Internet Engineering Task Force SIP WG
Internet Draft Rosenberg/Peterson/Schulzrinne/Camarillo
draft-rosenberg-sip-3pcc-01.txt dynamicsoft,Level3,Columbia U.,Ericsson
November 22, 2000
Expires: May 2001

Third Party Call Control in SIP

STATUS OF THIS MEMO

Abstract

This document discusses the usage of the Session Initiation Protocol (SIP) for third party call control. Third party call control refers to the ability of one entity to create a call in which communications is actually between other parties. We present a SIP mechanism for accomplishing third party call control that does not require any extensions or changes to SIP.

1 Introduction

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 is often used for operator services (where an operator creates a call that connects two participants together), and

conferencing.

On the Internet, a wider range of services are enabled through a third party session control mechanism. This is because other IP applications, such as web, email, presence, instant messaging, and chat can now be brought into the picture. An excellent example is click-to-dial. This service 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.

In order to support third party call control applications, a mechanism is needed that allows a controller to create, modify, and terminate calls with other entities. In this document, we present a mechanism using the Session Initiation Protocol (SIP) [1] which allows a controller to execute third party services. The mechanism is not an extension to SIP. It is merely an application of the tools enabled through RFC 2543. A controller can create calls between any entity that contains a normal SIP user agent. After desribing the mechanism, we present three third party services which take advtantage of this mechanism. One is click-to-dial, the second is a feature that enables a mid-call announcement for credit card authorization , and the third is a timed conference bridge initiation.

2 Third Party Control

The basic idea behind the third party mechanism is simple. Consider first the case of just connecting two users in a call. The controller first sends an INVITE to the first user whose phone is to ring. This is a standard INVITE, but its SDP contains a single audio media line, with one codec, a random port number (but not zero), and a connection address of 0.0.0.0. This creates an initial media stream "on hold".

When the first user answers, the controller sends an ACK. It then generates a second INVITE. This INVITE is addressed to the second user to be connected in the call. This INVITE contains the SDP as received from the 200 OK of the first user. When the 200 OK to this second INVITE arrives, the controller ACK s it, takes the SDP, and then re-INVITEs the first user with this updated SDP. A flow diagram for this mechanism is given in 1.

At this point, both participants believe they are in a single point-to-point call with some control system (assuming the controller identified itself as such in the From field of the INVITE). However, they are exchanging media directly with each other, rather than with

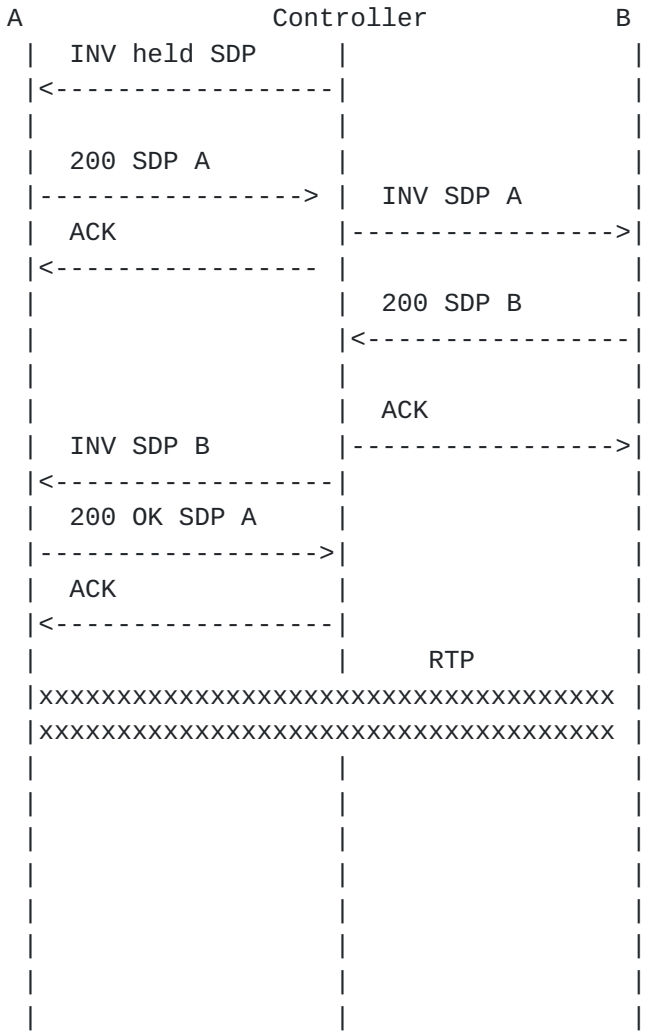


Figure 1: Basic Third Party Call Control

the controller. The result is that the controller has set up a call between both participants.

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 2.

As an alternative, when the controller receives a BYE from A, it can generate a new INVITE to a third party, C, using the SDP from B. When the 200 OK arrives from C, the controller sends a re-INVITE to B,

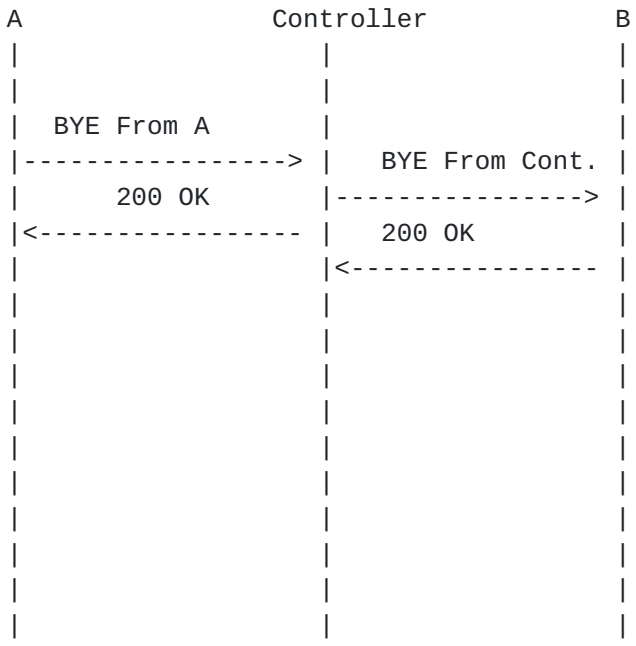


Figure 2: Hanging Up with 3PCC

using the SDP from C. If the 200 OK to the re-INVITE contains the same SDP as it used in the INVITE to C, the controller has sucessfully connected B to C, transparently to B. A call flow for this is shown in 3.

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

The general idea behind the mechanism is that there is a point to point SIP relationship between each participant and the controller. However, by passing the SDP it receives from one participant to another, it can causes users to actually communicate with each other rather than the controller.

3 Third party call control and SDP preconditions

In unicast sessions there is a number of media streams flowing between two entities. In order to perform resource reservation it is necessary to know the session descriptions from both parties. When third party call control is performed the information needed to establish the QoS required is not available from the beginning. The

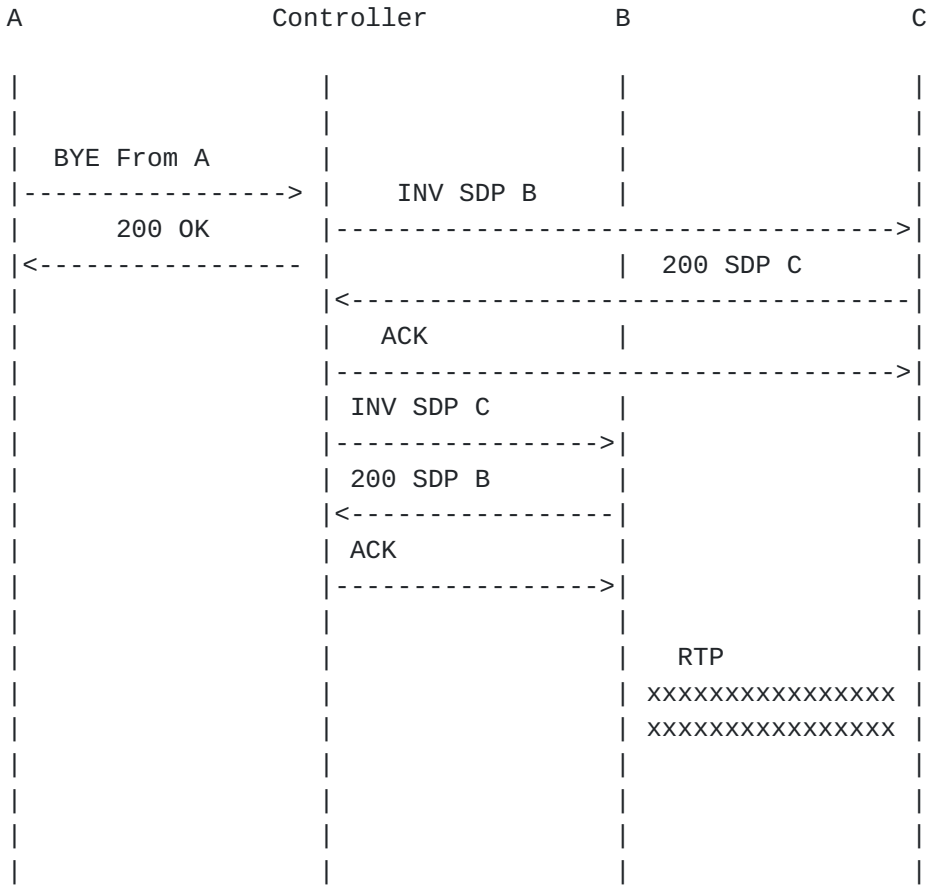


Figure 3: Alternative to Hangup

call flow shown in Figure 4 shows how the exchange of SDPs between both parties can be performed.

The controller INVITEs A in (1). At this point of time there is no information available about codecs to be used port numbers or IP addresses. The SDP of this INVITE just contains SDP preconditions and the media stream types (audio, video, etc...). As specified in [2], the called UAS returns a 183 immediately containing SDP information needed for QoS signaling (2).

INVITE (3) contains the SDP received from A. This INVITE is sent to B. When B responses with (4) 183 it is ready to perform resource reservation. However, B will not start resource reservation until the PRACK (7) is received. This allows B's SDP to be sent to A in (5). This way both parties have all the information needed to perform

resource reservation. Note that, since reliable provisional responses are used [3], the 183 (2) is retransmitted until the PRACK (5) arrives from the controller. This PRACK is transmitted only when the 183 arrives from B (4). Fortunately, this 183 is generated automatically, so that the first 183 (2) should not be retransmitted that much, if at all.

The PRACK matching (2) is sent at (5). This PRACK is not sent before because it is used to send B's SDP to A. The controller does not get this information until (4).

When the preconditions from B to the controller and from A to the controller are met two COMETs are received (9) and (11). At this point of time is up to the controller to let the session establishment go on sending a COMET to A (13). When A accepts joining the session (15), a COMET (16) is sent to B so B is alerted.

4 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 the phone on the desk (a normal telephone) rings. When the user picks up, the phone of the customer service representative (an IP phone) rings. When they pick up, the service representative is talking to the user.

We assume for purposes of this discussion that the web server is actually an applications server that contains an http interface. In this case, when the user clicks on the URL, the application server knows, through cookies or some other state mechanism, the addresses of the participants to be connected.

The call flow for this service is given in 5. Note that it is identical to that of Figure 1, 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 [4]. 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.

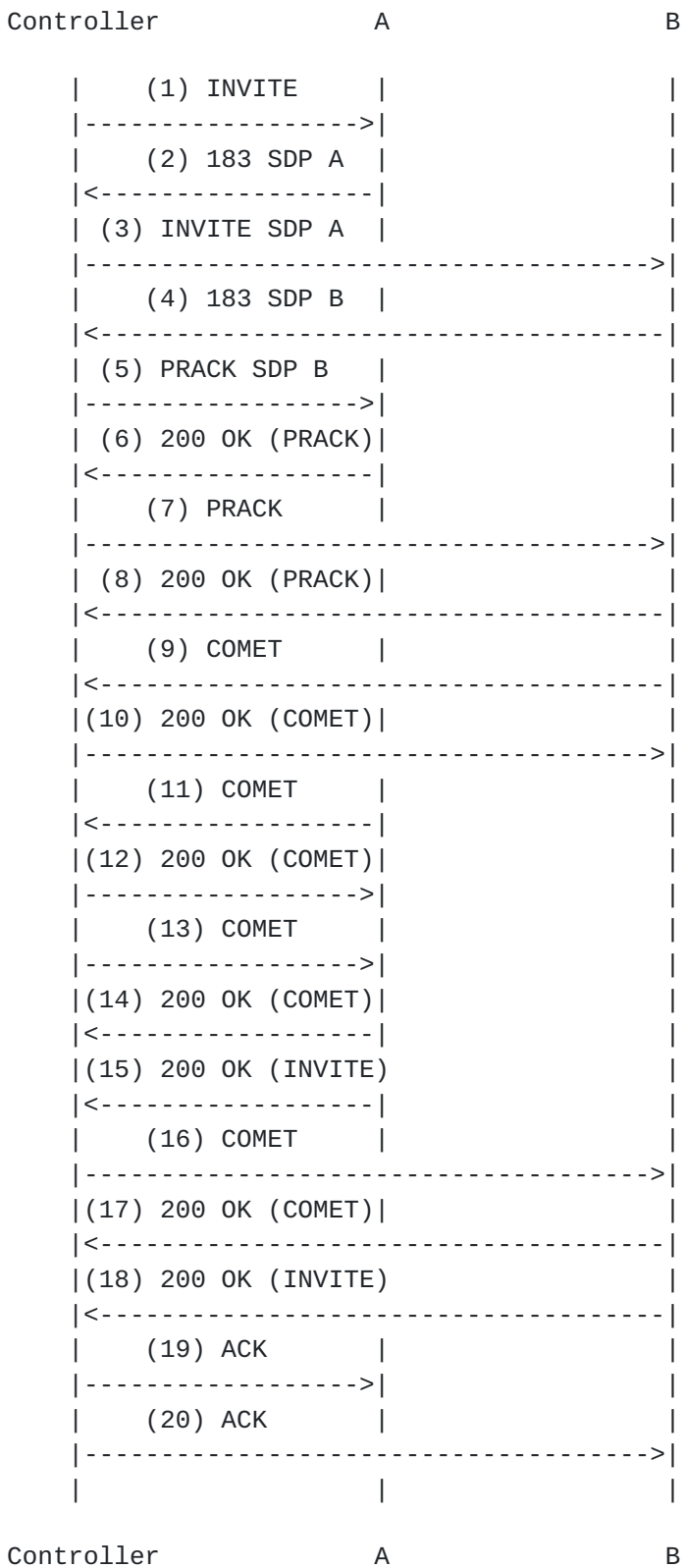

Controller
A
B
(1) INVITE
(2) 183 SDP A
(3) INVITE SDP A
(4) 183 SDP B
(5) PRACK SDP B
(6) 200 OK (PRACK)
(7) PRACK
(8) 200 OK (PRACK)
(9) COMET
(10) 200 OK (COMET)
(11) COMET
(12) 200 OK (COMET)
(13) COMET
(14) 200 OK (COMET)
(15) 200 OK (INVITE)
(16) COMET
(17) 200 OK (COMET)
(18) 200 OK (INVITE)
(19) ACK
(20) ACK
Controller
A
B

Figure 4: Call Flow for Preconditions

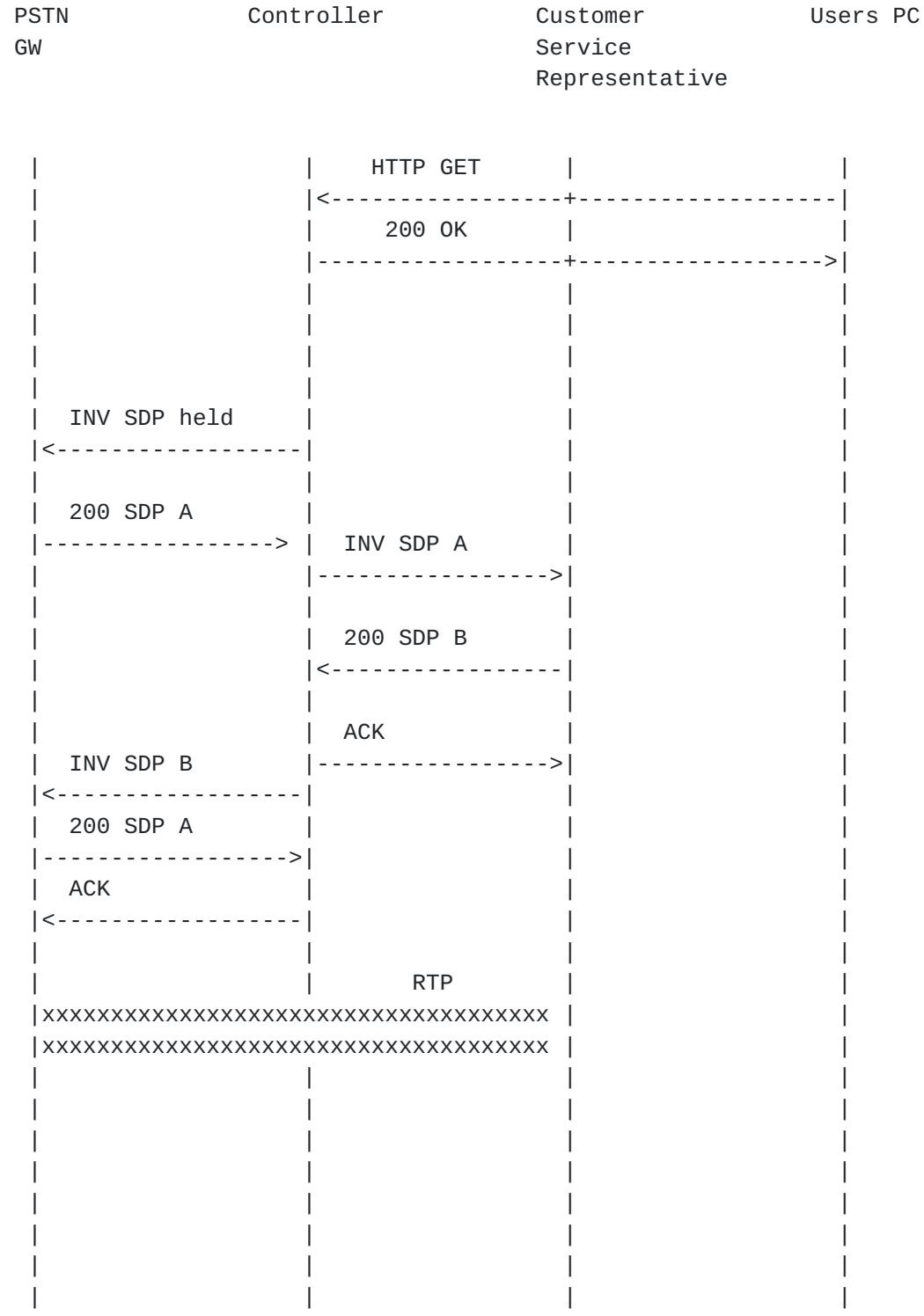


Figure 5: Click to Dial Call Flow

- 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.

5 Mid-Call Announcement Capability

The third party call control mechanism described here can also be used to enable mid-call announcements. The call is set up by the controller as desribed above.

It is actually not necessary for the controller to set up the call. However, if a participant initiates the call, the controller must step in as a virtual UAC/UAS, and act as a termination and re-initiation point

Perhaps the call is through a payphone, in which case the controller determines that the call is to be terminated after some amount of time if the user doesn't add more money to the phone. When this timer expires, the controller places the called party on hold. It then sends an INVITE to the media server which will be collecting digits. It then sends a re-INVITE to the user on the payphone, connecting its media streams with the media server. The media server plays an announcement, and prompts the user to enter a credit card number, for example. After collecting the number and validating the card, if the call can continue, the media server hangs up. The controller takes this as a cue and reconnects the user to the original called party, and takes the original called party off hold.

A call flow for this service is shown in 6.

We have assumed that the media server and the controller have agreed, ahead of time, that a hangup implies that the desired service (extending the lifetime of the call) has succeeded. This is effectively allowing a call control interface between the controller

and the media server. Parameters needed between the elements, such as the new expiration of the call, can be passed in the BYE. A separate draft, forthcoming, will discuss call control interfaces to media services in more detail.

6 Timed Conference Intitation

In this service, a conference bridge is booked for some number of participants. In order to make sure the conference begins on time, the conference bridge will call each participant at the time of the call. If a participant doesn't answer, the bridge tries to contact them again (unless they call in) five minutes later.

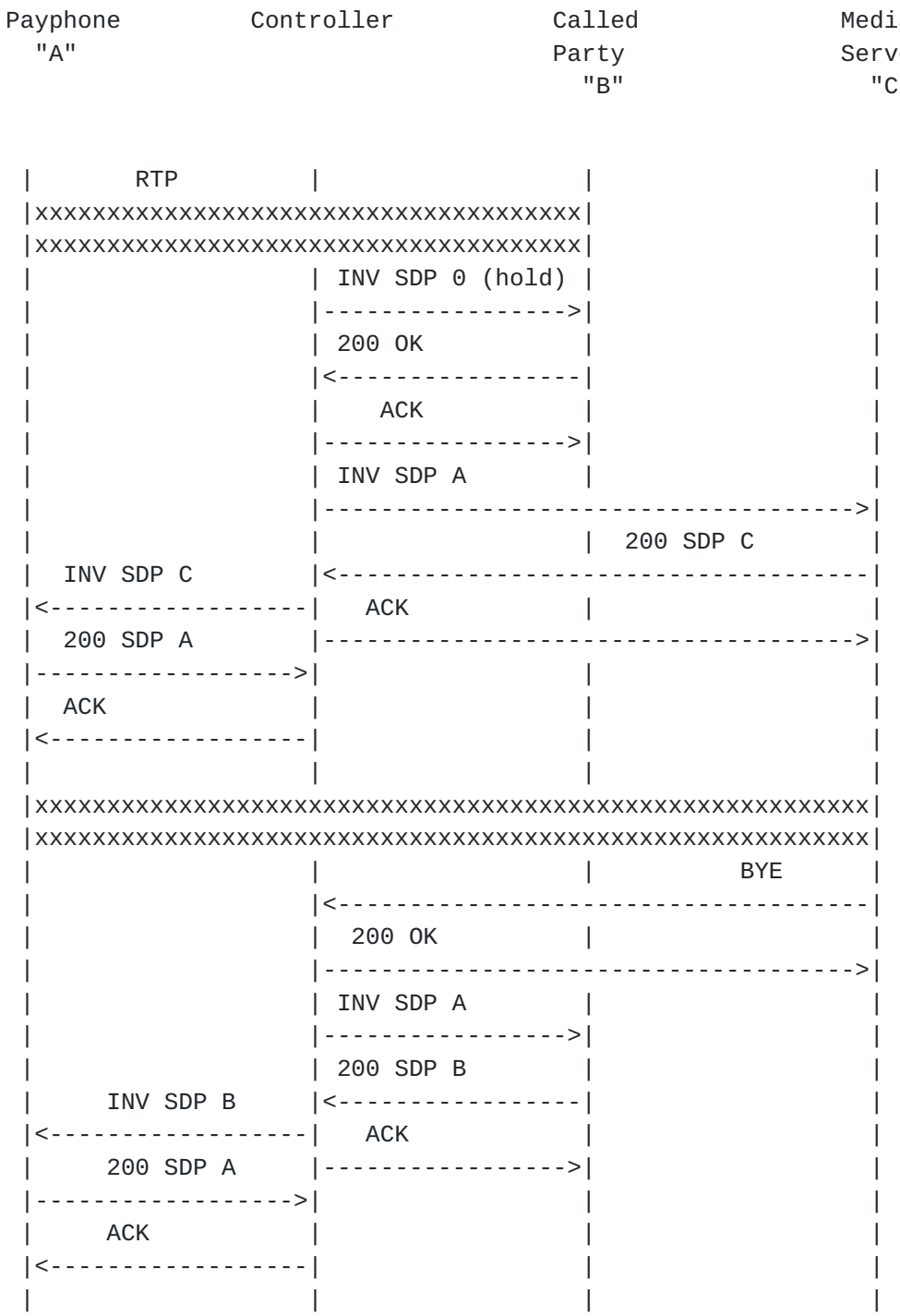
In the call flow described here, we assume that the controller acts as the media bridge. This is not strictly necessary; some kind of control interface could be used to separate the media function from the controller.

The call flow, shown in 7, is, not surprisingly, remarkably like that of Figure 1. The only difference is that the SDP listed in the INVITE s generated by the controller always contain SDP that points to the conference bridge, rather than one of the other participants. In the call flow diagram, user 1 is invited first, then user 2, and then user 3. User 3 is not available, but is called again five minutes later.

7 Implementation Notes

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

- Re-invites that change the port to which media should be send
- 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
- Hold (connection address of zero)


Payphone "A"
Controller
Called Party "B"
Media Server "C"
RTP
INV SDP 0 (hold)
200 OK
ACK
INV SDP A
200 SDP C
INV SDP C
ACK
200 SDP A
ACK
BYE
200 OK
INV SDP A
200 SDP B
INV SDP B
ACK
200 SDP A
ACK

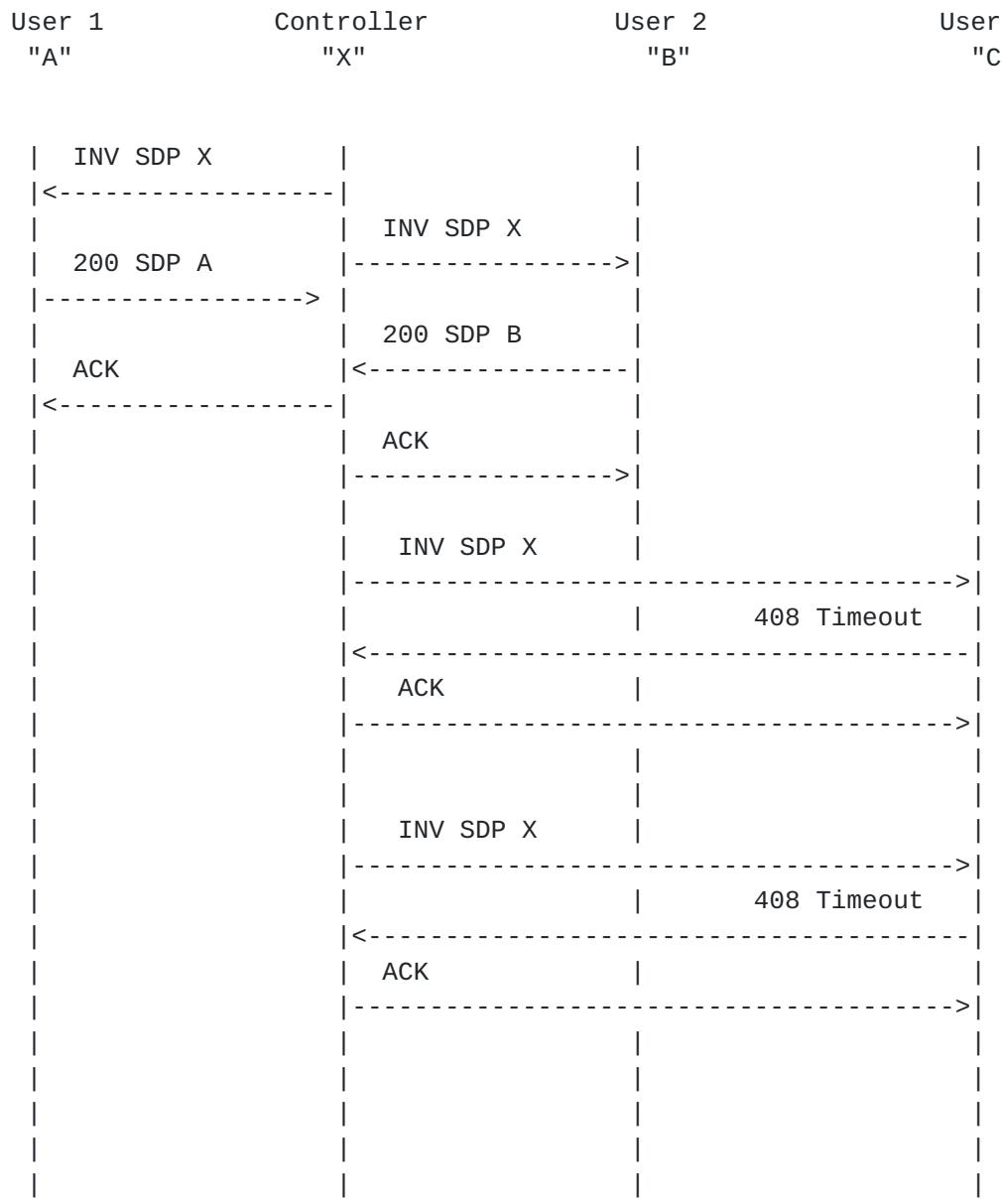


Figure 7: Timed Conference Initiation Call Flow

o Initial invites on hold

In addition, note that in Figure 1, the controller sends a re-INVITE to A with the SDP from B. The response to this re-INVITE is a 200 OK that contains "SDP A". If the SDP returned in the re-INVITE response were not the same, the controller would need to initiate a re-INVITE to B with that new SDP. This means that a UA which returns a

different SDP (for example, by changing the ports) in the response to every re-INVITE will trigger an infinite re-INVITE loop from the controller to each controller entity. As such, it is STRONGLY RECOMMENDED that if a re-INVITE does not require a UAS to modify the formulation of the SDP description for a specific stream in the response, the SDP description of that stream in the response MUST be the same. In other words, if a UA was in a session with a single stream, using codec A, and it received a re-INVITE modifying the port to send to, the re-INVITE response MUST be the same as it was to the initial request. However, if the re-INVITE forces the UAS to change codecs, it is acceptable in that case to use a different port in the SDP in the response.

In a previous draft, the flow of Figure 1 used a delayed ACK instead of a re-INVITE to update the SDP with the first user. The delayed ACK approach results in fewer messages (a savings of two), but has timeout problems. Specifically, If the second user does not answer within 32 seconds, the first user will timeout (as it did not receive an ACK), and the call will not be established. Since this will happen in practice, we strongly recommend the procedure described above rather than the delayed ACK mechanism.

8 Security Considerations

The mechanism described here introduces several security considerations. The first issue is the calling party identities delivered to the participants which the controller invites. 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, user agents SHOULD authenticate the requests. The controller MUST sign the request as itself, not as A or B. This will allow both parties to know that the call is actually being established through a controller. User agents SHOULD be configured to authorize requests from entities known to be controllers.

Note that this will result in SIP messages whose From field does not match the identity of the signator (as indicated in the signed-by field of the request).

The third party mechanism can also have an impact on encryption of the media that is part of the session. If negotiation of session keys is done through some kind of key exchange within SIP, the controller will, in all likelihood, not be able to set it up so that participants in the call arrive at the same key. This means that the controller may need to act as an RTP translator, decrypting with one key and re-encrypting with another.

Third party call control has unfortunate interactions with NATs and firewalls. The problems arise when the controller is on one side of a firewall/NAT that is being controlled by a proxy [5] [6] that receives the controller's requests, and the controlled users are on the other side. Pinholes in the firewall may be opened when, in fact, the media does not pass through the firewall. One way to avoid this is for the firewall controlling proxy to recognize that the address of the media is not within its private network, and so not perform NAT or firewall control in those cases.

9 Conclusions

We have presented a basic third party call control mechanism that uses SIP. This mechanism does not require any extensions to SIP and is completely backwards compatible.

10 Changes since -00

- Modified basic flow to use re-INVITE instead of delayed ACK.
- Included preconditions interactions.
- Added implementation considerations section.
- Updated security considerations to talk about NAT/firewall control, and to introduce the notion of signing requests when the signator is not the user in the From field.

11 Authors Addresses

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

Jon Peterson
Level 3 Communications
1025 Eldorado Blvd
Broomfield, CO 80021
email: Jon.Peterson@level3.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

12 Bibliography

[1] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[2] B. Marshall et al. , "Integration of resource management and SIP," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[3] J. Rosenberg and H. Schulzrinne, "Reliability of provisional responses in SIP," Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[4] S. Petrack and L. Conroy, "The PINT service protocol:extensions to SIP and SDP for IP access to telephone call services," Internet Draft, Internet Engineering Task Force, Feb. 2000. Work in progress.

[5] J. Kuthan and J. Rosenberg, "Firewall control protocol framework and requirements," Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.

[6] J. Rosenberg, D. Drew, and H. Schulzrinne, "Getting SIP through firewalls and NATs," Internet Draft, Internet Engineering Task Force, Feb. 2000. Work in progress.