Internet Engineering Task Force
Internet Draft

SIPPING WG
J. Rosenberg
dynamicsoft
J. Peterson
Neustar
H. Schulzrinne
Columbia U.
G. Camarillo
Ericsson

draft-ietf-sipping-3pcc-00.txt
May 10, 2002
Expires: November 2002

Best Current Practices for Third Party Call Control in the Session Initiation Protocol

STATUS OF THIS MEMO

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

Abstract

Third party call control refers to the ability of one entity to create a call in which communications is actually between other parties. Third party call control is possible using the mechanisms specified within the Session Initiation Protocol (SIP). However, there are several possible approaches, each with different benefits and drawbacks. This document discusses best current practices for the usage of the SIP for third party call control.

Table of Contents

1 Introduction

(Note to RFC Editor - please replace all instances of RFC BBBB with RFC 3261 when draft-ietf-sip-rfc2543bis is published as an RFC. Please replace all instances of RFC MMMM with the RFC number of draft-ietf-sip-manyfolks-resource when it issues as an RFC.)

In the traditional telephony context, third party call control allows one entity (which we call the controller) to set up and manage a communications relationship between two or more other parties. Third party call control (referred to as 3pcc) is often used for operator services (where an operator creates a call that connects two participants together), and conferencing.

Similarly, many SIP services are possible through third party call control. These include the traditional ones on the PSTN, but also new ones such as click-to-dial. Click-to-dial allows a user to click on a web page when they wish to speak to a customer service representative. The web server then creates a call between the user and a customer service representative. The call can be between two phones, a phone and an IP host, or two IP hosts.

Third party call control is possible using only the mechanisms specified within RFC BBBB [1]. Indeed, many different call flows are possible, each of which will work with SIP compliant user agents. However, there are benefits and drawbacks to each of these flows. The usage of third party call control also becomes more complex when aspects of the call utilize SIP extensions or optional features of SIP. In particular, the usage of RFC MMMM [2] (used for coupling of signaling to resource reservation) with third party call control is non-trivial. Similarly, the usage of early media (where session data is exchanged before the call is accepted) with third party call control is not trivial.

This document serves as a best current practice for implementing third party call control. Section 4 presents the known call flows that can be used to achieve third party call control, and provides guidelines on their usage. Section 8 discusses the interactions of RFC MMMM [2] with third party call control. Section 7 discusses the interactions of early media with third party call control. Section 9 provides example applications that make usage of the flows recommended here.

2 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [3] and

indicate requirement levels for compliant implementations.

3 Definitions

The following terms are used throughout this document:

3pcc: Third Party Call Control, which refers to the general ability to manipulate calls between other parties.

Controller: A controller is a SIP User Agent that wishes to create a session between two other user agents.

4 3pcc Call Establishment

The primary primitive operation of third party call control is the establishment of a session between participants A and B. Establishment of this session is orchestrated by a third party, referred to as the controller.

This section documents three call flows that the controller can utilize in order to provide this primitive operation.

4.1 Flow I

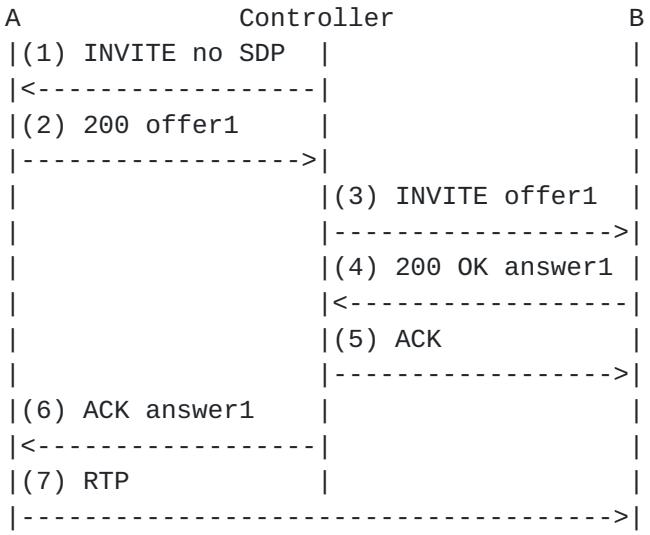


Figure 1: 3pcc Flow I

The call flow for Flow I is shown in Figure 1. The controller first

sends an INVITE A (1). This INVITE has no session description. A's phone rings, and A answers. This results in a 200 OK (2) that contains an offer [4]. The controller needs to send its answer in the ACK, as mandated by [1]. To obtain the answer, it sends the offer it got from A (offer1) in an INVITE to B (3). B's phone rings. When B answers, the 200 OK (4) contains the answer to this offer, answer1. The controller sends an ACK to B (5), and then passes answer1 to A in an ACK sent to it (6). Because the offer was generated by A, and the answer generated by B, the actual media session is between A and B. Therefore, media flows between them (7).

This flow is simple, requires no manipulation of the SDP by the controller, and works for any media types supported by both endpoints. However, it has a serious timeout problem. User B may not answer the call immediately. The result is that the controller cannot send the ACK to A right away. This causes A to retransmit the 200 OK response periodically. As specified in RFC BBBB Section 13.3.1.4, the 200 OK will be retransmitted for 64*T1 seconds. If an ACK does not arrive by then, the call is considered to have failed. This limits the applicability of this flow to scenarios where the controller knows that B will answer the INVITE immediately.

4.2 Flow II

An alternative flow, Flow II, is shown in Figure 2. The controller first sends an INVITE user A (1). This is a standard INVITE, containing an offer (sdp1) with a single audio media line, one codec, a random port number (but not zero), and a connection address of 0.0.0.0. This creates an initial media stream that is "black holed", since no media (or RTCP packets [8]) will flow from A. The INVITE causes A's phone to ring.

When A answers (2), the 200 OK contains an answer, sdp2. the controller sends an ACK (4). It then generates a second INVITE (3). This INVITE is addressed to user B, and it contains sdp2 as the offer to B. Note that the role of sdp2 has changed. In the 200 OK (message 2), it was an answer, but in the INVITE, it is an offer. Fortunatly, all valid answers are valid initial offers. This INVITE causes B's phone to ring. When it answers, it generates a 200 OK (5) with an answer, sdp3. The controller then generates an ACK (6). Next, it sends a re-INVITE to A (7) containing sdp3 as the offer. Once again, there has been a reversal of roles. sdp3 was an answer, and now it is an offer. Fortunately, an answer to an answer recast as an offer is, in turn, a valid offer. This re-INVITE generatea a 200 OK (8) with sdp2, assuming that A doesn't decide to change any aspects of the session as a result of this re-INVITE. This 200 OK is ACKed (9), and then media can flow from A to B. Media from B to A could already

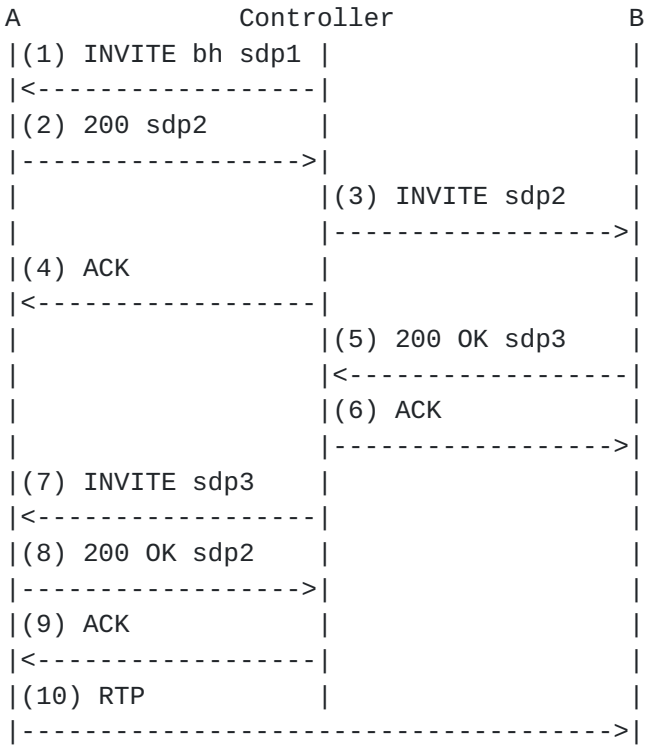


Figure 2: 3pcc Flow II

start flowing once message 5 was sent.

This flow has the advtange that all final responses are immediately ACKed. If therefore does not suffer from the timeout and message inefficiency problems of flow 1. However, it too has troubles. First off, it requires that the controller know the media types to be used for the call (since it must generate a "blackhole" SDP, which requires media lines). Secondly, the first INVITE to A (1) contains media with a 0.0.0.0 connection address. The controller expects that the response contains a valid, non-zero connection address for A. However, experience has shown that many UAs respond to an offer of a 0.0.0.0 connection address with an answer containing a 0.0.0.0 connection address. The offer-answer specification [4] now explicitly tells implementors not to do this, but at the time of publication of this document, many implementations still did. If A should respond with a 0.0.0.0 connection address in sdp2, the flow will not work.

However, the most serious flaw in this flow is the assumption that the 200 OK to the re-INVITE (message 8) contains the same SDP as in message 2. This may not be the case. If it is not, the controller

needs to re-INVITE B with that SDP (say, sdp4), which may result in getting a different SDP, sdp5 , in the 200 OK from B. Then, the controller needs to re-INVITE A again, and so on. The result is an infinite loop of re-INVITEs. It is possible to break this cycle by having very smart UAs which can return the same SDP whenever possible, or really smart controllers that can analyze the SDP to determine if a re-INVITE is really needed. However, we wish to keep this mechanism simple, and avoid SDP awareness in the controller. As a result, this flow is not really workable. It is therefore NOT RECOMMENDED.

4.3 Flow III

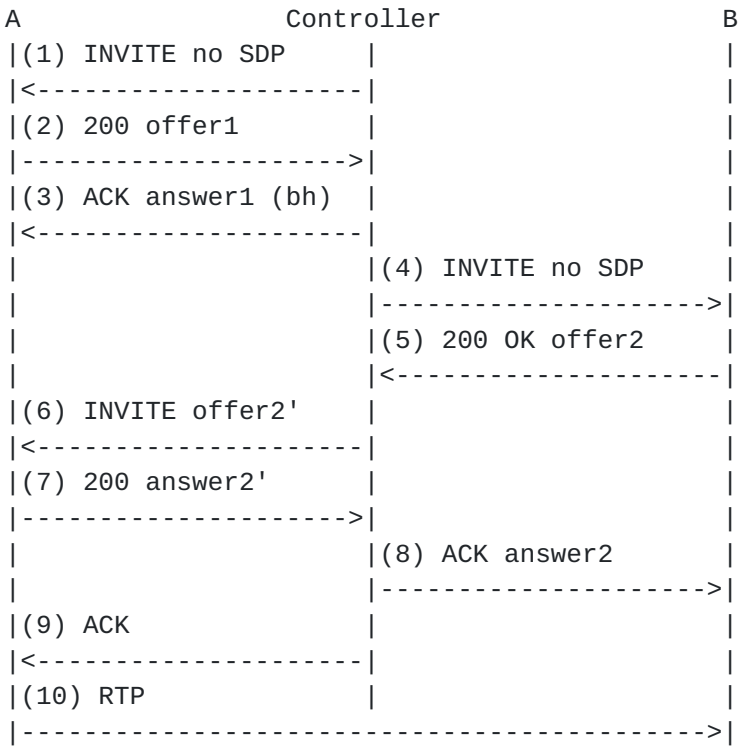


Figure 3: 3pcc Flow III

A thid flow, Flow III, is shown in Figure 3.

First, the controller sends an INVITE (1) to user A without any SDP (which is good, since it means that the controller doesn't need to assume anything about the media composition of the session). A's phone rings. When A answers, a 200 OK is generated (2) containing its

offer, offer1. The controller generates an immediate ACK containing an answer (3). This answer is a "black hole" SDP, with its connection address set to 0.0.0.0.

The controller then sends an INVITE to B without SDP (4). This causes B's phone to ring. When they answer, a 200 OK is sent, containing their offer, offer2 (5). This SDP is used to create a re-INVITE back to A (6). That re-INVITE is based on offer2, but may need to be reorganized to match up media lines, or to trim media lines. For example, if offer1 contained an audio and a video line, in that order, but offer2 contained just an audio line, the controller would need to add a video line to the offer (setting its port to zero) to create offer2'. Since this is a re-INVITE, it should complete quickly in the general case. Thats good, since user B is retransmitting their 200 OK, waiting for an ACK. The SDP in the 200 OK (7) from A, answer2', may also need to be reorganized or trimmed before sending it an the ACK to B (8) as answer2. Finally, an ACK is sent to A (9), and then media can flow.

This flow has many benefits. First, it will usually operate without any spurious retransmissions or timeouts (although this may still happen if a re-INVITE is not responded to quickly). Secondly, it does not require the controller to guess the media that will be used by the participants. Thirdly, it does not assume that a device responds properly to an INVITE with SDP containing a connection address of 0.0.0.0.

There are some drawbacks. The controller does need to perform SDP manipulations. Specifically, it must take some SDP, and generate another SDP which has the same media composition, but has connection addresses of 0.0.0.0. This is needed for message 3. Secondly, it may need to reorder and trim on SDP X, so that its media lines match up with those in some other SDP, Y. Thirdly, the offer from B (offer2) may have no codecs or media streams in common with the offer from A (offer 1). The controller will need to detect this condition, and terminate the call. Finally, the flow is far more complicated than the simple and elegant Flow I (Figure 1).

4.4 Flow IV

Flow IV shows a variation on Flow III that reduces its complexity. The actual message flow is identical, but the SDP placement and construction differs. The initial INVITE (1) contains SDP with no media at all, meaning that there are no m lines. This is valid, and implies that the media makeup of the session will be established later through a re-INVITE [4]. The 200 OK (2) has an answer with no media either. This is acknowledged by the controller (3). The flow

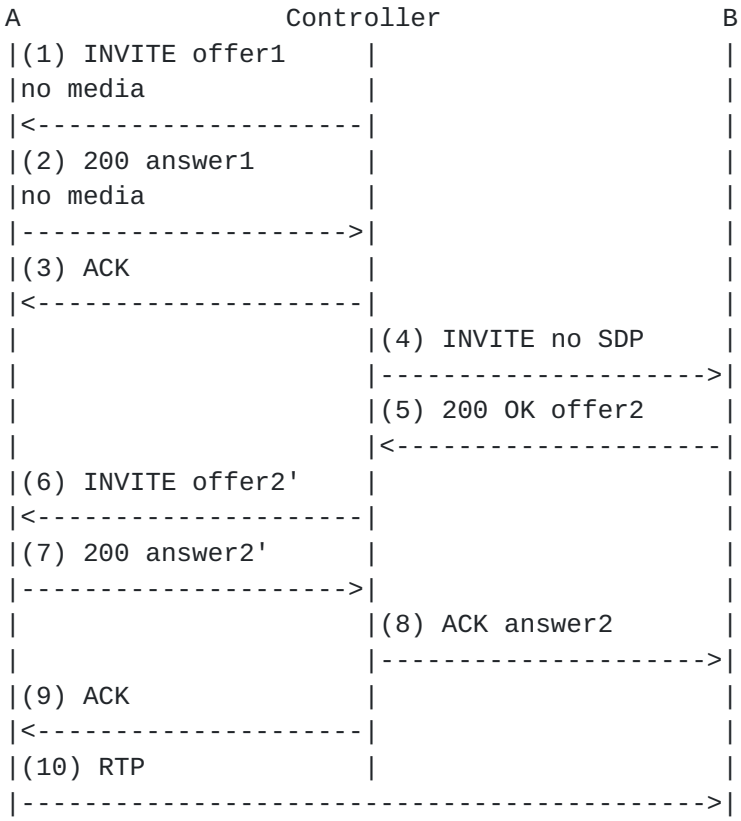


Figure 4: 3pcc Flow IV

from this point onwards is identical to Flow III. However, the manipuldations required to convert offer2 to offer2', and answer2' to answer2, are much simpler. Indeed, no media manipulations are needed at all. The only change that is needed is to modify the origin lines, so that the origin line in offer2' is valid based on the value in offer1 (validify requires that the version increments by one, and that the other parameters remain unchanged).

4.5 Recommendations

Flow I (Figure 1) represents the simplest and the most efficient flow. This flow SHOULD be used by a controller if it knows with certainty that user B is actually an automata that will answer the call immediately. This is the case for devices such as media servers, conferencing servers, and messaging servers, for example. Since we expect a great deal of third party call control to be to automata, special caseing this scenario is reasonable.

For calls to unknown entities, or to entities known to represent people, it is RECOMMENDED that Flow IV (Figure 4) be used for third party call control. Flow III MAY be used instead, but it provides no additional benefits over Flow IV. However, Flow II SHOULD NOT be used, because of the potential for infinite ping-ponging of re-INVITEs.

Several of these flows use a "black hole" connection address of 0.0.0.0. This is an IPV4 address with the property that packets sent to it will never leave the host which sent them; they are just discarded. Those flows are therefore specific to IPv4. For other network or address types, an address with an equivalent property SHOULD be used.

5 Error Handling

With all of the call flows in Section 4, one call is established to A, and then the controller attempts to establish a call to B. However, this call attempt may fail, for any number of reasons. User B might be busy (resulting in a 486 response to the INVITE), there may not be any media in common, the request may time out, and so on. If the call attempt to B should fail, it is RECOMMENDED that the controller send a BYE to A. This BYE SHOULD include a Reason header [5] which carries the status code from the error response. This will inform A of the precise reason for the failure. The information is important from a user interface perspective. For example, if A was calling from a black phone, and B generated a 486, the BYE will contain a Reason code of 486, and this could be used to generate a local busy signal so that A knows that B is busy.

6 Continued Processing

Once the calls are established, both participants believe they are in a single point-to-point call. However, they are exchanging media directly with each other, rather than with the controller. The controller is involved in two dialogs, yet sees no media.

Since the controller is still a central point for signaling, it now has complete control over the call. If it receives a BYE from one of the participants, it can create a new BYE and hang up with the other participant. This is shown in Figure 5.

Similarly, if it receives a re-INVITE from one of the participants, it can forward it to the other participant. Depending on which flow was used, this may require some manipulation on the SDP before passing it on.

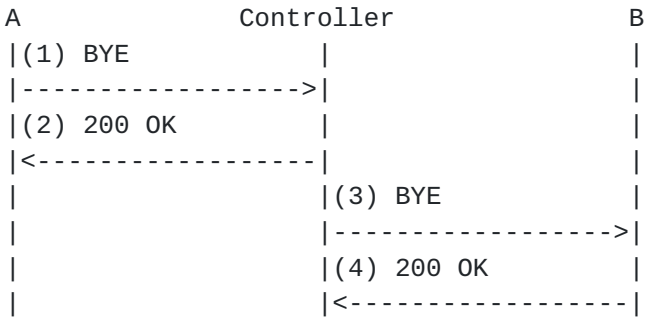


Figure 5: Hanging Up with 3PCC

However, the controller need not "proxy" the SIP messages received from one of the parties. Since it is a B2BUA, it can invoke any signaling mechanism on each dialog, as it sees fit. For example, if the controller receives a BYE from A, it can generate a new INVITE to a third party, C, and connect B to that participant instead A call flow for this is shown in Figure 6, assuming the case where C represents an end user, not an automata. Note that it is just Flow IV.

From here, new parties can be added, removed, transferred, and so on, as the controller sees fit.

It is important to point out that the call need not have been established by the controller in order for the processing of this section to be used. Rather, the controller could have acted as a B2BUA during a call established by A towards B (or vice a versa).

7 3pcc and Early Media

Early media represents the condition where the session is established (as a result of the completion of an offer/answer exchange), yet the call itself has not been accepted. This is usually used to convey tones or announcements regarding progress of the call. Handling of early media in a third party call is straightforward.

Figure 7 shows the case where user B generates early media before answering the call. The flow is almost identical to Flow IV from Figure 4. The only difference is that user B generates a reliable provisional response (5) [6] instead of a final response, and answer2 is carried in a PRACK (8) instead of an ACK. When party B finally

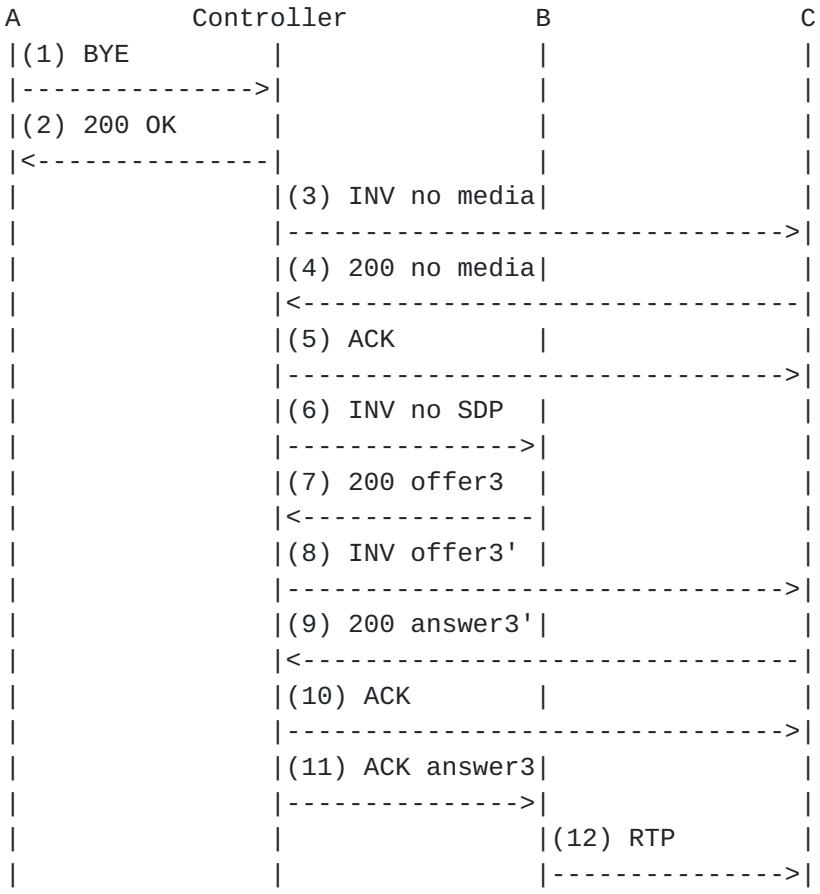


Figure 6: Alternative to Hangup

does accept the call (11), there is no change in the session state, and therefore, no signaling needs to be done with user A. The controller simply ACKs the 200 OK (12) to confirm the dialog.

The case where user A generates early media is more complicated, and is shown in Figure 8. The flow is based on Flow IV. The controller sends an INVITE to user A (1), with an offer containing no media streams. User A generates a reliable provisional response (2) containing an answer with no media streams. The controller PRACKs this provisional response (3). Now, the controller sends an INVITE without SDP to user B (5). User B's phone rings, and they answer, resulting in a 200 OK (6) with an offer, offer2. The controller now needs to update the session parameters with user A. However, since the call has not been answered, it cannot use a re-INVITE. Rather, it uses a SIP UPDATE request (7) [7], passing the offer (after modifying

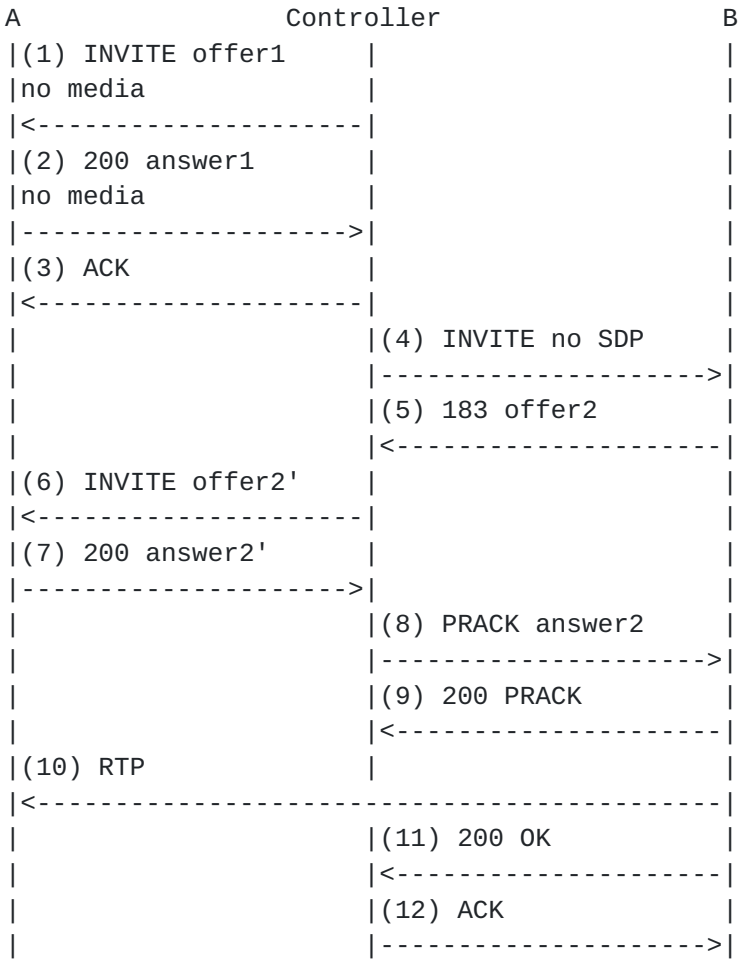


Figure 7: Early Media from User B

it to get the origin field correct). User A generates its answer in the 200 OK to the UPDATE (8). This answer is passed to user B in the ACK (9). When user A finally answers (11), there is no change in session state, so the controller simply ACKs the 200 OK (12).

Note that it is likely that there will be clipping of media in this call flow. User A is likely a PSTN gateway, and has generated a provisional response because of early media from the PSTN side. The PSTN will deliver this media even though the gateway does not have anywhere to send it, since the initial offer from the controller had no media streams. When user B answers, media can begin to flow. However, any media sent to the gateway from the PSTN up to that point will be lost.

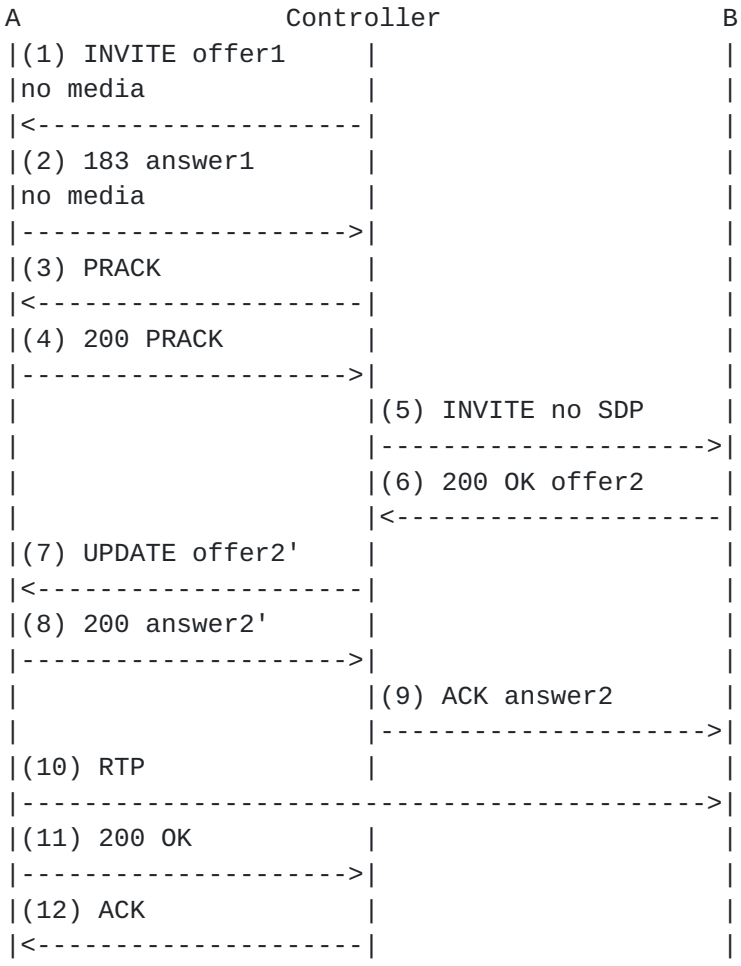


Figure 8: Early Media from User A

8 Third arty call control and SDP preconditions

A SIP extension has been specified that allows for the coupling of signaling and resource reservation [2]. This draft relies on exchanges of session descriptions before completion of the call setup. These flows are initiated when certain SDP parameters are passed in the initial INVITE. As a result, the interaction of this mechanism with third party call control is not obvious, and worth detailing.

Consider the call flow in Figure 9. The controller follows Flow IV; it has no specific requirements for support of the preconditions specification [2]. Indeed, there is no mechanism that can be used

with Flow IV which allows the controller to request preconditions. Therefore, it sends an INVITE (1) with SDP that contains no media lines. User A is interested in supporting preconditions, and does not want to ring its phone until resources are reserved. Since there are no media streams in the INVITE, it can't ring the phone until they are conveyed in a subsequent offer. Therefore, it generates a 183 with the answer, and doesn't alert the user (2). The controller PRACKs this (3) and A responds to the PRACK (4).

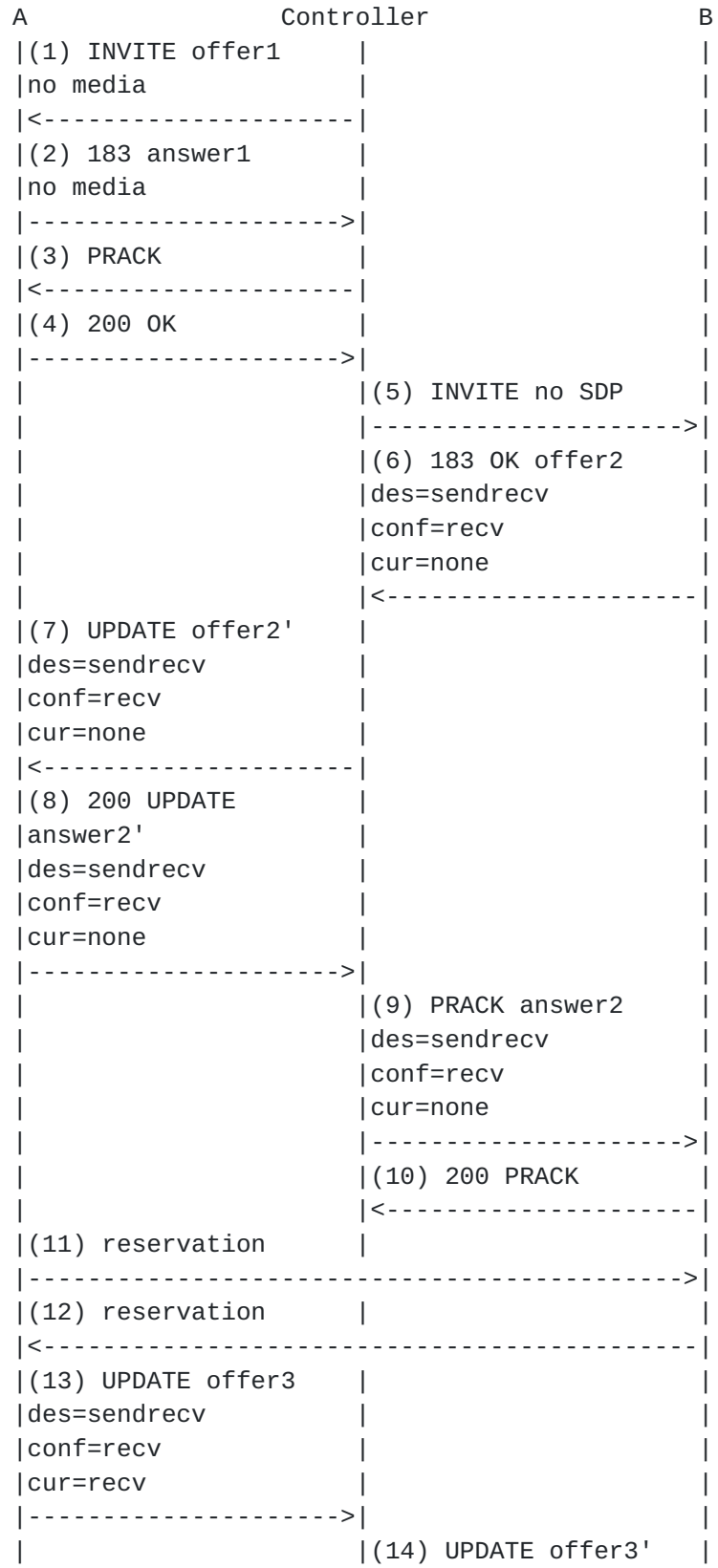
At this point, the controller attempts to bring B into the call. It sends B an INVITE without SDP (5). B is interested in having preconditions for this call. Therefore, it generates its offer in a 183 that contains the appropriate SDP attributes (6). The controller passes this offer to A in an UPDATE request (7). The controller uses UPDATE because the call has not been answered yet, and therefore, it cannot use a re-INVITE. User A sees that its peer is capable of supporting preconditions. Since it desires preconditions for the call, it generates an answer in the 200 OK (8) to the UPDATE. This answer, in turn, is passed to B in the PRACK for the provisional response (9). Now, both sides perform resource reservation. User A succeeds first, and passes an updated session description in an UPDATE request (13). The controller simply passes this to A (after the manipulation of the origin field, as required in Flow IV) in an UPDATE (14), and the answer (15) is passed back to A (16). The same flow happens, but from B to A, when B's reservation succeeds (17-20). Since the preconditions have been met, both sides ring (21 and 22), and then both answer (23 and 25), completing the call.

What is important about this flow is that the controller doesn't know anything about preconditions. It merely passes the SDP back and forth as needed. The trick is the usage of UPDATE and PRACK to pass the SDP when needed. That determination is made entirely based on the offer/answer rules described in [6] and [7], and is independent of preconditions.

9 Example Call Flows

9.1 Click to Dial

The first application of this capability we discuss is click to dial. In this service, a user is browsing the web page of an e-commerce site, and would like to speak to a customer service representative. They click on a link, and a call is placed to a customer service representative. When the representative picks up, the phone on the user's desk rings. When they pick up, the customer service representative is there, ready to talk to the user.


A
Controller
B
(1) INVITE offer1
no media
(2) 183 answer1
no media
(3) PRACK
(4) 200 OK
(5) INVITE no SDP
(6) 183 OK offer2
des=sendrecv
conf=recv
cur=none
(7) UPDATE offer2'
des=sendrecv
conf=recv
cur=none
(8) 200 UPDATE
answer2'
des=sendrecv
conf=recv
cur=none
(9) PRACK answer2
des=sendrecv
conf=recv
cur=none
(10) 200 PRACK
(11) reservation
(12) reservation
(13) UPDATE offer3
des=sendrecv
conf=recv
cur=recv
(14) UPDATE offer3'

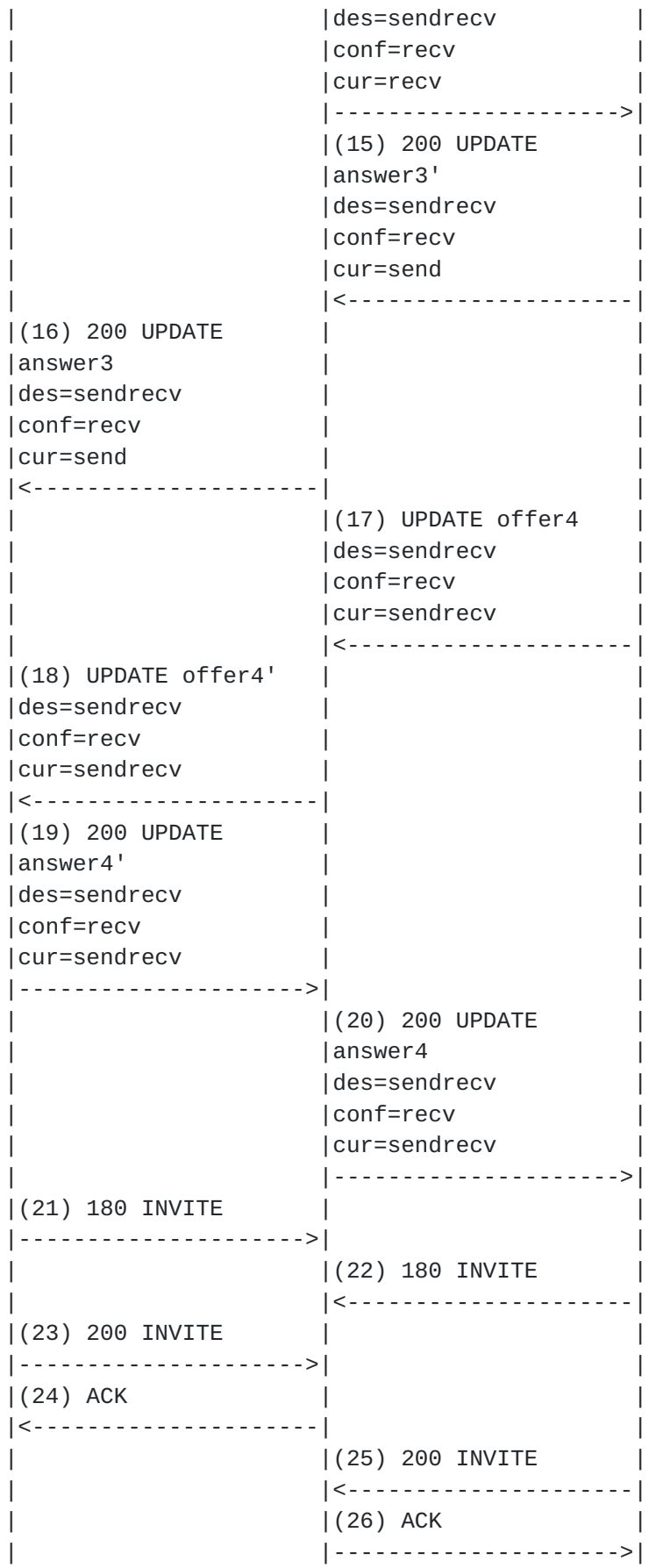

des=sendrecv
conf=recv
cur=recv
(15) 200 UPDATE
answer3'
des=sendrecv
conf=recv
cur=send
(16) 200 UPDATE
answer3
des=sendrecv
conf=recv
cur=send
(17) UPDATE offer4
des=sendrecv
conf=recv
cur=sendrecv
(18) UPDATE offer4'
des=sendrecv
conf=recv
cur=sendrecv
(19) 200 UPDATE
answer4'
des=sendrecv
conf=recv
cur=sendrecv
(20) 200 UPDATE
answer4
des=sendrecv
conf=recv
cur=sendrecv
(21) 180 INVITE
(22) 180 INVITE
(23) 200 INVITE
(24) ACK
(25) 200 INVITE
(26) ACK

Figure 9: Call Flow for Preconditions

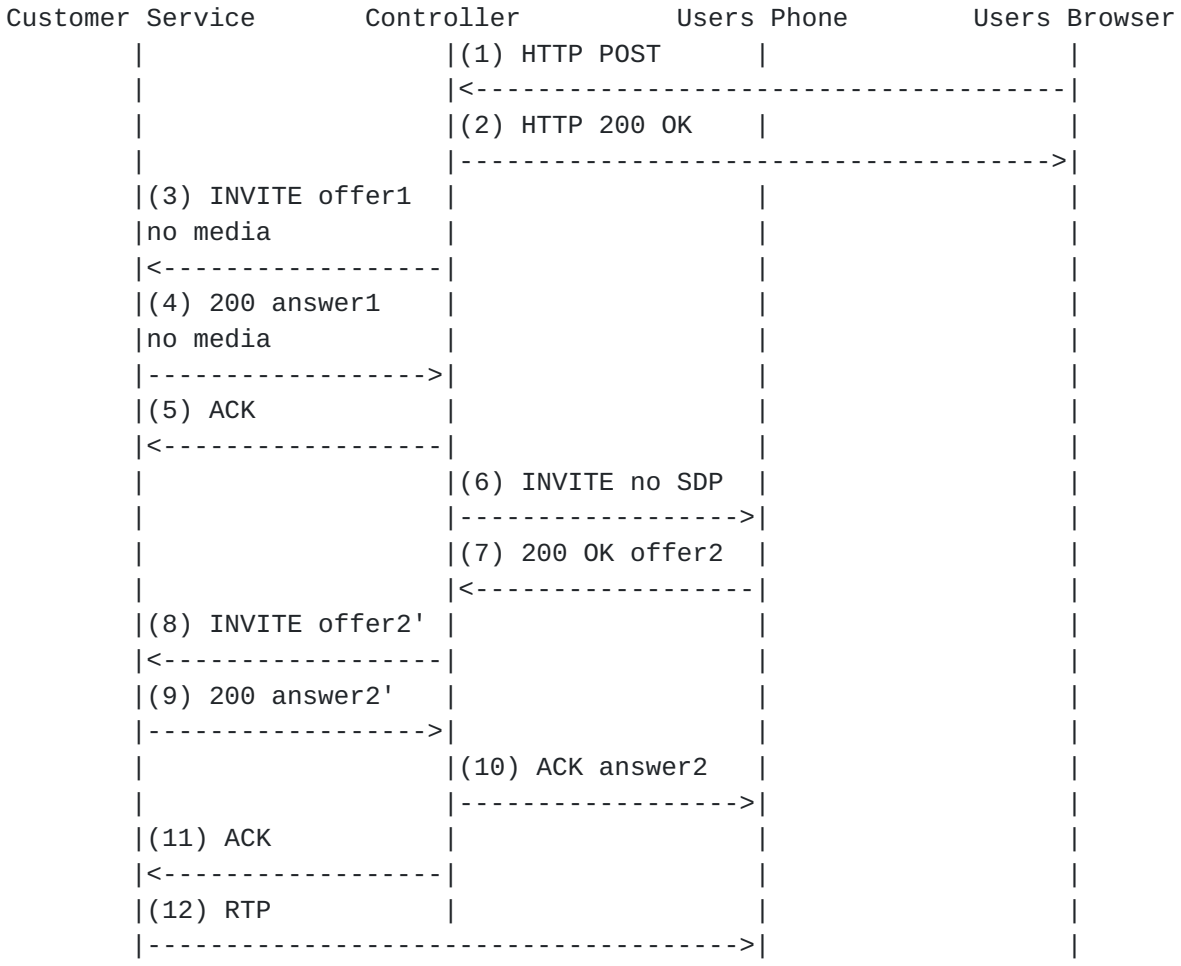


Figure 10: Click to Dial Call Flow

The call flow for this service is given in Figure 10. It is identical to that of Figure 4, with the exception that the service is triggered through an http GET request when the user clicks on the link.

We note that this service can be provided through other mechanisms, namely PINT [9]. However, there are numerous differences between the way in which the service is provided by pint, and the way in which it is provided here:

- The pint solution enables calls only between two PSTN endpoints. The solution described here allows calls between PSTN phones (through SIP enabled gateways) and native IP phones.

- When used for calls between two PSTN phones, the solution here may result in a portion of the call being routed over the Internet. In pint, the call is always routed only over the PSTN. This may result in better quality calls with the pint solution, depending on the codec in use and QoS capabilities of the network routing the Internet portion of the call.

- The PINT solution requires extensions to SIP (PINT is an extension to SIP), whereas the solution described here is done with baseline SIP.

- The PINT solution allows the controller (acting as a PINT client) to "step out" once the call is established. The solution described here requires the controller to maintain call state for the entire duration of the call.

9.2 Mid-Call Announcement Capability

The third party call control mechanism described here can also be used to enable mid-call announcements. Consider a service for pre-paid calling cards. Once the pre-paid call is established, the system needs to set a timer to fire when they run out of minutes. When this timer fires, we would like the user to hear an announcement which tells them to enter a credit card to continue. Once they enter the credit card info, more money is added to the pre-paid card, and the user is reconnected to the destination party.

We consider here the usage of third party call control just for playing the mid-call dialog to collect the credit card information.

We assume the call is set up so that the controller is in the call as a B2BUA. When the timer fires, we wish to connect the caller to a media server. The flow for this is shown in Figure 11. When the timer expires, the controller places the called party with a connection address of zero (1). This effectively "disconnects" the called party. The controller then sends an INVITE without SDP to the the pre-paid caller (4). The offer returned from the caller (5) is used in an INVITE to the media server which will be collecting digits (6). This is an instantiation of Flow II. This flow can only be used here because the media server is an automata, and will answer the INVITE immediately. If the controller was connecting the pre-paid user with another end user, Flow III would need to be used. The media server returns an immediate 200 OK (7) with an answer, which is passed to the caller in an ACK (8). The result is that the media server and the pre-paid caller have their media streams connected.

The media server plays an announcement, and prompts the user to enter

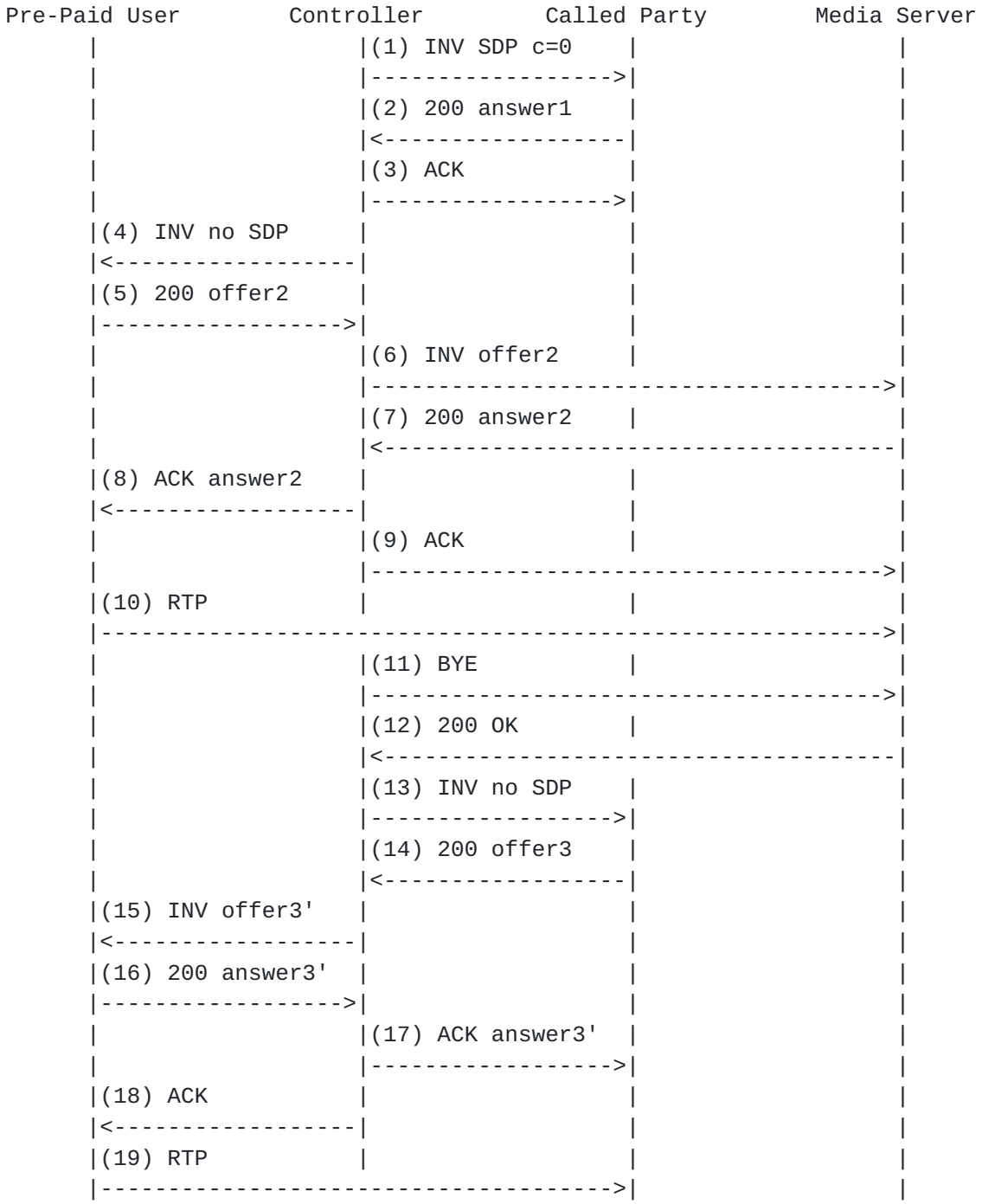


Figure 11: Mid-Call Announcement

a credit card number. After collecting the number, the card number is validated. The controller can then hang up the call to the media server (11). How the controller can know when to hang up the call is outside the scope of this document, and might have been done through an HTTP message from the media server to the controller, for example.

After hanging up with the media server, the controller reconnects the user to the original called party. To do this, the controller sends an INVITE without SDP to the called party (13). The 200 OK (14) contains an offer, offer3. The controller modifies the SDP (as is done in Flow III), and passes the offer in an INVITE to the pre-paid user (15). The pre-paid user generates an answer in a 200 OK (16) which the controller passes to user B in the ACK (17). At this point, the caller and called party are reconnected.

10 Implementation Recommendations

Most of the work involved in supporting third party call control is within the controller. A standard SIP UA should be controllable using the mechanisms described here. However, third party call control relies on a few features that might not be implemented. As such, we RECOMMEND that implementors of user agent servers to support the following:

- Re-invites that change the port to which media should be sent
- Re-invites that change the connection address
- Re-invites that add a media stream
- Re-invites that remove a media stream (setting its port to zero)
- Re-invites that add a codec amongst the set in a media stream
- SDP Connection address of zero
- Initial invites with a connection address of zero
- Initial invites with no SDP
- Initial invites with SDP but no media lines
- Re-invites with no SDP
- The UPDATE method [7]
- Reliability of provisional responses [6]

11 Security Considerations

The mechanism described here introduces several security considerations. The first issue is that of identity. When the controller initiates the call, what identity does it place in the From field of the INVITE? The controller could indicate that the call is from itself (From: sip:controller@company.com), but in many cases, the service is more usable if it "spoofs" the identity of the participant that is actually calling. However, to differentiate legitimate use of 3pcc from real attacks where a caller is faking an identity, user agents SHOULDauthenticate the requests. The controller will, of course, authenticate itself as the controller, rather than either participant. It is RECOMMENDED that user agents be configurable with credentials for entities that are legitimate controllers. Note that this will result in SIP messages whose From field does not match the identity of originator as determined from the authentication mechanism.

Some of the flows require the controller to manipulate the SDP. If S/MIME is used to encrypt or sign the bodies of the request end-to-end, third party call control will fail.

12 IANA Considerations

There are no IANA considerations associated with this specification.

13 Authors Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Jon Peterson
NeuStar, Inc
1800 Sutter Street, Suite 570
Concord, CA 94520
USA
email: jon.peterson@neustar.com

Henning Schulzrinne
Columbia University
M/S 0401
1214 Amsterdam Ave.

New York, NY 10027-7003
email: schulzrinne@cs.columbia.edu

Gonzalo Camarillo
Ericsson
Advanced Signalling Research Lab.
FIN-02420 Jorvas
Finland
Phone: +358 9 299 3371
Fax: +358 9 299 3052
Email: Gonzalo.Camarillo@ericsson.com

14 Normative References

[1] J. Rosenberg, H. Schulzrinne, et al. , "SIP: Session initiation protocol," Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

[2] W. Marshall, G. Camarillo, and J. Rosenberg, "Integration of resource management and SIP," Internet Draft, Internet Engineering Task Force, Apr. 2002. Work in progress.

[3] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[4] J. Rosenberg and H. Schulzrinne, "An offer/answer model with SDP," Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

[5] H. Schulzrinne, D. Oran, and G. Camarillo, "The reason header field for the session initiation protocol," Internet Draft, Internet Engineering Task Force, Apr. 2002. Work in progress.

[6] J. Rosenberg and H. Schulzrinne, "Reliability of provisional responses in SIP," Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

[7] J. Rosenberg, "The SIP UPDATE method," Internet Draft, Internet Engineering Task Force, Mar. 2002. Work in progress.

15 Informative References

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," RFC 1889, Internet Engineering Task Force, Jan. 1996.

[9] S. Petrack and L. Conroy, "The PINT service protocol: Extensions to SIP and SDP for IP access to telephone call services," RFC 2848, Internet Engineering Task Force, June 2000.

Full Copyright Statement

Copyright (c) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.