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 Outline

- **General requirements: management of conference-rooms**

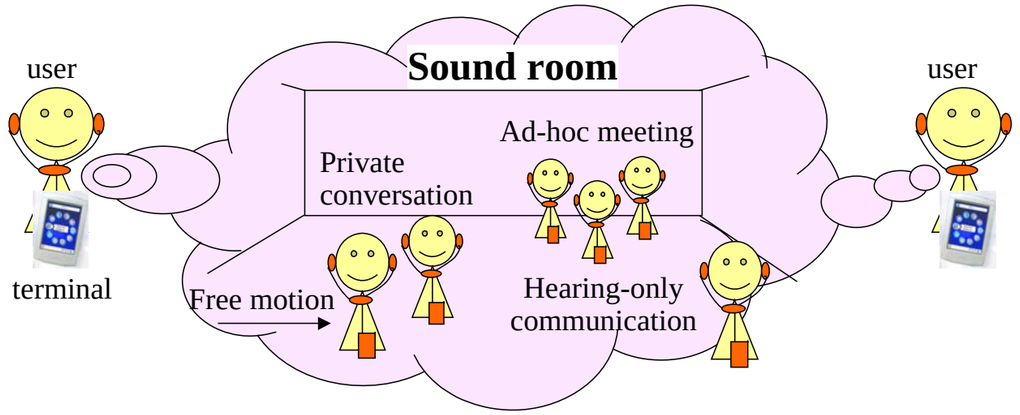
  We developed a type of teleconferencing systems called voiscape – a voice-centered communication medium with “sound rooms” (spatial audio based virtual spaces).

  - In voiscape, conferences (conference rooms) must be managed in a similar method to other conferencing systems.
  - The management functions include creation, deletion, and modification of a room, addition and deletion of users to a room, etc.

- **Our special requirements: auditory virtual space management**

  Voiscape is a voice-centered communication medium with “sound rooms” (i.e., spatial audio based virtual spaces). People freely enter a room and navigate around there.

  - Sound rooms (as virtual spaces) and their properties, such as room width or depth must be managed.
  - Presence and motion of users and objects in sound rooms must be managed – users can quickly move in the room.
  - Communication policies, such as connection/disconnection distance policies, must be managed.

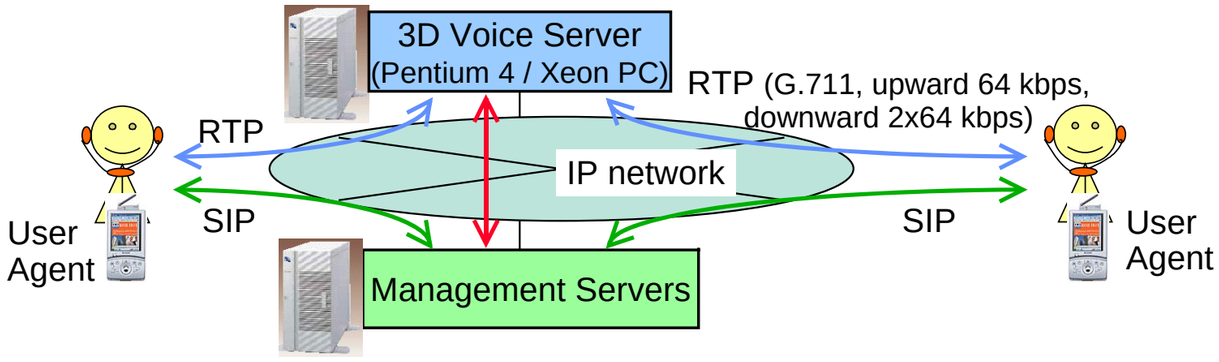


Sound room (auditory virtual space)

- **Our solution in VPII: SIP and SIMPLE based unified management**

  VPII (Voiscape Prototype II) is the second prototype of voiscape.

  - Every information to be managed is communicated by using SIP and SIMPLE* (SIP for Instant Messaging and Presence Leveraging Extensions), i.e., using presence-related event notification mechanism.

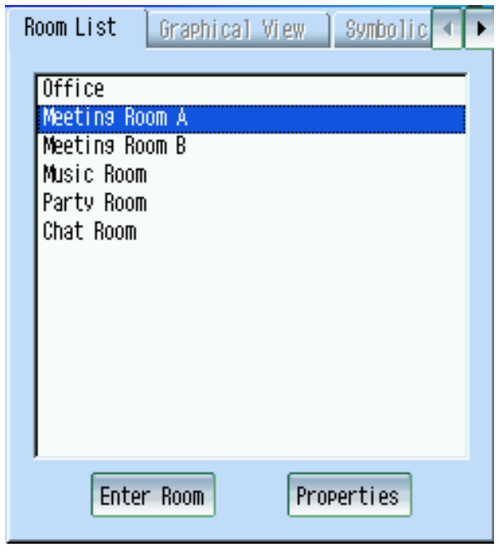


VPII System Structure

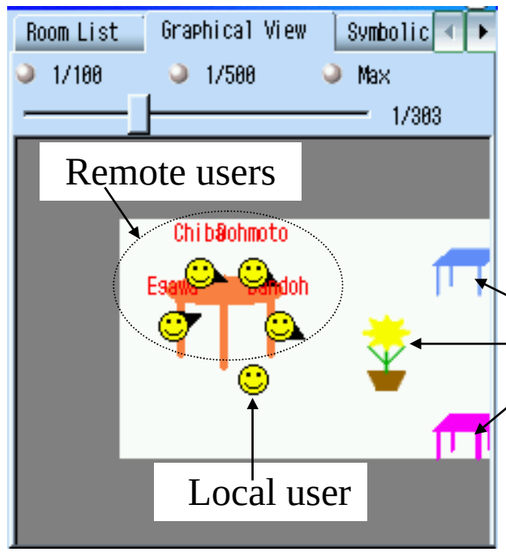
2 The user interface: map-based virtual space navigation

The sequence of user operations are as follows.

- 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 user can move in the room by using cursor keys or other pointing devices.
- 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.



Room list window



Map window

Virtual space navigation

3 Room Management Functions

- **Room-list management**
  A room-list manager (RLM) manages the room list by using a database manager.
  - When a UA is activated, it requests the room list from thr RLM.
  - The RLM also accepts requests for creating or deleting a room.
- **Room-property and object management**
  A room manager (RM) manages the room properties and objects in the room by using the database manager.
  - *Room property management*: Such as *room size* (e.g., width, depth, etc.) or the *reflection ratio* of the walls are managed.
  - *Room size*: used as a sound property and for the display.
  - *Reflection ratio*: used with the room size for calculating early reflections (reverberations).
  - *Silent objects*: objects such as tables or plants can be used as visual *landmarks* from which users can distinguish their orientation in the room and places to meet with other users.
  - *Speakers* that play a sound files have their addresses (URLs) as properties.
- **Room-user management**
  Room users are also managed by the RM.
  - *Membership management*: If the room is private, the membership must be managed. The RM manages the membership and authorize users when they request room entry.
  - *Current-user management*: 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.
  - *Member property management*: the URLs of user's visual and auditory icons if the user specifies non-default ones, and the default (virtual) locations and orientations of a member. This can be part of membership management.
- **Policy management**
  Communication between room users and that from a speaker to a room user can be controlled by policies. Policies can be selected by the user and the UA sends them to the RM.

4 Room Management Using SIMPLE

4.1 Outline

- **Presence event notification mechanism is used.**
  We use SIP and SIMPLE, which is an extension of SIP (Session Initiation Protocol).
  - SUBSCRIBE, NOTIFY, and PUBLISH requests are used.
- **Why a presence mechanism is used?**
  Because the properties and states of rooms, users, and objects that must be managed in VPII can be regarded as “presence”.
- **Representation of room, user, and object presence**
  - User presence is expressed using an extended PIDF (Presence Information Data Format).
  - Tags added to the standard PDF: `vs:property`, `vs:type`, `vs:room-size`, etc.
- **Use of TCP**
  - TCP, instead of UDP, is used for transmitting a presence message, because the message size is usually larger than the MTU of Ethernet.

4.2 Room-list management

SIP/SIMPLE is also used for room list delivery, although HTTP could be used.

- *Principle*: Each room is represented by an XML (PIDF) expression and room-related operations are transmitted by using SIP/SIMPLE.
- *An advantage of SIP compared to HTTP*: room list change (room addition, deletion, etc.) and room property change can be pushed to UAs.
- *Room creation/deletion by users*: A room creation or deletion can be ordered through a UA by using a PUBLISH request. The creating user is the owner of the room. (not yet implemented)
- *Soft-state approach*: The created room is kept alive only while the RM continues receiving PUBLISH requests from the owner. The interval of the PUBLISH requests can be a month, a year, or longer. (not yet implemented)

Example: A PIDF expression that represents a 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>
```

4.3 Room-user and object management

Both current room users and objects in the room are regarded as parts of the room presence.

- *User addition*: A room user (or object) can enter a room by sending an INVITE and a PUBLISH request to the room.
- *User deletion*: The user (or object) can exit from the room by sending a BYE and an un-PUBLISH request to it.
- *Presence/motion propagation*: The UA sends a SUBSCRIBE request concerning room presence changes to the room URI, and they are reported to users by sending NOTIFY requests that contain PIDF documents.

Example: A PIDF expression that represents a user:

```
<tuple id="George@userdomain">
 <nickname>George</nickname>
 <icon>http://domain/icons/George.bmp</icon>
 <vs:auditory-icon>http://domain/auditory-icons/Georg
e.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>
```

4.4 Optimization of room-user management using the partial notification mechanism

- **Problem: SIP messaging overhead for user presence propagation is large.**

  RFC 3856 states that the minimum interval for presence notification should be 5 sec, but the interval in the current VPII is 2 sec (and still not short enough) because users cannot wait for 5 sec.

- **Our solution using partial notification mechanism**
  - *Difference is sent*: When the presence is updated, only the difference from the previously reported presence is sent to the UAs by using a partial PIDF document.
  - *Full notification is sent occasionally*: Full notification messages are also sent to a UA when it sends a periodical SUBSCRIBE request to the RM.

Example: A PIDF document for updating user location:

```
<?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>
```

- **Experiments**
  - *The message size was reduced by 48-69% by the optimization*: The average TCP message size of a full notification was 2200 bytes, and that of a partial notification was 680-1150 bytes.
  - *Processing time was reduced by 66-87%*: 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.

5 Conclusion

- *Unified usage of SIP/SIMPLE*: It makes the room management natural and light-weight in terms of computational and programming resources.
- *Optimization method using partial notification mechanism*: It significantly reduced SIP message sizes and processing time in a simulated meeting.
- *Better solution?*: The message size of presence notification 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.