Internet Engineering Task Force
Internet Draft

Wu/Koskelainen/Schulzrinne
Columbia University

draft-wu-sipping-floor-control-04.txt
March 2, 2003
Expires: September 2003

# Use of Session Initiation Protocol (SIP) and Simple Object Access Protocol (SOAP) for Conference Floor Control

STATUS OF THIS MEMO



Abstract

During a conference, floor control coordinates simultaneous access to shared resource in multimedia conferences. Floor control allows applications and users to gain safe and mutually exclusive or non-exclusive access to the shared resources. This document defines an approach of using Session Initiation Protocol (SIP) event notification mechanism and Simple Object Access Protocol (SOAP) to perform floor control.

## 1 Introduction

Many existing conference management protocols, such as [11][12] have defined floor control functions. Floor control can be used to avoid or resolve conflicts among simultaneous media inputs. For example, at a given time, the moderator of a floor can ensure that only one person is heard by other participants or one person types into a shared document.

The conference models can be centralized or non-centralized [13]. In a centralized model, we assume there is one conference server acting as the root of a conference. The root conference server receives all floor requests and can control the propagation of media in the conference directly or through sending requests to other conference servers in a tree topology. There is no such root conference server in the non-centralized model. The non-centralized model does not work well with the mechanism in this document. In the rest of this document, we simply use conference server referring to the root conference server.

The conference server needs to be able to control the shared resources. For example, the mixer in a conference server can selectively choose the media sources for mixing. The moderators and participants of the conference should be able to send the floor control commands to the conference server to change floor status, and the conference server should notify the moderators and participants of changes.

A floor control protocol is used to convey the floor control messages among the moderators of the conference, the conference server and the participants of the conference. The floor control protocol does not deal with the conference management such as how to elect the moderator of the conference or how to add users to the conference.

We divide the floor control messages into two categories: floor control events and floor control commands.

The conference server sends floor control events to moderators or participants to report changes in the resource status. The SIP [1] events architecture [2] is well fitted for conveying floor control events. This document defines a new event package named floor-control for the floor control events.

Moderators or participants send floor control commands to the conference server to change the status of resources. Floor control commands are like remote-procedure calls from the moderators or the participants to the conference server. Since one of SOAP's [3] design goals is to encapsulate and exchange RPC calls, instead of creating

another RPC protocol, we consider of using SOAP to transmit floor control commands. SOAP messaging most commonly uses HTTP as the transport protocol. However, SIP can also be used to carry the SOAP content [14]. The same mechanism can be used for general conference control [15].

In the centralized conference model, floor claims are ordered in queues at the conference server. Floor moderators can assert priority or reorder the claims in the queues.

### 1.1 Conventions of This Document

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [4].

## 2 Using SIP and SOAP for Floor Control

### 2.1 SIP Event Notification for Tracking Floor Status

A new SIP event package 'floor-control' is introduced for floor status notification. The conference server can indicate that it supports the floor-control event package by including an 'Allow-Events: floor-control' header field. If a user wants to be informed of the floor status, the user's UA MUST send a SUBSCRIBE with the Event header set to 'floor-control' to the conference server. The events in the floor-control package will be described in detail in Section 4.

If the user's UA cannot understand the 'floor-control' package, the user may use web based floor control approach. To convey this URL for the web based floor control, the conference server MAY use the 'Call-Info' header to bring the URL. And a new value, named 'floor-control', SHOULD be used for the Call-Info header's purpose parameter.

### 2.2 Using SDP to Establish Floor Control Channel

We use Session Description Protocol (SDP) [5] to convey the floor control channel information since floor control channels and media channels are closely coupled. Any approach trying to convey floor control channel information must know the detail of the session description.

When a user joins a conference, the conference server uses SDP's 'a' line to indicate that the conference is moderated.

a=type:moderated

The new participants joining the moderated conference SHOULD start media tools as 'mute' so that they do not send media.

If only some of the media streams are moderated in a conference, the 'a=type:moderated' line MUST NOT be used because this attribute is a session level attribute, it indicates all the media streams in a session are moderated. As introduced later in this section, grouping media channels and a control channel can clearly indicate that the grouped media channels are moderated. For moderated media channels, the new participants SHOULD start the media tools as 'mute' accordingly.

As indicated in RFC2327 [5], the 'm' line can specify the conference control tools, the port and protocol used for control. The following is an example:

```
m=control 5060 SIP SOAP
```

However, the 'm' line does not provide the information of HTTP URL or SIP URI. A way of relaying the URI information is to use a session-level 'a=floorcontrol:' attribute. This is similar to how RTSP's 'a=control:' attribute [16] works. The following is an example:

```
a=floorcontrol:sip:floorcontrol@foo.example.com
```

Using m=control line cannot associate floor control channel with particular media channels. Thus, it cannot handle the case that a moderator can only control part of the resources, e.g., a moderator can control the cameras but not the microphones.

To solve this problem, we can group the control lines with the other media lines as described in the RFC3388 "Grouping of media lines in the Session Description Protocol (SDP) [6]. A new semantics named FL (floor control) is defined for this purpose. An example session description is shown below:

```
v=0
o=Foo 289083124 289083124 IN IP4 foo.example.com
s=SIP conferencing
t=0 0
c=IN IP4 224.2.17.12/127
a=group:FL 1 2 4
m=audio 10000 RTP/AVP 0
a=mid:1
m=video 20000 RTP/AVP 31
a=mid:2
m=application 30000 udp wb
a=mid:3
```

```
m=control 5060 SIP SOAP
a=floorcontrol:sip:floorcontrol@foo.example.com
a=mid:4
```

In this example, the control channel can only control audio and video, but not the whiteboard. The SIP URI for the control channel is sip:floorcontrol@foo.example.com.

The control line cannot indicate whether a user is a moderator or not. The conference server will use the floorCreated (Section 4.1) or configChanged (Section 4.3) event notification to indicate that.

If a participant dials in the conference, the INVITE message will not contain the m=control line for floor control because the participant's user agent does not know how many floor control channels are needed. The m=control line can only be initiated from the conference server's INVITE message. If a participant's user agent cannot support floor control, in the SIP response, the user agent will set the port as 0 for the m=control line.

### 2.3 Use of SOAP for Floor Control Commands

If a user wants to change the floor control status, the user's UA MAY use SOAP to carry the floor control commands. The SOAP message can be carried either in HTTP or SIP. The name and the parameters of the commands will be described in detail in Section 5.

Both the moderator and the participants can have control over the conference. However, they have different control command set. The conference server SHOULD have knowledge of the moderated resources (e.g., who can control the resources) and SHOULD be able to convey the knowledge to the users.

## 3 Datatypes in the floor control messages

We use an XML-based data format for the floor control messages. A floor control message contains information about floors, resources and floor claims. The namespace URI for elements defined by this specification is a URN [7], using the namespace identifier 'ietf' defined by [8] and extended by [9]. This URN is:

urn:ietf:params:xml:ns:floor-control

To represent such information, the following sections define several datatypes and provides the XML schema fragment for each datatype.

### 3.1 floorType

The floorType is used to define a floor. It contains an optional attribute and several sub-elements. The optional attribute 'maxHolders' defines how many users can hold the floor simultanuously. The default value of the 'maxHolders' is 1. The following XML schema fragment defines the floorType.

```
<xs:complexType name="floorType">
  <xs:sequence>
    <xs:element name="resources" type="resourcesType" minOccurs="0"/>
    <xs:element name="users" type="usersType" minOccurs="0"/>
    <xs:element name="moderators" type="moderatorsType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0"
            maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="maxHolders" type="xs:int" default="1"/>
</xs:complexType>
```

There are three sub-elements in the floorType. The element 'resources' has the type 'resourcesType'. It lists the resources the floor controls. If it is not provided, the floor controls all the resources of the conference. The element 'users' has the type 'usersType'. It defines who can hold the floor. If it is not provided, every user in the conference can hold the floor. The element 'moderators' has the type 'moderatorsType'. It defines who are the moderators of the floor. If it is not provided in the CreateFloor command, the user who sends the FloorCreate command will serve as the moderator. If the element 'moderators' is not provided in a floor control notification, it means this information is hidden by the conference server.

The wildcard xs:any in the floorType is used to provide additional information of the floor, such as floor control policies.

### 3.2 resourcesType

The resourcesType groups resources that can be controlled by one floor. It contains a sequence of elements with the type resourceType. The following XML schema fragment defines the resourceType and the resourcesType.

```
<xs:simpleType name="resourceType">
  <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:complexType name="resourcesType">
```

```
    <xs:sequence>
      <xs:element name="resource" type="resourceType"
                  maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

The value of the 'resource' element MUST be one of the mids defined in the SDP of the conference server's message. The SDP's 'a=mid:' attribute provides the value.

### 3.3 usersType

The usersType contains a list of users that can claim the floor or the URL of the user list. The following XML schema fragment defines the userType and the usersType.

```
<xs:simpleType name="userType">
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
<xs:complexType name="usersType">
  <xs:sequence>
    <xs:element name="user" type="userType" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="url" type="xs:anyURI"/>
</xs:complexType>
```

The string value in userType MUST be a valid SIP or SIPS URI. The attribute 'url' of usersType gives the web URL that contains the user list. If it is not provided, there MUST be at least one 'user' element presented.

### 3.4 moderatorsType

The moderatorsType contains a floor moderator list. The following XML schema fragment defines the moderatorType and the moderatorsType.

```
<xs:simpleType name="moderatorType">
  <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
<xs:complexType name="moderatorsType">
  <xs:sequence>
    <xs:element name="moderator" type="moderatorType"
```

```
                  maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
```

The string value in moderatorType MUST be a valid SIP or SIPS URI.

### 3.5 holdingType

The holdingType defines the relationship between the floor holders and the resources. Within the holdingType there are two elements, the resources and the users. The element 'resources' has the type 'resourcesType'. If it is not provided, the holding is for all the resources of the conference. The element 'users' has the type 'usersType'. If 'users' is not provided, there is no user holding the floor. The holdingType has two attributes, the required attribute timestamp gives the time the users get the access right of the resources and the optional attribute expiration defines when the holding will be expired. If the expiration is not provided, the holding will end until the users release the floor. The following XML schema fragment defines holdingType.

```
<xs:complexType name="holdingType">
  <xs:sequence>
    <xs:element name="resources" type="resourcesType" minOccurs="0"/>
    <xs:element name="users" type="usersType"/>
  </xs:sequence>
  <xs:attribute name="expiration" type="xs:dateTime"/>
  <xs:attribute name="timestamp" type="xs:dateTime" use="required"/>
</xs:complexType>
```

### 3.6 claimType

The claimType contains the information sent by the participants to request a floor. Floor claims are put in a queue at the conference server. Generally, the order of the queue is based on the timestamp of floor claims.

The required element 'user' defines who sends the claim. The optional element 'resources' defines what resources the user claims for. If not provided, the user wishes to claim for all the resources. The optional element 'subject' provides the reason for holding the floor. The optional element 'holdingTime' defines how long the user expects to hold the floor. The required attribute 'claimID' MUST be

globally unique for the conference, generated by the combination of a random string and the claimer's SIP URI. The claimer can modify an existing claim by sending a new claim with the same claimID as the existing one. When a claim is granted, before the claimer releases the floor, the claim is considered as a granted claim. Granted claims MUST be removed from the claim queue. The conference server MUST keep track of the granted claims. The current floor holder can send a claim with the same claimID as his granted claim to ask for the extension of the holding time. The optional attribute 'timestamp' provides the time that the claim is generated. The optional attribute 'expiration' defines when a claim expires. The conference server MUST remove expired floor claims from the floor claim queue. The optional attribute 'priority' defines the priority of the claim. There are four possible priority values, "non-urgent", "normal", "urgent" and "emergency". By default, the priority is "normal". The following XML schema fragment defines the claimType.

```
<xs:simpleType name="PriorityType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="non-urgent"/>
    <xs:enumeration value="normal"/>
    <xs:enumeration value="urgent"/>
    <xs:enumeration value="emergency"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="claimType">
  <xs:sequence>
    <xs:element name="user" type="userType"/>
    <xs:element name="resources" type="resourcesType" minOccurs="0"/>
    <xs:element name="subject" type="xs:string" minOccurs="0"/>
    <xs:element name="holdingTime" type="xs:duration" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="claimID" type="xs:string" use="required"/>
  <xs:attribute name="timestamp" type="xs:dateTime" use="optional"/>
  <xs:attribute name="expiration" type="xs:dateTime" use="optional"/>
  <xs:attribute name="priority" type="PriorityType"
                default="normal" use="optional"/>
</xs:complexType>
```

### 3.7 claimsType

The claimsType contains a list of claims. The following XML schema fragment defines the claimsType.

```
<xs:complexType name="claimsType">
  <xs:sequence>
    <xs:element name="claim" type="claimType" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

### 3.8 operationType

We defined several queue operations such as move up, move down, move to the top and move to the bottom to manipulate the floor claim queue. The 'up', 'down', 'top' and 'bottom' are the operators. The operation MAY have an argument. For 'up' and 'down', the argument means how many steps to move a claim. For 'top', the argument means to what position to move a claim counting down from the top of the queue. For 'bottom', the argument means to what position to move a claim counting up from the bottom of the queue. If the argument is not presented, the operation 'up' will move the claim up one position in the queue; the operation 'down' will move the claim down one position in the queue; the operation 'top' will move a claim at the top of the queue and the operation 'bottom' will move a claim to the bottom of the queue. The following XML schema fragment defines the operatorType and the operationType.

```
<xs:simpleType name="operatorType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="up"/>
    <xs:enumeration value="down"/>
    <xs:enumeration value="top"/>
    <xs:enumeration value="bottom"/>
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="operationType">
  <xs:sequence>
    <xs:element name="operator" type="operatorType"/>
    <xs:element name="argument" type="xs:int" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="timestamp" type="xs:dateTime"/>
</xs:complexType>
```

## 4 Floor control events

Table 1 shows the events for the floor control package. We specify an XML-based data format for the parameters of each event. The MIME

type for the format is application/floor-control+xml, consistent with the recommendations provided in RFC 3023 [10].

| Event name | Description | Issuer -> Receiver |
|---|---|---|
| floorCreated | A floor has been created for some resources and participants. | Server -> User |
| floorRemoved | A floor has been removed for some resources | Server -> User |
| configChanged | Floor configuration changed | Server -> User |
| floorChanged | Floor changed to different users | Server -> User |
| queueChanged | Floor claim queue changed | Server -> Moderator |

Table 1: Floor control events

In Table 1, the 'Server' refers to 'Conference server' and the 'User' refers to either 'Moderator' or 'Participant'.

### 4.1 floorCreated event

When a floor is created for some set of resources, the conference server SHOULD send a notification to the parties interested in this event.

The floorCreated event contains the information about what are the resources being controlled and who can access the floor.

The XML document for the floorCreated event starts with a 'floorCreated' tag. Within the tag are one or more 'floor' elements. The floor element has the type floorType. The following XML schema fragment defines the floorCreated event.

```
<xs:element name="floorCreated">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="floor" type="floorType" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 1 shows an example of the floorCreated event.

```
<floorCreated>
  <floor maxHolders="1">
    <resources>
      <resource>mid:1</resource>
    </resources>
    <users>
      <user>sip:user_a@foo.example.com</user>
      <user>sip:user_b@foo.example.com</user>
      <user>sip:user_c@foo.example.com</user>
    </users>
    <moderators>
      <moderator>sip:user_a@foo.example.com</moderator>
    </moderators>
  </floor>
</floorCreated>
```

Figure 1: floorCreated event example

### 4.2 floorRemoved event

When a floor is removed for some set of resources, the conference server SHOULD send a notification to the parties interested in this event.

The XML document for floorRemoved event starts with a 'floorRemoved' tag. Within the tag, there are zero or more 'resources' tag. If the resources tag is not provided, it means the floor for all the resources are removed. The following XML schema fragment defines the floorRemoved event.

```
<xs:element name="floorRemoved">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="resources" type="resourcesType"
                  minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
 </xs:element>
```

Figure 2 shows an example of the floorRemoved event.

```
<floorRemoved>
  <resources>
    <resource>mid:1</resource>
  </resources>
</floorRemoved>
```

Figure 2: floorRemoved event example

### 4.3 configChanged event

When the configuration of the floor is changed, the conference server SHOULD send a notification to the parties interested in this event. The event contains the updated floor information. The following XML schema fragment defines the configChanged event.

```
<xs:element name="configChanged">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="floor" type="floorType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 3 shows an example of the configChanged event.

### 4.4 floorChanged event

If the holders of one or more floors have changed, the conference server SHOULD send a notification to the parties interested in this event. At the same time, the conference server SHOULD send a re-INVITE to the new holders to enable the sending of media. The UA SHOULD not send media until it has received a floorChanged event. The following XML schema fragment defines the floorChanged event. The element 'holding' contains the new holders' information.

```
<xs:element name="floorChanged">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="holding" type="holdingType" maxOccurs="unbounded"/>
```

```
   <configChanged>
     <floor maxHolders="1">
       <resources>
         <resource>mid:1</resource>
       </resources>
       <users url="http://foo.example.com/conference1/user.html" />
       <moderators>
         <moderator>sip:user_a@foo.example.com</moderator>
       </moderators>
     </floor>
   </configChanged>
```

Figure 3: configChanged event example

```
       </xs:sequence>
     </xs:complexType>
   </xs:element>
```

Figure 4 shows an example of the floorChanged event.

```
   <?xml version="1.0" encoding="UTF-8"?>
   <floorChanged>
     <holding expiration="2003-01-12T09:40:47-05:00"
timestamp="2003-01-12T09:30:47-05:00">
       <resources>
         <resource>mid:1</resource>
       </resources>
       <users>
         <user>sip:user_a@foo.example.com</user>
       </users>
     </holding>
   </floorChanged>
```

Figure 4: floorChanged event example

### 4.5 queueChanged event

The conference server SHOULD send a notification to the parties interested in the queueChanged event when the claim queue changes, e.g., a new claim is added in or an existing claim is removed,

The queueChanged event contains the updated claim queue. The required attribute 'timestamp' defines when the event happens. The optional attribute 'url' provides the web URL having the updated claim queue. If the 'url' attribute is not provided, there MUST be one or more claims presented in the queueChanged tag. The following XML schema fragment defines the queueChanged event.

```
<xs:element name="queueChanged">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="claims" type="claimsType"
                  minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="timestamp" type="xsd:dateTime"/>
    <xs:attribute name="url" type="xsd:string"/>
  </xs:complexType>
</xs:element>
```

Figure 5 shows an example of the queueChanged event.

## 5 Floor control commands

Table 2 shows the floor control commands. Floor control commands are encapsulated in SOAP and are sent from the user to the conference server in order to change floor status.

In Table 2, the 'Server' refers to 'Conference server' and the 'User' refers to either 'Moderator' or 'Participant'.

### 5.1 CreateFloor command

The CreateFloor command creates a floor for some resources and users. Only moderators can execute this command.

boolean CreateFloor(floorType floor)

The CreateFloor command takes one parameter, floor, to create a new floor for some resources. The parameter 'floor' has the type 'floorType'. The response of the method is a boolean value indicating whether the floor is successfully created or not. The following XML schema fragment defines the CreateFloor command and the response of the command.

```
<queueChanged timestamp="2003-01-17T09:40:00-05:00">
  <claims>
    <claim claimID="1:sip:user_a@foo.example.com"
           timestamp="2003-01-12T09:30:00-05:00">
      <user>sip:user_a@foo.example.com</user>
      <resources>
        <resource>mid:1</resource>
      </resources>
      <subject>SIP conferencing</subject>
      <holdingTime>PT5M</holdingTime>
    </claim>
    <claim claimID="1:sip:user_b@foo.example.com"
           timestamp="2003-01-12T09:35:00-05:00">
      <user>sip:user_b@foo.example.com</user>
      <resources>
        <resource>mid:1</resource>
      </resources>
      <subject>Floor control</subject>
      <holdingTime>PT10M</holdingTime>
    </claim>
  </claims>
</queueChanged>
```

Figure 5: queueChanged event example

| Command name | Description | Issuer -> Receiver |
|---|---|---|
| CreateFloor | Create a floor for some resources and participants. | Moderator -> Server |
| RemoveFloor | Remove floors for some resources. | Moderator -> Server |
| ChangeConfig | Change the configuration of a floor | Moderator -> Server |
| ClaimFloor | Request a floor | User -> Server |
| ReleaseFloor | Give up a floor | User -> Server |
| GrantFloor | Grant a floor to some users | Moderator -> Server |
| RevokeFloor | Force release floors from some users | Moderator -> Server |
| RemoveClaims | Remove some existing floor claims | User -> Server |
| ReorderClaims | Reorder some claims in the queue | Moderator -> Server |

Table 2: Floor control commands

```
<xs:element name="CreateFloor">
  <xs:complexType name="CreateFloor">
    <xs:sequence>
      <xs:element name="floor" type="floorType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>



<xs:element name="CreateFloorResponse">
  <xs:simpleType>
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 6 shows an example of using SOAP to carry the CreateFloor command and the response of the CreateFloor command.

## 5.2 RemoveFloor command

The RemoveFloor command deletes floors for several resources. Only moderators can execute this command.

boolean RemoveFloor(resourcesType resources)

The RemoveFloor command takes one parameter, resources, which has the type resoursesType. The response of the method is a boolean value indicating whether the floor has been successfully removed or not. The following XML schema fragment defines the RemoveFloor command and the response of the command.

```
<xs:element name="RemoveFloor">
  <xs:complexType name="RemoveFloor">
    <xs:sequence>
      <xs:element name="resources" type="resourcesType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:CreateFloor xmlns:m="Some-URI">
         <floor max_holders=2>
           <resources>
             <resource>mid:1</resource>
             <resource>mid:2</resource>
           </resources>
           <moderators>
             <moderator>sip:foo@examples.com</moderator>
           </moderators>
         </floor>
       </m:CreateFloor>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
      <m:CreateFloorResponse xmlns:m="Some-URI">
        true
      </m:CreateFloorResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 6: Use SOAP to encapsulate CreateFloor command

```
<xs:element name="RemoveFloorResponse">
  <xs:simpleType name="RemoveFloorRespones">
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 7 shows an example of using SOAP to carry the RemoveFloor command.

### 5.3 ChangeConfig command

The ChangeConfig command changes a floor's configuration. Only

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:RemoveFloor xmlns:m="Some-URI">
         <resources>
           <resource>mid:1</resource>
           <resource>mid:2</resource>
         </resources>
       </m:RemoveFloor>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 7: Use SOAP to encapsulate RemoveFloor command

moderators can execute this command.

boolean ChangeConfig(floorType floor)

The parameters for the ChangeConfig command is the same as that for the CreateFloor command. The response indicates whether the configuration is successfully changed or not. The following XML schema fragment defines the ChangeConfig command and the response of the command.

```
<xs:element name="ChangeConfig">
  <xs:complexType name="ChangeConfig">
    <xs:sequence>
      <xs:element name="floor" type="floorType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="ChangeConfigResponse">
  <xs:simpleType name="ChangeConfigResponse">
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 8 shows an example of using SOAP to carry the ChangeConfig command.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:ChangeConfig xmlns:m="Some-URI">
         <floor max_holders=2>
           <resources>
             <resource>mid:1</resource>
             <resource>mid:2</resource>
           </resources>
           <moderators>
             <moderator>sip:foo@examples.com</moderator>
           </moderators>
         </floor>
       </m:ChangeConfig>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 8: Use SOAP to encapsulate ChangeConfig command

## 5.4 ClaimFloor command

When a user wants to request a floor, the user's UA SHOULD send a ClaimFloor command to the conference server. The holder of a floor can also use ClaimFloor command to extend the holding time. To ask for the extension, the new claim MUST have the same claimID as the granted claim which enables the current holding.

boolean ClaimFloor(claimsType claims)

For new claims, the response of the method is a boolean value indicating whether the claim has been successfully put into the claim queue. For updating an existing claim, the response indicates whether the existing claim get updated successfully. The following XML schema fragment defines the ClaimFloor command and the response of the command.

```
<xs:element name="ClaimFloor">
```

```
    <xs:complexType name="ClaimFloor">
      <xs:sequence>
        <xs:element name="claims" type="claimsType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>



  <xs:element name="ClaimFloorResponse">
    <xs:simpleType name="ClaimFloorResponse">
      <xs:restriction base="xs:boolean"/>
    </xs:simpleType>
  </xs:element>
```

Figure 9 shows an example of using SOAP to carry the ClaimFloor command.

### 5.5 ReleaseFloor event

To release a floor, the user's UA can send a ReleaseFloor command to the conference server. The ReleaseFloor command takes one parameter, holding, which has the type holdingType. The sender of the command SHOULD be the same as the sub-element 'user' of the holding parameter.

```
boolean ReleaseFloor (holdingType holding)
```

The following XML schema fragment defines the ReleaseFloor command.

```
  <xs:element name="ReleaseFloor">
    <xs:complexType name="ReleaseFloor">
      <xs:sequence>
        <xs:element name="holding" type="holdingType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>



  <xs:element name="ReleaseFloorResponse">
    <xs:simpleType name="ReleaseFloorResponse">
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:ClaimFloor xmlns:m="Some-URI">
         <claims>
           <claim claimID="1:sip:foo@examples.com"
                  timestamp="2003-01-14T09:30:47-05:00"
                  expiration="2003-01-14T09:40:47-05:00"
                  priority=emergency>
             <user>sip:foo@examples.com</user>
             <resources>
               <resource>mid:1</resource>
               <resource>mid:2</resource>
             </resources>
             <subject>The auditorium is on fire!</subject>
             <holdingTime>P30S</holdingTime>
           </claim>
         </claims>
       </m:ClaimFloor>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 9: Use SOAP to encapsulate ClaimFloor command

```
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 10 shows an example of using SOAP to carry the ReleaseFloor command.

### 5.6 GrantFloor command

The GrantFloor command grants floors to one or more users. The parameter 'holding', which has the type 'holdingType', defines the relationship between the floors and the holders. The parameter 'claim', which has the type 'claimType', specifies the claim that the floor is granted for. The claim MUST be removed from the queue. If the 'claim' parameter is not provided, the GrantFloor command does not affect the claim queue. Only moderators can execute this command.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:floor_release xmlns:m="Some-URI">
         <holding>
           <resources>
             <resource>mid:1</resource>
             <resource>mid:2</resource>
           </resources>
           <users>
             <user>sip:foo@examples.com</user>
           </users>
         </holding>
       </m:floor_release>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 10: Use SOAP to encapsulate ReleaseFloor command

```
boolean GrantFloor(holdingType holding, claimType claim)
```

The following XML schema fragment defines the GrantFloor command.

```
<xs:element name="GrantFloor">
  <xs:complexType name="GrantFloor">
    <xs:sequence>
      <xs:element name="holding" type="holdingType"/>
      <xs:element name="claim" type="claimType" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="GrantFloorResponse">
  <xs:simpleType name="GrantFloorResponse">
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 11 shows an example of using SOAP to carry the GrantFloor command.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:GrantFloor xmlns:m="Some-URI">
         <holding expiration="Sat Apr 6 11:53:24 EST 2002">
           <resources>
             <resource>mid:1</resource>
             <resource>mid:2</resource>
           </resources>
           <users>
             <user>sip:foo@examples.com</user>
           </users>
         </holding>
       </m:GrantFloor>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 11: Use SOAP to encapsulate GrantFloor command

## 5.7 RevokeFloor command

The RevokeFloor command forces the release of a floor from the current holders.

boolean RevokeFloor(holdingType holding)

The parameter holding indicates which floor should be revoked. Only moderators can execute this command.

The following XML schema fragment defines the RevokeFloor command.

```
<xs:element name="RevokeFloor">
  <xs:complexType name="RevokeFloor">
    <xs:sequence>
      <xs:element name="holding" type="holdingType"/>
    </xs:sequence>
  </xs:complexType>
```

```
</xs:element>



<xs:element name="RevokeFloorResponse">
  <xs:simpleType name="RevokeFloorResponse">
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 12 shows an example of using SOAP to carry the RevokeFloor command.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:RevokeFloor xmlns:m="Some-URI">
         <holding>
           <resources>
             <resource>mid:1</resource>
             <resource>mid:2</resource>
           </resources>
           <users>
             <user>sip:foo@examples.com</user>
           </users>
         </holding>
       </m:RevokeFloor>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 12: Use SOAP to encapsulate RevokeFloor command

## 5.8 RemoveClaims command

The RemoveClaims command removes several claims from the claim queue. Moderators can remove any claims. Participants can only remove their own claims.

```
boolean RemoveClaims(claimsType claims)
```

RemoveClaims command takes one parameter, claims. The return value indicates whether the claims have been removed successfully. The following XML schema fragment defines the RemoveClaims command.

```
<xs:element name="RemoveClaims">
  <xs:complexType name="RemoveClaims">
    <xs:sequence>
      <xs:element name="claims" type="claimsType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="RemoveClaimsResponse">
  <xs:simpleType name="RemoveClaimsResponse">
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 13 shows an example of using SOAP to carry the RemoveClaims command.

## 5.9 ReorderClaims command

The ReorderClaims command changes the order of the claims in the queue. Only moderators can execute this command.

This command supports some simple operations to change a single claim's position. It takes three parameters. The parameter 'resources' indicates the claim queue to operate. The parameter 'claim' indicates which claim to move. The parameter 'operation' defines how to move the claim.

```
boolean ReorderClaims(resourcesType resources,
                      claimType claim,
                      operationType operation)
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:RemoveClaims xmlns:m="Some-URI">
         <claims>
           <claim claimID="1:sip:foo@examples.com">
             <user>sip:foo@examples.com</user>
           </claim>
         </claims>
       </m:RemoveClaims>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 13: Use SOAP to encapsulate RemoveClaims command

The following XML schema fragment defines the RemoveClaims command.

```
<xs:element name="ReorderClaims">
  <xs:complexType name="ReorderClaims">
    <xs:sequence>
      <xs:element name="resources" type="resourcesType"/>
      <xs:element name="claim" type="claimType"/>
      <xs:element name="operation" type="operationType"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="ReorderClaimsResponse">
  <xs:simpleType name="ReorderClaimsResponse">
    <xs:restriction base="xs:boolean"/>
  </xs:simpleType>
</xs:element>
```

Figure 14 shows an example of using SOAP to carry the RemoveClaims command.

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
       <m:ReorderClaims xmlns:m="Some-URI">
         <claim claimID="1:sip:foo@examples.com">
           <user>sip:foo@examples.com</user>
         </claim>
         <operation timestamp="2003-01-14T09:30:47-05:00">
           <operator>up</operator>
           <argument>2</argument>
         </operation>
       </m:ReorderClaims>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure 14: Use SOAP to encapsulate ReorderClaims command

## 6 Floor Control Policies

Each resource SHOULD have its own floor claim queue so that people interested in one resource will not get notified by the other resource's claim. However, if multiple resources need to be granted in atomic mode, e.g., access to all resources is granted or denied as a group, the conference server MUST use one floor claim queue for all the resources. The floor claim queue is created when executing the CreateFloor command. The parameter of the command defines the resources the floor applys.

When a conference server receives a ClaimFloor command, the conference server SHOULD append the new claims at the end of the queue. If the current floor holder releases the floor, the claim at the front of the queue SHOULD automatically get the floor. The fulfilled claims MUST be removed from the claim queue.

In one claim request, a user may claim multiple resources in different floor claim queues. The claim will be appended to all the applicable queues. To avoid potential deadlock, the claims in different queues MUST be granted independently.

When a conference server receives a GrantFloor command, the conference server SHOULD queue the grant until there is an available

floor. Occupied floor can be released by ReleaseFloor and RevokeFloor commands.

A floor creator can specify some complicated floor control policies, for example, "senior members get 5 minutes to talk, junior members 2 minutes and visitors 1 minute". There should be a floor control language to describe the policies and the wildcard 'xs:any' in the floorType can carry the policies to the conference server.

## 7 Security consideration

Conference server SHOULD use appropriate authentication to ensure the commands and events originated from trusted parties. Other SIP considerations apply [1].

## 8 IANA considerations

### 8.1 New semantics for the "group" SDP Attribute

Name being registered: FL

Long-form name in English: Floor Control

Type of name: Semantics for the "group" SDP Attribute

Purpose: Group medias for floor control

Reference: RFCxxxx (this document)

Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>

### 8.2 URN Sub-Namespace Registration

This section registers a new XML namespace, as per the guidelines in [9]

URI: urn:ietf:params:xml:ns:floor-control

Registrant Contact: Xiaotao Wu <xiaotaow@cs.columbia.edu>

XML:

```
BEGIN
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
          "http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
  <meta http-equiv="content-type"
     content="text/html;charset=iso-8859-1"/>
  <title>Conference Floor Control Namespace</title>
</head>
<body>
  <h1>Namespace for Conference Floor Control</h1>
  <h2>application/floorcontrol+xml</h2>
  <p>See <a href="[[[URL of published RFC]]]">RFCXXXX</a>.</p>
</body>
</html>
END
```

# 9 Call flows

## 9.1 A user joins the conference and gets a floor

Figure 15 shows the call flow when a user joins a moderated conference, claims for a floor and gets the floor. In the call flow, initially, the participant first sends an INVITE to the conference server to join the conference. Since the participant does not know whether the conference is moderated or not, there is no m=control line in the initial INVITE. In the 200 response from the conference server, the conference server uses a=type:moderated line to mute all the media tools of the participant. The conference server then sends a re-INVITE with m=control lines to establish floor control channels. The participants can then send SUBSCRIBE for floor control events and uses floor control channels for floor control commands. When the participant gets the floor, the conference server notifies all the users about the floor change and send a re-INVITE to unmute the media tools of the participant.

# 10 Changes from Earlier Version

## 10.1 Changes from Draft -03

- Add IANA considerations for new semantics for "group" SDP attribute and URN Sub-Namespace registration

## 10.2 Changes from Draft -02

- Change the title from "Use of SIP and SOAP for Conference Floor Control" to "Use of Session Initiation Protocol (SIP) and Simple Object Access

Protocol (SOAP) for Conference Floor Control".
- If only part of the media are moderated, the 'a=type:moderated' line MUST NOT be used. The grouping of media channels and control channels can indicate which media are moderated.
- Fix the lost part at the bottom of page 5.
- Add ACK to Figure 15.
- Explicitly describe the floor claim queue model in section 1 and section 3.6
- Explain the reason to put floor control channel information in SDP.
- Explain how to use Allow-Events header for floor status notification.
- The m=control line can only be initiated from conference server's INVITE message.
- Add the missing 's' line in the SDP example in Section 2.2.
- Add "a=floorcontrol:" attribute in SDP for control URL information.
- Add re-INVITE to Figure 15.
- Add xs:any wildcard in floorType for additional information of a floor.
- In claimType, the claimID MUST be globally unique. Define the way of updating an existing claim and extending the current floor holding time.
- Add priority attribute to claimType.
- Add floor control event examples.
- Separate the references into normative and informative.
- Clarified some wording.
- Fixed some typographical errors.

### 10.3 Changes from Draft -01

- Reorganize section 2 into three subsections for clearer description of using SIP and SOAP for floor control
- Provide namespace definitation for the elements defined by this specification
- If the 'moderators' element is not provided in a floor, in floor control command, the person creates the floor will serve as the moderator, in floor control events, it means the information of moderators is hidden by conference server
- Clarify that the whole string of the value in 'a=mid' line is used as the value for 'resourceType'
- Clarify that for 'usersType', if the attribute 'url' is not provided, the list of the users MUST be provided.
- For 'holdingType', if the 'users' element is not provided, it means no user is holding the floor. Previously, it defined as 'all users holding the floor'.
- Remove 'maxOccurs=1' in schema fragment because by default, maxOccurs is 1
- In 'claimType', change the element 'resource' to 'resources', and the type 'resourceType' to 'resourcesType'.

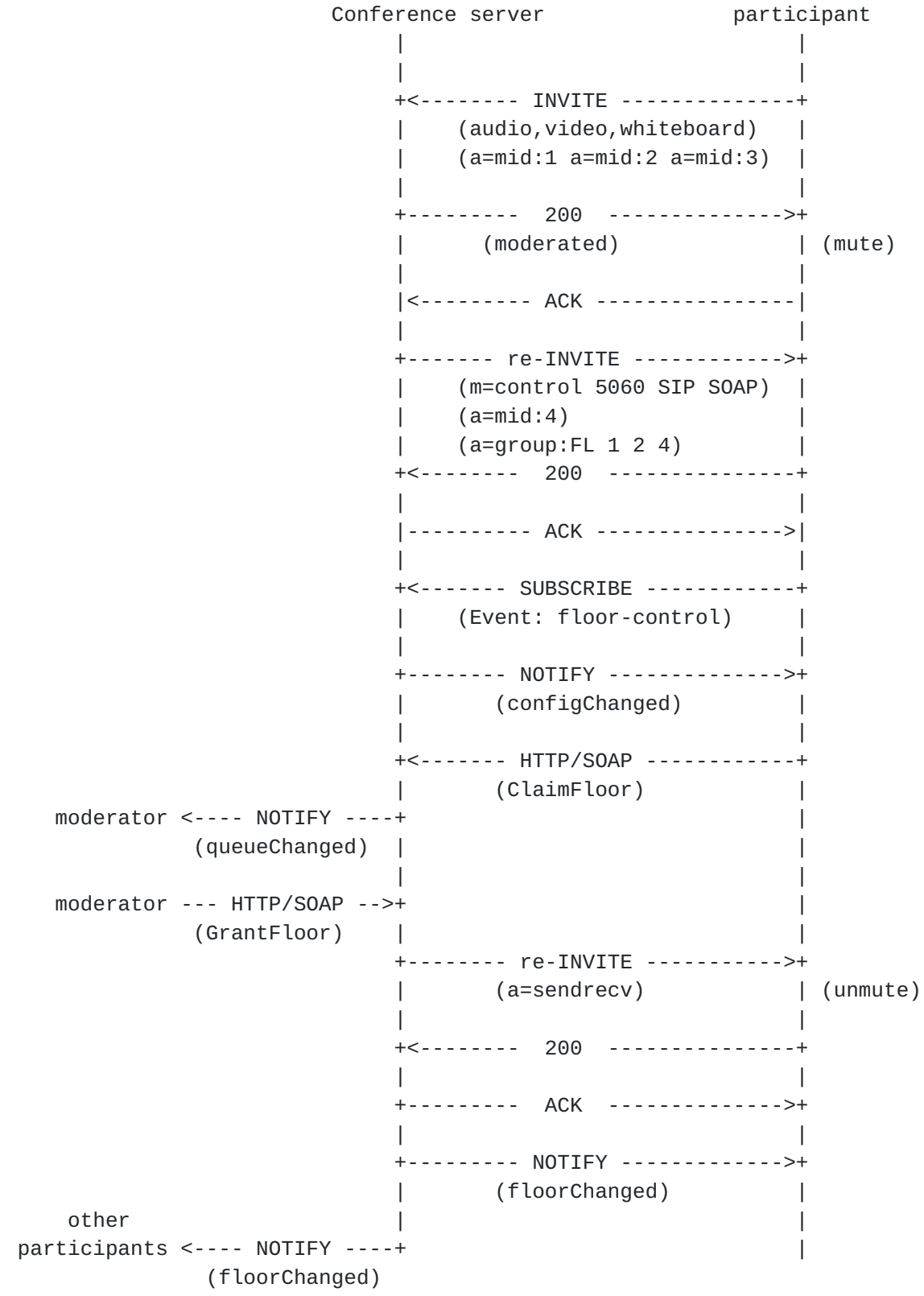
Conference server
participant
INVITE
(audio,video,whiteboard)
(a=mid:1 a=mid:2 a=mid:3)
200
(moderated)
(mute)
ACK
re-INVITE
(m=control 5060 SIP SOAP)
(a=mid:4)
(a=group:FL 1 2 4)
200
ACK
SUBSCRIBE
(Event: floor-control)
NOTIFY
(configChanged)
HTTP/SOAP
(ClaimFloor)
moderator <---- NOTIFY ----
(queueChanged)
moderator --- HTTP/SOAP -->
(GrantFloor)
re-INVITE
(a=sendrecv)
(unmute)
200
ACK
NOTIFY
(floorChanged)
other participants <---- NOTIFY ----
(floorChanged)

Figure 15: A user send INVITE to join the conference

- Add an example of the SOAP response of CreateFloor command
- Fix typo 'floor_remove' to 'RemoveFloor' in Figure 7

## 11 Authors' Addresses

Xiaotao Wu
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
electronic mail: xiaotaow@cs.columbia.edu

Petri Koskelainen
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
electronic mail: petkos@cs.columbia.edu electronic mail:
petri.koskelainen@nokia.com

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027
USA
electronic mail: schulzrinne@cs.columbia.edu

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

[3] W3C, "Simple object access protocol (soap) 1.1."

[4] S. Bradner, "Key words for use in rfcs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[5] M. Handley and V. Jacobson, "SDP: session description protocol," RFC 2327, Internet Engineering Task Force, Apr. 1998.

[6] G. Camarillo, G. Eriksson, J. Holler, and H. Schulzrinne, "Grouping of media lines in the session description protocol (SDP)," RFC 3388, Internet Engineering Task Force, Dec. 2002.

[7] R. Moats, "URN syntax," RFC 2141, Internet Engineering Task Force, May 1997.

[8] R. Moats, "A URN namespace for IETF documents," RFC 2648, Internet Engineering Task Force, Aug. 1999.

[9] M. Mealling, "The IETF XML registry," internet draft, Internet Engineering Task Force, July 2002. Work in progress.

[10] M. Murata, S. S. Laurent, and D. Kohn, "XML media types," RFC 3023, Internet Engineering Task Force, Jan. 2001.

## 13 Informative References

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

[12] R. Malpani and L. A. Rowe, "Floor control for large-scale mbone seminars," in
ACM Multimedia, (Seattle, Washington), Nov. 1997.

[13] J. Rosenberg and H. Schulzrinne, "Models for multi party conferencing in SIP," internet draft, Internet Engineering Task Force, July 2002. Work in progress.

[14] N. Deason, "SIP for SOAP sessions," internet draft, Internet Engineering Task Force, Apr. 2002. Work in progress.

[15] H. S. P. Koskelainen and X. Wu, "A sip-based conference control framework," in NOSSDAV, (Miami, Florida), May 2002.

[16] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (RTSP)," RFC 2326, Internet Engineering Task Force, Apr. 1998.

Table of Contents