SIP/SIMPLE-based Conference Room Management Method for the Voice Communication Medium "voiscape"

Yasusi Kanada
Central Research Laboratory, Hitachi, Ltd.
Higashi-Koigakubo 1-280, Kokubunji, Tokyo 185-8601, Japan
kanada@crl.hitachi.co.jp

1 Introduction

Teleconferencing systems, including audio-only and audio-and-video conferencing systems, enable conversation between three or more persons. In many teleconferencing systems, there are multiple virtual conference rooms; i.e., two or more independent conferences can be held simultaneously. Each conference room has different users and different resources that must be managed by the teleconferencing system.

We are developing a voice-centered communication medium called voiscape [Kan 04]. Voiscape creates an auditory virtual environment with spatial audio technologies. The sound environment thus created is close to a face-to-face conversation environment. People enter a 3-D space, which is shared among the people in that space, to communicate with each other. This space is called a sound room. Each person in a sound room is represented by a spatially located sound, and people can move freely (see **Figure 1**). If a person moves close to another person, a localized conversation can be naturally initiated. A voiscape system is an audio conferencing system and the sound rooms that belong to the system are conference rooms. The sound rooms must be managed by the voiscape system.

A voiscape system prototype called VPII (Voiscape Prototype II) has been developed [Kan 05]. Because the properties and states of rooms, users, and objects that must be managed in VPII can be regarded as "presence", VPII manages them by using the presence-related event-notification mechanism of SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) [Roa 02][Ros 04]. This mechanism enables a presence watcher to subscribe to a presence server, which notifies presence changes, and enables a presentity to "publish" its presence change to a presence server [Nie 04].

This paper explains the conference-room management functions of voiscape and describes their implementation using SIMPLE.

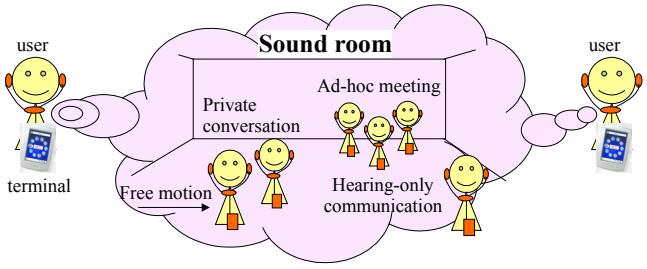

Figure 1. Sound-room concept

2 Outline of Communication Using Voiscape

The user interface of a user agent (UA, i.e., a terminal program) is illustrated in **Figure 2**. The user first selects a room to enter from the room list, shown on the left. The UA then displays a map of the sound room, shown on the right. The walls are displayed in gray. The scale of the map can be changed using radio buttons (or a slider). A unique icon can be used for each user.

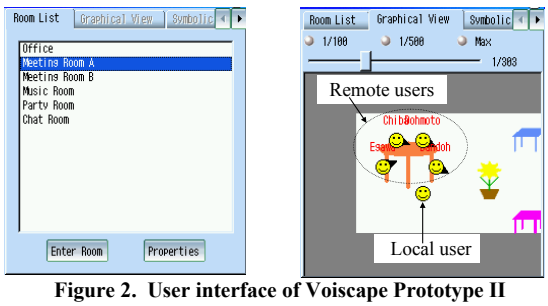

Figure 2. User interface of Voiscape Prototype II

A user moves one-foot forward by pushing the forward (arrow) key and one-foot backward by pushing the backward key. The user turns left (18 degrees) by pushing the left (arrow) key and right by pushing the right key. The user's icon is always displayed immediately below the screen center, and its orientation remains fixed. By moving, the user can select a person or persons in the room and talk with them, or can select a sound source in the room and hear them.

In the prototype, a Sharp Zaurus (a Linux-based PDA) or PCs with Microsoft Windows or Linux are used as the terminals. Qt middleware developed by TrollTech is used in the Zaurus to provide a light-weight window system and some additional functions such as XML parsing.

3 Room Management Functions

3.1 Room-list management

When a UA is activated, it requests the room list from a room-list manager (RLM), which then returns the requested list to the agent. The RLM stores the list in the room-list database. The RLM also accepts requests for creating or deleting a room and records them in the room database.

3.2 Room-property and object management

Room properties are managed by the room manager (RM). A sound room has many properties. It is not just an abstract conference room, but it is a 3-D virtual space and has sound-related properties. The virtual space properties include room size (e.g., width, depth, and height). The room size is used as a sound property and for the display. Another sound-related property is the reflection ratio of the walls. This is used with the room size for calculating early reflections (reverberations) [Kan 05].

A sound room can also contain silent objects such as tables or plants (see Figure 2). They can be used as visual *landmarks* from which users can distinguish their orientation in the room and places to meet with other users. The objects in the room have their locations and icons as their properties. A sound room can contain a speaker object that plays a sound file has its address (URL) as a property.

3.3 Room-user management

Room users are also managed by the RM. Two types of user management are required.

(1) *Membership management*: Every user can enter a public room, but users can enter a private room only if they are members of that room. The RM manages the membership and authorize users when they request room entry. The member properties include the URLs of user's visual and auditory icons if the user specifies icons that are not default. The visual icon is used in the map, and the auditory icon may be used when the user moves close to another user [Kan 05]. The member properties may include the default (virtual) locations and orientations of a member. The default location represents the location where he/she appears when he/she enters a room.

(2) *Current-user management*: A member can enter a room or exit from it. The RM has a list of users currently in the room. This list contains not only the identifiers of the users but also contains their current virtual locations and orientations. These properties are updated when the users move or turn around.

3.4 Policy management

Communication between room users and that from a speaker to a room user can be controlled by policies [Kan 04], which are also managed by the RM. Policies can be selected by the user and the UA sends them to the RM.

4 Room Management Using SIMPLE

4.1 Outline

The properties and states of rooms, users, and objects that must be managed in VPII can be regarded as "presence". VPII thus manages them by using the presence-related event notification mechanism of SIMPLE. SIMPLE is an extension of SIP (Session Initiation Protocol). Two request messages, i.e., SUBSCRIBE and NOTIFY, are added in this extension, and PUBLISH request is added by another document [Nie 04]. In VPII, the user agent (UA) sends a PUBLISH request to the RM or RLM when a presence changes. The RM or RLM stores the presence in a database and sends the updated presence by using a NOTIFY request. The UA requests the room-presence infor-

mation, which includes the presence of users and objects in the room, to the RM by sending a SUBSCRIBE request.

User presence is expressed using an extended PIDF (Presence Information Data Format [Sug 04]). Presence is usually regarded as a status and is thus indicated by `status` tags. While a status can be changed easily, presence, in a general sense, contains properties that are not easily changed. Although it is not necessary to propagate unchanged properties every time a presence message produced by a status change is sent, they must be propagated the first time the user or object appears. Such properties should be distinguished from the status. However, PIDF has no tag for properties. A new tag, `vs:property`, was thus introduced. This tag includes new tags such as `vs:type` and `vs:room-size`. The former indicates the type of entity, and the latter indicates the coordinates of the object.

TCP, instead of UDP, is used for transmitting a presence message, because the message size is usually larger than the MTU of Ethernet. SIMPLE and PIDF are computationally quite heavy, so if the status is updated frequently, presence propagation requires much computational resource. This problem and a solution are explained in Section 4.4.

4.2 Room-list management

In addition to room users and objects, a room list is also regarded as "presence" in VPII, and it is requested and sent by using SIMPLE. If the UA requests one-time notification, the effect is similar to a request-response protocol such as HTTP. However, if the UA requests subsequent notifications, the RLM reports room additions, changes, and deletions to the UA.

The following part of the PIDF document describes the presence of a sound room.

```
<tuple id="Office@serverdomain">
 <nickname>Office</nickname>
 <contact>sip:Office@1.2.3.4:5060</contact>
 <status><basic>open</basic></status>
 <vs:property><vs:type>room</vs:type>
  <vs:room-size>50,30,5</vs:room-size>
 </vs:property></tuple>
```

The identifier (SIP URI) of this room is `Office@serverdomain` and its short name is `Office`. The contact address is the IP address and port of the RM. This enables direct access to the RM and avoids high load of SIP proxies and the delay they cause. If the IP address or port of the RM is changed, UAs are informed about this change by sending a room-list notification message. The type of this entity is room, and its size is $50 \times 30 \times 5$ m.

A room creation or deletion can be ordered by a UA by using a PUBLISH request. If the user is allowed to create a room with the specified properties, a new room is created and its URI becomes ready to receive messages such as INVITE, BYE, or SUBSCRIBE. The creating user is the owner of the room. Because this event notification mechanism is based on a soft-state approach, i.e., the effects of SUBSCRIBE, NOTIFY, and PUBLISH requests disappear if these messages expire, the created room is kept alive only while the RM continues receiving PUBLISH requests from the owner. This means that the room lifecycle is managed by a soft-state approach. However, the interval of the PUBLISH requests can be a month, a year, or longer. This approach is applied to users and objects too.

Currently, the RLM and RM are integrated into a single machine. The room-creation and deletion requests have not yet been implemented. The rooms are thus permanent and, at present, can be added or removed only by the system administrator.

4.3 Room-user and object management

Both current room users and objects in the room are regarded as parts of the room presence. A room PIDF document thus contains both. An object can be added to or removed from a room by a PUBLISH request. A room user can enter a room by sending a PUBLISH request to the room, and the user can exit from the room by sending an un-PUBLISH request to it in VPII. When entering or exiting a room, INVITE and BYE requests are used for opening or closing a voice stream between the UA and the media server. The UA sends a SUBSCRIBE request concerning room presence changes to the room URI, and they are reported to users by sending NOTIFY requests. The NOTIFY requests contain PIDF documents.

The following part of PIDF document (a tuple) describes the presence of a user.

```
<tuple id="George@userdomain">
 <nickname>George</nickname>
 <icon>http://domain/icons/George.bmp</icon>
 <vs:auditory-icon>http://domain/auditory-icons/Geor
ge.wav</vs:auditory-icon>
 <status><basic>open</basic>
  <vs:location>10,5,0</vs:location></status>
 <vs:property><vs:type>human</vs:type></vs:property>
</tuple>
```

This tuple describes the user's properties and status. The 2-D and auditory icons of George are specified by their URLs. The status includes George's location and orientation.

A NOTIFY request must contain a list of all the users and objects even if there is no presence change [Roa 02]. This means that if the number of users and/or objects is large, the message size becomes very large. In addition, if a user (or users) moves very frequently, notification requests are sent very frequently. The overheads of transmission, analysis, and creation of such a message is thus very large. This problem has been solved by the method described in the next subsection.

Currently, objects cannot be added/removed by using SIMPLE messaging but only by the system administrator, and room membership management functions are not yet implemented.

4.4 Partial notification of users and objects

To reduce the overhead of communication and computation, a partial notification mechanism [Lon 04a][Lon 04b] is used. When the presence is updated, only the difference from the previously reported presence is sent to the UAs by using a partial PIDF document; i.e., it does not contain unchanged parts. If there is only a small change, the document size is much smaller than the full notification message. For example, an update of user location and orientation is notified by the following text.

```
<?xml version="1.0" encoding="UTF-8"?>
<pidf-diff xmlns="urn:ietf:params:xml:ns:pidf"
 xmlns:vs="urn:ietf:params:xml:ns:virtual-space"
 entity="pres:meetingA@serverdomain" version="2">
 <replace sel="presence/tuple[id=&quot;Listener51&
quot;]/status/vs:location/text()">5,4,0</replace>
</pidf-diff>
```

Presence changes always generate partial notification messages. However, full notification messages are also sent to a UA when it sends a SUBSCRIBE request to the RM. Because SUBSCRIBE requests are sent periodically, full notification messages are also sent periodically.

Although the size of partial notification message is small, RFC 3856 [Ros 04] states that the minimum interval for presence notification should be five seconds. Thus, if a presence change such as user motion is quick, the system cannot track the change properly. In the current VPII, the interval is set to two seconds. However, this response time is still not sufficient for some applications. A protocol different from SIP/SIMPLE, such as RTP (Real-time Transport Protocol), should be used for user location and orientation notification in such applications.

Message sizes and corresponding processing times were measured in the case of a simulated conference of five people with four objects. The average TCP message size of a full notification was 2200 bytes, and that of a partial notification was 680-1150 bytes (with 1-5 replace tags). The message size was thus reduced by 48-69%. The average elapsed time from when a Zaurus UA receives a NOTIFY request to when an 200 OK response is received for a full notification was 130 ms, and that for a partial notification was 17-31 ms. Processing time was thereby reduced by 66-87%. This reduction ratio is smaller than the size reduction ratio because the XML part of the message, which requires much more parsing time, is much smaller.

5 Conclusion

In VPII, SIMPLE was used for all the user, object, room, and room list operations between an RMS and UAs. This makes the room management natural and light-weight in terms of computational and programming resources. To reduce the messaging overhead, the partial notification mechanism of SIMPLE is used. This significantly reduced SIP message sizes and processing time in a simulated meeting.

References

[Kan 04] Kanada, Y., "Multi-Context Voice Communication Controlled by using an Auditory Virtual Space", *2nd Int'l Conf. on Communication and Computer Networks* (*CCN 2004*), pp. 467-472, 2004.

[Kan 05] Kanada, Y., "Multi-Context Voice Communication In A SIP/SIMPLE-Based Shared Virtual Sound Room With Early Reflections", *NOSSDAV 2005*, pp. 45-50, 2005.

[Lon 04a]Lonnfors, M., Costa-Requena, J., Leppanen, E., and Khartabil, H., "Partial Notification of Presence Information", Work in progress, IETF.

[Lon 04b]Lonnfors, M., Leppanen, E., and Khartabil, H., "Presence Information Data Format (PIDF) Extension for Partial Presence", Work in progress, IETF.

[Nie 04] Niemi, A., Ed., "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, IETF, October 2004.

[Roa 02] Roach, A. B., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 2543, IETF, June 2002.

[Ros 04] Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)", RFC 3856, IETF, August 2004.

[Sug 04] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and Peterson, J., "Presence Information Data Format (PIDF)", RFC 3863, IETF, August 2004