog is Terminated state from
   Alice's point of view, but the dialog is Confirmed state
   from Bob's point of view. A race condition occurs. */
```

F7 200 OK Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
 ;received=192.0.2.201
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 2 BYE
Content-Length: 0
```

F8 481 Call/Transaction Does Not Exist Alice -> Bob

```
SIP/2.0 481 Call/Transaction Does Not Exist
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

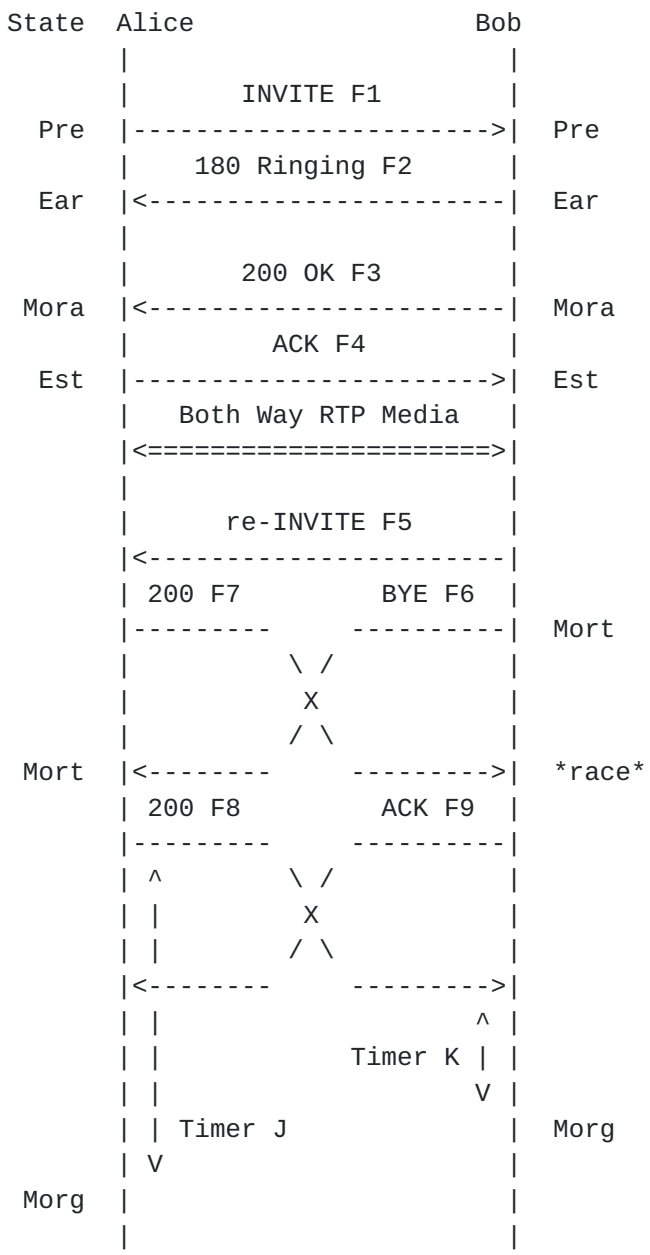
```
/* Since Alice is in Mortal state, she responds with a 481 to the
```

```
      re-INVITE.  */
```

F9 ACK Bob -> Alice

/* ACK for an error response is handled by Bob's INVITE client transaction. */

3.2.3 Receiving 200OK for re-INVITE (Established state) in Mortal state



This scenario illustrates the race condition which occurs when

UAS receives an Established message (200 to re-INVITE) in Mortal

state. Bob sends a BYE immediately after sending a re-INVITE.
(A user is not conscious that refresher sends a re-INVITE
automatically. For example, in the case of a telephone application,
it is possible that a user places a receiver immediately after
refresher.)
Bob sends ACK for a 2xx response when he receives 200 to INVITE
in the Mortal state, so that he completes the INVITE transaction.

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 re-INVITE Bob -> Alice

```
INVITE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F6 BYE Bob -> Alice

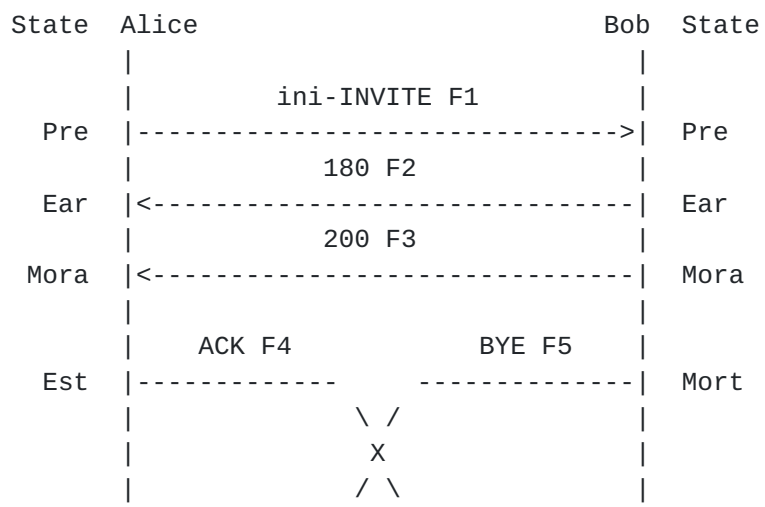
```
BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds8
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

```
/* Bob sends a BYE immediately after sending of a re-INVITE.
   Bob terminates a session and transits from Established
   state to Mortal state. */
```

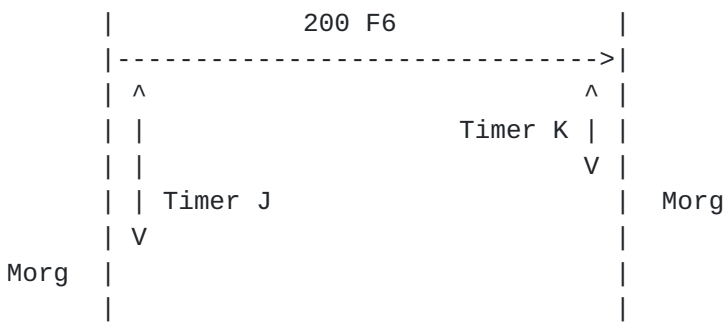
F7 200 OK (re-INVITE) Alice -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0

/* Bob sends a BYE, and Alice responds with a 200 OK to re-INVITE.
   A race condition occurs.  */


F8 200 OK (BYE) Alice -> Bob

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds8
 ;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0


F9 ACK Bob -> Alice

/* Bob sends ACK in the Mortal state to complete the three-way handshake
   of the INVITE transaction */
```

3.2.4 Receiving ACK (Moratorium state) in Mortal state



```
   Mort  |<------------     ------------->|  *race*
```



This scenario illustrates the race condition which occurs when UAS receives an Established message (ACK to 200) in Mortal state. Alice sends an ACK and Bob sends a BYE at the same time. When the offer is in a 2xx, and the answer is in an ACK, this example is in a race condition. Do not begin the session by receiving an ACK because Bob has already terminated the session by sending the BYE. The answer of ACK is just ignored.

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bd5
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

/* RTP streams are established between Alice and Bob */

F5 BYE Alice -> Bob

```
BYE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
```

```
   Content-Length: 0
```

```
/* Alice sends a BYE and terminates a session and dialog.  */
```

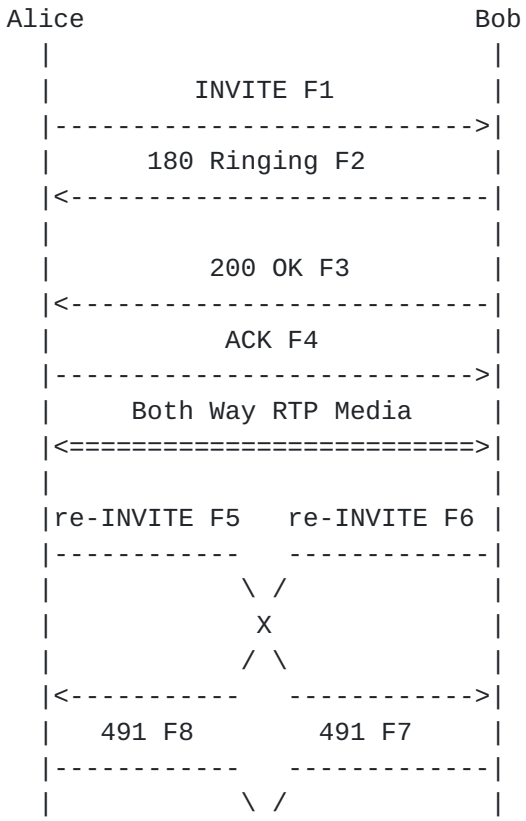
```
F6 200 OK Bob -> Alice
```

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKnashds8
 ;received=192.0.2.201
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

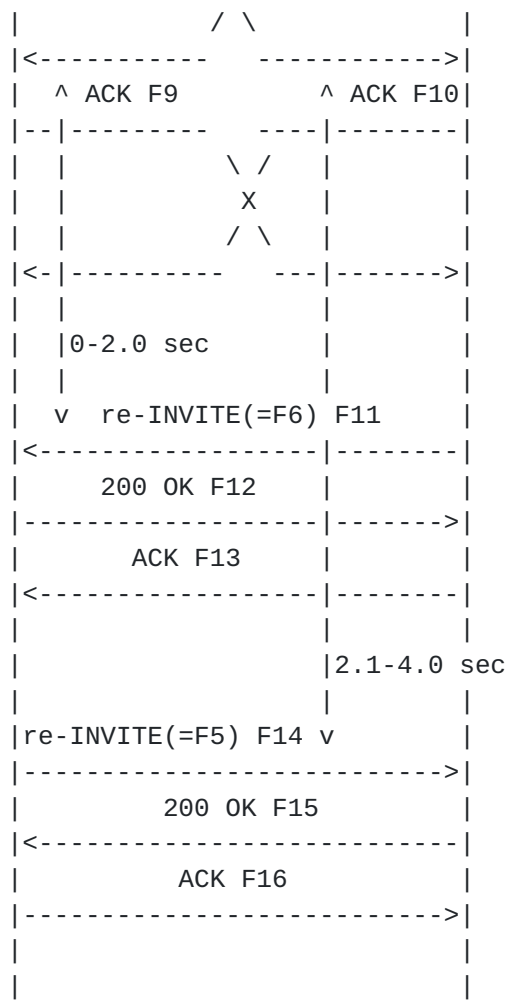
3.3 Other race condition

Here, examples in race condition that doesn't relate directly to the dialog state transition are shown. In this section, it is shown that how to treat the race condition which generated when UAs treat "What is established by SIP" which related closely with dialog.

3.3.1 re-INVITE crossover



| X |



In this scenario, Alice and Bob send a re-INVITE at the same time. When two re-INVITEs cross in the same dialog, they resend re-INVITEs after different intervals. (RFC3261, 14.1) When Alice sends an initial INVITE, an INVITE will be sent again after 2.1-4.0 seconds because she generated the Call-ID (owner of the Call-ID). Bob will send an INVITE again after 0.0-2.0 seconds, because Bob isn't the owner of the Call-ID. Therefore, each user agent must remember whether they has generated the Call-ID of the dialog or not, in case INVITEs may be crossed by another INVITE. In this example, Alice's re-INVITE is for session modification and Bob's re-INVITE is for session refresh. In this case, after the 491 responses, Bob retransmits re-INVITE for session refresh earlier than Alice. If Alice was to retransmit her re-INVITE (that is, if she was not the owner of Call-ID), the request would refresh and modify the session at the same time. Then Bob would know that he would not need to retransmit his re-INVITE to refresh the session. In another instance where two re-INVITEs for session modification cross over, retransmitting the same re-INVITE again after 491 by the

Call-ID holder (the UA which retransmits his re-INVITE after the other UA) may result in a behavior different from what the user

originally intended to, so the UA needs to decide if the retransmission of re-INVITE is necessary.
(For example, when a call hold and an addition of video cross over, mere retransmission of the re-INVITE at the firing of the timer may result in the situation where the video is transmitted immediately after the holding of the audio. This behavior is probably not intended by the users.)

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 re-INVITE Alice -> Bob

```
INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly

/* re-INVITE for session modification. (a=sendrecv -> sendonly) */
```

F6 re-INVITE Bob -> Alice

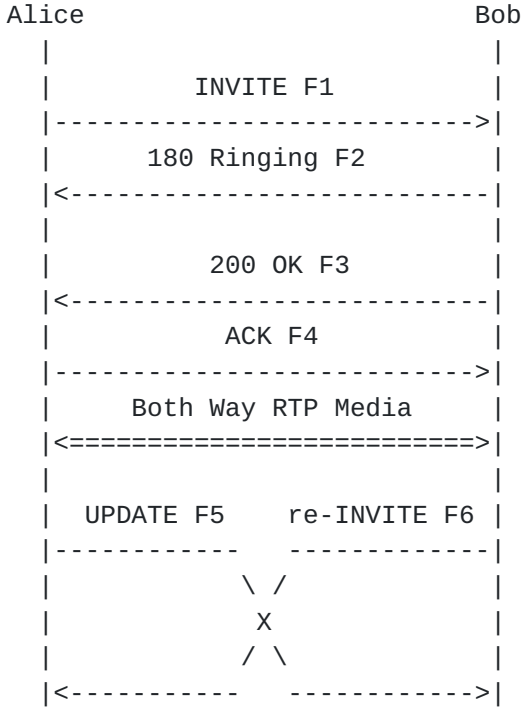
```
INVITE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70
```

```
   From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
```

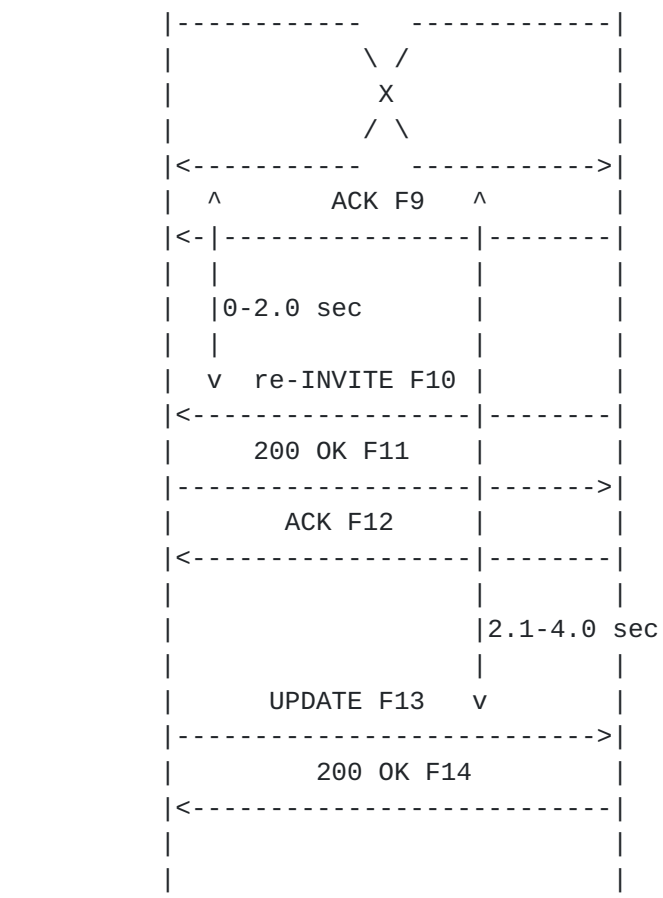
```
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 INVITE
   Content-Length: 0

   /* A case where a re-INVITE for a session refresh and a re-INVITE for
      hold are sent at the same time.  */


   F7 491 Request Pending Bob -> Alice
   /* Since an INVITE is in progress, a 491 response are returned.  */

   F8 491 Request Pending Alice -> Bob

   F9 ACK (INVITE) Alice -> Bob

   F10 ACK (INVITE) Bob -> Alice

   F11 re-INVITE Bob -> Alice

   INVITE sip:alice@client.atlanta.example.com SIP/2.0
   Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7.1
   Session-Expires: 300;refresher=uac
   Supported: timer
   Max-Forwards: 70
   From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 2 INVITE
   Content-Type: application/sdp
   Content-Length: 133

   v=0
   o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
   s=-
   c=IN IP4 192.0.2.201
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* Since Bob is not the owner of Call-ID, Bob sends an INVITE again
      after 0.0-2.0 seconds.  */


   F12 200 OK Alice -> Bob

   F13 ACK Bob -> Alice

   F14 re-INVITE Alice -> Bob
```

```
INVITE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 3 INVITE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly

/* Since Alice is the owner of Call-ID, Alice sends an INVITE again
   after 2.1-4.0 seconds.  */


F15 200 OK Bob -> Alice

F16 ACK Alice -> Bob
```

3.3.2 UPDATE and re-INVITE crossover



```
     |   491 F8        491 F7     |
```



In this scenario, the UPDATE contains SDP offer, therefore UPDATE and re-INVITE are returned error response (491) as in the case of "re-INVITE crossover". When an UPDATE for refresher which doesn't contain a session description and the re-INVITE crossed each other, both request don't fail by 491 and succeed with 200 because 491 means that UA have a pending request. Moreover, the same is equally true of UPDATE crossover, in case that either UPDATE contains a session description fail with 491, other cases succeed with 200.

Editor's Note:
A 491 response is considered a result that UA judged the effectiveness of request to "What is established by SIP". Therefore, it is considered that 491 will be used in all the requests that demand operation to "What is established by SIP".

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 UPDATE Alice -> Bob

```
UPDATE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 UPDATE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly
```

F6 re-INVITE Bob -> Alice

```
INVITE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0

/* A case where a re-INVITE for a session refresh and a re-INVITE for
   hold are sent at the same time.  */
```

F7 491 Request Pending Bob -> Alice

```
/* Since an INVITE is in process, a 491 response are returned.  */
```

F8 491 Request Pending Alice -> Bob

F9 ACK (re-INVITE) Alice -> Bob

F10 re-INVITE Bob -> Alice

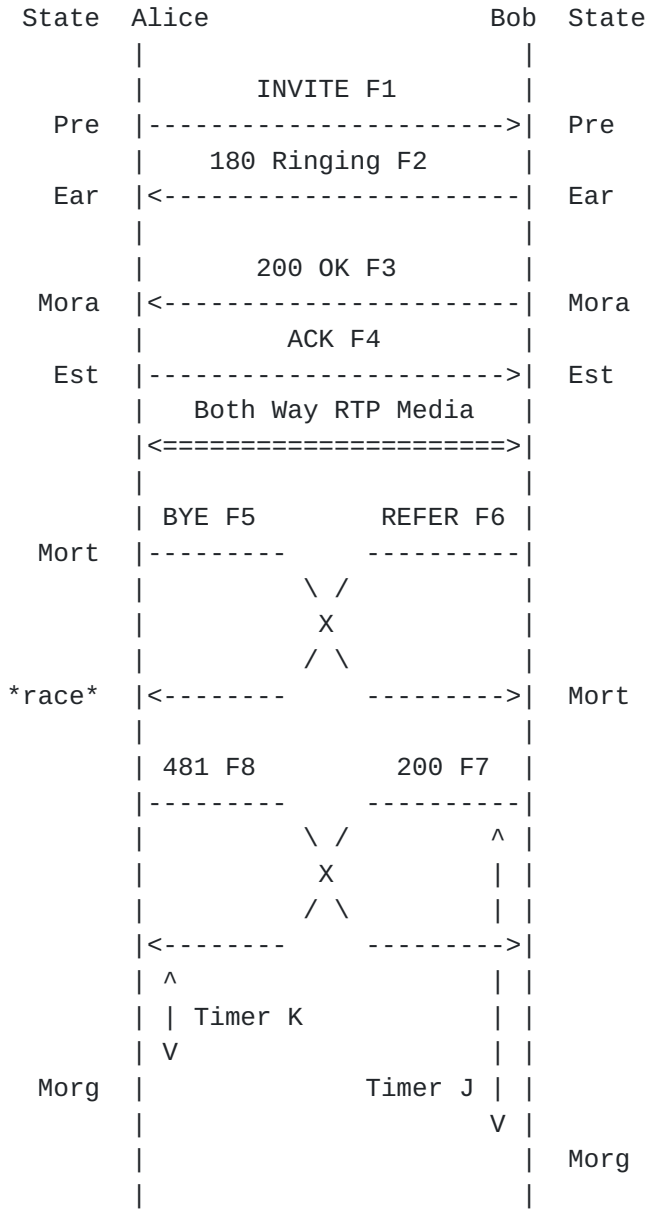
```
INVITE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7.1
Session-Expires: 300;refresher=uac
Supported: timer
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Type: application/sdp
Content-Length: 133

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Since Bob is not the owner of Call-ID, Bob sends an INVITE again
   after 0.0-2.0 seconds.  */


F11 200 OK Alice -> Bob

F12 ACK Bob -> Alice

F13 UPDATE Alice -> Bob

UPDATE sip:sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 3 UPDATE
Content-Length: 147

v=0
o=alice 2890844526 2890844527 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=sendonly
```

```
   /* Since Alice is the owner of Call-ID, Alice sends an INVITE again
```

after 2.1-4.0 seconds. */

F14 200 OK Bob -> Alice

3.3.3 Receiving REFER (Establish state) in Mortal state



This scenario illustrates the race condition which occurs when UAS receives an Established message (REFER) in Mortal state. Bob sends a REFER, and Alice sends a BYE at the same time. Bob

send a REFER in the same dialog. Alice sends an error response
to request like a REFER which operates the session, because,

by sending a BYE, Alice had terminated the session which would have corresponded to the REFER. For handling of dialogs with multiple usages, as can be seen in the use of REFER method, see the draft on dialog usage [8].

Message Details

F1 INVITE Alice -> Bob

F2 180 Ringing Bob -> Alice

F3 200 OK Bob -> Alice

F4 ACK Alice -> Bob

F5 BYE Alice -> Bob
/* Alice sends a BYE and terminates a session, and transits from Confirmed state to Terminnated state. */

F6 REFER Bob -> Alice

```
REFER sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
Refer-To: sip:carol@cleveland.example.org
Contact: <sip:bob@client.biloxi.example.com;transport=udp>
CSeq: 1 REFER
Content-Length: 0
```

/* Alice sends a BYE, and Bob sends a REFER at the same time.
Bob sends a REFER on the INVITE dialog.
The state of dialog transits to Mortal state at the moment
Alice sends a BYE, but Bob doesn't know it until he receives
the BYE. A race condition occurs. */

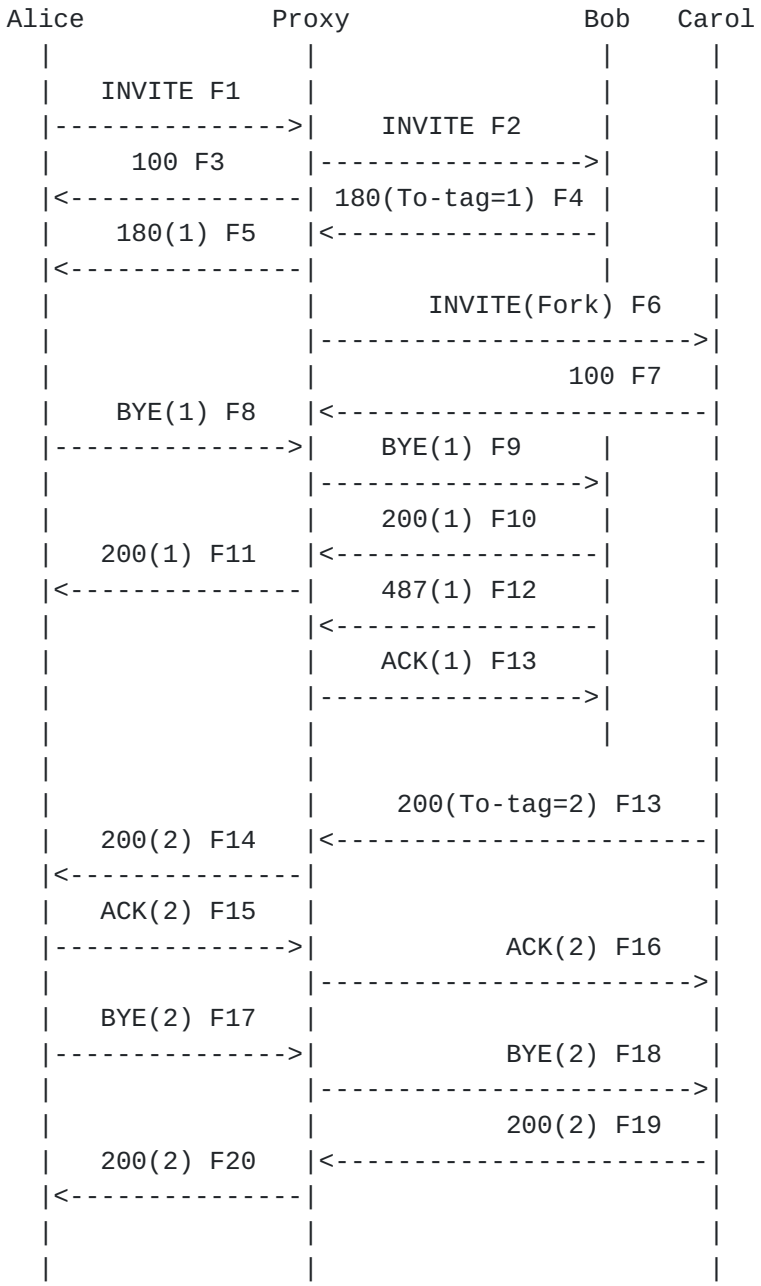
F7 200 OK Bob -> Alice

F8 481 Call/Transaction Does Not Exist Alice -> Bob
/* Since Alice is terminated the session, she responds with a 481 to the REFER. */

Appendix A - BYE on the Early Dialog

This section, related to 3.1.3, explains why BYE is not recommended

in Early state, illustrating the case in which BYE in Early dialog triggers confusion.



Care is advised in sending of BYE in Early state when a proxy may fork. In the example, the BYE request progresses normally, and it succeeds in correctly terminating the dialog with Bob. After Bob terminates the dialog by sending BYE, he sends 487 to the ini-INVITE. According to Section 15.1.2 of RFC3261 [1], it is RECOMMENDED for UAS to

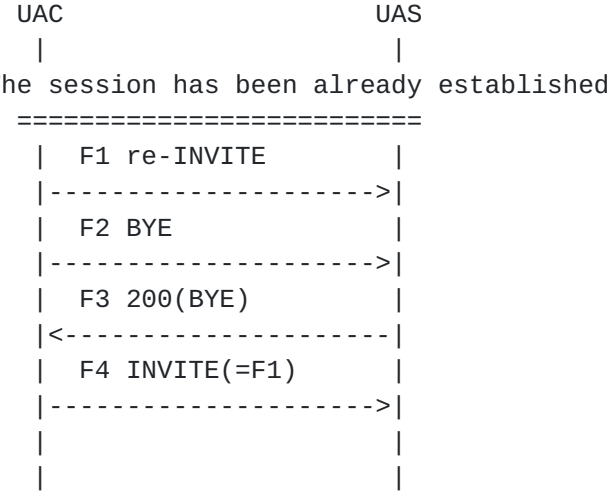
generate 487 to any pending requests after receiving BYE.
In the example, Bob sends 487 to ini-INVITE since he receives

BYE with the ini-INVITE in pending state.

However, Alice receives a final response for INVITE (a 200 from Carol) even though she has successfully terminated the dialog with Bob. This means that, regardless of the success/ failure of the BYE in Early state, Alice MUST be prepared for the establishment of a new dialog until receiving the final response for the INVITE and terminating the INVITE transaction.

The choice of BYE or CANCEL in the early state must be made carefully. CANCEL is appropriate when the goal is to abandon the call attempt entirely. BYE is appropriate when the goal is to abandon a particular early dialog while allowing the call to be completed to other destinations. When using either BYE or CANCEL the UAC must be prepared for the possibility that a call may still be established to one (or more) destinations.

Appendix B - BYE request overlapped on re-INVITE

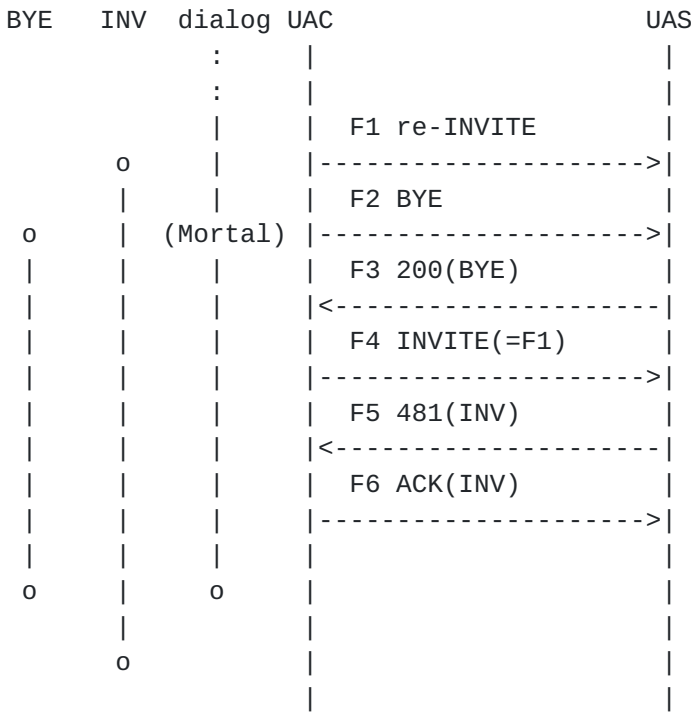


This case could look similar to the one in 3.2.3. However, it is not a race condition but the behavior where there is no response for INVITE for some reasons. The appendix explains the behavior in this case and its rationale behind, since this case is likely to cause confusion.
First of all, it is important not to confuse the behavior of the transaction layer and that of the dialog layer. RFC3261 details the Transaction layer behavior, and this document explains the dialog layer behavior. It has to be noted that these behaviors are independent of each other, even though the both layers transit their states triggered by sending or receiving of the same SIP messages (A dialog can be terminated even though a transaction is still remaining, and vice versa).

In the sequence above, there is no response for F1, and F2 (BYE) is sent immediately after F1 (F1 is a mid-dialog request. If F1

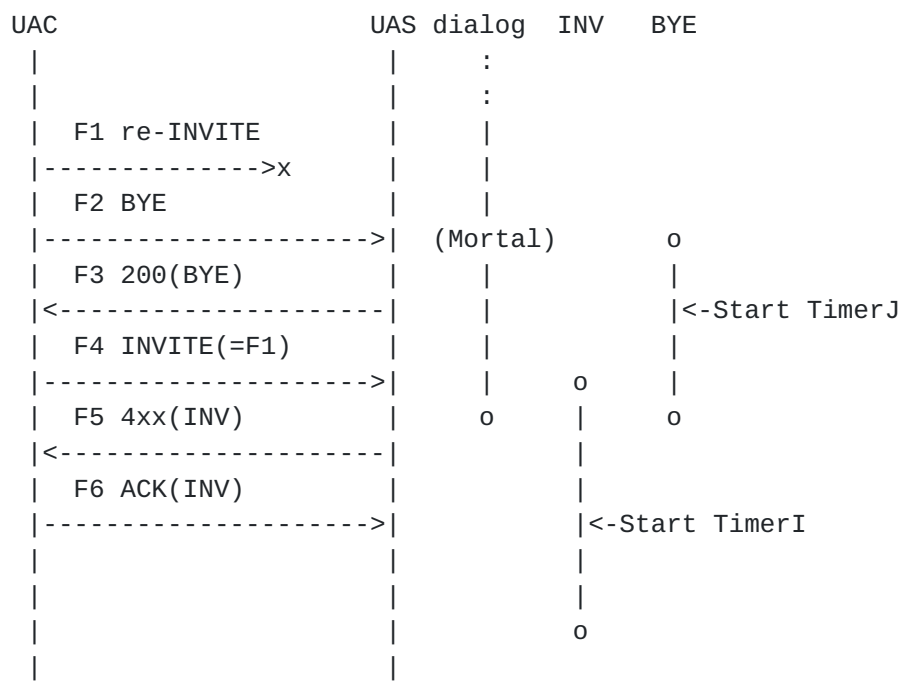
was ini-INVITE, BYE could not be sent before UAC received a provisional response with To-tag for the request).

Below is a figure which illustrates UAC's dialog state and transaction state.



For UAC, the INVITE client transaction begins at the point F1 is sent. The UAC sends BYE (F2) immediately after F1. This is a legitimate behavior. (Usually the usage of each SIP method is independent, for BYE and others. However, it should be noted that it is prohibited to send a request with a SDP offer while the previous offer is in progress.)
After that, F2 triggers the BYE client transaction. At the same time, the dialog state transits to the Mortal state and since then only a BYE or its response can be handled.
It is permitted to send F4 (a retransmission of INVITE) in the Mortal state, because the retransmission of F1 is handled by the transaction layer, and the INVITE transaction has not yet transited to the Terminated state. As it is mentioned above, the dialog and the transaction behave independently each other. Therefore the transaction handling has to be continued even though the dialog moved to the Terminated state.

Next, UAS's state is shown below.



For UAS, it can be regarded that F1 packet is lost or delayed. (Here the behavior is explained for the case UAS receives F2 BYE before F1 INVITE) Therefore, F2 triggers the BYE transaction for UAS, and simultaneously the dialog moves to the Mortal state. Then, upon the reception of F4 the INVITE server transaction begins. (It is allowed to start the INVITE server transaction in the Mortal state. The INVITE server transaction begins for handling received SIP request regardless of dialog state.)
UAS's TU sends an appropriate error response (probably 481) for F4 INVITE, because the TU knows that the dialog which matches to the INVITE is in the Terminated state.
(It is mentioned above that F4 (and F1) INVITE is a mid-dialog request. Mid-dialog requests have a To-tag. It should be noted that UAS's TU does not begin a new dialog upon the reception of INVITE with a To-tag.)

Appendix C - UA's behaviour for CANCEL

This section explains the CANCEL and the Expires header behaviors which indirectly involve in the dialog state transition in the Early state. CANCEL does not have any influence on UAC's dialog state. However, the request has indirect influence on the dialog state transition because it has a significant effect on ini-INVITE. Similarly, the Expires header does not have direct influence on the dialog state transition, but it indirectly affect the state transition because its expiration triggers the sending of CANCEL.

For UAS the CANCEL request and the Expires header timeout have more direct effects on the dialog than the sending of CANCEL by UAC,

because they can be a trigger to send the 487 response. The figure3 explains UAS's behavior in the Early state. This flow diagram is only explanatory figure, and the actual dialog state transition is as illustrated in Figure 1, 2.

In the flow, full lines are related to dialog state transition, and dotted lines are involved with CANCEL. (r) represents the reception of signals, and (s) means sending. There is no dialog state for CANCEL, but here the Cancelled state is virtually handled just for the ease of understanding of UA's behavior when it sends and receives CANCEL.

Next, UAS's flow is explained.

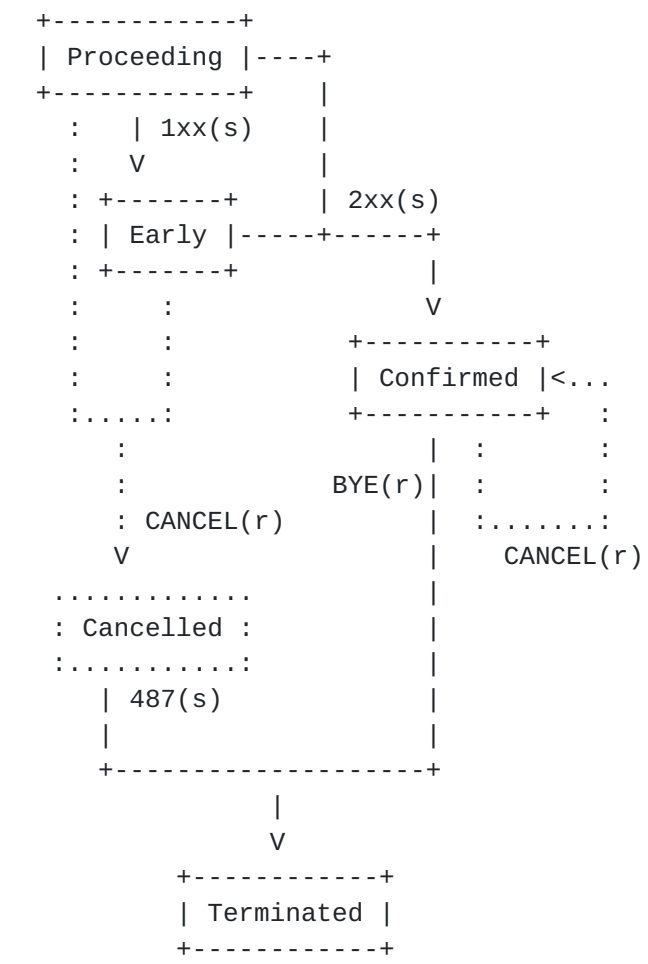


figure 3. CANCEL flow diagram for UAS

There are two cases for UAS depending on the state when it receives CANCEL.

One is when UAS receives CANCEL in the Proceeding and Early states. In this case the UAS immediately sends 487 for the INVITE, and the

dialog transits to the Terminated state.
The other is the case in which UAS receives CANCEL in the Confirmed state. In this case the dialog state transition does not occur because UAS has already sent the final response for the INVITE to which the CANCEL is targeted.
(Note that, from the point of UAC's behavior, it can be expected that UAS receives BYE immediately after the reception of CANCEL and moves to the Terminated state. However, the UAS's state does not transit until it actually receives BYE.)

Appendix D - Notes on the request in Mortal state

This section describes UA's behavior in the Mortal state which need careful attention.
In the Mortal state, only a BYE or its response can be handled, and no other messages can be received. However, sending of ACK and the authentication procedure for BYE are conducted in this state.

ACK for error responses is handled by the transactiuon layer, so the handling is not related to the dialog state. Unlike ACK for error responses, ACK for 2xx responses is a request newly generated by TU. However, the ACK for 2xx and the one for error responses are both a part of the INVITE transaction, even though their hadlings differ. (Section 17.1.1.1 of RFC3261 [1])
Therefore, the INVITE transaction is completed by three-way handshake, which includes ACK, even in the Mortal state.

BYE authentication procedure shall be processed in the Mortal state. When anthentication is requested by 401 or 407 response, UAC resends BYE with an appropriate certificate. Also UAS handles the retransmission of the BYE which it requested anthentication itself.

References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with SDP", RFC 3264, April 2002.

[3] Johnston, A., Donovan, S., Sparks, R., Cunningham, C. and K. Summers, "Session Initiation Protocol (SIP) Basic Call Flow Examples", BCP 75, RFC 3665, December 2003.

[4] Johnston, A., Donovan, S., Sparks, R., Cunningham, C. and K. Summers, "Session Initiation Protocol (SIP) Public Switched

Telephone Network (PSTN) Call Flows", BCP 76, RFC 3666, December

2003.

[5] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[6] Rosenberg, J. and H. Schulzrinne, "Reliability of Provisional Responses in the Session Initiation Protocol (SIP)", RFC 3262, June 2002.

[7] Rosenberg, J., Schulzrinne, H., Mahy, R., "An INVITE-Initiated Dialog Event Package for the Session Initiation Protocol (SIP)", RFC 4235, November 2005.

[8] Sparks, R., "Multiple Dialog Usages in the Session Initiation Protocol", draft-ietf-sipping-dialogusage-03 (work in progress), August 29, 2006.

Author's Addresses

All listed authors actively contributed large amounts of text to this document.

Miki Hasebe
NTT-east Corporation
19-2 Nishi-shinjuku 3-chome Shinjuku-ku Tokyo 163-8019 Japan

EMail: hasebe.miki@east.ntt.co.jp

Jun Koshiko
NTT-east Corporation
19-2 Nishi-shinjuku 3-chome Shinjuku-ku Tokyo 163-8019 Japan

EMail: j.koshiko@east.ntt.co.jp

Yasushi Suzuki
NTT-east Corporation
19-2 Nishi-shinjuku 3-chome Shinjuku-ku Tokyo 163-8019 Japan

EMail: suzuki.yasushi@east.ntt.co.jp

Tomoyuki Yoshikawa
NTT-east Corporation
19-2 Nishi-shinjuku 3-chome Shinjuku-ku Tokyo 163-8019 Japan

EMail: tomoyuki.yoshikawa@east.ntt.co.jp

Paul H. Kyzivat
Cisco Systems, Inc.
1414 Massachusetts Avenue
Boxborough, MA 01719
USA

Email: pkyzivat@cisco.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

to the rights, licenses and restrictions contained in BCP 78, and

Acknowledgment

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