Network Working Group
Request for Comments: 4353
Category: Informational

J. Rosenberg
Cisco Systems

February 2006

A Framework for Conferencing with the Session Initiation Protocol (SIP)

Status of This Memo

Copyright Notice

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 the other participants, 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 presents the overall framework for tightly coupled SIP conferencing, referred to simply as "conferencing" from this point forward. This framework presents a general architectural model for these conferences and presents terminology used to discuss such conferences. It also discusses the ways in which SIP itself is involved in conferencing. The aim of the framework is to meet the general requirements for conferencing that are outlined in [3]. This specification alludes to non-SIP-specific mechanisms for achieving several conferencing functions. Those mechanisms are outside the scope of this specification.

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, that 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 implement non-SIP-specific mechanisms for additional functionality.

Conference State: The state of the conference includes the state of the focus, the set of participants connected to the conference, and the state of their respective dialogs.

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.

Conference Policy Server: A conference policy server is a logical function that can store and manipulate the conference policy. This logical function is not specific to SIP, and may not physically exist. It refers to the component that interfaces a protocol to the conference policy.

Conference Policy: The complete set of rules governing a particular conference.

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 3550, 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 that are linked such that the user agent that 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. A conference-aware participant can use the conference notification service or additional non-SIP-specific mechanisms for additional functionality.

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

Mass Invitation: An attempt to add a large number of users into a conference.

Mass Ejection: An attempt to remove a large number of users from a 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.

3. Overview of Conferencing Architecture

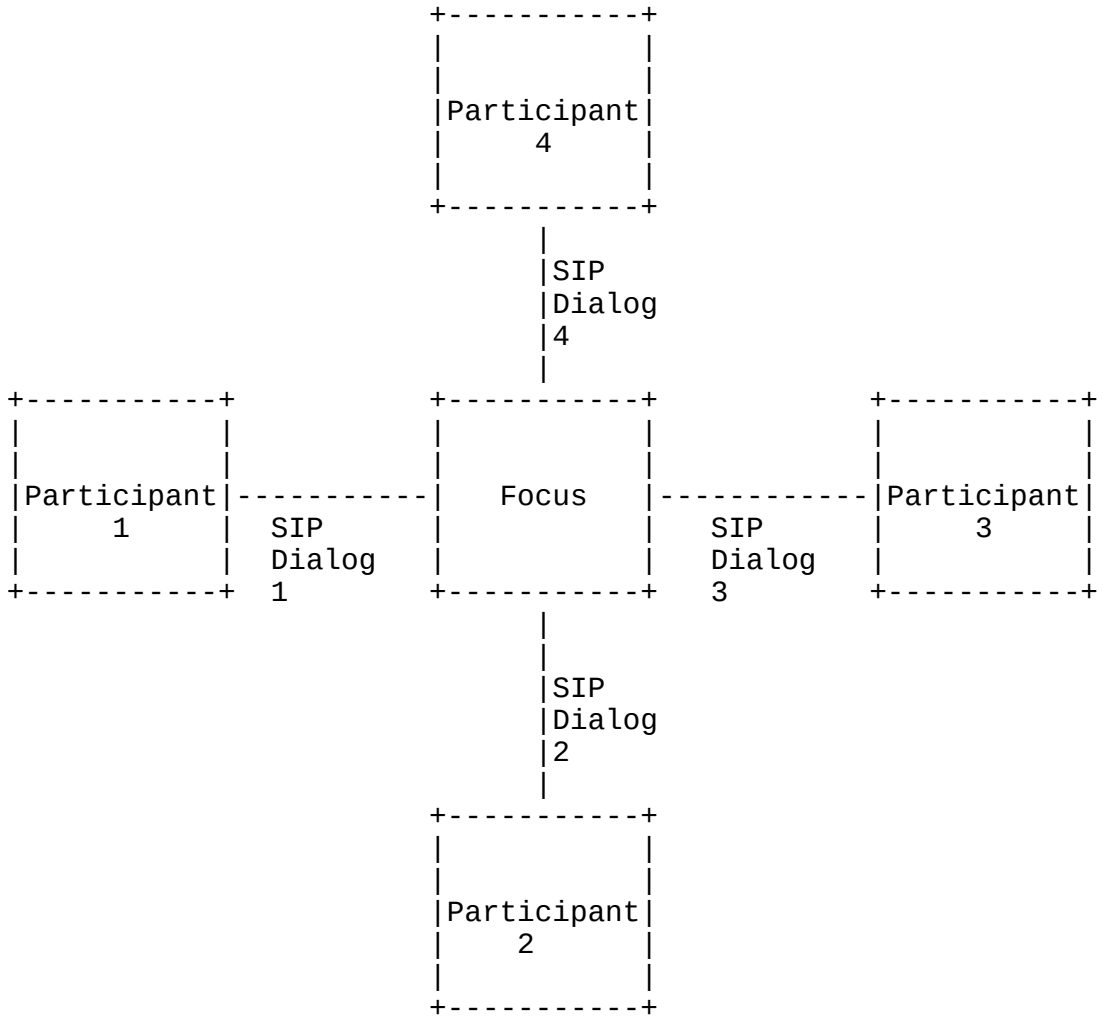

Figure 1

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, as shown in Figure 1.

The focus is responsible for making sure that the media streams that 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.

The focus has access to the conference policy, an instance of which exists for each conference. Effectively, the conference policy can be thought of as a database that 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 those changes that affect the conference state will require a notification to be sent to subscribers using the conference notification service.

The conference is represented by a URI that 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 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, the UA appears like any other UA. The focus, of course, knows that it’s 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 [6]. However, if the participant wishes to explicitly control aspects of the conference using functional signaling protocols, the participant must be conference-aware.

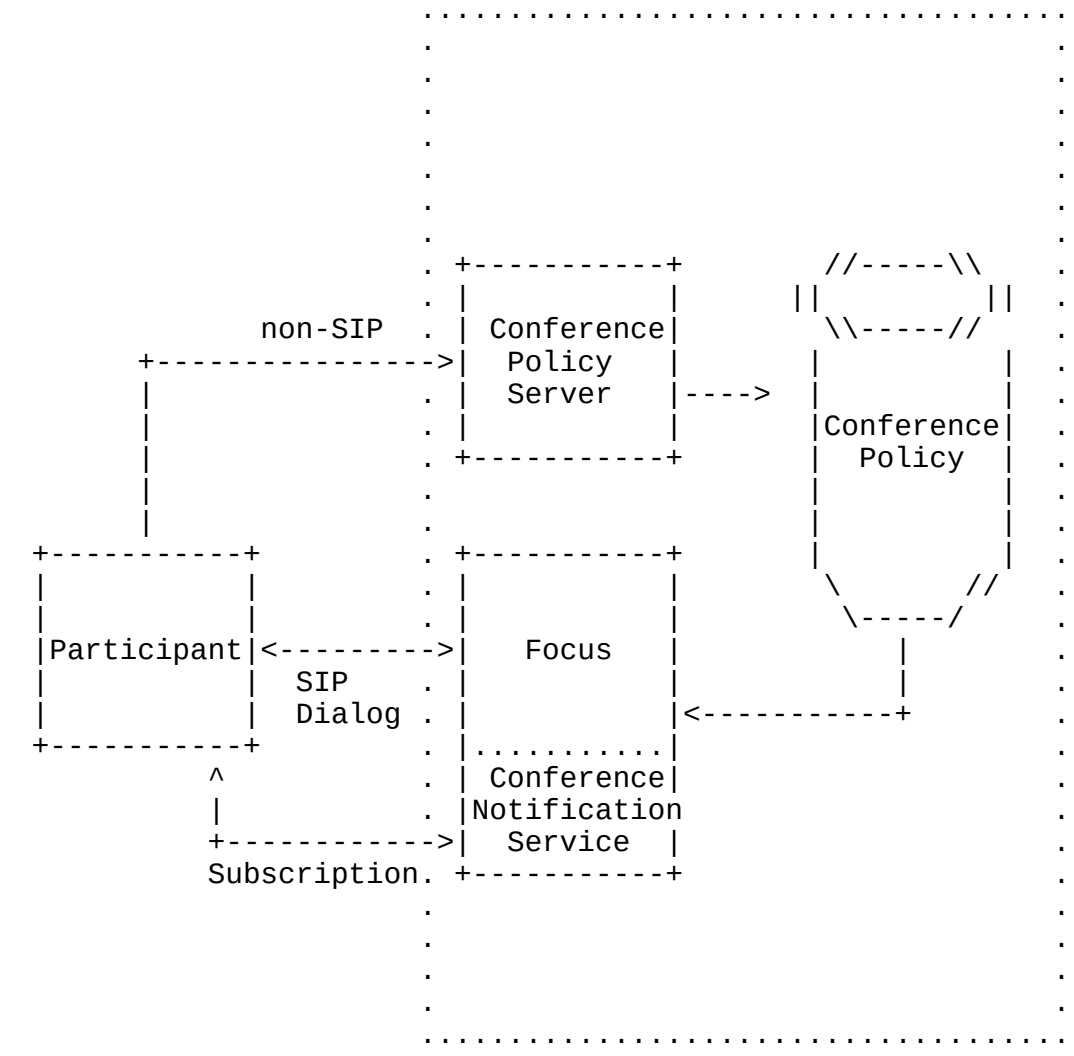
non-SIP
Conference Policy Server
Conference Policy
Participant
SIP Dialog
Focus
Conference Notification Service
Subscription
Conference Functions

Figure 2

A conference-aware participant is one that has access to advanced functionality through additional protocol interfaces, which may include access to the conference policy through non-SIP-specific mechanisms. 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 [7]. 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 and the media.

The participant can communicate with the conference policy server using some kind of non-SIP-specific mechanism by which 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 between the conference policy server and the conference policy are non-SIP-specific. For the purposes of SIP-based conferencing, they serve as logical roles involved in a conference, as opposed to representing a physical decomposition. The separation of these functions is documented here to encourage clarity in the requirements. This approach provides individual SIP implementations the flexibility to compose a conferencing system in a scalable and robust manner without requiring the complete development of these interfaces.

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 that 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 that 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 [8]. 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. 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 also be obtained from some non-SIP mechanism.

To determine that a SIP URI does represent a focus, standard techniques for URI capability discovery can be used. Specifically, the callee capabilities specification [9] provides the "isfocus" feature tag to indicate that the UA is acting as focus in this dialog. Callee capability 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.

Other functions in a conference may be represented by URIs. If the conference policy is exposed through a web application, it is identified by an HTTP URI. 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 conference policy. The 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 could then allow this new participant to join, and the focus could then accept the INVITE (or unpark it from the music-on-hold server). The interpretation of policy by the focus is, itself, a matter of local policy, and not subject to standardization.

When it is necessary to remove a SIP participant (with a confirmed dialog) from a conference, the focus would send a BYE to that participant to remove the participant. This is often referred to as "ejecting" a user from the conference, and is called "mass ejection" when done for many users. Similarly, if it is necessary to add a new SIP participant to a conference, the focus would send an INVITE request to that participant. When done for a large number of users, this is called mass invitation. Finally, if it is necessary to change the properties of the media of a session (for example to remove video) for a SIP participant, the focus can update the session description for that participant by sending a re-INVITE or UPDATE [15] request with a new offer to that participant.

In many cases, the signaling actions performed by the focus, such as ejection or addition of a participant, will change the media composition of the conference. To affect these changes, the focus interacts with the mixer. Through that interaction, it makes sure that all valid participants received a copy of the media streams, and that each participant sends media to an IP address and port on the mixer that cause it to be appropriately mixed with the other media in the conference. The means by which the focus interacts with the mixer are outside the scope of this specification.

4.2. Conference Policy Server

The conference policy server is a logical component of the system. It represents the interface between clients and the conference policy that governs the operation of the conference. Clients communicate with the conference policy server using a non-SIP-specific mechanism.

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, either directly or indirectly. 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. If the focus is indirectly controlling a mixer, it delegates the mixing to the participants, each of which has its 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.

The state of the conference includes the participants connected to the focus, and also 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.

If a participant is anonymous, the conference notification service will either withhold the identity of a new participant from other conference participants, or will neglect to inform other conference participants about the presence of the anonymous participant. The choice of approach depends on the level of anonymity provided to the anonymous participant.

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 that 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 non-SIP-specific standard or proprietary protocols.

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 [7].

As well as providing an overview of the common conferencing operations, each of the subsections in this section of the document provides a description of the SIP mechanism for supporting the operation. Non-SIP mechanisms are also possible, but not discussed here.

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 that 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 a random URI with sufficiently low probabilities of collision).

When conference policy is created, it is established with default rules that are implementation-dependent. If the creator of the conference wishes to change those rules, they would do so using a non-SIP mechanism.

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 that will 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 or UPDATE 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 [10]. The UA that 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 or UPDATE.

5.2. Adding Participants

There are many mechanisms for adding participants to a conference. In all cases, participant additions can be first party (a user adds themself) or third party (a user adds another user).

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 [11]), the UA can join the conference by using the Join header to join the dialog.

Third party additions with SIP are done using REFER [12]. 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.

5.3. Removing Participants

As with additions, there are several mechanisms for departures. Removals can also be first person or third person.

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. The focus can also remove a participant from the conference by sending it a BYE. In either case, the focus interacts with the mixer to make sure that the departed participant ceases receiving conference media, and that media from that participant are no longer mixed into the conference.

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

5.4. 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 policy associated with it is destroyed. Any attempts to read or write the policy 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.

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 or a specific user leaves, when that user does leave (using a SIP BYE request), the conference is destroyed.

5.5. 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 in many ways.

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.

5.6. 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 the stream is no longer part of the conference, only that the participant is not involved in it.

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 [13]. This will trigger the focus to generate its own re-INVITEs.

5.7. Conference Announcements and Recordings

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

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

- Allowing a user to request a roll call, so they can hear who else is in the conference
- Allowing a user to press some keys on their keypad to record the conference
- Allowing a user to press some keys on their keypad to be connected with a human operator
- Allowing a user to press some keys on their keypad to mute or unmute their line

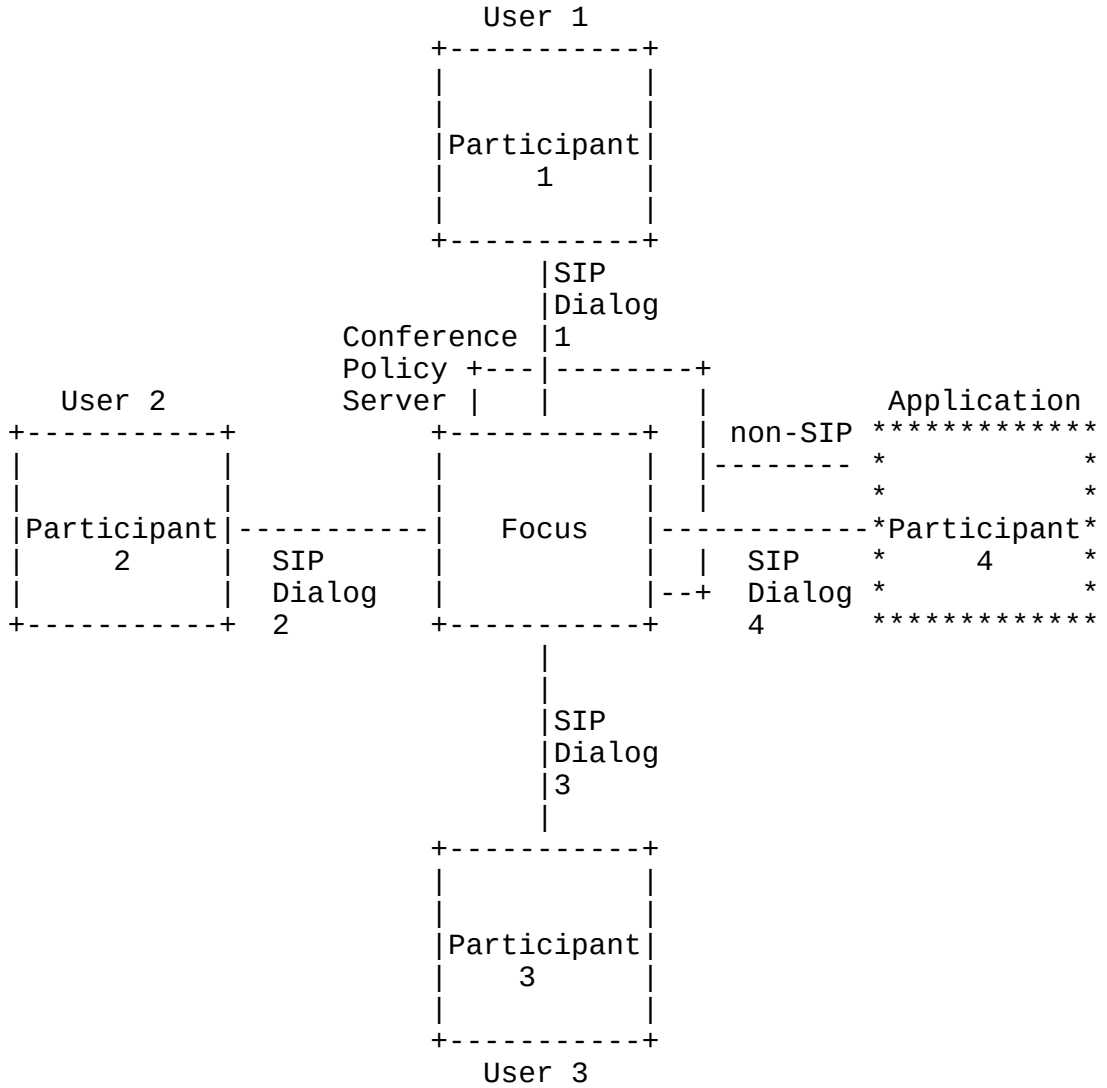

Figure 3

In this framework, these capabilities are modeled as an application that 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 configuring the conference 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. To do this, it can use the application interaction framework [6]. This allows it to collect user input, possibly through keypad stimulus, and to take actions.

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.

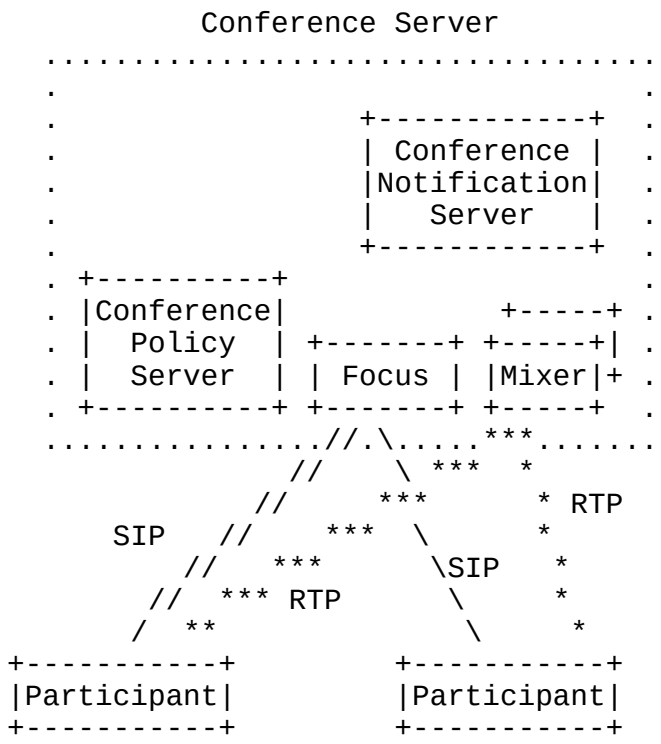

Figure 4

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 that 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.

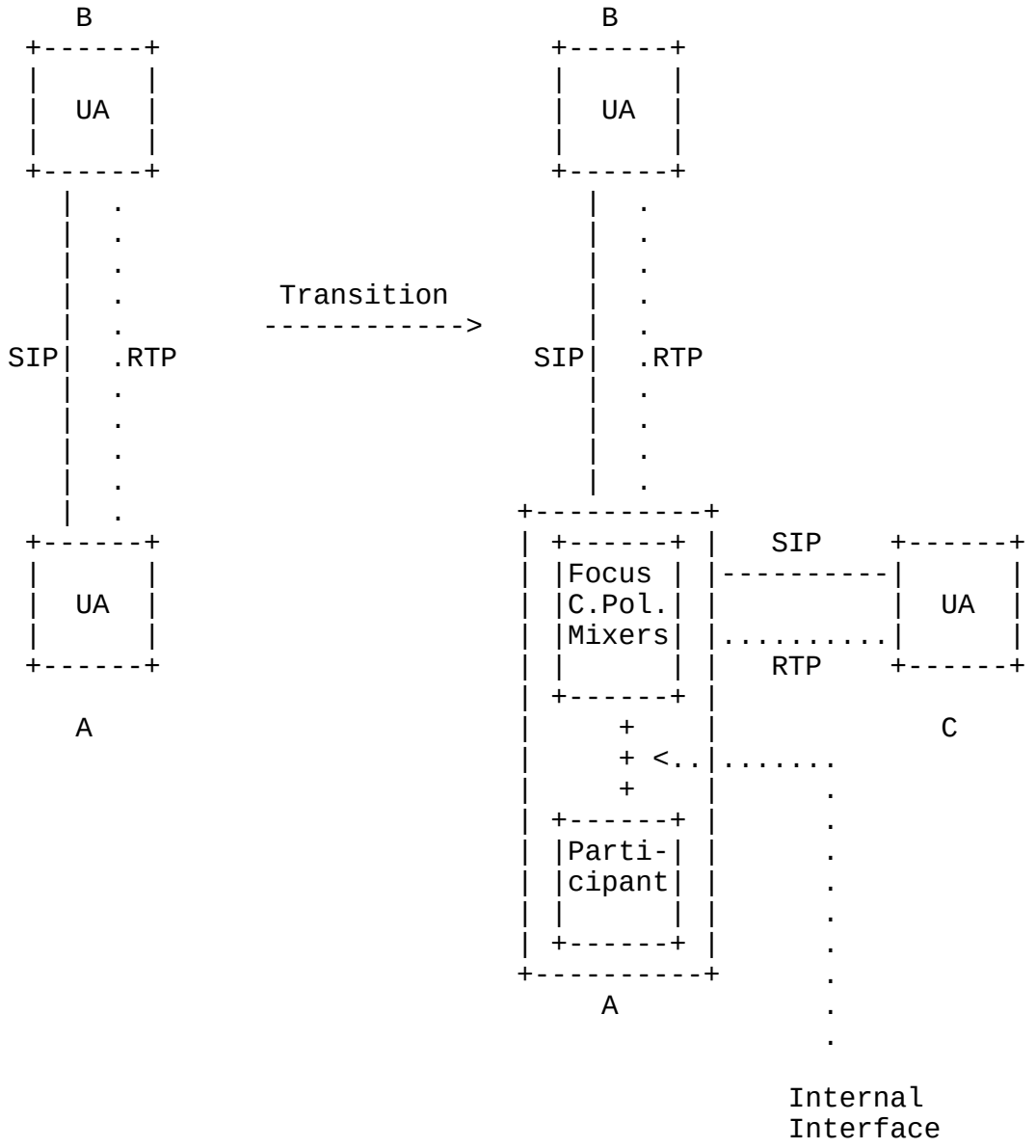

Figure 5

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. User A could also implement a conference policy and a 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

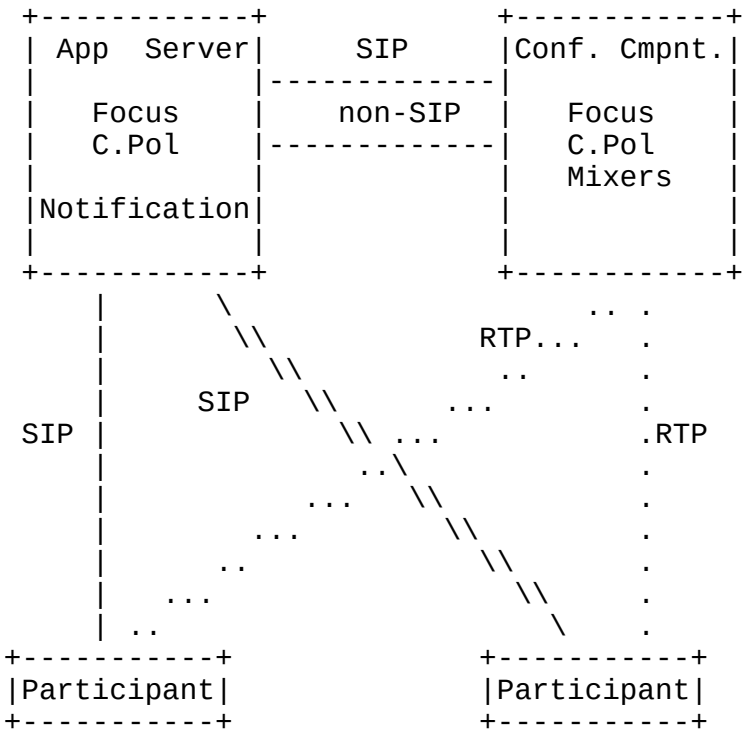

Figure 6

In this model, shown in Figure 6, each conference involves two 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, called the "mixing server". This server includes a focus, and implements a conference policy, but has no conference notification service. Its conference policy tells it to accept all invitations from the top-level focus. The focus in the application 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 conference policy; it is merely a "slave" to the top-level server, doing whatever it asks.

6.4. Distributed Mixing

In a distributed mixed conference, there is still a centralized server that 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 [14] 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.

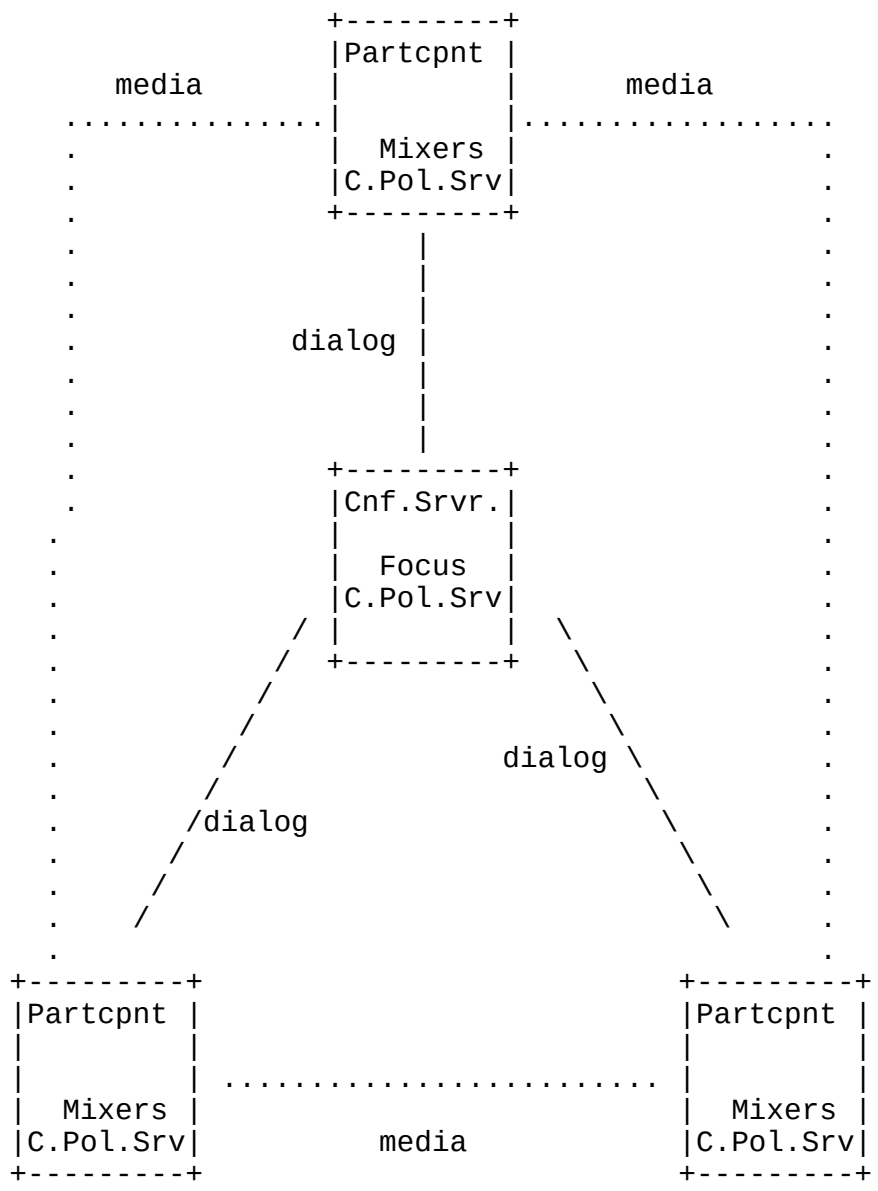

Figure 7

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 versa.

The central conference server also exposes an interface to the conference policy. Of course, the central conference server cannot implement any of the media operations or policies directly. Rather, it would delegate the implementation to each 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 that are co-resident with each participant, using some non-SIP-specific mechanism, 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.

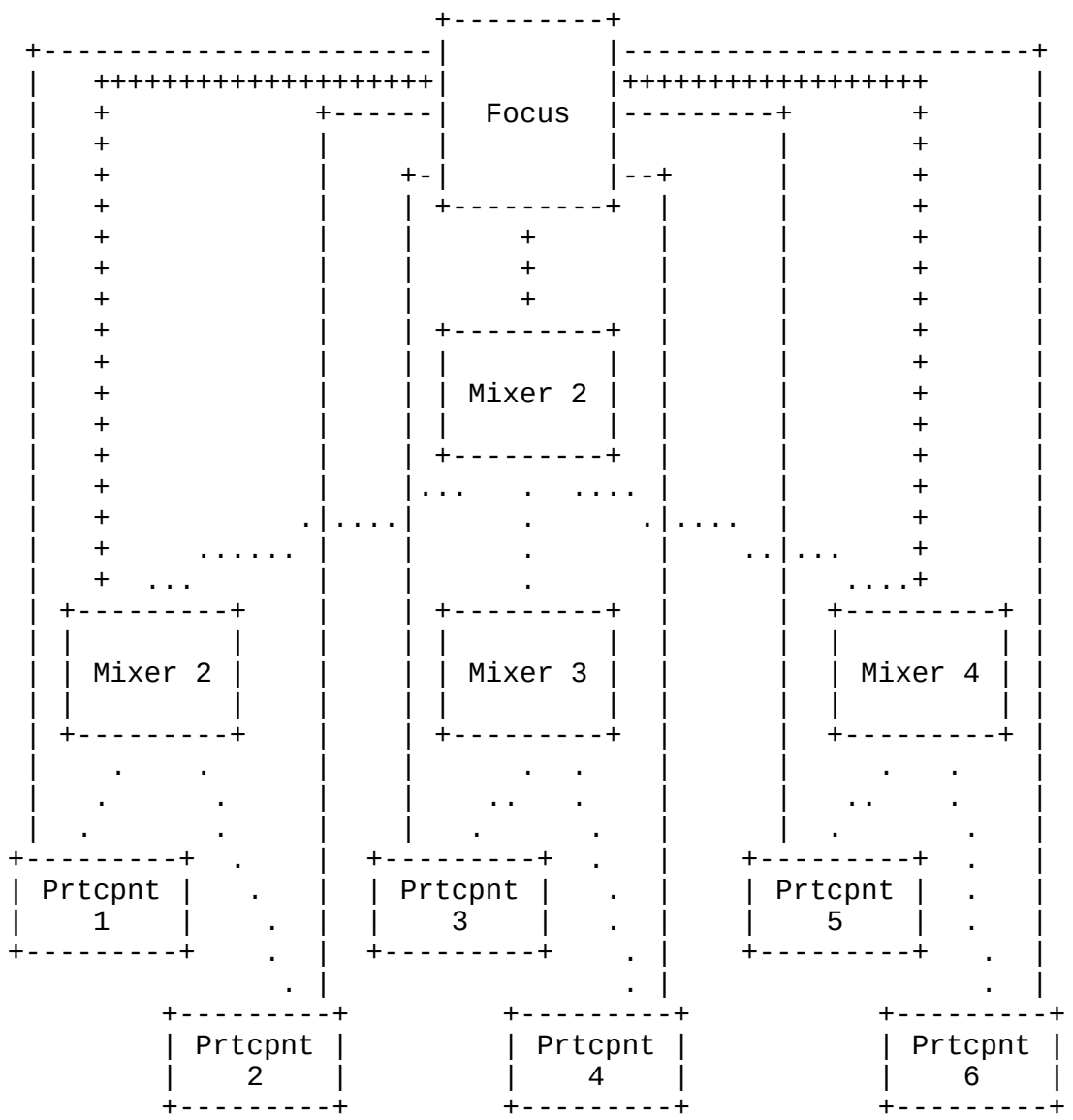
Focus
Mixer 2
Mixer 2
Mixer 3
Mixer 4
Prtcpnt 1
Prtcpnt 2
Prtcpnt 3
Prtcpnt 4
Prtcpnt 5
Prtcpnt 6
------- SIP Dialog
..... Media Flow
+++++++ Control Protocol

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

9. Acknowledgements

The authors would like to thank Mary Barnes, Chris Boulton and Rohan Mahy for their comments. Thanks to Allison Mankin for her comments and support of this work.

10. Informative References

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

[2] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[3] Levin, O. and R. Even, "High-Level Requirements for Tightly Coupled SIP Conferencing", RFC 4245, November 2005.

[4] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[5] Campbell, B., "The Message Session Relay Protocol", Work In Progress, October 2004.

[6] Rosenberg, J., "A Framework for Application Interaction in the Session Initiation Protocol (SIP)", Work In Progress, February 2005.

[7] Johnston, A. and O. Levin, "Session Initiation Protocol (SIP) Call Control - Conferencing for User Agents", Work in Progress, February 2005.

[8] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[9] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, August 2004.

[10] Mahy, R. and D. Petrie, "The Session Initiation Protocol (SIP) "Join" Header", RFC 3911, October 2004.

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

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

[13] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[14] Rosenberg, J., Peterson, J., Schulzrinne, H., and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)", BCP 85, RFC 3725, April 2004.

[15] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, October 2002.

Author's Address

Jonathan Rosenberg
Cisco Systems
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@cisco.com
URI: http://www.jdrosen.net

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).