A SIP-based Conference Control Framework

Petri Koskelainen
Nokia Research Center
Visiokatu 1
33720 Tampere, Finland
petri.koskelainen@nokia.com

Henning Schulzrinne and Xiaotao Wu
Dept. of Computer Science
Columbia University
New York, NY 10027, USA
hgs@cs.columbia.edu,xiaotaow@cs.columbia.edu

ABSTRACT

Conference control has been an area of intensive research over the years but widely accepted robust and scalable solutions and standards are still lacking. The main conference control components are conference management and floor (resource) control. We identify the requirements for conference control and propose a component-based scalable conference control framework employing the Session Initiation Protocol (SIP) and the Simple Object Access Protocol (SOAP). The framework assumes a single control point, but our architecture can scale to large groups by distributing media via a tree-shaped hierarchy of conference servers.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Computer conferencing, teleconferencing, and video conferencing

General Terms

Conference control and management

Keywords

Conference control; floor control; multimedia conferencing; packet audio; packet video; SIP; SOAP

1. INTRODUCTION

Conference control is one of the core building blocks of the Internet multimedia conferencing architecture. It is required not just by traditional voice and video conferencing, but also for multiparty network gaming. There currently is no application-independent conference control mechanism, so that each application is forced to re-invent solutions to subsets of the problem. Below, we attempt to take a first step towards defining a common, interoperable framework for creating and controlling multi-user centralized conferences. The paper is organized as follows. Section 2 describes related work. Section 3 gives a problem statement. In Section 4, we present requirements for conference control. Section 5 describes the framework and abstract operations needed for the framework. In section 6, we finally present a mapping from the requirements onto real protocols and message formats. Section 7 gives brief examples. In section 8, we will describe how the framework can be utilized for other services, such as presence authorization and allow/block lists now being discussed in the IETF.

NOSSDAV'02 May 12-14, 2002, Miami Beach, Florida, USA.

2. RELATED WORK

Problems related to conference control have been studied extensively over the years [1, 35, 7, 23, 43, 36, 4, 14, 27, 3, 29, 5, 42, 30, 24, 6, 13, 21, 33, 41]. However, most of the earlier work discusses only the floor control aspects of conference control. Furthermore, some papers focus on multicast conferences [25, 40, 37]; in the absence of a widely available Internet multicast service, we will focus on the currently more common centralized conferencing architecture.

Standardization efforts have met with limited success. ITU-T developed conference control protocols as a part of the H.323 series of recommendations [16] but they had several problems. For example, they have limited scalability due to the inefficient T.124 [17] database replication protocol and they lack some key features. Data and audiovisual conferences use different concepts, adding complexity [41].

In addition, they are based on binary ASN.1 formats instead of the more Internet-friendly text protocols used in current Internet protocols such as SIP and HTTP. Hence, deployed H.323 systems rarely use the protocol's conference control features.

The IETF's Multi-Party Multimedia (MMUSIC) working group has also considered this topic [1]. However, its main target was lightweight conference management for multicast instead of tightly controlled models. In the year 2000, IETF MMUSIC WG gave up and removed conference control from the WG charter. Work on enumerating conference architectures that can be established via the Session Initiation Protocol (SIP) has begun [33]. The draft defines several conferencing scenarios such as end-system based conferences, dial-in, dial-out and large-scale multicast scenarios. Of these, the dial-in and dial-out scenarios are the most interesting. SIP mechanisms can initiate such conferences, allow users to join and leave conferences and make it possible for a participant to ask a third party to join. However, SIP by itself does not offer configurable conference policies, participant access lists, floor control, or user privilege levels. Despite the problems, the dial-in and the dial-out models are sufficient for some scenarios, easy to implement and do not require any

protocol changes.

3. PROBLEM STATEMENT

We define conference as a set of members that exchange one or more media streams or share some other state, such as an application. Conferencing includes two main components. First, participants must be able to join and leave a conference. Signaling protocols can generally support these operations if conferences are addressed like regular endpoints. The second component, conference control, can be considered as consisting of three parts: creating, modifying and deleting conferences as a whole, user management (adding and deleting conference participants and modifying their conference privileges), and resource contention management, also known as floor control. Many different kinds of conference control scenarios are in use. The simplest one is an open voice conference without any control. The member identities or group size are not known to the participants and anyone can speak or join at any time. Controlling the right to speak is the most typical example of floor control. In general, floor control provides a conflict-free access to shared resources. In a controlled voice conference with one member acting as a chairperson, requests to speak are sent to the chairperson who grants or denies the request. Participants may differ in their claim on the shared resource, based on a *policy*. In a panel session, for example, the panel members may have the right to speak at any time, while the audience members can listen unless called upon by the moderator. Access rights and user groups are part of user management.

While conference control is typically associated with audio and video conferences, shared network games have similar requirements: access to the game needs to be controlled and there are usually a number of resources that can only be used by a limited number of players at the same time.

Conference control is a broad topic with no agreed-upon definition [1, 13]. For example, Borman et al. [1] describe an abstract SCCP (Simple Conference Control Protocol) framework that defines the necessary SCCP tasks: conference management, application session management, and floor control. In this paper, we will implement the same functionality and follow the same definition with minor exceptions. We do not consider operations such as joining a conference or joining an application session inside a conference part of conference control since they are often indistinguishable from regular one-to-one calls. We are also omitting joining and splitting conferences. The effect of these operations can be approximated by creating conferences and then transferring members from one to the other or having members leave one and join another. Also, we do not preclude that a conference can itself be part of another conference.

4. REQUIREMENTS

4.1 General Requirements

We define a conference control framework as the set of protocols that implement conference control and the architecture of network elements that implement them. A conference control framework needs to balance being powerful enough for most practical scenarios, yet avoid being too complex to be widely accepted and implemented on devices with limited computational resources and user interfaces.

Like any distributed system, a conference control framework should be scalable, easy to extend, generic, reliable and secure. The scalability requirement means that the conference control framework must support reasonably large, possibly geographically distributed, conferences. Moreover, it should be modular so that new components can be easily added or existing components can be changed. For example, it must be possible to use different floor control protocols within this framework. Modularity also allows clients to implement only relevant parts of the framework so that they can make use of basic conference control services without implementing all components. The conference control framework must also be generic so that it is not tied to any particular application. Unlike the loose conference control model [11], most messages need to be delivered reliably. (Some event notifications may be considered less critical.) Since moderators may be disconnected unexpectedly, the conference needs to be able to recover from moderator failures.

Since conference control manages shared resources, requests need to be authenticated and protected against replay attacks. In some scenarios, privacy of membership information and conference behavior may be important. Authorization of actions should be possible either manually or via a suitable automated policy language. Automating policy also makes it less likely that an attacker will overwhelm the moderator with repeated join attempts that have to be rejected individually.

Conference control can potentially serve as a means of a denial-of-service attack. For example, if all join attempts generate notifications to all participants, an attacker could easily amplify a single message into a large number of event notifications.

While the conference control protocols are likely to consume significantly less bandwidth than the media streams, some care needs to be taken for large conferences. Also, some conference participants may be behind low-bandwidth access links, such as a wireless network. Since the conference description and policy information can be large, incremental updates are preferred over having to re-send the whole description after each change. Similarly, changes in the participant list should be distributed as additions and removals. Also, not all participants care about the same level of detail; for example, some may only be interested when new members join or leave, but not when a participant adds herself to the floor queue. In some cases, network-layer multicast would be useful for distributing events, but it is not likely to be widely available particularly in mobile environments such as 3G networks. Thus, the protocol cannot depend on IP multicast. (RTCP [38] can be used to announce membership changes for multicast groups, but since the protocol is not reliable, notification of joins and leaves may take a while.)

We want to support hybrid conferences, where conference participants receive media via audio bridge or circuit-switched conferences (e.g., H.320 [19] or H.324 [18]), while conference notifications and requests are delivered via IP networks.

4.2 Conference Management Requirements

Conference management includes features such as conference creation, modification and termination. Conference

properties, such as conference model, policy and user rights lists, are established during conference creation.

Different conferencing models, such as dial-in and dial-out, must be supported. In a dial-in conference, users join the conference themselves, while the conference server calls up participants in a dial-out conference. For dial-out, the initiator can provide the server with a list of invitees, rather than having to complete signaling exchanges with each participant individually. The latter consumes far more time and scarce end system bandwidth. (We generally assume that the conference server is located on a high-speed access link, at least relative to the participants.)

The user rights list governs the privileges of potential participants. The user rights lists might include information about who can authorize the admission and expelling of participants and who can act on floor control requests. These functions are often combined into the role of the moderator, but a flexible system should allow these to be distributed among a set of participants. The policy may describe which users are pre-authorized to join, are explicitly forbidden from joining or may join, but in listen-only mode. Since Internet-based signaling protocols offer a variety of authentication mechanisms, a policy is also likely to define at what strength the participant has to authenticate. Unauthenticated users may be rejected or relegated to audience status.

Not all policy decisions can be made in advance. Decision makers need to be notified of events such as join requests, so that they can approve or deny them.

Conference management must also support the initialization of a floor control component. For example, the conference server must have the knowledge of who can create a floor prior to executing the floor creation commands.

It must also be possible to find out the conference status and member identities. The framework should support both occasional notifications to the clients and specific status inquiries from the clients.

Application session management must support the configuration of the conference-wide media settings. For example, it must be possible to add a video channel into the conference. After that, each user can decide whether he or she wants to participate in the session.

4.3 Floor Control Requirements

Conference applications often have shared resources such as the right to talk, access to a limited-bandwidth video channel, a pointer or input focus in a shared application, access to shared files or game rooms. Floor control enables applications or users to gain safe and mutually exclusive or non-exclusive access to the shared object or resource.

Floor control should support different floor control policies such as moderator-controlled or first-come-first-served. A moderator-controlled policy is relatively easy to implement, but needs to deal with the disconnection of the moderator. Automated queueing policies may cause starvation if one user holds the floor indefinitely. Time limits and renewable soft-stare solutions to prevent indefinite blocking.

For automated policies, many of the same algorithms used for packet scheduling apply. For example, some users may have higher priority than others, with access to the resource on a round-robin basis within each priority class. More detailed floor control requirements can be found in [6]. The framework presented here requires support of a basic first-in, first-out and moderated policy, but can support any number of more sophisticated allocation mechanisms.

5. FRAMEWORK FOR CONFERENCE CONTROL

Below, we describe an abstract framework based on the requirements and how these can be mapped onto real protocols.

5.1 Components of Conference Control

The conference control framework includes two principal components, conference management and floor control. The conference management component defines basic conference parameters such as name, type, subject and authentication information. It may include optional sub-components such as user management or distribution. The distribution subcomponent defines how the conference can be distributed across conference servers. The application session management component defines conference wide media configuration. Figure 1 clarifies the components. In addition to conference control components, conferencing requires also standard participant operations such as joining and leaving a conference. Current Internet session control protocols already provide most of the services needed for participant operations. For example, SIP can be used to initiate, modify and terminate sessions or transfer members. The Session Description Protocol (SDP) [12, 34] allows participants to negotiate media parameters; SDPng [22] adds additional capabilities. Standard SIP mechanism such as digest authentication enforce user authentication.

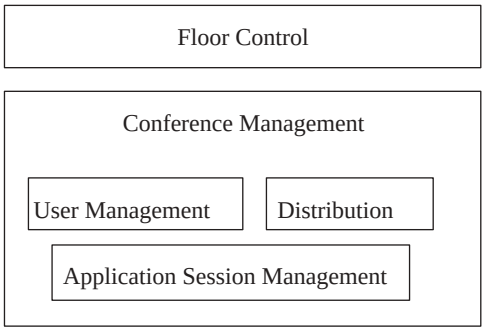

Figure 1: Conference control components

In this paper, we will assume that SIP is used for participant operations since it is the de facto session control protocol in the Internet.

The SIP architecture for conference control is straightforward. The conference server is a SIP entity that provides conference control services. One conference server usually supports several conferences, which are identified by a SIP URL such as sip:conf34@server.com. Depending on local policy, a server may allow a SIP INVITE with a currently unused SIP URL to create an ad-hoc conference. Each conference at a server may have different conference policies. One or more members of the conference may act as moderators for the conference. The conference server may also create subconferences elsewhere for scalability reasons.

5.2 Conference Management

In a conference, the conference manager controls the conferencing server. The conference manager may be a human being or an application instance that may or may not be co-located at the server. The conferencing server offers the conferencing services to the members. A conference server may be split into a control (signaling) and a media component, but we simply assume that the control component has the necessary mechanism to influence which member can send and receive media.

For large conferences, it may be necessary to split the conference members across multiple conference servers, organized as a tree. Conference servers can be selected based on the geographic distribution of the members, approximating a network-layer multicast tree. Conference control is simplified if all media streams and conference control commands reach the root of the conference distribution tree, similar to single-source multicast. If that assumption is made, conference servers in the tree can be treated like regular participants.

We assume that the conference is created by a conference creator who designates the initial set of moderators. Moderators can then designate others. More complicated election protocols are possible, but not pursued here.

The conference management messages can be roughly categorized as either commands or notifications. The commands are instructions or requests sent to the conference server, while notifications are informative announcements, typically sent by the server to the members. The following abstract commands for conference management are defined:

Create: create a conference and sets the initial conference parameters, such as the moderation style, join policy and maximum conference size;

Terminate: terminate an existing conference;

Modify: modify conference parameters.

These commands are sent from the conference creator or, if authorized, participant to the conference server. This supports basic clients since only the server is required to support the commands. A basic client can still participate to the conference as a regular participant.

In addition to commands, there are notifications. For example, a notification is used to inform the conference members about changes in membership, conference policy or media configuration.

5.3 Application Session Management

SIP can control application sessions only between one user and the conference server. It cannot change the conference state or the conference media settings at the server. For example, deleting media from a conference is difficult: if a participant removes the corresponding media line from the session description and sends a re-INVITE to the conference server, it is not clear whether that means that the participant would like to personally not receive that media session or would like to request that the server stop replicating that particularly media session for all participants. Similarly, when changing media sessions, it is impossible to distinguish whether the participant wants to restrict media codec choices, for example, for itself or the whole conference. Adding media is less problematic, except that the conference server may well have transcoding capabilities that make it unwise to restrict the media stream capability to that of the initiator. Thus, a separate control mechanism that explicitly adds, deletes and modifies conference server media sessions is desirable.

5.4 User Management

The user management sub-component governs user-related properties such as user access right groups and policies. For example, it may define that the conference has two user right groups, panel members and audience. The access control list (ACL) includes allow and deny definitions for incoming join attempts. Wildcards may be used in the definitions. User management also supports real-time decisions during the conference. Unresolved join attempts are kept in a table until the moderator, notified by the system, sends a request making a decision.

User management also supports a mass-invitation feature, in which a user may ask the conference server to invite several users at once into the conference. This saves bandwidth for low-bandwidth users since they do not have to issue separate invitations for each user.

Conference policy can be arbitrarily complex, expressible only in a full programming language. However, we claim that the majority of cases can be handled by defining a few simple operations that are directly tied to conference and floor management and giving users rights to those. Operations include the ability to terminate the conference, invite users, change user groups, expel users, and act as moderator. Since conferences are often divided into roles, we adopt the standard notion of user groups with privileges and assign members to groups.

We restrict policies to be stateless, i.e., we do not limit the number of operations of a given type that a member or group of members can perform. We also do not condition the operation on the identity of the member affected. For example, it is not possible to grant senior members the right to only expel five members during a conference, or that they cannot expel users from the domain example.com. We have considered, but not pursued, the notion of privilege levels, so that each level can only perform actions on members belonging to a lower level.

One motivation for limiting the complexity of policies is that devices with limited interfaces, such as mobile phones, should be able to control conferences.

The conference server may also have a default policy for all new conferences and may restrict the operations that new conferences can support.

Individual aspects of the conference description are managed by individual commands, rather than maintaining a single conference description. This allows clients to only concern themselves with components that they want to manipulate.

5.5 Floor Control

Floor control manages access to shared conference resources, generally by restricting the number of concurrent users that can modify a particular resource. Several resources can be managed by one floor. For example, sending sounds to the audio channel and a shared pointer may be managed by the same floor. A conference may have any number of floors, possibly created and deleted during the conference. Each floor has an associated queue. The queue can be managed either automatically, with a pre-defined queueing policy, or

via manual intervention by one or more moderators. Each floor may have a different moderator. The moderator can re-order requests in the queue, add and remove entries. A single conference participant can have multiple entries in the queue.

Detailed examples of floor control primitives can be found in [1, 6, 44], among others. We will follow earlier definitions with only minor modifications. We define the following floor control primitives: create floor, remove floor, claim floor, release floor, grant floor, revoke floor, remove floor claim, reorder claims and change floor settings. These primitives are exchanged between the conference participant, the conference server and the moderator. Initial floor rights are typically defined in the conference description when the conference is created.

As for conference management, there are commands and event notifications. Floor control commands such as "claim floor" are commands, and announcements such as "floor granted" are notifications. All messages either terminate or originate on the conference server and are distributed reliably.

5.6 Scaling to Large Conferences

Multicast-based conferences scale well to large group sizes, but multicast support is spotty, particularly in next-generation wireless networks and in residential networks (e.g., DSL using PPPoE). Most existing PSTN-based teleconferences also tend to be small, with an average of around five members, so that the replication efficiency has to be weighed against the substantial amount of state that the network has to maintain. Also, since only source-specific multicast [15] seems to have a chance to be deployed, users will have to send data to the root of the tree, making the delay the same as a central-server solution. In some cases, a central server may even be more efficient. For example, if audio streams are mixed at the server and multiple sources are talking simultaneously, a single server reduces the overall bandwidth and makes bandwidth reservation easier. (There is no need to allocate bandwidth for the occasional multiple talker case.)

We do envision that for large conferences, hybrid solutions will be attractive. Here, a central conference server acts as the root of the SSM tree. The conference server mixes media streams and sends out replicas to participants that do not have multicast connectivity.

Single-server conferences scale to about 100 members using modest computing resources [39]. For larger or geographically distributed conferences where multicast is unavailable, we propose a tree architecture, essentially approximating a multicast distribution mechanism at the application layer. Such trees can grow dynamically as members join. When the group size exceeds a pre-defined limit, or there are clusters of members in certain geographic areas, the conference server may decide to create a server tree for the conference. First, the conference server creates one or more subconferences at the remote servers. After that, a number of existing or new members are transferred to the other servers in order to decrease the load at the original server and to improve reliability and response times in the subconferences. Each subconference may create new subconferences. Thus, a distributed server tree (DST) has been created. Each sub-conference may have local communication such as a local voice channel or other local resources. However, it also has access to the global conference resources such as the right to talk to the conference-wide voice channel.

The DST can be configured, via the session description, to have members send media to the local server or to the root server. Floor grant commands propagate down the tree, while floor claims are sent up the tree or enter the tree at the root. In either case, each tree node knows the current floor holder and can block other media streams.

Figure 2 shows a server tree in which the main server has created two subconferences, S2 and S3, at different servers. There are four members in the conference (A, B, C, and D).

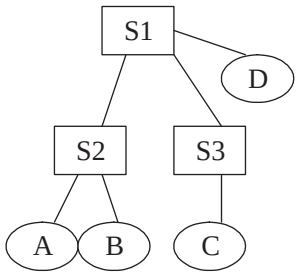

Figure 2: Distributed server tree

Server failures and tree-balancing operations complicate the model slightly. For example, if server S2 dies suddenly, users A and B must be transferred elsewhere immediately. A relatively efficient solution is that clients always know the root server address S1 and can re-join there. The root server may then redirect users elsewhere, if necessary.

SIP already has the means for redirecting and transferring users. A prospective conference participant can be sent to a different server using the "302 Moved Temporarily" response; a participant already in a conference is sent a SIP REFER request and then moves to another conference server. (Alternatively, the new server can contact the user on behest of the original one.)

We assume that the conference server has a local policy that decides when to graft another server into the DST, as described above. Once a decision has been made to enlarge the DST, the appropriate conference server has to find a suitable server, that is, a server willing to host the subconference that has sufficient resources. Clearly, this is an instance of the wide-area service location problem [32, 2] or application-layer routing problem [8], but simpler solutions may be sufficient. For example, if the domain of the new candidate server is known, we can use directed service location [45] (Fig. 3). There, the conference server queries the DNS SRV [9] record for a service entry for the Service Location Protocol (SLP) [10]. The conference server then sends an SLP query to the SLP directory agent (DA), obtaining a list of suitable servers. The DA may also keep track of the current server load, or the requesting server can query servers. The SLP information also contains the SOAP conference management address.

Since the search process may take several seconds, a server in the DST that needs to shed load should start the process well before it becomes overloaded.

6. IMPLEMENTING THE FRAMEWORK

6.1 Commands and Noti®cations

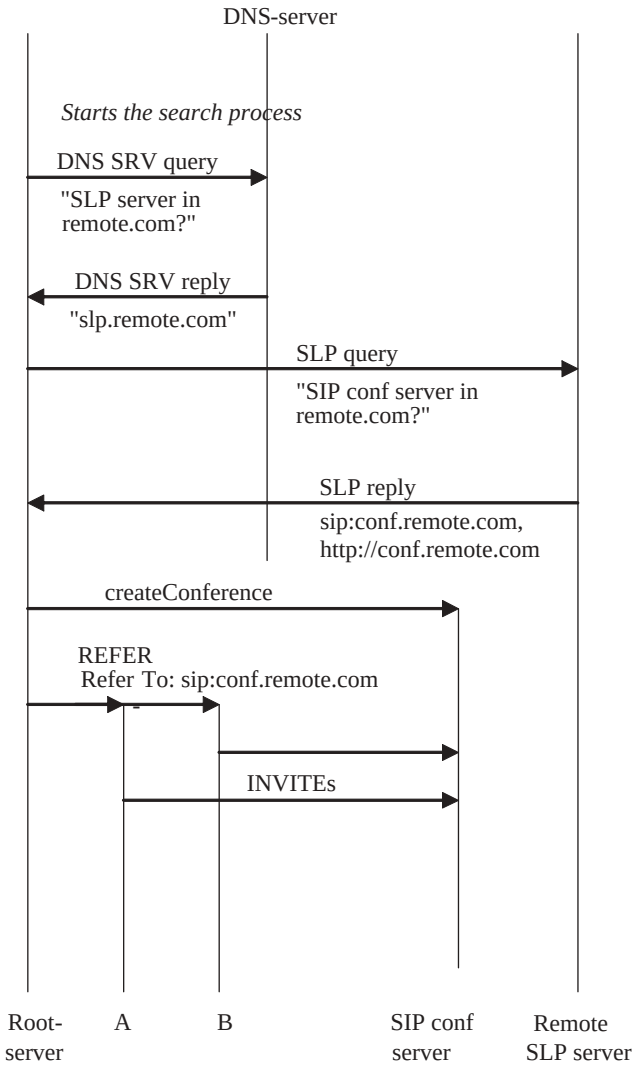

Figure 3: Conferencing server search process

Earlier in this paper, we stated that all abstract conference control messages are either commands or notifications. The latter is easy to implement with SIP since the SIP events framework [28] fits perfectly for this purpose. Members use the SUBSCRIBE method to ask for event notifications, which are delivered via NOTIFY requests. The event mechanism makes it easy to separate the implementation of the conference media client from the control client. Also, several entities representing a single participant can subscribe to events.

Separate event names are used for different modules. For example, floor control and conference management have separate event names. Beyond subscribing to certain events, the subscription may indicate the level of detail that the subscriber desires to receive in notifications. For large groups, limiting the number of different notifications that have to be sent is desirable, so that we prefer finer-grained event names, rather than an unbounded number of XML pattern matches. The first NOTIFY request in the subscription includes a full state while later NOTIFY requests include only the changes.

Commands can be implemented in several ways. The simple solution is to use HTTP to carry an XML document to the server, as implemented by schemes such as SOAP (Simple Object Access Protocol) [26]. We selected SOAP to implement commands because it provides a mechanism for carrying remote procedure calls and it is already widely used for web services. Typically, the SOAP request is carried in a HTTP POST request but it is possible to use other transports as well, such as a SIP or SMTP. Compared to HTTP, SIP would offer the possibility to use UDP for transport and location-independent addressing, but the benefit is small; the payload size may exceed the typical unfragmented UDP message size and commands are always addressed to a single server, rather than potentially moving destinations.

HTTP and SIP can readily co-exist in the same system. They can share the same authentication mechanism and keys. Both can use TLS (Transport Layer Security) strong hop-by-hop encryption.

Each conference control (sub-)component such as conference management, user management, and floor control has its own namespace for SOAP commands. This keeps them independent and allows the usage of different components. For example, a new floor control module can be invoked by referencing a new SOAP namespace.

The SOAP commands are generally executed asynchronously. That is, the result returned by the SOAP request simply indicates that the request was parsed correctly, not that it was executed successfully. This approach is taken since many operations may take a long time and succeed incrementally. Consider, as an example, a group invitation, where individual invitations are likely answered by humans, on timescales of tens of seconds. The issuer of the SOAP request will receive SIP notifications indicating the progress of the request.

One-time decisions are also implemented using SOAP and SIP events. As an example, consider the case where several non-members are trying to join the conference approximately at the same time. The server sends out an event notification to the designated set of conference managers. To save bandwidth, it may delay notification until a sufficient number of requests have accumulated, bundling them into one message. The conference manager then uses SOAP requests to approve or reject pending requests. The request might look something like this:

```
<join_requests>
  <accept>
   <user>sip:user1@domain.com</user>
  </accept>
  <deny>
   <user>sip:user@spam.com</user>
   <user>sip:user1@example.com</user>
  </deny>
<join_requests>
```

The floor control component is currently being implemented for Columbia University's CINEMA server and the sipc client [20]. Messaging details have been submitted [44] to the IETF for standardization.

7. EXAMPLES

Figure5 gives a signaling diagram for a fictional conference. First, a conference is created using the SOAP com-

mand createConference, using HTTP POST as the underlying transport mechanism. The SOAP command includes a conference description specifying conference type, policy, and other relevant information. In this example the conference type was dial-out and the names of the invited users (A and moderator) were given. The conference server establishes a session with user A and the moderator by inviting them into the conference. Standard SIP authentication may be used. All invited clients may subscribe to the conference events and specify the level of details they are willing to receive. Occasional SIP notifications are sent to the conference members.

Next, a non-member B tries to join the conference by sending a SIP INVITE request to the conference server. Since B was not in the ACL, the conference server asks the moderator to make a decision whether B can be accepted. The moderator makes the decision and sends a SOAP command UpdateJoinTable to the server. The server can now continue with the on-going SIP transaction with B by sending the "200 OK" response to confirm admission. Next, B makes a floor control request using the SOAP command floorClaim. The floor control moderator is then notified about the request and a decision is made using the floorGrant SOAP command.

The following example (Fig. 4) is a createConference command that includes several sub-components. Each component defines the SOAP namespace in use. This allows different components to be used within the framework. Separate SOAP commands are used to define each component. For example, the setModerator command is used inside UserManagement namespace to define the conference moderator. The SDPng format is used to describe the session media details.

```
<cc:createConference xmlns:cc="http://schemas.cc.org/cc/cc">
  <contact>sip:meeting12@cs.columbia.edu</contact>
  <subject>IRT group meeting</subject>
  <type>moderated</type>
  <info>
     <webpage>http://www.cs.columbia.edu/IRT</webpage>
     <email>mailto:irt@cs.columbia.edu</email>
  </info>
  <authentication>
     <method>digest</method>
     <password encode=base64>d3h0Ond4dA==</password>
  </authentication>

  <UserManagement>
    <commands>
      <um:setModerator xmlns:um="http://schemas.cc.org/cc/um">
          <moderator>sip:hgs@cs.columbia.edu</moderator>
      </um:setModerator>
      <um:accessControl xmlns:um="http://schemas.cc.org/cc/um">
        <ACL>
          <block>
            <user>anonymous@*</user>
          </block>
          <allow>
            <user password=no>*@cs.columbia.com</user>
            <user password=yes>*@nokia.com</user>
          </allow>
        </ACL>
      </um:accessControl>
      <um:createGroups xmlns:um="http://schemas.cc.org/cc/um">
        <groups>
          <group name="senior-member" priority=1>
            <user>hgs@cs.columbia.edu</user>
            <user>nieh@cs.columbia.edu</user>
          </group>
          <group name="irt-people" priority=2>
            <user>lennox@cs.columbia.edu</user>
            <user>wenyu@cs.columbia.edu</user>
          </group>
        </groups>
      </um:createGroups>
      <um:invite xmlns:um="http://schemas.cc.org/cc/um">
        <users>
          <user>Bob</user>
          <user>Tom</user>
          <user>Alice</user>
        </users>
        <group name="irt-people" />
      </um:invite>
    </commands>
  </UserManagement>

  <Media type=application/sdpng>
    <def>
      <audio:codec name="audio-basic" encoding="PCMU"
                sampling="8000" channels="1"/>
      <rtp:pt name="rtp-avp-0" pt="0" format="audio-basic"/>
    </def>

    <cfg>
      <component name="interactive-audio" media="audio">
        <alt name="AVP-audio-0">
          <rtp:session format="rtp-avp-0">
            <rtp:udp role="receive" endpoint="A" addr="192.168.1.1"
             rtp-port="7800"/>
          </rtp:session>
        </alt>
       </component>
    </cfg>
  </Media>
<FloorControl>.........</Floorcontrol>
.........
</m:createConference>
```

Figure 4: SOAP signaling example

8. USE WITH OTHER SERVICES

In addition to the conferencing services, the conference control framework may be used as an example for other applications as well. For example, Rosenberg [31] describes some of the requirements for presence and instant messaging applications. The requirements include buddy list management, block and allow list management, buddy list subscription, presence status management, and real-time authorization. Buddy list subscription assumes that the group list can somehow be created and modified. These requirements are almost identical to the conference control requirements. In both cases the client must be able to manage the resources in a server. As for conferencing, buddy list management can be divided into commands and notifications. For user authorization, it may make sense to define a unified approach that identifies the request to be allowed or issued (SUBSCRIBE for presence, INVITE for conferences).

9. CONCLUSION

We have described a component-based framework for SIP conference control. It was shown that conference control can be implemented with two kinds of operations, commands and notifications. SOAP is used for commands, while the SIP event framework delivers notifications. In addition to conference control, standard participant operations are needed for conferencing. Currently, SIP is the de-facto session control protocol and can be used for participant operations between the server and the client. Conference control includes two main components. First, a floor control component is used to provide a conflict-free access to shared resources such as the right to talk to the speech channel. Second, a conference management component defines the conference parameters and provides real-time authorizations, if necessary. Moreover, it includes optional sub-components such as user management, application session management and distribution.

We claim that combinations of network-layer source-specific multicast and a server tree can scale Internet conferences to arbitrary sizes, even for the common case of limited deployment of multicast. Requests and notifications have a much lower volume than media streams, but a distributed server tree can also ensure that they scale to large groups.

The similarities and differences between managing conferences and buddy lists remain to be explored in detail.

10. REFERENCES

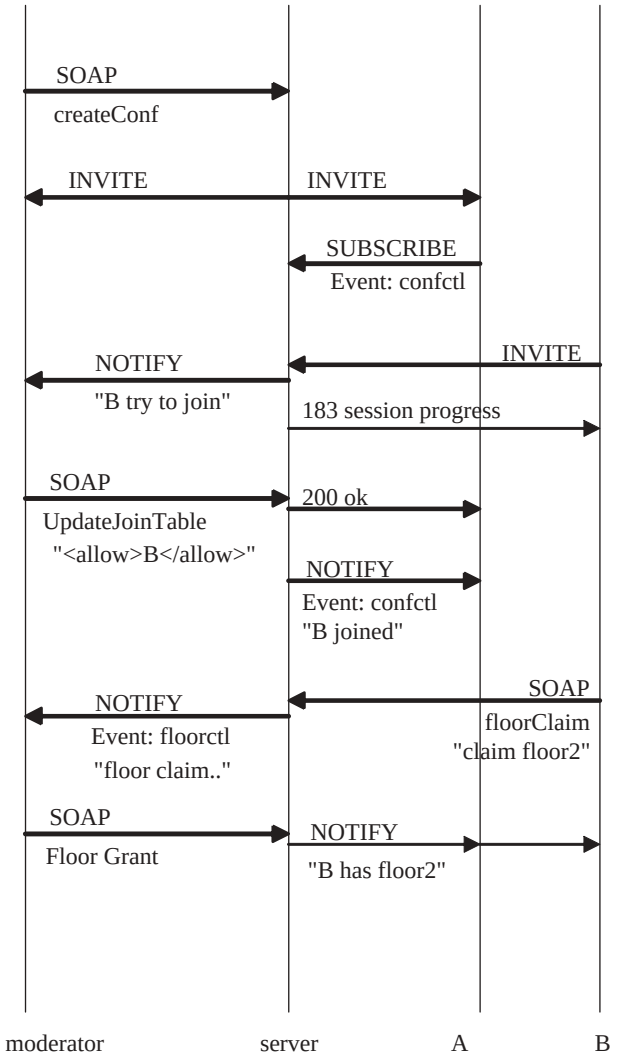

Figure 5: An example of conference control signaling

[1] Bormann, C., Kutscher, D., Ott, J., and Trossen, D. Simple conference control protocol service specification. Internet Draft, Internet Engineering Task Force, Mar. 2001. Work in progress.

[2] Castro, P., Greenstein, B., Muntz, R., Kermani, P., Bisdikian, C., and Papadopouli, M. Locating application data across service discovery domains. In *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)* (Rome, Italy, July 2001), pp. 28–42.

[3] Chen, M.-S., Barzilai, T., and Vin, H. M. Software architecture of DiCE: a distributed collaboration environment. *ACM Computer Communication Review 22*, 3 (Mar. 1992), 51–52.

[4] Crowley, T., Milazzo, P., Baker, E., Forsdick, H., and Tomlinson, R. MMConf: an infrastructure for building shared multimedia applications. In *CSCW 90 Proceedings* (Oct. 1990), ACM, pp. 329–342.

[5] Depaoli, F., and Tisato, F. Coordinator: a basic building block for multimedia conferencing systems. In *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)* (Phoenix, Arizona, Dec. 1991), IEEE, pp. 2049–2053 (58.1).

[6] Dommel, H.-P., and Garcia-Luna-Aceves, J. Floor control for activity coordination in networked multimedia applications. In *Proc. of 2nd Asian-Pacific Conference on Communications (APCC)* (Osaka, Japan, June 1995).

[7] Forgie, J. W. Voice conferencing in packet networks. In *Conference Record of the International Conference on Communications (ICC)* (Seatle, Washington, June 1980), IEEE, pp. 21.3.1–21.3.4.

[8] Ghosh, A., Fry, M., and Crowcroft, J. An architecture for application layer routing. In *Proc. of International Working Conference on Active Networks (IWAN)* (Philadelphia, Pennsylvania, September/October 2001).

[9] Gulbrandsen, A., Vixie, P., and Esibov, L. A DNS RR for specifying the location of services (DNS SRV). Request for Comments 2782, Internet Engineering Task Force, Feb. 2000.

[10] Guttman, E., Perkins, C., Veizades, J., and Day, M. Service location protocol, version 2. Request for Comments 2608, Internet Engineering Task Force, June 1999.

[11] Handley, M., Crowcroft, J., Bormann, C., and Ott, J. Very large conferences on the internet: the internet multimedia conferencing architecture. *Computer Networks 31* (1999).

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

[13] Handley, M., Wakefield, I., and Crowcroft, J. CCCP: confrerence control channel protocol-a scalable base for building conference control applications. *ACM Computer Communication Review 25*, 4 (Oct. 1995), 275–287.

[14] Hebrard, P., Karmouch, A., and Georganas, N. D. Management in multimedia cooperative applications. In *Proceedings of the IEEE Conference on Global Communications (GLOBECOM)* (Phoenix, Arizona, Dec. 1991), IEEE, pp. 2054–2058 (58.2).

[15] Holbrook, H., and Cain, B. Source-specific multicast for IP. Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

[16] International Telecommunication Union. Data protocols for multimedia conferencing. Recommendation T.120, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, July 1996.

[17] International Telecommunication Union. Generic conference control. Recommendation T.124, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.

[18] International Telecommunication Union. Terminal for low bit-rate multimedia communication. Recommendation H.324, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.

[19] International Telecommunication Union. Narrow-band visual telephone systems and terminal equipment. Recommendation H.320, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1999.

[20] Jiang, W., Lennox, J., Schulzrinne, H., and Singh, K. Towards junking the PBX: deploying IP telephony. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)* (Port Jefferson, New York, June 2001).

[21] Kausar, N., and Crowcroft, J. An architecture of conference control functions. In *Proc. of Photonics East* (Boston, Massachusetts, Sept. 1999), SPIE.

[22] Kutscher, D., Ott, J., and Bormann, C. Session description and capability negotiation. Internet Draft, Internet Engineering Task Force, Mar. 2002. Work in progress.

[23] Lauwers, J. C., and Lantz, K. A. Collaboration awareness in support of collaboration transparency: requirements for the next generation of shared window systems. In *Proceedings of CHI'90* (Apr. 1990), ACM, pp. 303–311.

[24] Malone, T. W., and Crowston, K. The interdisciplinary study of coordination. *ACM Computing Surveys 26*, 1 (Nov. 1993), 87–119.

[25] Malpani, R., and Rowe, L. A. Floor control for large-scale Mbone seminars. In *Proc. of ACM Multimedia* (Seattle, Washington, Nov. 1997).

[26] Mitra, N. Soap version 1.2 part 0: Primer. W3C working draft, World Wide Web Consortium (W3C), Dec. 2001.

[27] Newman-Wolfe, R., Ramirez, C. L., Pelimuhandiram, H. Montes, M., Webb, M., and Wilson, D. A brief overview of the dcs distributed conferencing system. In *Proc. of Usenix Summer Conference* (Nashville, TN, June 1991), Usenix, pp. 437–452.

[28] Roach, A. Sip-specific event notification. Request for Comments 3265, Internet Engineering Task Force, May 2002.

[29] Roseman, M., and Greenberg, S. GroupKit: a groupware toolkit for building real-time conferencing applications. In *Proceedings of the Conference on Computer Supported Cooperative Work (CSCW)* (Toronto, Canada, Nov. 1992), ACM, pp. 43–50.

[30] Roseman, M., and Greenberg, S. Building flexible groupware through open protocols. In *Proceedings COSC'93* (1993), ACM, pp. –.

[31] Rosenberg, J. A component model for SIMPLE. Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

[32] Rosenberg, J., and Schulzrinne, H. Internet telephony gateway location. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)* (San Francisco, California, March/April 1998), pp. 488–496.

[33] Rosenberg, J., and Schulzrinne, H. Models for multi party conferencing in SIP. Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

[34] Rosenberg, J., and Schulzrinne, H. An offer-answer model with SDP. Request for Comments 3264, Internet Engineering Task Force, May 2002.

[35] Rubin, D., Craighill, E., and Raphael, R. Topics in the design of a natural teleconferencing system. In *Conference record of the IEEE National Telecommunications Conference* (Birmingham, Alabama, Dec. 1978), vol. 1, IEEE, pp. 12.4.1 – 12.4.5.

[36] Sakata, S. Development and evaluation of an in-house multimedia desktop conference system. *IEEE Journal on Selected Areas in Communications 8*, 3 (Apr. 1990), 340–347.

[37] Schubert, I., Sisalem, D., and Schulzrinne, H. A session floor control scheme. In *Proc. of International Conference on Telecommunications* (Chalkidiki, Greece, June 1998).

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

[39] Singh, K., Nair, G., and Schulzrinne, H. Centralized conferencing using SIP. In *Internet Telephony Workshop 2001* (New York, Apr. 2001).

[40] Sisalem, D., and Schulzrinne, H. The multimedia internet terminal (MInT). *Telecommunications Systems 9*, 3-4 (Sept. 1998), 423–444.

[41] Trossen, D. *Scalable Group Communications in Tightly Coupled Environments*. PhD thesis, University of Technology, Aachen, Germany, Sept. 2000.

[42] Watabe, K., Sakata, S., Maeno, K., Fukuoka, H., and Ohmori, T. Distributed desktop conferencing system with multiuser multimedia interface. *IEEE Journal on Selected Areas in Communications 9*, 4 (May 1991), 531–539.

[43] Watabe, K., Sakata, S., Maeno, K., Fukuoko, H., and Ohmori, T. Distributed multipatry desktop conferencing system: MERMAID. In *CSCW 90 Proceedings* (Oct. 1990), ACM, pp. 27–38.

[44] Wu, X., et al. Use SIP and SOAP for conference floor control. Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.

[45] Zhao, W., et al. The SLP service and remote discovery in SLP. Internet Draft, Internet Engineering Task Force, Mar. 2002. Work in progress.