Internet Engineering Task Force SIPPING WG
Internet Draft

J. Rosenberg
dynamicsoft

draft-ietf-sipping-conferencing-framework-00.txt
May 1, 2003
Expires: October 2003

A Framework for Conferencing with the Session Initiation Protocol

STATUS OF THIS MEMO

Abstract

The Session Initiation Protocol (SIP) supports the initiation, modification, and termination of media sessions between user agents. These sessions are managed by SIP dialogs, which represent a SIP relationship between a pair of user agents. Because dialogs are between pairs of user agents, SIP's usage for two-party communications (such as a phone call), is obvious. Communications sessions with multiple participants, generally known as conferencing, are more complicated. This document defines a framework for how such conferencing can occur. This framework describes the overall architecture, terminology, and protocol components needed for multi-party conferencing.

Table of Contents

1 Introduction

The Session Initiation Protocol (SIP) [1] supports the initiation, modification, and termination of media sessions between user agents. These sessions are managed by SIP dialogs, which represent a SIP relationship between a pair of user agents. Because dialogs are between pairs of user agents, SIP's usage for two-party communications (such as a phone call), is obvious. Communications sessions with multiple participants, however, are more complicated. SIP can support many models of multi-party communications. One, referred to as loosely coupled conferences, makes use of multicast media groups. In the loosely coupled model, there is no signaling relationship between participants in the conference. There is no central point of control or conference server. Participation is gradually learned through control information that is passed as part of the conference (using the Real Time Control Protocol (RTCP) [2], for example). Loosely coupled conferences are easily supported in SIP by using multicast addresses within its session descriptions.

In another model, referred to as fully distributed multiparty conferencing, each participant maintains a signaling relationship with each other participant, using SIP. There is no central point of control; it is completely distributed amongst the participants. This model is outside the scope of this document.

In another model, sometimes referred to as the tightly coupled conference, there is a central point of control. Each participant connects to this central point. It provides a variety of conference functions, and may possibly perform media mixing functions as well. Tightly coupled conferences are not directly addressed by RFC 3261, although basic participation is possible without any additional protocol support.

This document is one of a series of specifications that discusses tightly coupled conferences. Here, we present the overall framework for tightly coupled conferencing, referred to simply as "conferencing" from this point forward. This framework presents a general architectural model for these conferences, presents terminology used to discuss such conferences, and describes the sets of protocols involved in a conference. The aim of the framework is to meet the general requirements for conferencing that are outlined in [3].

2 Terminology

Conference: Conference is an overused term which has different meanings in different contexts. In SIP, a conference is an instance of a multi-party conversation. Within the context

of this specification, a conference is always a tightly coupled conference.

Loosely Coupled Conference: A loosely coupled conference is a conference without coordinated signaling relationships amongst participants. Loosely coupled conferences frequently use multicast for distribution of conference memberships.

Tightly Coupled Conference: A tightly coupled conference is a conference in which a single user agent, referred to as a focus, maintains a dialog with each participant. The focus plays the role of the centralized manager of the conference, and is addressed by a conference URI.

Focus: The focus is a SIP user agent that is addressed by a conference URI and identifies a conference (recall that a conference is a unique instance of a multi-party conversation). The focus maintains a SIP signaling relationship with each participant in the conference. The focus is responsible for ensuring, in some way, that each participant receives the media that make up the conference. The focus also implements conference policies. The focus is a logical role.

Conference URI: A URI, usually a SIP URI, which identifies the focus of a conference.

Participant: The software element that connects a user or automata to a conference. It implements, at a minimum, a SIP user agent, but may also include a conference policy control protocol client, for example.

Conference Notification Service: A conference notification service is a logical function provided by the focus. The focus can act as a notifier [4], accepting subscriptions to the conference state, and notifying subscribers about changes to that state. The state includes the state maintained by the focus itself, the conference policy, and the media policy.

Conference Policy Server: A conference policy server is a logical function which can store and manipulate the conference policy. The conference policy is the overall set of rules governing operation of the conference. It is broken into membership policy and media policy. Unlike the focus, there is not an instance of the conference policy server for each conference. Rather, there is an instance of

the membership and media policies for each conference.

Conference Policy: The complete set of rules for a particular conference manipulated by the conference policy server. It includes the membership policy and the media policy. There is an instance of conference policy for each conference.

Membership Policy: A set of rules manipulated by the conference policy server regarding participation in a specific conference. These rules include directives on the lifespan of the conference, who can and cannot join the conference, definitions of roles available in the conference and the responsibilities associated with those roles, and policies on who is allowed to request which roles.

Media Policy: A set of rules manipulated by the conference policy server regarding the media composition of the conference. The media policy is used by the focus to determine the mixing characteristics for the conference. The media policy includes rules about which participants receive media from which other participants, and the ways in which that media is combined for each participant. In the case of audio, these rules can include the relative volumes at which each participant is mixed. In the case of video, these rules can indicate whether the video is tiled, whether the video indicates the loudest speaker, and so on.

Conference Policy Control Protocol (CPCP): The protocol used by clients to manipulate the conference policy.

Mixer: A mixer receives a set of media streams of the same type, and combines their media in a type-specific manner, redistributing the result to each participant. This includes media transported using RTP [2]. As a result, the term defined here is a superset of the mixer concept defined in RFC 1889, since it allows for non-RTP-based media such as instant messaging sessions [5].

Conference-Unaware Participant: A conference-unaware participant is a participant in a conference that is not aware that it is actually in a conference. As far as the UA is concerned, it is a point-to-point call.

Cascaded Conferencing: A mechanism for group communications in which a set of conferences are linked by having their focuses interact in some fashion.

Simplex Cascaded Conferences: a group of conferences which are

linked such that the user agent which represents the focus of one conference is a conference-unaware participant in another conference.

Conference-Aware Participant: A conference-aware participant is a participant in a conference that has learned, through automated means, that it is in a conference, and that can use a conference policy control protocol, media policy control protocol, or conference subscription, to implement advanced functionality.

Conference Server: A conference server is a physical server which contains, at a minimum, the focus. It may also include a conference policy server and mixers.

Mass Invitation: A conference policy control protocol request to invite a large number of users into the conference.

Mass Ejection: A conference policy control protocol request to remove a large number of users from the conference.

Sidebar: A sidebar appears to the users within the sidebar as a "conference within the conference". It is a conversation amongst a subset of the participants to which the remaining participants are not privy.

Anonymous Participant: An anonymous participant is one that is known to other participants through the conference notification service, but whose identity is being withheld.

Hidden Participant: A hidden participant is one that is not known to other participants in the conference. They may be known to the moderator, depending on conference policy.

3 Overview of Conferencing Architecture

The central component (literally) in a SIP conference is the focus. The focus maintains a SIP signaling relationship with each participant in the conference. The result is a star topology, shown in Figure 1.

The focus is responsible for making sure that the media streams which constitute the conference are available to the participants in the conference. It does that through the use of one or more mixers, each of which combines a number of input media streams to produce one or more output media streams. The focus uses the media policy to determine the proper configuration of the mixers.

```
                           +-----------+
                           |           |
                           |           |
                           |Participant|
                           |     4     |
                           |           |
                           +-----------+
                                 |
                                 |SIP
                                 |Dialog
                                 |4
                                 |
   +-----------+           +-----------+            +-----------+
   |           |           |           |            |           |
   |           |           |           |            |           |
   |Participant|-----------|   Focus   |------------|Participant|
   |     1     |  SIP      |           |   SIP      |     3     |
   |           |  Dialog   |           |   Dialog   |           |
   +-----------+  1        +-----------+   3        +-----------+
                                 |
                                 |
                                 |SIP
                                 |Dialog
                                 |2
                                 |
                           +-----------+
                           |           |
                           |           |
                           |Participant|
                           |    2      |
                           |           |
                           +-----------+
```

Figure 1: SIP Conference Architecture

The focus has access to the conference policy (composed of the membership and media policies), an instance of which exist for each conference. Effectively, the conference policy can be thought of as a

database which describes the way that the conference should operate. It is the responsibility of the focus to enforce those policies. Not only does the focus need read access to the database, but it needs to know when it has changed. Such changes might result in SIP signaling (for example, the ejection of a user from the conference using BYE), and most changes will require a notification to be sent to subscribers using the conference notification service.

The conference is represented by a URI, which identifies the focus. Each conference has a unique focus and a unique URI identifying that focus. Requests to the conference URI are routed to the focus for that specific conference.

Users usually join the conference by sending an INVITE to the conference URI. As long as the conference policy allows, the INVITE is accepted by the focus and the user is brought into the conference. Users can leave the conference by sending a BYE, as they would in a normal call.

Similarly, the focus can terminate a dialog with a participant, should the conference policy change to indicate that the participant is no longer allowed in the conference. A focus can also initiate an INVITE, should the conference policy indicate that the focus needs to bring a participant into the conference.

The notion of a conference-unaware participant is important in this framework. A conference-unaware participant does not even know that the UA it is communicating with happens to be a focus. As far as it's concerned, its a UA just like any other. The focus, of course, knows that its a focus, and it performs the tasks needed for the conference to operate.

Conference-unaware participants have access to a good deal of functionality. They can join and leave conferences using SIP, and obtain more advanced features through stimulus signaling, as discussed in However, if the participant wishes to explicitly control aspects of the conference using functional signaling protocols, the participant must be conference-aware.

A conference-aware participant is one that has access to advanced functionality through additional protocol interfaces. The client uses these protocols to interact with the conference policy server and the focus. A model for this interaction is shown in Figure 2. The participant can interact with the focus using extensions, such as REFER, in order to access enhanced call control functions [6]. The participant can SUBSCRIBE to the conference URI, and be connected to the conference notification service provided by the focus. Through

this mechanism, it can learn about changes in participants (effectively, the state of the dialogs), the media policy, and the membership policy.

The participant can communicate with the conference policy server using a conference policy control protocol. Through this protocol, it can affect the conference policy. The conference policy server need not be available in any particular conference, although there is always a conference policy.

The interfaces between the focus and the conference policy, and the conference policy server and the conference policy, are not subject to standardization at the time of this writing. They are intended primarily to show the logical roles involved in a conference, as opposed to suggesting a physical decomposition. The separation of these functions is documented here to encourage clarity in the requirements and to allow individual implementations the flexibility to compose a conferencing system in a scalable and robust manner.

3.1 Usage of URIs

It is fundamental to this framework that a conference is uniquely identified by a URI, and that this URI identifies the focus which is responsible for the conference. The conference URI is unique, such that no two conferences have the same conference URI. A conference URI is always a SIP or SIPS URI.

The conference URI is opaque to any participants which might use it. There is no way to look at the URI, and know for certain whether it identifies a focus, as opposed to a user or an interface on a PSTN gateway. This is in line with the general philosophy of URI usage [7]. However, contextual information surrounding the URI (for example, SIP header parameters) may indicate that the URI represents a conference.

When a SIP request is sent to the conference URI, that request is routed to the focus, and only to the focus. The element or system that creates the conference URI is responsible for guaranteeing this property.

The conference URI can represent a long-lived conference or interest group, such as "sip:discussion-on-dogs@example.com". The focus identified by this URI would always exist, and always be managing the conference for whatever participants are currently joined. Other conference URIs can represent short-lived conferences, such as an ad-hoc conference.

Ideally, a conference URI is never constructed or guessed by a user.

```
                         .....................................
                         .                                   .
                         .                                   .
                         .                                   .
                         .                                   .
                         .                      Conference   .
                         .                        Policy     .
            Conference   .                                   .
            Policy       . +-----------+        //-----\\    .
            Control      . |           |      ||         ||  .
            Protocol     . | Conference|        \\-----//    .
         +---------------->|  Policy   |       |         |   .
         |               . |  Server   |---->  |Membership   .
         |               . |           |       |         |   .
         |               . +-----------+       |    &    |   .
         |               .                     |         |   .
         |               .                     | Media   |   .
   +-----------+         . +-----------+       |   Policy|   .
   |           |         . |           |        \       //   .
   |           |         . |           |         \-----/     .
   |Participant|<--------->|   Focus   |            |        .
   |           |  SIP    . |           |            |        .
   |           |  Dialog . |           |<-----------+        .
   +-----------+         . |...........|                     .
             ^           . | Conference|                     .
             |           . |Notification                     .
             +------------>|  Service  |                     .
             Subscription. +-----------+                     .
                         .                                   .
                         .                                   .
                         .                                   .
                         .                                   .
                         .....................................

                                     Conference
                                      Functions
```

Figure 2: Conference-Aware Participant

Rather, conference URIs are learned through many mechanisms. A conference URI can be emailed or sent in an instant message. A conference URI can be linked on a web page. A conference URI can be obtained from a conference policy control protocol, which can be used to create conferences and the policies associated with them.

To determine that a SIP URI does represent a focus, standard techniques for URI capability discovery can be used. Specifically, the caller preferences specification [8] provides the "isfocus" feature tag to indicate that the URI is a focus. Caller preferences parameters are also used to indicate that a focus supports the conference notification service. This is done by declaring support for the SUBSCRIBE method and the relevant package(s) in the caller preferences feature parameters associated with the conference URI.

The other functions in a conference are also represented by URIs. If the conference policy server is implemented through web pages, this server is identified by HTTP URIs. If it is accessed using an explicit protocol, it is a URI defined for that protocol.

Starting with the conference URI, the URIs for the other logical entities in the conference can be learned using the conference notification service.

4 Functions of the Elements

This section gives a more detailed description of the functions typically implemented in each of the elements.

4.1 Focus

As its name implies, the focus is the center of the conference. All participants in the conference are connected to it by a SIP dialog. The focus is responsible for maintaining the dialogs connected to it. It ensures that the dialogs are connected to a set of participants who are allowed to participate in the conference, as defined by the membership policy. The focus also uses SIP to manipulate the media sessions, in order to make sure each participant obtains all the media for the conference. To do that, the focus makes use of mixers.

When a focus receives an INVITE, it checks the membership policy. The membership policy might indicate that this participant is not allowed to join, in which case the call can be rejected. It might indicate that another participant, acting as a moderator, needs to approve this new participant. In that case, the INVITE might be parked on a music-on-hold server, or a 183 response might be sent to indicate progress. A notification, using the conference notification service, would be sent to the moderator. The moderator then has the ability to

manipulate the policies using the conference policy control protocol. If the policies are changed to allow this new participant, the focus can accept the INVITE (or unpark it from the music-on-hold server). The interpretation of the membership policy by the focus is, itself, a matter of local policy, and not subject to standardization.

If a participant manipulated the membership policy to indicate that a certain other participant was no longer allowed in the conference, the focus would send a BYE to that other participant to remove them. This is often referred to as "ejecting" a user from the conference. The process of ejecting fundamentally constitutes these two steps - the establishment of the policy through the conference policy protocol, and the implementation of that policy (using a BYE) by the focus.

Similarly, if a user manipulated the membership policy to indicate that a number of users need to be added to the conference, the focus would send an INVITE to those participants. This is often referred to as the "mass invitation" function. As with ejection, it is fundamentally composed of the policy functions that specify the participants which should be present, and the implementation of those functions. A policy request to add a set of users might not require an INVITE to execute it; those users might already be participants in the conference.

A similar model exists for media policy. If the media policy indicates that a participant should not receive any video, the focus might implement that policy by sending a re-INVITE, removing the media stream to that participant. Alternatively, if the video is being centrally mixed, it could inform the mixer to send a black screen to that participant. The means by which the policy is implemented are not subject to specification.

4.2 Conference Policy Server

The conference policy server allows clients to manipulate and interact with the conference policy. The conference policy is used by the focus to make authorization decisions and guide its overall behavior. Logically speaking, there is a one-to-one mapping between a conference policy and a focus.

The conference policy is represented by a URI. There is a unique conference policy for each conference. The conference policy URI points to a conference policy server which can manipulate that conference policy. A conference policy server also has a "top level" URI which can be used to access functions that are independent of any conference. Perhaps the most important of these functions is the creation of a new conference. Creation of a new conference will

result in the construction of a new focus and a corresponding conference URI, which can then be used to join the conference itself, along with a media policy and conference policy.

The conference policy server is accessed using a client-server transactional protocol. The client can be a participant in the conference, or it can be a third party. Access control lists for who can modify a conference policy are themselves part of the conference policy.

The conference policy server is responsible for reconciliation of potentially conflicting requests regarding the policy for the conference.

The client of the conference policy control protocol can be any entity interested in manipulating the conference policy. Clearly, participants might be interested in manipulating them. A participant might want to raise or lower the volume for one of the other participants it is hearing. Or, a participant might want to add a user to the conference.

A client of the conference policy protocol could also be another server whose job is to determine the conference policy. As an example, a floor control server is responsible for determining which participant(s) in a conference are allowed to speak at any given time, based on participant requests and access rules. The floor control server would act as a client of the conference policy server, and change the media policy based on who is allowed to speak.

The client of the conference policy control protocol could also be another conference policy server.

4.3 Mixers

A mixer is responsible for combining the media streams that make up the conference, and generating one or more output streams that are distributed to recipients (which could be participants or other mixers). The process of combining media is specific to the media type, and is directed by the focus, under the guidance of the rules described in the media policy.

A mixer is not aware of a "conference" as an entity, per se. A mixer receives media streams as inputs, and based on directions provided by the focus, generates media streams as outputs. There is no grouping of media streams beyond the policies that describe the ways in which the streams are mixed.

A mixer is always under the control of a focus. The focus is

responsible for interpreting the media policy, and then installing the appropriate rules in the mixer. If the focus is directly controlling a mixer, the mixer can either be co-resident with the focus, or can be controlled through some kind of protocol.

However, a focus need not directly control a mixer. Rather, a focus can delegate the mixing to the participants, each of which has their own mixer. This is described in Section 6.4.

4.4 Conference Notification Service

The focus can provide a conference notification service. In this role, it acts as a notifier, as defined in RFC 3265 [4]. It accepts subscriptions from clients for the conference URI, and generates notifications to them as the state of the conference changes.

This state is composed of two separate pieces. The first is the state of the focus and the second is the conference policy. A subscriber to the conference notification service can use capabilities defined in the SIP events framework [4] to request that it receive focus state changes only, conference policy changes only, or both.

The state of the focus includes the participants connected to the focus, and information about the dialogs associated with them. As new participants join, this state changes, and is reported through the notification service. Similarly, when someone leaves, this state also changes, allowing subscribers to learn about this fact.

As described previously, the conference policy includes the membership policy and the media policy. As those policies change, due to usage of the CPCP, direct change by the focus, or through an application, the conference notification service informs subscribers of these changes.

4.5 Participants

A participant in a conference is any SIP user agent that has a dialog with the focus. This SIP user agent can be a PC application, a SIP hardphone, or a PSTN gateway. It can also be another focus. A conference which has a participant that is the focus of another conference is called a simplex cascaded conference. They can also be used to provide scalable conferences where there are regional sub-conferences, each of which is connected to the main conference.

4.6 Conference Policy

The conference policy contains the rules that guide the operation of the focus. The rules can be simple, such as an access list that

defines the set of allowed participants in a conference. The rules can also be incredibly complex, specifying time-of-day based rules on participation conditional on the presence of other participants. It is important to understand that there is no restriction on the type of rules that can be encapsulated in a conference policy.

The conference policy can be manipulated using web applications or voice applications. It can also be manipulated with proprietary protocols. However, the conference policy control protocol can be used as a standardized means of manipulating the conference policy. By the nature of conference policies, not all aspects of the policy can be manipulated with the conference policy control protocol.

The conference policy includes the membership policy and the media policy. The membership policy includes per-participant policies that specify how the focus is to handle a particular participant. These include whether or not the participant is anonymous, for example.

The media policy describes the way in which the set of inputs to a mixer are combined to generate the set of outputs. Media policies can span media types. In other words, the policy on how one media stream is mixed can be based on characteristics of other media streams. Media policies can be based on any quantifiable characteristic of the media stream (its source, volume, codecs, speaking/silence, etc.), and they can be based on internal or external variables accessible by the media policy.

Some examples of media policies include:

- o The video output is the picture of the loudest speaker (video follows audio).

- o The audio from each participant will be mixed with equal weight, and distributed to all other participants.

- o The audio and video that is distributed is the one selected by the floor control server.

5 Common Operations

There are a large number of ways in which users can interact with a conference. They can join, leave, set policies, approve members, and so on. This section is meant as an overview of the major conferencing operations, summarizing how they operate. More detailed examples of the SIP mechanisms can be found in [6].

5.1 Creating Conferences

There are many ways in which a conference can be created. The creation of a conference actually constructs several elements all at the same time. It results in the creation of a focus and a conference policy. It also results in the construction of a conference URI, which uniquely identifies the focus. Since the conference URI needs to be unique, the element which creates conferences is responsible for guaranteeing that uniqueness. This can be accomplished deterministically, by keeping records of conference URIs, or by generating URIs algorithmically, or probabilistically, by creating random URI with sufficiently low probabilities of collision.

When a media and conference policy are created, they are established with default rules that are implementation dependent. If the creator of the conference wishes to change those rules, they would do so using the conference policy control protocol (CPCP), for example.

Of course, using the CPCP requires that an element know the URI for manipulating the policy. That requires a means to learn the conference policy URI from the conference URI, since the conference URI is frequently the sole result returned to the client as a result of conference creation. Any other URIs associated with the conference are learned through the conference notification service. They are carried as elements in the notifications.

5.1.1 SIP Mechanisms

SIP can be used to create conferences hosted in a central server by sending an INVITE to a conferencing application that would automatically create a new conference and then place a user into it.

Creation of conferences where the focus resides in an endpoint operates differently. There, the endpoint itself creates the conference URI, and hands it out to other endpoints which are to be the participants. What differs from case to case is how the endpoint decides to create a conference.

One important case is the ad-hoc conference described in Section 6.2. There, an endpoint unilaterally decides to create the conference based on local policy. The dialogs that were connected to the UA are migrated to the endpoint-hosted focus, using a re-INVITE to pass the conference URI to the newly joined participants.

Alternatively, one UA can ask another UA to create an endpoint-hosted conference. This is accomplished with the SIP Join header [9]. The UA which receives the Join header in an invitation may need to create a new conference URI (a new one is not needed if the dialog that is being joined is already part of a conference). The conference URI is then handed to the recently joined participants through a re-INVITE.

5.1.2 CPCP Mechanisms

Another way to create a conference is through interaction with the conference policy server. Using the conference policy control protocol, a client can instruct the conference policy server to create a new conference and return the conference URI and conference policy URI.

5.1.3 Non-Automated Mechanisms

One way to create a conference is through interaction with an IVR application. The user would send a SIP INVITE to the conferencing application. This application would interact with the user, collect information about the desired conference, and create it. The user can then be placed into their newly created conference.

Of course, a user can also create conferences by interacting with a web server. The web server would prompt the user for the neccessary information (start and stop times of the conference, participants, etc.) and return the conference URI to the user. The user would copy this URI into their SIP phone, and send it an INVITE in order to join the newly-created conference.

5.2 Adding Participants

There are many mechanisms for adding participants to a conference. These include SIP, the conference policy control protocol, and non-automated means. In all cases, participant additions can be first party (a user adds themself) or third party (a user adds another user).

5.2.1 SIP Mechanisms

First person additions using SIP are trivially accomplished with a standard INVITE. A participant can send an INVITE request to the conference URI, and if the conference policy allows them to join, they are added to the conference.

If a UA does not know the conference URI, but has learned about a dialog which is connected to a conference (by using the dialog event package, for example [10]), the UA can join the conference by using the Join header to join the dialog.

Third party additions with SIP are done using REFER [11]. The client can send a REFER request to the participant, asking them to send an INVITE request to the conference URI. Additionally, the client can send a REFER request to the focus, asking it to send an INVITE to the participant. The latter technique has the benefit of allowing a

client to add a conference-unaware participant that does not support the REFER method.

5.2.2 CPCP Mechanisms

A basic function of the conference policy control protocol is to add participants. A client of the protocol can specify any SIP URI (which may identify themself) that is to be added. If the URI does not identify a user that is already a participant in the conference, the focus will send an INVITE to that URI in order to add them in.

5.2.3 Non-Automated Mechanisms

There are countless non-automated means for asking a participant to join the conference. Generally, they involve conveying the conference URI to the desired participant, so that they can send an INVITE to it. These mechanisms all require some kind of human interaction.

As an example, a user can send an instant message [12] to the third party, containing an HTML document which requests the user to click on the hyperlink to join the conference:

```
   <html>
   Hey, would you like to <a href="sip:
9sf88fk-99sd@conferences.example.com">join
   </a> the conference now?
   </html>
```

5.3 Conditional Joins

In many cases, a new participant will not wish to join the conference unless they can join with a particular set of policies. As an example, a participant may want to join anonymously, so that other participants know that someone has joined, but not who. To accomplish this, the conference policy control protocol is used to establish these policies prior to the generation or acceptance of an invitation to the conference. For example, if a user wishes to join a conference with a known conference URI, the user would obtain the URI for the conference policy, manipulate the policy to set themself as an anonymous participant, and then actually join the conference by sending an INVITE request to the conference URI.

5.4 Removing Participants

As with additions, there are several mechanisms for departures. These include SIP mechanisms and CPCP mechanisms. Removals can also be

first person or third person.

5.4.1 SIP Mechanisms

First person departures are trivially accomplished by sending a BYE request to the focus. This terminates the dialog with the focus and removes the participant from the conference.

Third person departures can also be done using SIP, through the REFER method.

5.4.2 CPCP Mechanisms

The CPCP can be used by a client to remove any participant (including themself). When CPCP is used for this purpose, the focus will send a BYE request to the participant that is being removed. The focus will execute any other signaling that is needed to remove them (for example, manipulate other dialogs in order to manage the change in media streams).

The conference policy control protocol can also be used to remove a large number of users. This is generally referred to as mass ejection.

5.4.3 Non-Automated Mechanisms

As with the other common conferencing functions, there are many non-automated ways to remove a participant. The identity of the participant can be entered into a web form. When the user clicks submit, the focus sends a BYE to that participant, removing them from the conference. Alternatively, the conference can expose an IM interface, where the user can send an IM to the conference saying "remove Bob", causing the conference server to remove Bob.

5.5 Approving Policy Changes

> OPEN ISSUE: The basic mechanism described here depends on the actual protocols used for conference and media policy manipulation. If the protocol itself provides change notifications, sip-events may not be needed for that purpose. Thus, this description here is tentative.

A conference policy for a particular conference may designate one or more users as moderators for some set of media policy or conference policy change requests. This means that those moderators need to approve the specific policy change. Typically, moderators are used to approve member additions and removals. However, the framework allows

for moderators to be associated with any policy change that can be made.

Moderating a policy request is done using a combination of the conference notification service and the CPCP protocol.

First, a client makes a policy change. This can be directly, using the CPCP, or indirectly. An indirect policy change request is any non-CPCP action that requires approval. The simplest example is an INVITE to the focus from a new participant. That represents a request to change the membership of the conference. From a moderation perspective, it is handled identically to the case where a client used the CPCP to request that the same user to be added to the conference.

Part of the conference policy itself may designate any policy change as moderated. This means that they change cannot be performed by the client directly. As a result, the CPCP request will be answered with a response saying that the action will be done pending authorization. That completes the CPCP transaction. In the case of a policy change requested indirectly through some other means, the behavior depends on the mechanism. For example, if a user sends a SIP INVITE request to the conference in order to join, and that join request is moderated, the focus would normally accept it and play music-on-hold until the request is approved.

Even though the CPCP transaction failed, it does result in a change in internal state. Specifically, the requested change shows up as a "pending" state within the media and conference policies. This means that the change has been requested, but has not taken effect. It is almost a form of change request history. However, because it is a state change, it is something that can result in notifications through the conference notification service.

Therefore, in order to moderate requests, the moderator subscribes to the conference policy notification service. Normally, the notifications from the focus do not reflect pending state changes. That is, the service will not normally send a notification informing a subscriber that a policy change request was made and failed due to lack of authorization. However, notifications to the moderator do reflect these changes. That is because the policy of the focus is to inform moderators, and only moderators, of these changes. Indeed, different users can be moderators for different parts of the conference and media policies. For example, one user can be a moderator for membership changes, and another, a moderator for whether users can be anonymously joined or not.

There are two ways that the focus knows whether a subscriber to the

conference notification service is a moderator. The first is configured policy (once again through CPCP). That policy can specify that a particular user is the moderator for a particular piece of policy. Therefore, if that user subscribes to the conference notification service, any notification sent to that user will include pending changes to that piece of policy. As an alternative, a SUBSCRIBE request from a user can include a filter [13] that requests receipt of these pending state changes. If the conference policy allows, that request is honored, and the subscriber will receive notifications about pending state changes.

Once the moderator receives a notification about the pending state change, they use the CPCP to implement their decision. If the moderator decides to approve the change, they use the CPCP or MPCP to actually perform the change themselves. Since the moderator for a piece of policy is allowed to change that piece of policy, by definition, their change is accepted and performed. If the moderator decides to reject the change, they use the CPCP to remove the pending state from the database.

The pending state persists in the database for a period of time which is, itself, part of the conference policy. If the moderator does not either approve or reject the change, the pending state eventually disappears, as if the change was explicitly rejected.

If the pending state is approved, a real change to the conference or media policy takes place, and this change will be reflected in the conference notification service. In this way, if a client makes a policy change, and their request is rejected because they are not authorized, the client can subscribe to the conference notification service to learn if their change is eventually approved or rejected.

This general mechanism for moderating policy requests is consistent with the moderation of presence subscriptions [14] [15].

5.6 Creating Sidebars

A sidebar is a "conference within a conference", allowing a subset of the participants to converse amongst themselves. Frequently, participants in a sidebar will still receive media from the main conference, but "in the background". For audio, this may mean that the volume of the media is reduced, for example.

A sidebar is represented by a separate conference URI. This URI is a type of "alias" for the main conference URI. Both route to the same focus. Like any other conference, the sidebar conference URI has a conference policy and a media policy associated with it. Like any other conference, one can join it by sending an INVITE to this URI,

or ask others to join by referring them to it. However, it differs from a normal conference URI in several ways. First, users in the main conference do not need to establish a separate dialog to the sidebar conference. The focus recognizes the sidebar as a special URI, and knows to use the existing dialog to the main conference as a "virtual" connection to the sidebar URI.

The second difference is the way in which conference and media policies are implemented. If the conference policy control protocol is used to add a user to a normal conference, the focus will typically send an INVITE to the participant to ask them to join. For a sidebar conference, it is done differently. If the conference policy control protocol is used to add a user to it, and that user is already part of the main conference, the focus will use the conference notification service to alert the existing participant that they have been asked to join the sidebar. The invited user can then make use of the CPCP to formally add themselves to the sidebar.

5.7 Destroying Conferences

Conferences can be destroyed in several ways. Generally, whether those means are applicable for any particular conference is a component of the conference policy.

When a conference is destroyed, the conference and media policies associated with it are destroyed. Any attempts to read or write those policies results in a protocol error. Furthermore, the conference URI becomes invalid. Any attempts to send an INVITE to it, or SUBSCRIBE to it, would result in a SIP error response.

Typically, if a conference is destroyed while there are still participants, the focus would send a BYE to those participants before actually destroying the conference. Similarly, if there were any users subscribed to the conference notification service, those subscriptions would be terminated by the server before the actual destruction.

5.7.1 SIP Mechanisms

There is no explicit means in SIP to destroy a conference. However, a conference may be destroyed as a by-product of a user leaving the conference, which can be done with BYE. In particular, if the conference policy states that the conference is destroyed once the last user leaves, when that user does leave (using a SIP BYE request), the conference is destroyed.

5.7.2 CPCP Mechanisms

The CPCP contains mechanisms for explicitly destroying a conference.

5.7.3 Non-Automated Mechanisms

As with conference creation, a conference can be destroyed by interacting with a web application or voice application that prompts the user for the conference to be destroyed.

5.8 Obtaining Membership Information

A participant in a conference will frequently wish to know the set of other users in the conference. This information can be obtained many ways.

5.8.1 SIP Mechanisms

The conference notification service allows a conference aware participant to subscribe to it, and receive notifications that contain the list of participants. When a new participant joins or leaves, subscribers are notified. The conference notification service also allows a user to do a "fetch" [4] to obtain the current listing.

5.8.2 CPCP Mechanisms

The CPCP contains mechanisms for querying for the current set of conference participants.

5.8.3 Non-Automated Mechanisms

Users can also interact with applications to obtain conference membership. There may be a conference web page associated with the conference, which has a link that will fetch the current list of participants and display them in the browser. Similarly, an interactive voice response application connected to the focus can be used to obtain the current membership. A user in the conference could press the pound key on their phone, and hear a listing of the current participants.

5.9 Adding and Removing Media

Each conference is composed of a particular set of media that the focus is managing. For example, a conference might contain a video stream and an audio stream. The set of media streams that constitute the conference can be changed by participants. When the set of media in the conference change, the focus will need to generate a re-INVITE to each participant in order to add or remove the media stream to each participant. When a media stream is being added, a participant can reject the offered media stream, in which case it will not

receive or contribute to that stream. Rejection of a stream by a participant does not imply that that the stream is no longer part of the conference - just that the participant is not involved in it.

There are several ways in which a media stream can be added or removed from a conference.

5.9.1 SIP Mechanisms

A SIP re-INVITE can be used by a participant to add or remove a media stream. This is accomplished using the standard offer/answer techniques for adding media streams to a session [16]. This will trigger the focus to generate its own re-INVITEs.

5.9.2 CPCP Mechanisms

The CPCP can be used to add or remove a media stream. This too will trigger the focus to generate a re-INVITE to each participant in order to affect the change.

5.9.3 Non-Automated Mechanisms

As with most of the other common functions, addition and removal of media streams can be accomplished with a web application or interactive voice application.

5.10 Conference Announcements and Recordings

Conference announcements and recordings play a key role in many real conferencing systems. Examples of such features include:

1. Asking a user to state their name before joining the conference, in order to support a roll call

2. Allowing a user to request a roll call, so they can hear who else is in the conference

3. Allowing a user to press some keys on their keypad in order to record the conference

4. Allowing a user to press some keys on their keypad in order to be connected with a human operator

5. Allowing a user to press some keys on their keypad to mute or unmute their line

In this framework, these capabilities are modeled as an application

```
                              User 1
                           +-----------+
                           |           |
                           |           |
                           |Participant|
                           |     1     |
                           |           |
                           +-----------+
                                 |SIP
                                 |Dialog
                      Conference |1
                      Policy +---|--------+
      User 2          Server |   |        |          Application
   +-----------+           +-----------+  |   CPCP  *************
   |           |           |           |  |-------- *           *
   |           |           |           |  |         *           *
   |Participant|-----------|   Focus   |------------*Participant*
   |     2     |  SIP      |           |  |  SIP    *     4     *
   |           |  Dialog   |           |--+  Dialog *           *
   +-----------+  2        +-----------+     4      *************
                                 |
                                 |
                                 |SIP
                                 |Dialog
                                 |3
                                 |
                           +-----------+
                           |           |
                           |           |
                           |Participant|
                           |    3      |
                           |           |
                           +-----------+
                              User 3
```

Figure 3: Conference announcement application

which acts as a participant in the conference. This is shown pictorially in Figure 3. The conference has four participants. Three of these participants are end users, and the fourth is the announcement application.

If the announcement application wishes to play an announcement to all the conference members (for example, to announce a join), it merely sends media to the mixer as would any other participant. The announcement is mixed in with the conversation and played to the participants.

Similarly, the announcement application can play an announcement to a specific user by using the CPCP to configure its media policy so that the media it generates is only heard by the target user. The application then generates the desired announcement, and it will be heard only by the selected recipient.

The announcement application can also receive input from a specific user through the conference. The announcement application would use the CPCP to cause in-band DTMF to be dropped from the mix, and sent only to itself. When a user wishes to invoke an operation, such as to obtain a roll call, the user would press the appropriate key sequence. That sequence would be heard only by the announcement application. Once the application determines that the user wishes to hear a roll call, it can use the CPCP to set the media policy so that media from that user is delivered only to the announcement application. This "disconnects" the user from the rest of the conference so they can interact with the application. Once the interaction is done, and announcement application uses the CPCP to "reconnect" the user to the conference.

5.11 Floor Control

Floor control is similar to a conference announcement application. Within this framework, floor control is managed by an application (possibly one that is not a participant) that uses the CPCP to enforce the resulting floor control decisions.

[[Need more work here]]

5.12 Camera and Video Controls

> OPEN ISSUE: Originally, I was just going to say that this is outside the scope of conferencing. But, it does impact conferencing. Effectively, camera control is treated like a media stream. The mixer would combine the various requests across participants and direct them to the appropriate

device. How does that work though? In a video conference with 4 participants, the camera control needs to identify the specific user whose camera is to be controlled. That is something unique to conferencing.

6 Physical Realization

In this section, we present several physical instantiations of these components, to show how these basic functions can be combined to solve a variety of problems.

6.1 Centralized Server

In the most simplistic realization of this framework, there is a single physical server in the network which implements the focus, the conference policy server, and the mixers. This is the classic "one box" solution, shown in Figure 4.

6.2 Endpoint Server

Another important model is that of a locally-mixed ad-hoc conference. In this scenario, two users (A and B) are in a regular point-to-point call. One of the participants (A) decides to conference in a third participant, C. To do this, A begins acting as a focus. Its existing dialog with B becomes the first dialog attached to the focus. A would re-INVITE B on that dialog, changing its Contact URI to a new value which identifies the focus. In essence, A "mutates" from a single-user UA to a focus plus a single user UA, and in the process of such a mutation, its URI changes. Then, the focus makes an outbound INVITE to C. When C accepts, it mixes the media from B and C together, redistributing the results. The mixed media is also played locally. Figure 5 shows a diagram of this transition.

It is important to note that the external interfaces in this model, between A and B, and between B and C, are exactly the same to those that would be used in a centralized server model. B could also include a conference policy server and conference notification service, allowing the participants to have access to them if they so desired. Just because the focus is co-resident with a participant does not mean any aspect of the behaviors and external interfaces will change.

6.3 Media Server Component

In this model, shown in Figure 6, each conference involves two

```
           Conference Server
  ...................................
  .                                 .
  .                 +------------+  .
  .                 | Conference |  .
  .                 |Notification|  .
  .                 |   Server   |  .
  .                 +------------+  .
  . +----------+                    .
  . |Conference|            +-----+ .
  . |  Policy  | +-------+ +-----+| .
  . |  Server  | | Focus | |Mixer|+ .
  . +----------+ +-------+ +-----+  .
  ................//.\.....***.......
                //    \ ***  *
              //     ***      * RTP
      SIP   //    ***  \      *
          //   ***      \SIP   *
        //  *** RTP      \     *
       /  **              \     *
+-----------+         +-----------+
|Participant|         |Participant|
+-----------+         +-----------+
```

Figure 4: Centralized server architecture

centralized servers. One of these servers, referred to as the "application server" owns and manages the membership and media policies, and maintains a dialog with each participant. As a result, it represents the focus seen by all participants in a conference. However, this server doesn't provide any media support. To perform the actual media mixing function, it makes use of a second server,

```
         B                              B
      +------+                       +------+
      |      |                       |      |
      |  UA  |                       |  UA  |
      |      |                       |      |
      +------+                       +------+
        |  .                           |  .
        |  .                           |  .
        |  .                           |  .
        |  .         Transition        |  .
        |  .        ------------>      |  .
     SIP|  .RTP                     SIP|  .RTP
        |  .                           |  .
        |  .                           |  .
        |  .                           |  .
        |  .                           |  .
        |  .                       +----------+
      +------+                     | +------+ |   SIP    +------+
      |      |                     | |Focus | |----------|      |
      |  UA  |                     | |C.Pol.| |          |  UA  |
      |      |                     | |Mixers| |..........|      |
      +------+                     | |      | |   RTP    +------+
                                   | +------+ |
         A                         |     +    |             C
                                   |     + <..|.......
                                   |     +    |      .
                                   | +------+ |      .
                                   | |Parti-| |      .
                                   | |cipant| |      .
                                   | |      | |      .
                                   | +------+ |      .
                                   +----------+      .
                                        A            .
                                                     .

                                                   Internal
                                                   Interface
```

Figure 5: Transition from two-party call to conference

called the "mixing server". This server includes a focus, and a conference policy server, but has no conference notification service. It has a default membership policy, which accepts all invitations

```
                   +------------+             +------------+
                   | App  Server|  SIP        |Conf. Cmpnt.|
                   |            |-------------|            |
                   |   Focus    | Conf. Proto |   Focus    |
                   |   C.Pol    |-------------|   C.Pol    |
                   |            | Media Proto |   Mixers   |
                   |Notification|-------------|            |
                   |            |             |            |
                   +------------+             +------------+
                       |      \                    .. .
                       |       \\            RTP...   .
                       |         \\           ..      .
                       |     SIP   \\      ...        .
                   SIP |             \\ ...           .RTP
                       |              ..\             .
                       |           ...   \\           .
                       |        ...        \\         .
                       |      ..             \\       .
                       |   ...                 \\     .
                       | ..                      \    .
                  +-----------+              +-----------+
                  |Participant|              |Participant|
                  +-----------+              +-----------+
```

Figure 6: Media server component model

server uses third party call control to connect the media streams of each user to the mixing server, as needed. If the focus in the application server receives a conference policy control command from a client, it delegates that to the media server by making the same media policy control command to it.

This model allows for the mixing server to be used as a resource for a variety of different conferencing applications. This is because it

is unaware of any conference or media policies; it is merely a "slave" to the top-level server, doing whatever it asks. This is consistent with the SIP Application Server Component Model [17].

6.4 Distributed Mixing

In a distributed mixed conference, there is still a centralized server which implements the focus, conference policy server, and media policy server. However, there are no centralized mixers. Rather, there are mixers in each endpoint, along with a conference policy server. The focus distributes the media by using third party call control [18] to move a media stream between each participant and each other participant. As a result, if there are N participants in the conference, there will be a single dialog between each participant and the focus, but the session description associated with that dialog will be constructed to allow media to be distributed amongst the participants. This is shown in Figure 7.

There are several ways in which the media can be distributed to each participant for mixing. In a multi-unicast model, each participant sends a copy of its media to each other participant. In this case, the session description manages N-1 media streams. In a multicast model, each participant joins a common multicast group, and each participant sends a single copy of its media stream to that group. The underlying multicast infrastructure then distributes the media, so that each participant gets a copy. In a single-source multicast model (SSM), each participant sends its media stream to a central point, using unicast. The central point then redistributes the media to all participants using multicast. The focus is responsible for selecting the modality of media distribution, and for handling any hybrids that would be necessitated from clients with mixed capabilities.

When a new participant joins or is added, the focus will perform the necessary third party call control to distribute the media from the new participant to all the other participants, and vice-a-versa.

The central conference server also includes a conference policy server. Of course, the central conference server cannot implement any of the media policies directly. Rather, it would delegate the implementation to the conference policy servers co-resident with a participant. As an example, if a participant decides to switch the overall conference mode from "voice activated" to "continuous presence", they would communicate with the central conference policy server. The conference policy server, in turn, would communicate with the conference policy servers co-resident with each participant, using the same conference policy control protocol, and instruct them

to use "continuous presence".

This model requires additional functionality in user agents, which may or may not be present. The participants, therefore, must be able to advertise this capability to the focus.

6.5 Cascaded Mixers

In very large conferences, it may not be possible to have a single mixer that can handle all of the media. A solution to this is to use cascaded mixers. In this architecture, there is a centralized focus, but the mixing function is implemented by a multiplicity of mixers, scattered throughout the network. Each participant is connected to one, and only one of the mixers. The focus uses some kind of control protocol to connect the mixers together, so that all of the participants can hear each other.

This architecture is shown in Figure 8.

7 Security Considerations

Conferences frequently require security features in order to properly operate. The conference policy may dictate that only certain participants can join, or that certain participants can create new policies. Generally speaking, conference applications are very concerned about authorization decisions. Mechanisms for establishing and enforcing such authorization rules is a central concept throughout this document.

Of course, authorization rules require authentication. Normal SIP authentication mechanisms should suffice for the conference authorization mechanisms described here.

Privacy is an important aspect of conferencing. Users may wish to join a conference without anyone knowing that they have joined, in order to silently listen in. In other applications, a participant may wish just to hide their identity from other participants, but otherwise let them know of their presence. These functions need to be provided by the conferencing system.

8 Contributors

This document is the result of discussions amongst the conferencing design team. The members of this team include:

Alan Johnston
Brian Rosen

```
                             +---------+
                             |Partcpnt |
                 media       |         |      media
              ...............|         |..................
              .              |  Mixers |                 .
              .              |C.Pol.Srv|                 .
              .              +---------+                 .
              .                   |                      .
              .                   |                      .
              .                   |                      .
              .            dialog |                      .
              .                   |                      .
              .                   |                      .
              .                   |                      .
              .              +---------+                 .
              .              |Cnf.Srvr.|                 .
             .               |         |                 .
             .               |  Focus  |                 .
             .               |C.Pol.Srv|                 .
             .             / |         |  \              .
             .            /  +---------+   \             .
             .           /                  \            .
             .          /                    \           .
             .         /               dialog \          .
             .        /                        \         .
             .       /dialog                    \        .
             .      /                            \       .
             .     /                              \      .
             .    /                                \     .
             .                                           .
           +---------+                           +---------+
           |Partcpnt |                           |Partcpnt |
           |         |                           |         |
           |         | ......................... |         |
           |  Mixers |                           |  Mixers |
           |C.Pol.Srv|          media            |C.Pol.Srv|
           +---------+                           +---------+
```

Figure 7: Dialog and media streams in a distributed mixed conference

Rohan Mahy
Henning Schulzrinne
Orit Levin
Roni Even
Tom Taylor
Petri Koskelainen
Nermeen Ismail
Andy Zmolek
Joerg Ott
Dan Petrie

9 Changes since draft-rosenberg-sipping-conferencing-framework-01

- o Clarified that the conference notification service uses a single package with some kind of filtering to select whether you get the focus or policy state

10 Changes since draft-rosenberg-sipping-conferencing-framework-00

- o Rework of terminology.
- o More details on moderating policy changes.
- o Rework of the overview, and in particular, a shift of focus from basic/complex conferences (a term which has been removed) to conference aware/unaware participants.
- o Removal of explicit reference to megaco for controlling a mixer.
- o Discussion of a lot more conferencing operations.
- o New sidebar mechanism.

11 Authors Addresses

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

12 Normative References

13 Informative References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: session initiation protocol," RFC 3261, Internet Engineering Task Force, June 2002.

[2] 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.

[3] O. Levin and R. K. Even, "High level requirements for tightly coupled SIP conferencing," internet draft, Internet Engineering Task Force, Mar. 2003. Work in progress.

[4] A. B. Roach, "Session initiation protocol (sip)-specific event notification," RFC 3265, Internet Engineering Task Force, June 2002.

[5] B. Campbell and J. R. et. al., "Instant message sessions in SIMPLE," internet draft, Internet Engineering Task Force, Mar. 2003. Work in progress.

[6] A. R. Johnston and O. Levin, "Session initiation protocol call control - conferencing for user agents," internet draft, Internet Engineering Task Force, Feb. 2003. Work in progress.

[7] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource identifiers (URI): generic syntax," RFC 2396, Internet Engineering Task Force, Aug. 1998.

[8] H. Schulzrinne, J. Rosenberg, and P. Kyzivat, "Caller preferences and callee capabilities for the session initiation protocol (SIP)," internet draft, Internet Engineering Task Force, Mar. 2003. Work in progress.

[9] R. Mahy and D. Petrie, "The session inititation protocol (SIP) 'join' header," internet draft, Internet Engineering Task Force, Mar. 2003. Work in progress.

[10] J. Rosenberg and H. Schulzrinne, "An INVITE inititiated dialog event package for the session initiation protocol (SIP," internet draft, Internet Engineering Task Force, Mar. 2003. Work in progress.

[11] R. Sparks, "The SIP refer method," internet draft, Internet Engineering Task Force, Dec. 2002. Work in progress.

```
                           +---------+
   +-----------------------|         |------------------------+
   |   ++++++++++++++++++++|         |++++++++++++++++++      |
   |   +            +------|  Focus  |---------+       +      |
   |   +            |      |         |         |       +      |
   |   +            |    +-|         |--+      |       +      |
   |   +            |    | +---------+  |      |       +      |
   |   +            |    |      +       |      |       +      |
   |   +            |    |      +       |      |       +      |
   |   +            |    |      +       |      |       +      |
   |   +            |    | +---------+  |      |       +      |
   |   +            |    | |         |  |      |       +      |
   |   +            |    | | Mixer 2 |  |      |       +      |
   |   +            |    | |         |  |      |       +      |
   |   +            |    | +---------+  |      |       +      |
   |   +            |    |...   .  .... |      |       +      |
   |   +           .|....|      .      .|....  |       +      |
   |   +     ...... |    |      .       |    ..|...    +      |
   |   +  ...       |    |      .       |      |   ....+      |
   | +---------+    |    | +---------+  |      |  +---------+ |
   | |         |    |    | |         |  |      |  |         | |
   | | Mixer 2 |    |    | | Mixer 3 |  |      |  | Mixer 4 | |
   | |         |    |    | |         |  |      |  |         | |
   | +---------+    |    | +---------+  |      |  +---------+ |
   |    .    .      |    |      .  .    |      |     .   .    |
   |   .      .     |    |    ..   .    |      |   ..    .    |
   |  .       .     |    |   .      .   |      |  .       .   |
  +---------+  .    |  +---------+  .   |    +---------+  .   |
  | Prtcpnt |   .   |  | Prtcpnt |   .  |    | Prtcpnt |  .   |
  |    1    |    .  |  |    1    |   .  |    |    1    |  .   |
  +---------+    .  |  +---------+    . |    +---------+   .  |
                  . |                 . |                  .  |
           +---------+         +---------+           +---------+
           | Prtcpnt |         | Prtcpnt |           | Prtcpnt |
           |    1    |         |    1    |           |    1    |
           +---------+         +---------+           +---------+


     -------  SIP Dialog
     .......  Media Flow
     +++++++  Control Protocol
```

Figure 8: Cascaded Mixers

[12] "Session initiation protocol (SIP) extension for instant messaging," RFC 3428, Internet Engineering Task Force, Dec. 2002.

[13] T. Moran and S. Addagatla, "Architecture for event notification filters," internet draft, Internet Engineering Task Force, Oct. 2002. Work in progress.

[14] J. Rosenberg, "A presence event package for the session initiation protocol (SIP)," internet draft, Internet Engineering Task Force, Jan. 2003. Work in progress.

[15] J. Rosenberg, "A watcher information event template-package for the session initiation protocol (SIP)," internet draft, Internet Engineering Task Force, Jan. 2003. Work in progress.

[16] J. Rosenberg and H. Schulzrinne, "An offer/answer model with session description protocol (SDP)," RFC 3264, Internet Engineering Task Force, June 2002.

[17] J. Rosenberg, P. Mataga, and H. Schulzrinne, "An application server component architecture for SIP," internet draft, Internet Engineering Task Force, Mar. 2001. Work in progress.

[18] J. Rosenberg, J. L. Peterson, H. Schulzrinne, and G. Camarillo, "Best current practices for third party call control in the session initiation protocol," internet draft, Internet Engineering Task Force, Mar. 2003. Work in progress.

Intellectual Property Statement

Full Copyright Statement