Internet Engineering Task Force
Internet Draft
draft-rosenberg-sip-unify-00.txt
January 17, 2002
Expires: May 2002

SIP WG
J.Rosenberg
dynamicsoft

Unifying Early Media, Manyfolks, And HERFP

STATUS OF THIS MEMO

Abstract

The SIP working group has been grappling with a variety of very difficult problems lately. These include early media, coupling of resource reservation and call signaling, and the so-called Heterogeneous Error Response Forking Problem (HERFP). Additional problems being considered include supporting capability negotiation for indicating support any one of one-of-N codecs, upgrading the security on RTP streams, and demultiplexing media from a forked INVITE. In this document, we present a unified view of all of these problems, and present a unified solution that solves all of them in the same way.

Table of Contents

1 Introduction

The SIP working group has been grappling with a variety of very difficult problems lately. These include early media, coupling of resource reservation and call signaling, and the so-called Heterogeneous Error Response Forking Problem (HERFP). Additional problems being considered include supporting capability negotiation for indicating support any one of one-of-N codecs, upgrading the security on RTP streams, and demultiplexing media from a forked INVITE. In this document, we present a unified view of all of these problems, and present a unified solution that solves all of them in the same way.

Section 2 overviews each of these problems in more detail, and overviews the solutions proposed to date. Section 3 presents our unified view of these problems, and Section 4 presents a single mechanism for solving all of them at the same time. Section 5 shows how the proposed solution is applied to each of the problems discussed in Section 2.

2 Problems

2.1 One-of-N Codecs

It is very common for SIP gateways and hardphones to use DSP chips for the speech compression and decompression. A common characteristic of these DSPs is that they can support a wide variety of codecs, but only one codec can be loaded at a time. Unfortunately, there is no clear way in SIP to signal that this is the case. A caller can include, in their INVITE, SDP which lists all of their codecs, but the offer-answer specification [1] mandates that this means the caller supports the ability to dynamically change between them mid-call, which is not possible here.

A variety of solutions have been proposed for this problem:

Pick One, Try Again: In this approach, the caller picks one of the codecs, its preferred one, and includes that in the INVITE. If the callee supports it, great. If not, it returns a 488 error. According to the SIP specification [2], this error response contains a list of the codecs supported by the caller in an SDP message [3]. The caller then picks from amongst those, and issues a new call request. This approach has numerous problems, including increased call setup time, and the possibility that the second call goes to a different device due to routing changes. It is also inelegant.

Say and Pray: In this approach, the caller lists all of their codecs in the INVITE, but simply hopes that the callee sticks with one of them. This clearly has disadvantages.

a=pickone: It has been proposed on the mailing lists that the caller could include all of their codecs in the SDP, but include a new attribute, "a=pickone" which tells the callee to just pick one of them when responding. This approach is somewhat specialized, and is not backwards compatible.

Simcap restrictions: It has been proposed on the mailing lists that the caller include all of their codecs in the SDP, and include additional parameters in the SDP which more concretely define their capability constraints, using the simcap specification [4]. However, simcap cannot express the constraint that only one of N codecs can be supported at once.

Three-Way Handshake: It has been proposed [5] that a three-way handshake be used for this. The INVITE contains a full listing of codecs. The callee sends back a SIP provisional response (183 Progress) with a restricted set, which is those codecs amongst the list in the INVITE which the callee can do. The PRACK request (used for acknowledging the 183) contain another SDP with a single codec in it, and that is the one used for the session. This mechanism has the problem that it is not backwards compatible

Thus, none of the proposed solutions seem satisfactory.

2.2 Change Ports on forked 2xx

When the caller issues an INVITE, it contains the port and IP address in the SDP where media should be sent to. If this INVITE should "fork", meaning it is delivered to multiple recipients, each of them can answer. According to the specifications, each callee will begin sending media to the address and port provided in the INVITE. The caller will therefore receive two distinct media streams on the same port and address. In many cases, for RTP, they will be distinguishable with the SSRC parameter [6]. However, it is possible (although remotely) that both will pick the same SSRC, resulting in serious media confusion.

It has been proposed that a three way handshake be used to solve this problem. The INVITE contain an initial port/address, or none at all, since they are not used. The 200 OK contains the port and address of the called party, and the SIP ACK message, sent from the caller back to the callee, contain an update port and address for that particular

recipient. This would avoid the media confusion. However, the approach has the problem that it results in clipping of an RTT worth of any media sent by the callee immediately after they answer the phone.

Furthermore, the approach is not backwards compatible with existing SIP deployments.

2.3 Secure Media

SIP can establish media that runs over RTP [6]. More recently, a secure version of RTP, called SRTP [7], has been developed. However, SRTP is not backwards compatible (obviously). We would desire for the caller to be able to indicate that they can do either, and then the called party picks one, followed by the caller responding with the specific address and port for the media stream.

Robert Fairlie Cuninghame has proposed (http://www1.ietf.org/mail-archive/working-groups/sip/current/msg00664.html) that a three way exchange be used for this also. This is a different exchange than the one discussed above for the one-of-N codecs case; rather, in this exchange, the INVITE contains a complete set of capabilities, and an offer comes in the 200 OK, followed by an answer in the ACK.

This approach suffers the problem that it is not backwards compatible, and also clips an RTT worth of media after the caller picks up.

2.4 Early Media

Early media has received a tremendous amount of attention, with several drafts [8] [9] [10] and countless email discussions and presentations.

The essence of the problems are outlined in [8]. At the core is a single problem - there is a need for aspects of the media sessions to be changed before the call is accepted. This includes rejecting streams, modifying streams, placing them on hold, adding media to them, and so on. Numerous solutions have been proposed. Rosenberg has proposed [8] the addition of a re-INVITE before session completion to update the media, all following the offer/answer exchange. Sen et. al. have proposed that in the case of calls with preconditions, there be two separate media negotiations, one for early media, and one for the final media, both of which would use manyfolks [9]. Mahy has proposed on the mailing lists to modify Rosenberg's preferred proposal to use a separate method, OFFER, to update the media stream.

Also on the mailing lists, Robert Fairlie-Cuninghame has proposed

that a UAC "prod" the UAS into sending a new offer in the 1xx, which can be answered in the PRACK. This avoids the problem with overlapping INVITE, but has a separate problem, which is that the UAC is restricted to providing answers, not offers. This means that it can never add a media stream, for example.

All of the other proposals have problems. Rosenberg's preferred solution, as pointed out by Mahy, suffers in that it breaks the ability of proxies to provide services. Sen's proposal, in the case of preconditions, is extremely complex, with vague semantics. Mahy's proposal seems workable, but its interactions with preconditions are undefined.

2.5 Coupling of Resource Reservation and Signaling

The most complex of all of the problems is the coupling of resource reservation and signaling, which has been the result of literally years of back and forth proposals, compromises and specification work. It is now codified in the so-called "manyfolks" draft [5]. The essence of the problem it is trying to solve is the coupling of resource reservation to call state. We would like to be able to setup a call, and ring the callee, ONLY if resources can be reserved for the call. However, we need signaling to the callee in order to obtain the information required to perform resource reservation. The result is a chicken-and-egg problem. The solution is to include preconditions in the INVITE, which effectively tell the callee that it shouldn't ring the phone until resources have been reserved, security has been set up, or some other precondition has been met. Once the conditions are met, a COMET request is sent by either side to indicate this fact, so that ringing can occur and call setup can proceed.

Manyfolks accomplishes this by a call exchange shown in Figure 1, copied from Figure 1 of [5]. The INVITE contains SDP with a precondition on a media stream. The precondition typically indicates that the call should not proceed unless bidirectional resources can be reserved for the media stream. The UAS returns a 183 response indicating acceptance of this precondition, and providing enough information for the UAC to begin performing resource reservation. This 183 is acknowleged with PRACK, said PRACK can contain additional SDP which updates the codec sets against which reservations are done. Once reservations are made, the UAC sends a COMET to the UAS, with SDP as well. This SDP is used solely to indicate that the precondition has been met (i.e., resources have been reserved). This is done by an attribute in the SDP which indicates that the reservation is confirmed. Once that has been sent, the UAS can begin alerting the callee, and it therefore sends a 180 ringing message

(which is also acknowleged with PRACK). Once the caller accepts, a 200 OK is sent, also with SDP, mirroring that of the 183. This is ACKed as well.

We have observed some problems with the current approach. Most of them have to do with an underspecification of behavior, including no clear path for smooth integration with early media. Some of the issues:

- The specification does not say what happens if the SDP in the COMET is different beyond just indication of a confirmation tag. Can media streams be added or deleted? Can codecs be changed?

- There is no visible way to support early media; Sen has proposed a dual-exchange for it, but this is extremely complex.

- There is no way to indicate that, when a call is initially established, resources have already been reserved (for example, link resources in the case of access-only reservations). This has been raised on the list as an issue.

- There is no way to support the case of failure of media in the middle of a call causing termination of the call, or other action.

- There is no clear way to handle the case of addition of streams mid-call, which may require further reservation which could also fail. We believe it is possible with the documented approach, but is not specified or discussed.

2.6 The Heterogeneous Error Response Forking Problem (HERFP)

HERFP, as it is called, is, in our opinion, the most complex remaining problem with the SIP specification.

It relates to the rules for response processing at a forking proxy. A proxy never forwards more than one error response back to the UAC. This is needed to prevent response implosion, but more importantly, to support services at proxies. A forking proxy only returns an error response upstream if all forked requests generate an error response. However, a 200 OK is always forwarded upstream immediately.

The problem is that if a request forks, and one UAS generates an error because the INVITE is not acceptable for some reason (no credentials, bad credentials, bad body type, unsupported extension, etc.), that response is held at the forking proxy until the other

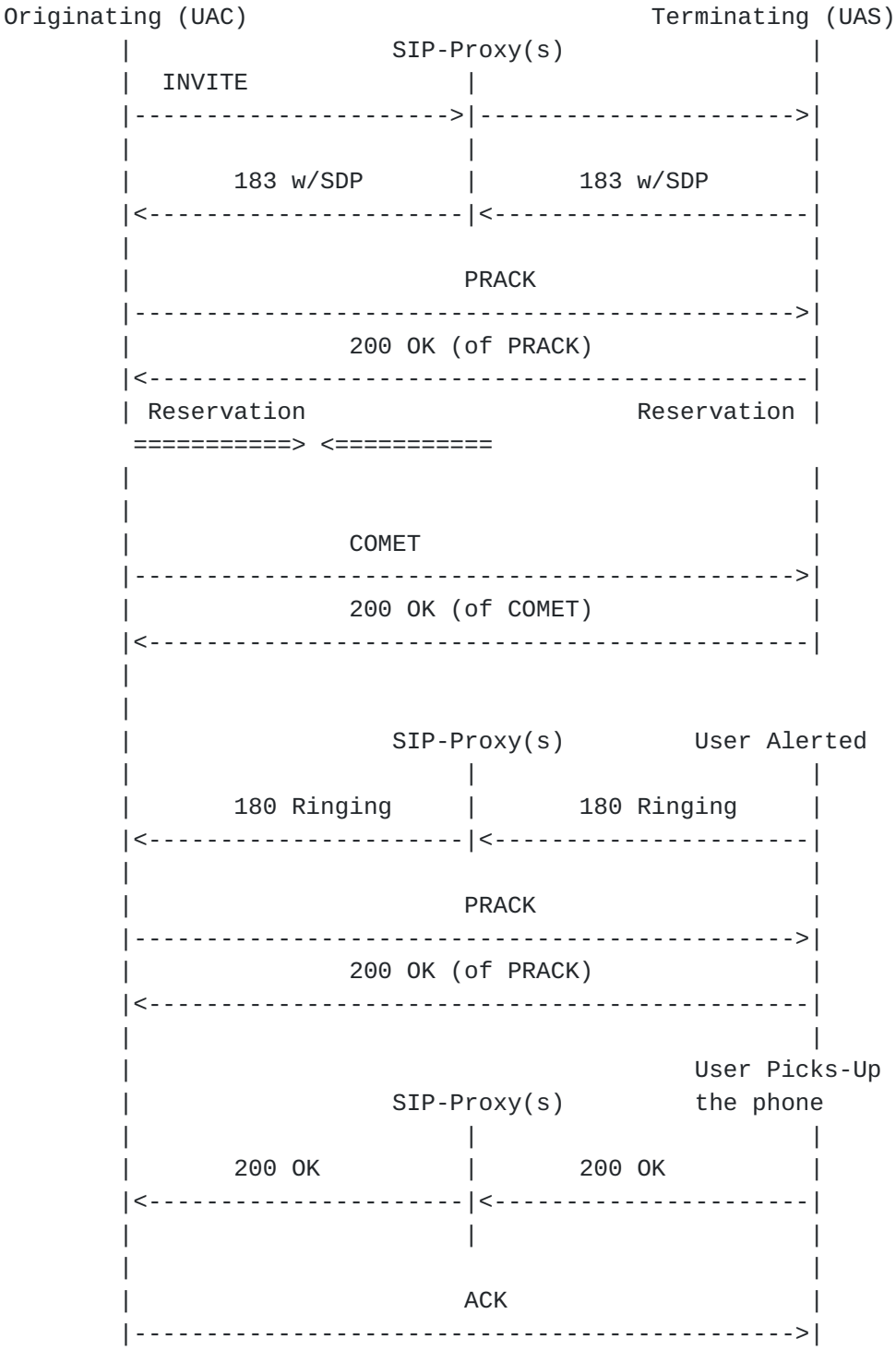


Figure 1: Manyfolks Basic Flow

forks respond. Of course, another branch may find the request acceptable, and therefore never generate an error response. The effect is to cancel out the benefits of forking.

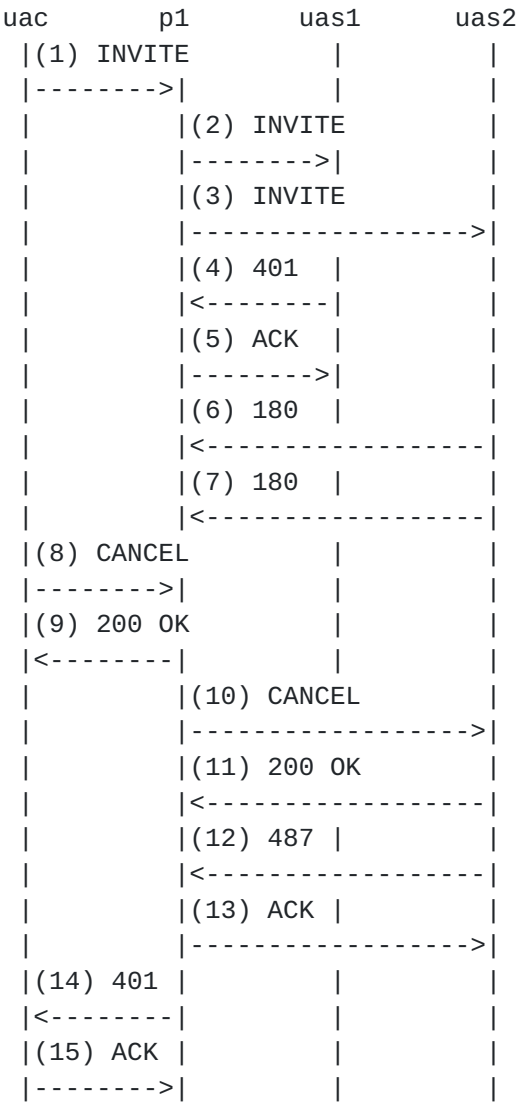


Figure 2: Basic HERFP Case

Figure 2 shows the simplest form of the problem. In this flow, the UAC sends an INVITE to proxy P1, which forks to UAS1 and UAS2. UAS1 might be a cell phone, and UAS2 a business phone. UAS1 rejects with a 401, and so never rings. However, UAS2 does not require credentials (or the request already had them), and therefore it rings. However, the user is not at their business phone, although they are available at the cell phone. After ringing for 20s, the caller gives up, and therefore sends CANCEL. This stops UAS2 from ringing, and results in the proxy forwarding the now-old 401 to the UAC. The UAC is not

likely to retry, since the user just hung up. Thus, no call is

established when it should have been.

Another HERFP case is shown in Figure 3. This is a case of sequential forking for a call forwarding service. The UAC calls a user, and the proxy first forks the call to UAS1. The user is not there, so the phone rings for 5s, and is then cancelled by the proxy, which forks to UAS2. UAS2 challenges, resulting in a 401 being returned to the UAC. The UAC tries again, which causes re-invocation of the call forwarding service! UAC1 rings once more for another 5s, and then finally the call is connected to UAS2. Interestingly, if the first UAC doesn't challenge but the others do, and there are N phones tried before completion, the first phone will ring N times! A user standing by UAS1 but electing to not answer will probably view it as a prank or malicious call.

The problem is that information needs to be propagated back to the UAC immediately, and the UAC needs to resubmit it, but the resubmission should not affect services somehow, e.g., should not re-invoke them as above.

3 Unifying the Problems and the Solution

To solve these problems together, we observed the following:

Session state and dialog State are NOT the same: The state of the media sessions, in terms of its media makeup, codec support, and so on, is orthogonal to the state of the call (referred to now as a dialog). A session can be active before an INVITE is even sent (in the case of an invitation to a multicast session), and it can be active after the call has ended in the same way. The instant a offer and an answer exhchange have taken place for a unciast session, the session exists, irregardless of the state of the dialog. There is no distinction between early media and final media. They are the same as far as the IP network is concerned, so any distinctions would be purely artificial. Because of this, it makes little sense to separately negotiate early media parameters from final media parameters, for example. Media communications begin when the session is established, and ends when it ends, irregardless of the state of the dialog. Early media is only early in the sense that it happens to take place before the dialog is established. If one wishes to account for communications services for billing purposes, those are tied to the establishment of media sessions, NOT the dialog, since it is the media session that enables communications. One can force them to be the same in order

to use the dialog state for accounting, but the fact remains that it is the media sessions that facilitate communications.

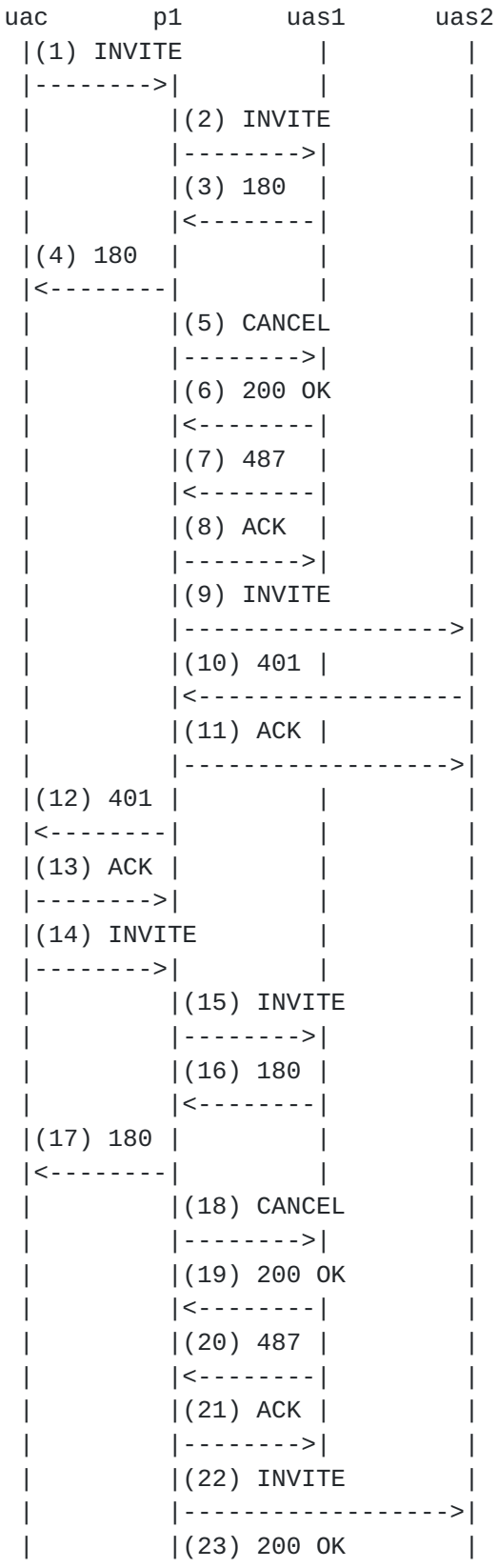
Services are driven off of dialog state, not session state: Call services, such as call-forward-busy, personal mobility, routing and screening services, are all based on the state of the dialog, which is manipulated through SIP requests and responses. They are not driven at all by the session state (generally).

INVITE affects both session and dialog state: The INVITE method, as specified, has the dual role of facilitating the exchange of SDP to establish a session, and also exchange messages for signaling the state of the dialog. It thus drives services and session state.

We need a way to manipulate session state independent of the dialog: The essence of the early media problem is that we need a way to manipulate the state of the session without impacting the dialog. The essence of the various problems with motivated a three-way handshake, is also the need to modify characteristics of the session without impacting the dialog. This is the key.

Manyfolks incorrectly views reservations: Manyfolks takes the wrong model for viewing the reservation. It views a precondition as special, requiring a specific method to indicate its achievement. The SDP in the COMET, for example, is used only for tunneling the bit in the SDP which indicates that the precondition is met. However, that is a statement of a CHANGE in state. SDP is meant to convey the full state of a session, not its changes. It is our view that the reservation of a media stream is just another piece of session state which can be manipulated. A media stream can be send-only, it can be receive-only. In the same exact way, a media stream can be reserved or not. Like the directionality of the stream, it can change at any time, and that should be signaled in the same way a new codec would be signaled. A precondition is nothing more than a statement that the dialog state should remain fixed in a particular state until the session state achieves a certain value. This, too, is a piece of state associated with a media stream, which can be manipulated in the exact same way.

Since sessions are independent of dialogs, the process of modification should not differ inside or outside of a


uac
p1
uas1
uas2
(1) INVITE
(2) INVITE
(3) 180
(4) 180
(5) CANCEL
(6) 200 OK
(7) 487
(8) ACK
(9) INVITE
(10) 401
(11) ACK
(12) 401
(13) ACK
(14) INVITE
(15) INVITE
(16) 180
(17) 180
(18) CANCEL
(19) 200 OK
(20) 487
(21) ACK
(22) INVITE
(23) 200 OK

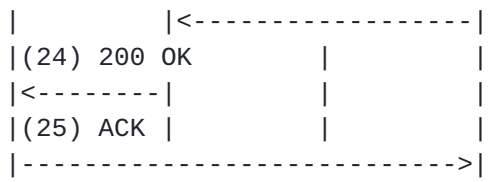


Figure 3: Forking HERFP Case

dialog: The process of modification of session state should not depend on the state of a dialog. We should have one way to manipulate session state. That mechanism should be independet of the SIP messages which carry it. This is the essence of the offer/answer model, and it is critical that this be the case for session manipulation before the call is accepted.

HERFP is about updating state too: HERFP is primarily about passing information from UAS to UAC, and getting additional information. The problems caused by the current approaches is that the additional information is provided in a request (INVITE) which invokes services and creates dialog state. What we need is a way to pass additional information, which is not session state, but state nonetheless, to the UAS, without affecting the initial call setup. We argue that this is, in fact, the same problem as the others.

Based on these observations, the solution became clear. We need a new method that can manipulate session state independent of the dialog state. That manipulation must follow the same rules for manipulation after the dialog is accepted. We must cast manyfolks into a light which views it as a manipulation of a different sort of session state, the state of the reservation for the session.

To solve HERFP, we must pass back information from the UAS to UAC without terminating the initial dialog, and get information back to the UAS without affecting intermediaries.

4 The new COMET

Our solution proposes that we take the one new method defined that sort-of affects session state, which is COMET, and generalize it. COMET is now a method, only within the context of a dialog (early or stablished), which may update some aspect of the session state, or other peripheral state relating to the dialog, but has no impact on the dialog itself. Reuse of the COMET name assists in backwards compatibility with manyfolks.

The offer/answer model remains as defined [1]. However, a UAS can generate a 1xx with an answer to the offer in the INVITE. The UAC can generate additional offers by sending a COMET, and the answer appears in the 2xx. Similarly, the callee can generate additional offers in COMET back to the callee, with the answer in its 2xx. This means that the 2xx to the INVITE need not contain SDP, nor the ACK. We would propose to relax the constraints in bis in order to allow this. This may strand some ALGs that are implemented, but this is exactly the reason we feel that embedded ALGs inside of layer 3 boxes are a bad

idea.

A session is considered established solely based on the offer/answer model, e.g., when an answer to the offer arrives. The dialog is created by the first 1xx or 2xx, and the call answered on 2xx.

COMET can be sent at any time, even after the dialog is established. In that sense, it would work like a re-INVITE, but unlike a re-INVITE, would never establish a new dialog, which re-INVITE can, if the UAS is reconstituting state [11].

COMETs cannot overlap, and the request glare procedures of [2] need to be applied to it in order to ensure that COMETs in each direction do not occur simultaneously.

The general rule for offers and answers would then be the following:

- A UA cannot send an offer when it has sent a previous offer to which it has not yet gotten an answer.
- A UA cannot send an offer when it has received an offer to which it has not generated an answer.
- A UA can only send an offer in a message that is sent reliably.
- The answer to an offer has to occur in a message which is correlated to the message the offer came in.
- If an offer does not take place in the first reliable message from UAC to UAS, it must be present in a reliable message from UAS to UAC.

The result is that, for basic UA, the offer is in the INVITE, answer in 2xx, or offer in 2xx, answer in ACK, just as is specified today. Both of those cases are covered by the above rules. When a UA can do COMET and 100rel, the rules would allow for a broader range of exchanges.

Any of the three-way handhake mechanisms described above can be done with two two-pass exchanges, as we will show below. If the UA wants to make sure that the session parameters of the first exchange are never used, it can set the stream to inactive in its offer or answer and then reactivate it later. It is possible to use a series of two pass exchanges both before and after the dialog is established.

To solve HERFP, we further specify that any of the SIP error responses which solicit additional information (401, 415, 420, 484,

etc.) can instead be sent as reliable provisional responses. These provisional responses would use response code 155 Please Update, and would contain a Reason header [12] indicating the specific error and its reason phrase. We use a provisional responses since they are sent end-to-end through existing proxies, and because they do not terminate the transaction. That is the key to not breaking forking and related services.

A benefit of this approach, everything else aside, is that it provides full disclosure to the caller about what has happened with their call. Progress on all the tried branches and their results are communicated back to the UAC. We believe this is very much in line with recent IAB directives on OPES [13], which has mandated that intermediaries inform the client about what is being done with their request.

Upon receipt of the provisional response, the UAC can send a COMET message providing the updated information, whether it be credentials, a new body type, or a new session description, as needed. This information would supplant whatever was sent in the initial INVITE, as if the COMET were a re-INVITE, except it has no impact on the dialog state itself.

To determine whether or not the UAS should send a provisional or final response, a proxy along the path which is invoking services on the request, or which forks, would insert a Require header indicating that this should be done. This Require would only be inserted by the proxies if the INVITE itself contained a Supported header indicating the ability to understand 155. If the UAS doesn't support it, the proxy can retry the request without the Require header.

We also elect to rename the expansion of COMET, from "Conditions Met" to "Conditional Modification of Endpoint Thinking", as a real stretch, but better than "Conditions Met" for its enhanced role.

We propose to restructure manyfolks, so that it is instead presented as a set of new attributes that can describe the reservation state of a stream, along with a new piece of state, called a precondition, which can itself be negotiated.

To summarize, the key ideas are:

- Generalize COMET to a method for updating session state or dialog-related state, but not dialog state itself.

- Allow COMET to be sent in either direction within any dialog at any time, subject to overlapping and glare rules, even if the dialog is an early one.

- Allow UAS to generate provisional responses instead of error responses containing the error information.

- Modify manyfolks so that its attributes are now just pieces of state about a media stream, like any other.

- Use a=inactive to ensure that intermediate session parameters are not used when a multi-pass exchange is required.

5 Application to Problems

We will now show below how this simple approach of separation solves all of the problems above in a unified way.

5.1 One-of-N Codecs

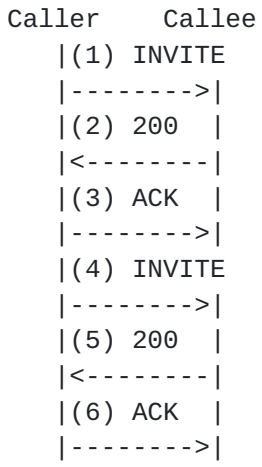


Figure 4: Basic Flow for One-of-N Codec Selection

The call flow for the two-pass solution to this problem is shown in Figure 4. The initial INVITE (message 1) contains SDP with the desired streams. Each stream lists ALL of the codecs supported by the UA, even if they cannot be supported simultaneously. The media stream has a direction of inactive, to indicate it is not yet started.

When the 200 OK arrives (message 2), it is ACKed normally. The 200 OK will have an m line with a subset of the codecs supported by the callee. If simcap is used [4], the caller will have additional information on other codecs as well. In parallel with the ACK, the caller sends a re-INVITE, this time with an m line with a single

codec, which is one of the ones supported by the UAS. THe media stream also changes to sendrecv, activating it. This generates its own 200 OK and ACK.

This flow has exactly the same properties as the proposed three-way handshake. However, it is backwards compatible. The only drawback is an additional message exchange, although there is no latency associated with that, just message overheads. We think this is a reasonable price to pay for a unified mechanism.

The ACK overhead can be avoided by using a COMET request instead of the second re-INVITE.

In Section 5.5, we show a flow that combines one-of-N selection with preconditions.

5.2 Change Ports on forked 2xx

The call flow for this case is shown in Figure 5. The initial INVITE (message 1) contains an m line with a valid port and address, which the UAC is listening on for media. The proxy forks this request to UAS1 and UAS2, each of which respond with a 200 OK. The proxy forwards both of these to the UAC. The UAC ACKs both of them. However, with the one from UAS2, in parallel with the ACK, it generates a re-INVITE that changes its receive port for the session with UAS2. This results in a 200 OK an ACK as normal.

The result is that media from UAS1 and UAS2 will go on separate ports, allowing the UA to associate a media stream with a particular dialog. The latency for such establishment is identical to the proposed three-way handhsake for this case. However, this solution has important benefits. First, its backwards compatible with existing UA. Second, if there was no forking, the flow will avoid media clipping. The proposed three way handshake will always suffer media clipping. Media clipping only occurs here if there is forking, in which case the media may be mixed or jarbled for 1 RTT until the port is changed. If such jarbling is not acceptable, the SDP in the INVITE can indicate an inactive stream, and then the stream can be changed to active in the re-INVITE. This avoids jarbling but will clip the media, just as the alternate proposed three way handshake will.

The cost of the approach is an additional INVITE/200/ACK exchange. That cost can be reduced by using COMET instead, at the expense of backwards compatibility.

5.3 Secure Media

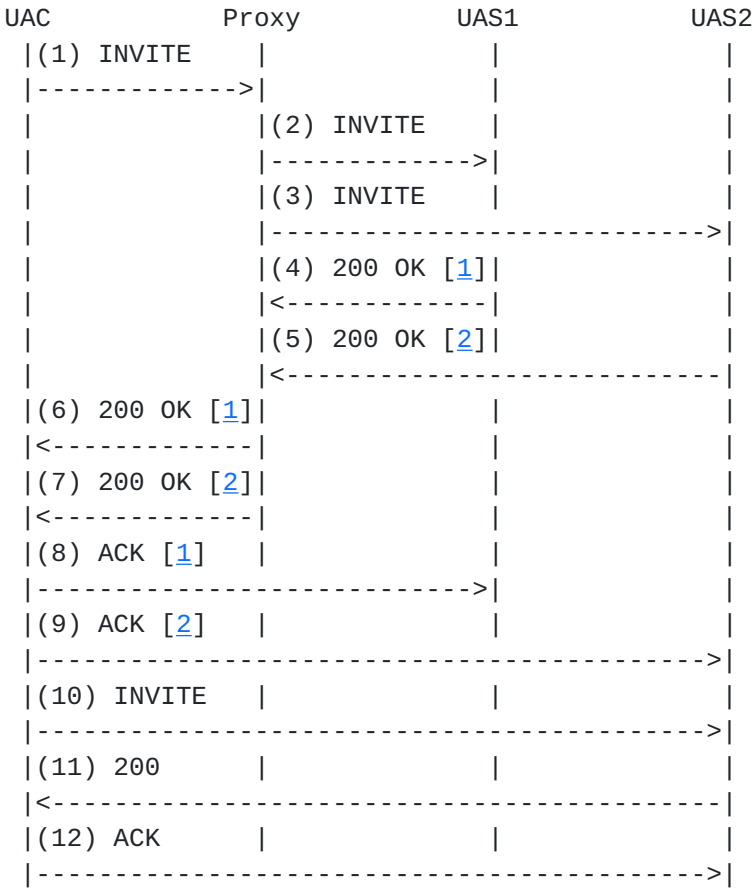


Figure 5: Call flow for changing ports

The flow for the secure media case is nearly identical to the one-of-N flow in Figure 4, and is shown in Figure 6. The initial INVITE uses regular RTP, but simcap is used to indicate support for SRTP. The 200 OK and ACK result in establishment of a media session using regular RTP. However, the callee knows that the caller supports SRTP (from simcap). So, it initiates a re-INVITE, changing the profile to SRTP.

5.4 Early Media

The basic flow for an early media session is shown in Figure 7. The caller sends an INVITE with SDP 1 in it. This is a normal SDP, providing a receive address and port for a single sendrecv audio stream. The caller supports 100rel, so it includes a Supported header

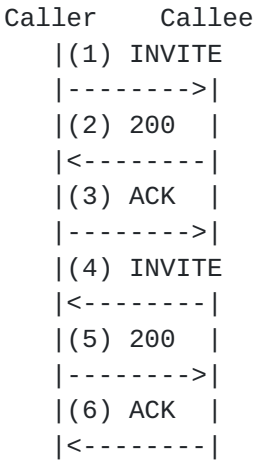


Figure 6: Secure Media Flow

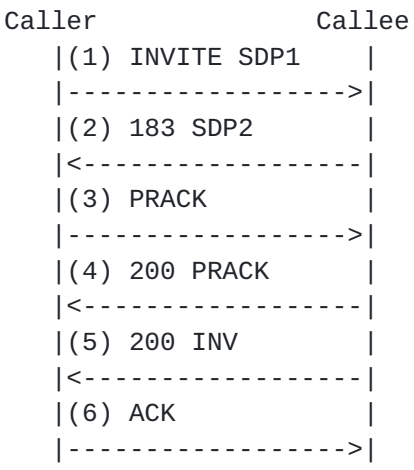


Figure 7: Basic Early Media Flow

in the INVITE indicating that. The Allow header also indicates support for COMET, which implies early media capability.

The UAS decides to generate early media. So, it answers the original INVITE with a 183, which is sent reliably. This 183 contains SDP, which is an offer to the initial answer. The answer does not change the directionality of the stream, implying that media can be sent in

either direction. Upon receipt of the answer in the 183, the session

is established (although the dialog is not). The caller hears the early media. Finally, the callee answers, generating a 200 OK. Since there is no need to modify any aspect of the session description, the 200 OK does not contain any SDP, and therefore neither does the ACK. The impact of the 200/ACK is only to update the dialog state, moving it to established.

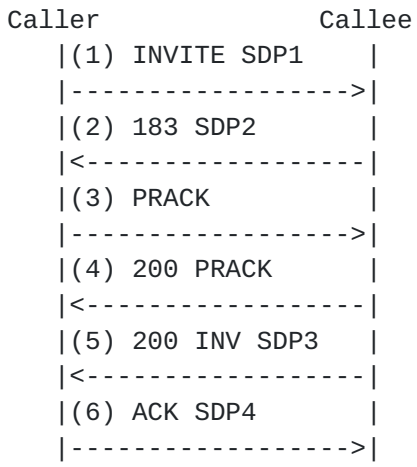


Figure 8: Unidirectional Early Media Flow

A slightly more complex flow is shown in Figure 8. In this case, the UAS would like to use unidirectional early media, from the callee back to the caller. The initial INVITE from the caller is identical to that in 7. However, the SDP in the 183 differs. The media stream is marked as sendonly, indicating a unidirectional flow from caller to callee. The user hears the early media. Once the call is setup, the callee needs to change the directionality to bidirectional. So, it issues a new offer in its 200 OK, which is identical to its previous SDP, but changes the direction from sendonly to sendrecv. The ACK contains an answer to this, which in this case indicates no changes. There is a smooth transition of media from early to bidirectional in this case.

The flow for refusing an early media session is shown in Figure 9. The flow starts like Figure 7. However, when the UAC gets the answer in a 183, which establishes the session before the dialog is answered, it decides to change it. It immmediately issues a COMET (message 6) with a new SDP that sets that media stream to inactive. This is answered in the SDP in the 200 OK to COMET. The stream is now

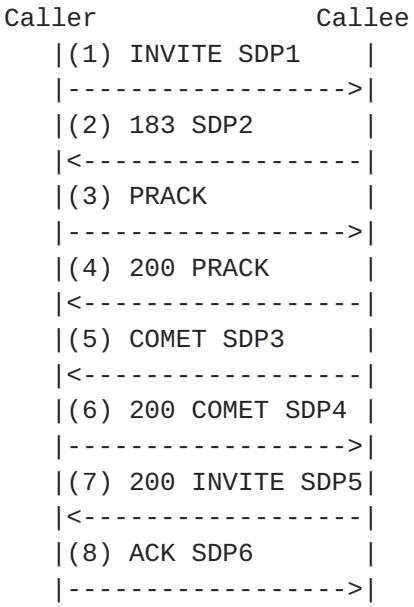


Figure 9: Refusing Early Media

inactive, so the callee sends no early media. However, when the call is answered, the callee decides to try to restart the session. So, the 200 OK to the INVITE (message 7) contains an offer, changing the stream to sendrecv. Since the offer is an a 200 OK, the callee decides to accept it, and generates an answer in the ACK, which keeps the stream sendrecv.

There is nothing special in this flow about the 200 OK; the callee could have decided to delay turning the media stream back on until after the 200 OK. Alternatively, if the UAS doesn't do anything to re-enable the stream, the UAC can offer to do so in a COMET or re-INVITE immediately following the receipt of the 200 OK (which would presumably have no SDP).

It is important to note that refusing early media can lead to clipping of the "regular" media once the call is established. However, the model proposed here, of a totally separated notion of media sessions from dialogs, would argue that there is nothing special about the content in the media stream upon acceptance of the dialog.

To demonstrate how the proposal works nicely with services, we show a call forward no-answer service in a proxy, where the initial UAS uses

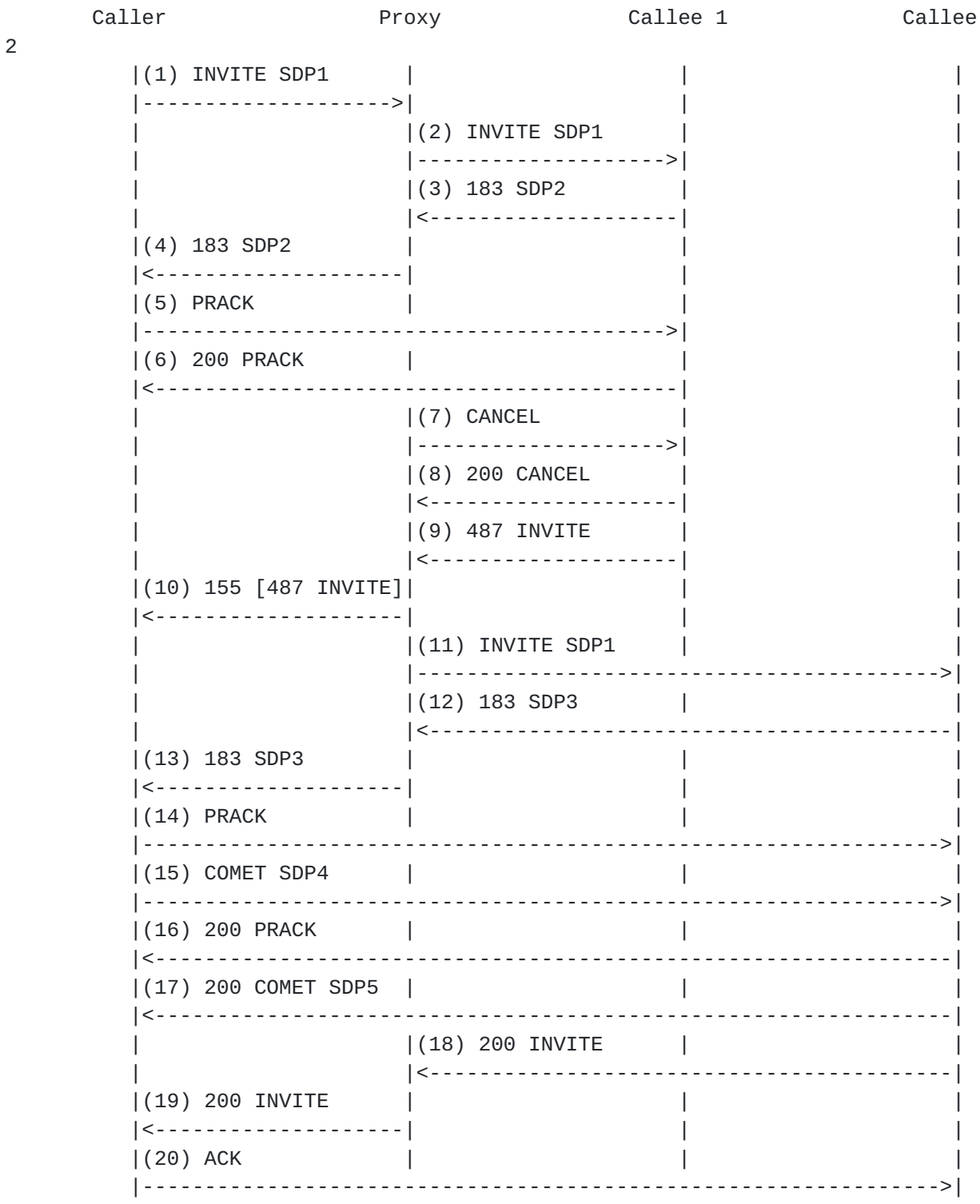


Figure 10: Early Media with Services

early media to providing ringback, as does the second UAS that is rung. The initial INVITE (message 1) has a single audio stream listing the address and port for receiving media. This is sent to a

session as in the flows above. Since the proxy does not record-route, the PRACK is sent directly to the callee. No COMET is issued by the UAC, and it therefore receives the media on the address and port from SDP1. At some point later, the proxy decides to try the next address. So, it CANCELs the INVITE (message 7). This generates a 487 to the initial INVITE. Now, the proxy decides to use the new 155 response code, to pass on the 487 (message 10). The primary advantage of this is to give the UAC some indication that the branch has failed. This UAC can use this in any way it chooses; for UI benefits, or simply discard it. In this instance, it is useful for the UAC to determine that the early dialog with Callee 1 is terminated. One use for such information is to know that no more media will come, and therefore, the port from SDP1 is safely reusable for any further media from a different UAS. However, in this flow, the 155 is ignored by the UAC and not used for that purpose.

The proxy now proceeds to contact the second callee (message 11). This callee also responds with a 183 with SDP. Since this 183 is provided on a separate dialog, the UAC knows that this is a different answer to its initial offer. So, it sends a PRACK, and at the same time, a COMET (message 15) to provide a new port for receiving the second media stream. The COMET is responded to with an answer (message 17), which changes no aspect of the media sessions.

At some point, callee 2 answers, generating a 200 OK and ACK, both without SDP. There is no change in media, and thus a smooth transition from early to final media with the callee.

This flow emphasizes how the various exchanges to manipulate the media composition of the "early sessions" is done effectively behind the back of the proxy, which doesn't know, and doesn't care. The services it provides are unaffected by the media changes, since its services are based on the INVITE. In fact, an off-the-shelf RFC2543 compliant proxy would function perfectly in providing this service.

5.5 Coupling of Resource Reservation and Signaling

Our proposal is to model the parameters described in manyfolks as just additional pieces of state about the media stream. In particular, the strength-tag and direction tag combined indicate the current reservation state of the stream. Mandatory and optional indicate that there is no reservation in place, but one is needed. The value of succeeded indicates that a reservation exists, failed means the attempt was made but failed.

We would argue that it is more correct to have one variable that describes the state of the actual reservation, which is a boolean flag (on or off) plus a direction. This would be a shared parameter

(that is, a media parameter whose value is negotiated since both sides have input to it), much like the direction attribute in SDP (a=sendonly, a=recvonly, a=sendrecv). When either side learns information about the state of the reservation, they would update this parameter and send it in a COMET. The reservation state can change in many ways; it can fail, for example, and therefore revert from bidirectionally reserved to no reservation at all. This is quite separate from the precondition, which we believe is a separate variable. The precondition includes a value of the reservation state that must be achieved before proceeding with the call (or more generally, alerting the user, since it can happen during a re-INVITE), and a strength (mandatory, optional). The precondition itself can be negotiated, since it, too, like the reservation state, is a shared parameter.

The precondition is met when the state of the reservation equals, or exceeds, the precondition. It can certainly exceed it if the precondition is for one direction, but the reservation state gets established bidirectionally. Therefore, we believe that manyfolks needs to define an absolute order of the values of the reservation state in order to correctly operate. Once that is done, a concise statement on preconditions can be made. Specifically, the state of the precondition and the state of the reservation can both evolve over time through updates in COMET, initiated by either side. Once the states reach a point where the reservation state meets or exceeds the precondition value, the callee can be alerted and the processing continues.

Even though we believe the reservation state and the precondition are ideally separate attributes, we believe that this semantic can be associated with the existing syntax, although awkwardly.

The confirm tag in manyfolks does not actually describe reservation state. It can be viewed as a single-ended parameter (that is, a parameter whose state is not negotiated, but rather, each side provides its own instance of the parameter) that requests for a new offer when the state of the reservation changes to a particular value.

Because we propose that the various SDP exchanges are not different from normal SDP offer/answers, there is no longer any need for the separate "precondition" Content-Disposition described in manyfolks.

We begin with the simplest case, which is a basic call, without early media. The flow is nearly identical to the one in manyfolks. However, using a pure offer/answer model as proposed moves the SDP from PRACKs to COMETs.

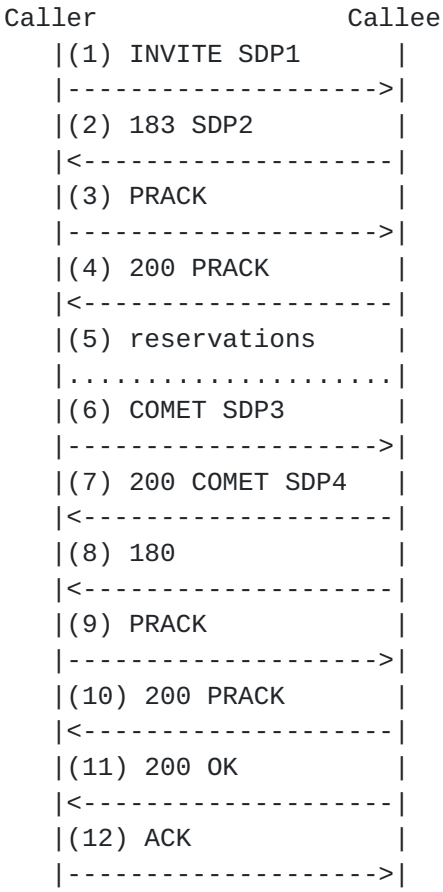


Figure 11: Manyfolks basic call

The caller sends an INVITE with SDP1. This SDP has a single audio stream, which indicates a strength of mandatory, a direction of sendrecv, and a confirmation. The INVITE has a Require header to indicate that the UAS has to support preconditions. The UAS receives the INVITE, and notices the required precondition processing. The precondition indicates that sendrecv reservation has to be in place before proceeding with the call. The UAS returns a 183 with an answer to the SDP. This answer also indicates a mandatory precondition, with sendrecv direcitonality, and confirmation. This is PRACKed. Both sides perform resource reservation, and both succeed. The UAC issues a COMET request. From its perspective, the state of the reservation state for the media stream is now successful in the receive direction, so the SDP in the COMET indicates that. Since the UAS has also succeeded, it knows that the state of the reservation is actually sendrecv, and so the answer in the SDP updates the state to

successful in the sendrecv direction. Since this is the precondition,

the UAS can now proceed with the call. It alerts the user, returns a 180 ringing, and eventually answers the call. Note that neither the 200 or ACK contain SDP.

Although the flow does not appear to support early media, it does. Once the response to the COMET arrives, the stream is active and the reservations have succeeded, and thus, early media can be sent. If the caller wished to avoid early media, the initial offer would indicate an inactive media stream, which could be updated in an offer in the 2xx, as in the exchanges above (excepting that the offer in the 2xx would continue to indicate that the reservation state was established). The result is that there is no special casing for early media here, and that is because early media is not being treated specially, its just the normal session.

Since the exchanges are just the normal offer/answer, other aspects of the media streams can be updated, in addition to the reservation state. For example, the codec set can be modified, in order to perform resource reservation for the bandwidth requirements of a specific codec, rather than a set. The flow for this case is shown in Figure 12. The call proceeds initially as in Figure 11. However, once the answer comes in the 183, the UAC decides to modify the codec set. So, it generates a new offer in a COMET. The reservation state and preconditions remain unchanged, but now there is a single codec. The answer in the 200 to the COMET also indicates no change in reservation or precondition state, but confirms that a single codec is in use, and that is used to guide the reservations.

Once preconditions are met, an additional COMET is sent (message 8) with an updated offer with the new values of the reservation state (success in the receive direction, in the case of RSVP, for example). This arrives at the callee. Since the callee has also succeeded in their reservation (which was in the caller to callee direction), the callee now knows that bidirectional reservations exist. Thus, the SDP in the answer further updates the reservation state to succeeded in both directions. This meets the precondition, and so it alerts and the call proceeds as in Figure 11.

5.6 The Heterogeneous Error Response Forking Problem (HERFP)

One aspect of our proposal is that a UAS can send a 155 response, instead of a final response, when supported by the UAC, to support services that are complicated by HERFP. The COMET can then be used to provide whatever information is requested by the error response. The COMET would operate just like the a re-INVITE would operate if the actual final response had been sent.

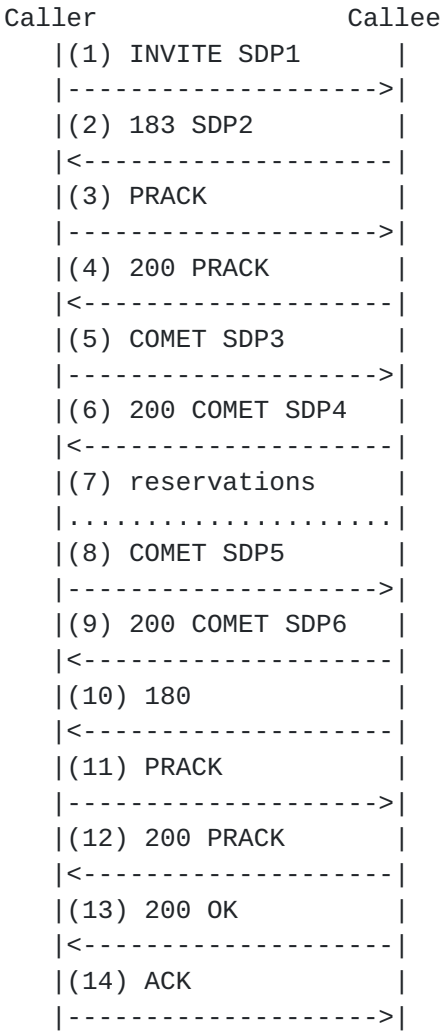


Figure 12: Modifying codecs in manyfolks

To show the effectiveness of our proposal, we once again consider the two scenarios of Section 2.6. The call flow for the first case is now shown in Figure 13, except now this time with our proposed solution. For brevity, we ignore the PRACKs for the provisional responses. As before, the caller sends an INVITE, and the proxy forks it. This time, the proxy inserts a Require header in the request that indicates services are being offered based on dialog state, and so the UAS should send provisionals instead of finals. UAS1 challenges for credentials, but this time, it sends a 155 response that contains the challenge in the WWW-Authenticate header (message 4). The proxy passes this upstream to the UAC. The UAC formulates the response, and

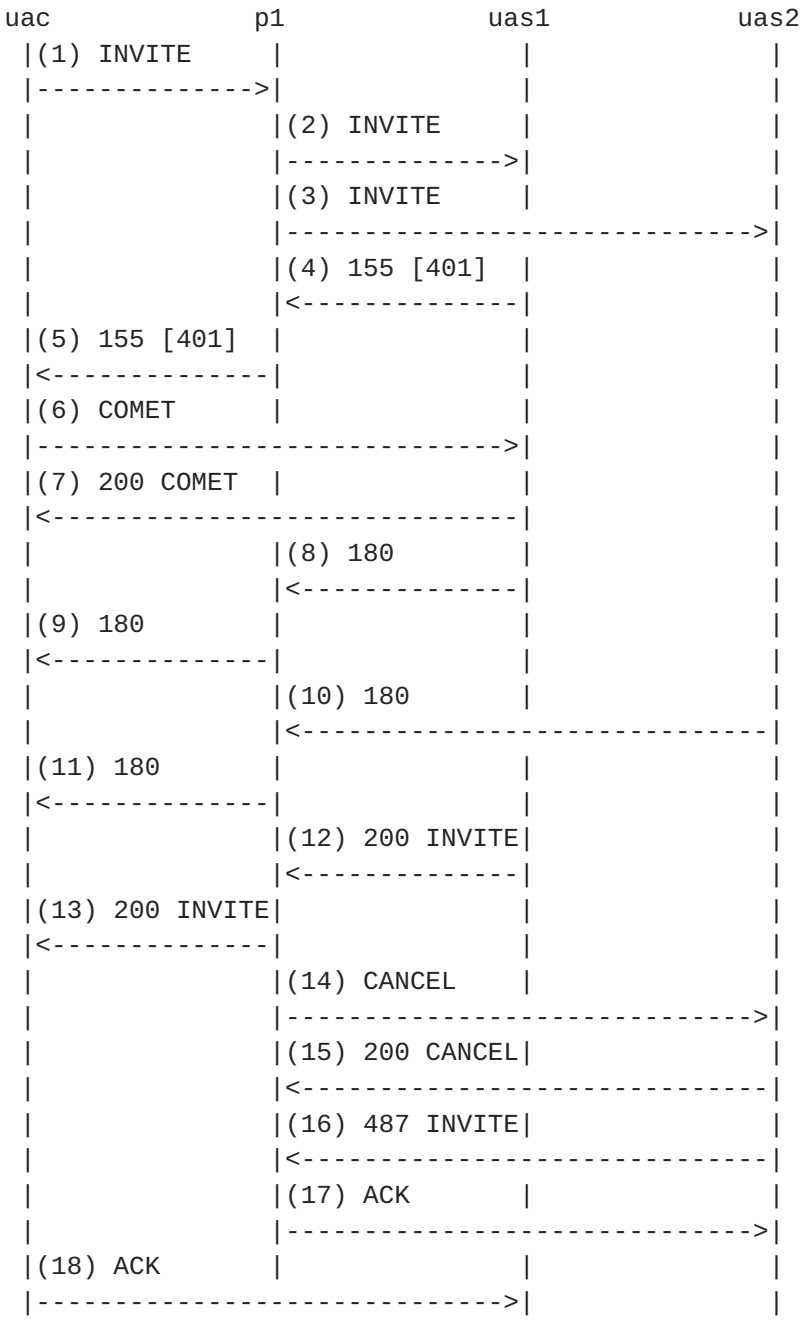


Figure 13: HERFP fix, scenario 1

places it in an Authorization header in the COMET (message 6). This goes directly to the UAS (the proxy did not record-route). Since the credentials are valid, the UAS proceeds with the session and rings

the phone. It therefore generates a 180 response to the intial INVITE

(message 8), which is passed to the UAC. UAS2 does not challenge, and generates an immediate 180, which is passed to the UAC as well. In this example, as discussed in Section 2.6, the user is at UAS1, the call is answered there, resulting in a 200 OK (message 12). The proxy cancels the branch towards UAS2, and the call completes successfully this time!

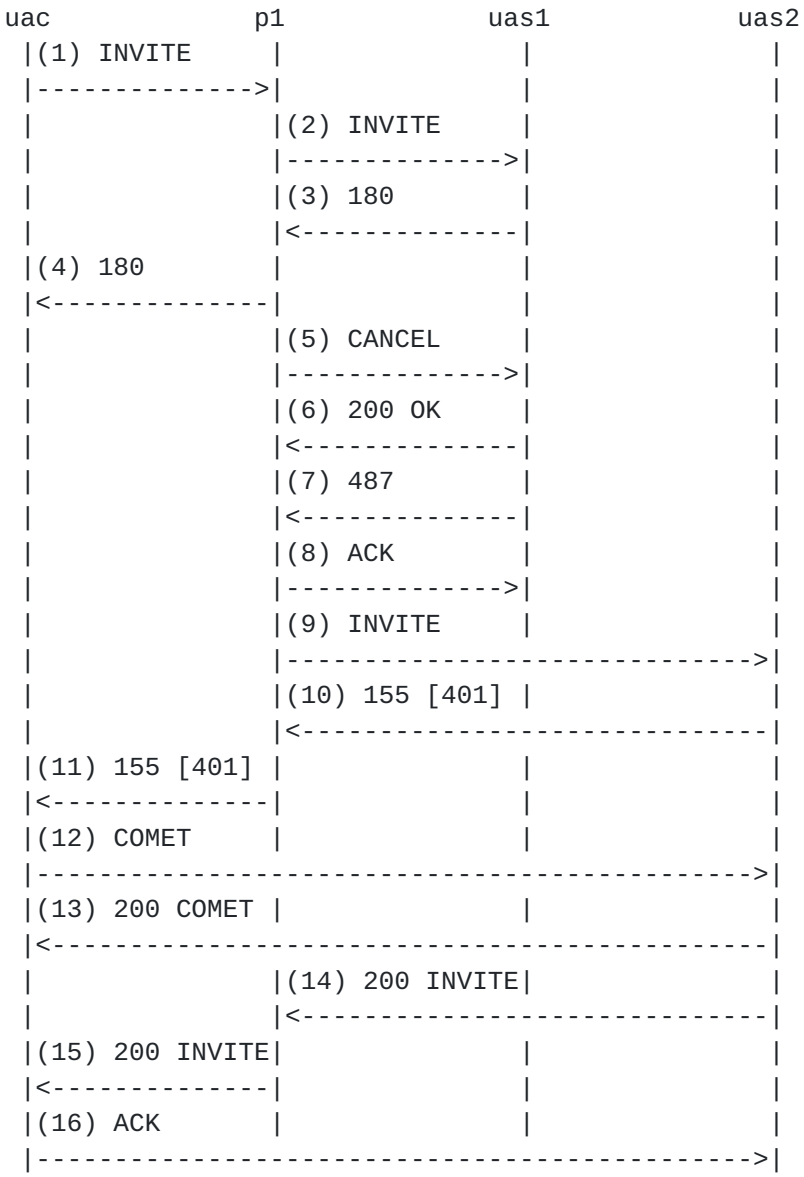


Figure 14: HERFP fix, scenario 2

Consider the second example of Section 2.6. The flow for this example, this time with our proposed solution, is shown in Figure 14. The initial flow proceeds as in Figure 3. UAS1 is rung, and there is no answer, resulting in a cancellation and an attempt to ring UAS2. UAS2 wishes to challenge. However, this time, it issues a 155 that otherwise looks like a 401, which contains a WWW-Authenticate header with the challenge. This response is passed to the proxy and forwarded to the UAC (once again, PRACK requests are not shown). The UAC generates credentials for the challenge, and sends a COMET with the response to the challenge. This is sent directly to UAS2, since the proxy did not record-route. The credentials are accepted, causing the phone to ring. The user is there, so they pick up, generating a 200 OK, which is passed to the UAC, which sends an ACK to complete the call. Once again, a successful call setup!

6 Musings on COMET vs. re-INVITE

It is worth discussing the differences between COMET and re-INVITE in this proposed solution. The current proposal would allow COMET to effectively modify session state, just like re-INVITE. COMET can be used outside of a transaction, just like re-INVITE. Just like re-INVITE, it would need to include full session state when used that way. However, unlike re-INVITE, it cannot usefully query for user input, since it must be answered immediately (it is a non-INVITE request). In principle, COMET could establish session and dialog state on crashed-and-rebooted or backup servers, although only when used outside of an INVITE transaction (since within one, it need not have full state, as in the HERFP flows above, where no SDP is present). COMET would never be used for services at proxies or intermediaries because of its pure end-to-end significance. However, it is worth noting that proxies do not provide services on re-INVITE anyway, for much the same reason - it makes little sense for in-progress calls. In hindsight, a separate method for re-INVITE, although with the same semantics as re-INVITE, might have been a better choice. However, we feel it is too widely used to be deprecated, and since COMET provides no real benefit for outside-of-a-transaction updates above and beyond re-INVITE, we believe re-INVITE should continue to exist and be used.

7 Proposed Process for the SIP WG

Our proposed action plan for IETF, should this proposal be accepted, is the following:

- COMET would NOT be integrated into bis. The rules in bis on where offer and answer can be placed need to be generalized, such that its still INVITE/200 or 200/ACK for baseline operation, but broader when COMET is used. No other changes to

bis are needed.

- Manyfolks would be split into a COMET definition spec, and a preconditions specification. The COMET specification would enable useful early media, although it would discuss it more from an example perspective. Early media, as a problem, becomes less complex as a result of our proposal to not view it as special in any way. We would also not opposed keeping them in one specification to speed up the process.

- The precondition specification would depend on the COMET specification. It would capture just the SDP attributes and the allowed evolutions in the precondition and reservation state. It would discuss how the precondition states impact call processing. The draft would be significantly shorter as a result of its more limited scope, although it would do much more because of the generalization that would be accomplished.

- There are no changes to the offer/answer specification.

A result of this is that HERFP would not be solved in bis, but only through an extension. We feel this is reasonable given the relatively large scope of the change required to fix it. The advantage of our process proposal is that it doesn't delay bis at all.

8 Open Issues

There are definitely some open issues left. They are:

- Do we also allow an offer/answer exchange in PRACK? It might save some messages. I chose not to in this specification, primarily for simplicity.

- Is the SDP in 1xx to INVITE an offer or answer? This specification says its an answer. This is different from our earlier proposals for early media. Keeping it as an answer is needed to really unify early and final media. However, it has the side effect of introducing transient conditions of multiple media streams on the same port in the case of forking. The alternative, making it an offer, avoids that problem, but introduces others. More specifically, it guarantees clipping of media, and requires substantial additional protocol machinery to signal whether an SDP is an offer or answer. Since it is possible to handle multiple media streams on the same port, but the clipping is unfixable, we chose to go for making it an answer.

- More thinking needs to happen on security. It is unclear to me

if the usage of COMET to provide credentials for a different request creates any problems. The only one I came up with was that it can introduce a TCP-SYN style attack at a UAS, since the UAS retains transaction state even for an unauthetnicated request. However, this problem is less of an issue in a UAS than a proxy. The UAS could use 155 for the first two calls, and then statelessly reject any other unauthenticated requests. That would work for single user devices. Gateways typically don't challenge individual requests at all, so there is not likely to be an issue there either. Thus, the TCP-SYN style problem may not be applicable in this case because its a UAS, not a proxy.

- With all of the various places one can find offers and answers in SIP message, it might get confusing about where one can find each of them, and whether a particular SDP is an answer to an offered one. Historically, we have gotten into trouble whenever there was no explicit matching between things (m lines in SDP). Therefore, it might be valuable to specify a mechanism for matching answers to offers. This would provide substantial flexibility in which messages they can occur, beyond the rules specified above. We believe this would have very positive impacts on 3pcc, for example, which could become more robust through the use of COMET to exchange the various offers before call completion.

9 Acknowledgements

We'd like to thank Robert Sparks for his review and comments.

10 Author's Addresses

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

11 Bibliography

[1] J. Rosenberg and H. Schulzrinne, "An offer/answer model with SDP," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[2] J. Rosenberg, H. Schulzrinne, et al. , "SIP: Session initiation protocol," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[3] M. Handley and V. Jacobson, "SDP: session description protocol," Request for Comments 2327, Internet Engineering Task Force, Apr. 1998.

[4] F. Andreasen, "SDP simple capability negotiation," Internet Draft, Internet Engineering Task Force, Aug. 2001. Work in progress.

[5] W. Marshall et al. , "Integration of resource management and SIP," Internet Draft, Internet Engineering Task Force, Aug. 2001. Work in progress.

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

[7] R. Blom et al. , "The secure real time transport protocol," Internet Draft, Internet Engineering Task Force, Feb. 2001. Work in progress.

[8] J. Rosenberg, "SIP early media," Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.

[9] S. Sen et al. , "Early media issues and scenarios," Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.

[10] E. Burger, "Why early media in sip," Internet Draft, Internet Engineering Task Force, Oct. 2001. Work in progress.

[11] J. Rosenberg, "Reconsituting call state in SIP user agents," Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.

[12] H. Schulzrinne, G. Camarillo, and D. Oran, "The reason header field for the session initiation protocol," Internet Draft, Internet Engineering Task Force, Dec. 2001. Work in progress.

[13] S. Floyd and L. Daigle, "IAB architectural and policy considerations for OPES," Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

Full Copyright Statement