Integrating VoiceXML with SIP services

Kundan Singh, Ajay Nambi and Henning Schulzrinne
Department of Computer Science, Columbia University
{kns10,an2029,hgs}@cs.columbia.edu

Abstract— **We describe our Session Initiation Protocol (SIP)-based VoiceXML browser, sipvxml, that allows programming interactive voice response applications that are accessible from telephones as well as IP phones. We also describe how we have used sipvxml in our multi-party multimedia conferencing server. We propose other applications and extensions that can benefit from this technology in our IP telephony test bed. (keywords: Internet telephony; interactive voice response; SIP; VoiceXML; sipvxml)**

I. Introduction

People are familiar with traditional interactive voice response (IVR) systems found in voice mail access, dial-in conferences, phone-based customer support and tele-banking. VoiceXML is an XML-based language developed by the W3C [1] to create voice dialogs that feature synthesized speech, digitized audio, recognition of spoken and DTMF key input and recording of audio for telephony applications. It enhances the traditional proprietary and closed IVR systems to an open programmable architecture. It brings the advantage of web technologies to a telephony user by providing programmable dialogs, similar to HTML forms or CGI scripts.

The Session Initiation Protocol (SIP [7]) is an Internet telephony signaling protocol used for establishing and terminating Internet multimedia sessions. A SIP-based VoiceXML browser (or SIP-VoiceXML browser) allows a SIP user to take part in application-specific IVR systems, e.g., voice mail or tele-banking. It also brings the advantage of VoiceXML technology to a regular telephone user via a SIP-PSTN gateway. We have developed a SIP-VoiceXML browser, sipvxml, to enhance the services of our CINEMA test-bed [3], [4], [11]. In particular, we have extended our multimedia conferencing server, sipconf [12], and unified messaging (voicemail) server, sipum [13] to provide enhanced services and convenience to a telephone user.

We describe the architecture of sipvxml in Section II. Section III describes use of sipvxml with our conferencing server. More examples of SIP services enabled by VoiceXML are described in Section IV. We list some other related work in Section V and conclude in Section VI.

II. Architecture

A SIP-VoiceXML browser is similar to a web browser for a telephone instead of a desktop PC. The browser fetches the VoiceXML pages or pre-recorded media files from a web server and presents an interactive dialog to the telephone user. Fig. 1 shows an example scenario where the browser can be accessed from SIP phones as well as a regular telephone. The VoiceXML pages can either be statically stored on the web server or dynamically generated based on some server side programming logic like HTTP-CGI (Common Gateway Interface), Java servlet or Java server pages. The media files can either be stored on the web server or can be streamed in real-time from an RTSP [10] media server, such as rtspd, directly to the SIP caller using RTP [8].

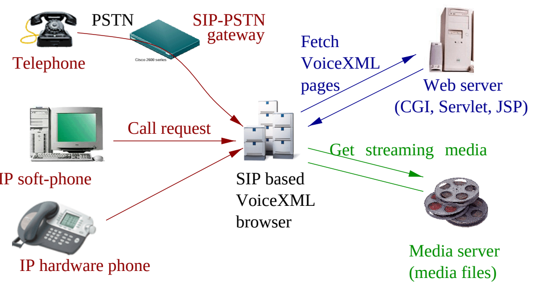

Fig. 1. Example sipvxml scenario

A. VoiceXML page

The following example VoiceXML page prompts the caller with spoken audio: "Enter the ZIP code ...". When the user presses a sequence of digits, say 10027#, the variable zipcode gets the value "10027" that gets passed to the URL http://myserver.com/weather.cgi?zipcode=10027. It is up to the script weather.cgi to process the input and generate further VoiceXML pages. If there is some error or user doesn't press anything, then the prompt is repeated.

```
<?xml version="1.0"?>
<vxml version="1.0">
<form>
 <field name="zipcode">
  <prompt>Enter the ZIP code of the location for which you
   want weather information.</prompt>
 </field>
 <catch event="noinput error help">
 Enter the ZIP code again followed by the pound key.
 </catch>
 <block>
  <submit
   next="http://myserver.com/weather.cgi" namelist="zipcode"/>
 </block>
</form>
</vxml>
```

We have implemented a very simple DTMF grammar. A typical explicit dtmf tag in the VoiceXML page looks like:

```
 <dtmf type="application/x-dtmf">
   1 | 2 | 3 | 4 | *
 </dtmf>
```

The MIME type for this grammar is "application/x-dtmf". Input is either a fixed length string or terminated by a "#". An implicit timeout of 5 seconds is implemented so that the input is automatically accepted if the user does not press the terminating "#" key for some time. If no grammar is specified, then the interpreter will accept any input. Users can press "**#" anytime to signal the help event.

We have implemented only a subset of VoiceXML tags as needed in our application: assign, audio, block, catch, clear, disconnect, dtmf, error, exit, field, filled, form, goto, help, noinput, nomatch, prompt, submit, value, var and vxml. We do not support any client side script (e.g., JavaScript) usually needed for arithmetic or string operations in the browser, as the same effect can be achieved using server side processing.

B. Operation of the browser

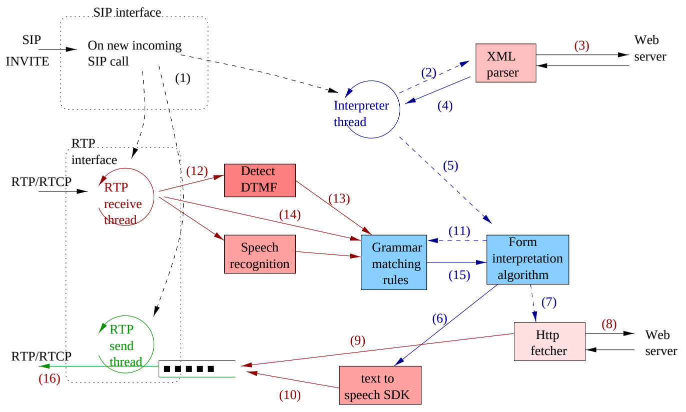

Fig. 2. Operation of sipvxml

Fig. 2 shows the components of our SIP-VoiceXML browser, sipvxml. We use our SIP library[1] for implementing a SIP interface, the RTP library[2] for the RTP/RTCP interface, the Apache's XML parser[3] with DOM interface, an HTTP fetcher[4] for getting non-XML pages and IBM ViaVoice Text-To-Speech SDK[5] for speech synthesis.

1. When the browser receives a new incoming SIP call it creates three different threads: RTP receive thread, RTP send thread, and the VoiceXML interpreter thread. The RTP receive thread receives media packets from the caller and invokes the DTMF detection module. The RTP send thread streams out media packets to the caller. A separate send thread helps in maintaining the constant bandwidth (e.g., 64 kb/s for G.711 audio) for outgoing packets and irrespective of the speed of the speech synthesizer. The initial VoiceXML page URL can be preconfigured in the browser or encoded in the SIP request [6]. For example, if the caller dials sip:dialog.vxml.http%3a//dialogs.server.com/script32.vxml@vxmlservers.com then the call will reach the browser running at *vxmlservers.com* and it will fetch the initial VoiceXML page from `http://dialogs.server.com/script32.vxml`. On the other hand, if the request-uri is `sip:7137@cs.columbia.edu`, then the interpreter is invoked with the default pre-configured initial VoiceXML URL, e.g., that of the conferencing service.
2. The interpreter thread calls the XML parser with the initial URL.
3. The XML parser fetches the page from the web server or a local file system (based on the initial URL).
4. It presents the returned XML document into a tree data structure.
5. The interpreter thread invokes the *Form Interpretation Algorithm* (FIA [1]) on the selected form from the VoiceXML document.
6. FIA invokes various other modules based on the content of the VoiceXML document. For example, it can invoke the text-to-speech SDK to synthesize any prompts. The current implementation does not use any speech recognition engine because user input is via touch-tone keys.
7. FIA can also invoke the HTTP fetcher module to fetch an external grammar file or a media file for an audio prompt. XML parser internally has its own HTTP client to fetch VoiceXML pages.
8. The HTTP fetcher implements a simple HTTP GET method to retrieve a document.
9. The media file retrieved from the web server using HTTP fetcher is fragmented into 20 ms packets for interactive telephony, and enqueued for streaming out to the caller by the send thread.
10. The speech synthesizer output is also fragmented and enqueued for sending out to the caller.
11. The VoiceXML document can specify the grammar rules in various scopes in the document. FIA can set the active grammar for the matching engine based on the current execution scope in the VoiceXML page.
12. The RTP receive thread receives the RTP media packets and invokes the DTMF detector.
13. Any detected DTMF digit is passed to the grammar matching engine.
14. DTMF tones can be transported from the caller to the browser in a number of ways. One approach is to not distinguish them from the spoken voice by encoding them using the same audio codec. However, a low bandwidth audio codec may distort the properties of the in-band DTMF tones making them hard to detect. A second, preferred way is to use "telephone-event" [9] containing the digit codes instead of the encoded audio in RTP packets. In the first case, the browser has to do the DTMF detection, whereas in the second case the caller or the gateway has to do the DTMF detection. The RTP receive module forwards telephone-events directly to the grammar matching engine. We have implemented both these methods. A third method of transporting DTMF in SIP INFO message is not used in our implementation.
15. The grammar matching engine tries to match the received digits with any active grammar, and informs

[1]http://www.cs.columbia.edu/~ kns10/software/siplib
[2]http://www.cs.columbia.edu/~ hgs/rtp/rtp-library.html
[3]http://xml.apache.org
[4]http://cs.nmu.edu/~ lhanson/http_fetcher/
[5]http://www-4.ibm.com/software/speech/dev/ttssdk_linux.html

the FIA if a match is found.

16. The RTP send thread periodically sends media packets to the caller. No packets are sent during silence.

III. Multi-party Conferencing

Consider a SIP conferencing system where users join the conference by dialing in a conference URI sip:staffmeet@conference.com. A regular telephone user with only a touch-tone phone cannot dial such a generic URI. We can assign one phone number per conference for direct inward dialing. However, it is preferred that the user always dials the number of the VoiceXML browser that in turn prompts him for the authentication PIN (personal identification number) and conference number. Once the user is authenticated the browser transfers the call to the selected conference. One can also use a single PIN to identify both the participant as well as the conference.

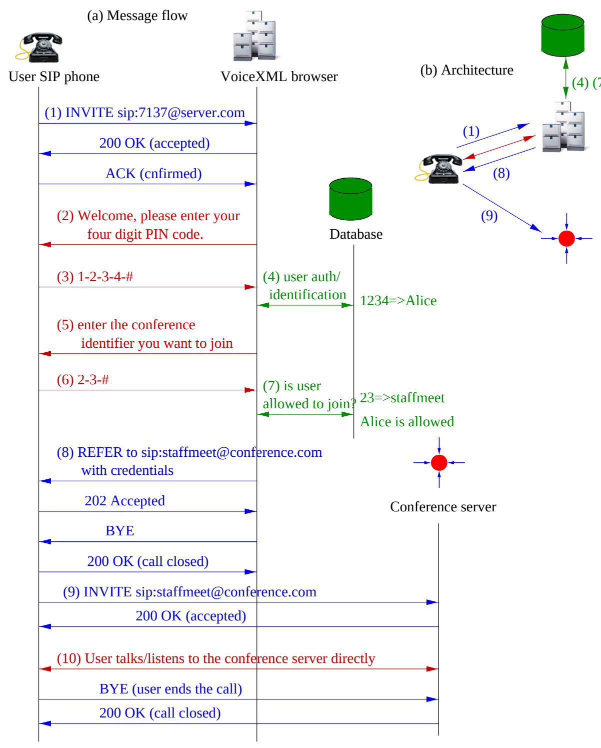
(a) Message flow
User SIP phone
VoiceXML browser
(b) Architecture
(4) (7)
(1)
(8)
(9)
Database
(1) INVITE sip:7137@server.com
200 OK (accepted)
ACK (cnfirmed)
(2) Welcome, please enter your four digit PIN code.
(3) 1-2-3-4-#
(4) user auth/ identification
1234=>Alice
(5) enter the conference identifier you want to join
(6) 2-3-#
(7) is user allowed to join?
23=>staffmeet
Alice is allowed
(8) REFER to sip:staffmeet@conference.com with credentials
202 Accepted
Conference server
BYE
200 OK (call closed)
(9) INVITE sip:staffmeet@conference.com
200 OK (accepted)
(10) User talks/listens to the conference server directly
BYE (user ends the call)
200 OK (call closed)

Fig. 3. Method 1: Joining conference in blind transfer mode

Fig. 3 shows how an user, say Alice, interacts with the browser before joining the conference.

1. Alice dials the browser's phone number (212-9397137) or SIP URI (sip:7137@server.com).
2. The browser accepts the call and prompts the caller to enter the PIN for identification.
3. Alice keys in her PIN, 1-2-3-4, followed by a terminating # key. The DTMF digits are sent in RTP.
4. The browser looks up the database and identifies the caller as "Alice".
5. Based on the privileges, the browser prompts her with a list of conferences to choose from.
6. Alice picks up the conference with identifier 23.
7. The browser again checks if Alice is allowed to join the conference identified by number 23, which in this example is sip:staffmeet@conference.com.
8. Once the authentication is done, the browser transfers the call to the actual conference server using the SIP REFER method [14] containing the SIP URI of the conference.
9. Alice's phone accepts the transfer and initiates a new call to the conference server.
10. Alice's phone exchanges audio with the conference server directly, without going through the browser.

Note that the user authentication, conference look up and transfer are actually invoked by the conference service CGI scripts, whereas the browser just interprets the VoiceXML pages generated by the scripts to do the actual transfer or prompt the caller. For instance, the service script may generate the following transfer tag for the call transfer in step (9).

```
<block><prompt>Your call is being transferred,
please wait.</prompt></block>
<transfer dest="sip:staffmeet@conference.com" bridge="false" />
```

The transfer can be done in two modes: blind and bridged. The former is the transfer of the call to the conference server without consulting the server whereas the latter is the transfer after consulting such that the browser may choose to be in the media path. We have implemented the blind call transfer as shown in Fig. 3.

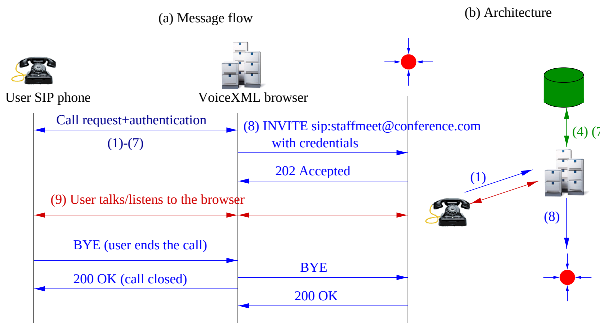

Fig. 4. Method 2: Joining conference using bridged mode

Fig. 4 shows the bridged transfer case with the browser as a "back-to-back-user-agent" bridging the audio path between the user phone and the conference server. Steps 1 to 7 are same as the blind transfer case. Instead of sending REFER, the browser sends a new call request to the conference server identifying the conference sip:staffmeet@conference.com in the Request-URI of the SIP INVITE message. The browser acts as an application level packet forwarder in both directions for RTP/RTCP media traffic.

The advantage of bridged transfer is that the browser remains in the media path and can accept future control commands (using DTMF) from the user phone. For conferencing, it may be useful to interpret DTMF, e.g., 6-6-# to mute your audio or 6-8-# to join another virtual chat/conference room. Secondly, the browser needs to forward other signaling messages also, e.g., re-INVITE from the caller to the conference server. Moreover, maintaining packet forwarding states for the duration of the conference limits the scalability of the browser on how many simultaneous callers it can handle. The browser may issue re-INVITEs with updated transport addresses for RTP/RTCP to both the caller and the conference server such that the media path is direct. However, this still needs the signaling state to be maintained for the duration of the call. On the other hand, a blind transfer does not require any call state in the browser for the duration of the conference. But it expects that the caller's IP phone supports the REFER method.

IV. Other services

This section describes some of the current and future services that can be provided using VoiceXML in our SIP environment.

A. Unified messaging – voice mail

Sipum is a SIP/RTSP-based unified messaging system that provides a centralized voice mail and answering machine service. For example, when Alice calls bob@cs.columbia.edu, the SIP server on cs.columbia.edu domain forks the call request to both Bob's IP phone and the answering machine (sipum). If Bob picks up the phone, the call request to sipum is cancelled. If Bob does not pick up the phone after 10 seconds, sipum accepts the call on Bob's behalf and prompts the caller, Alice, to leave any voice message. The same application, sipum, is also used by Bob to retrieve his voice mails, for example, by dialing a URL sip:bob-672-retrieve@sipum-host.cs.columbia.edu to retrieve the message with ID 672. There are other ways of retrieving voice mails, for instance, using a web browser or a media client. However none of these are appropriate for a telephone user with limited touch tone capability. We use sipvxml with application-level logic for voice mail service to allow more interactive interface to access and manage the voice mail box. From a user's perspective this is similar to the traditional voice mail service. However, use of SIP and VoiceXML allows easy integration with web, email, instant messaging, and telephone.

The application logic to perform voice mail service is built as CGI scripts executed in the browser's context. Once the user is authenticated using PIN, the main menu is spoken out. This includes options for playing out new voice messages and other details like sender, subject and timestamp of the message. An RTSP media server (e.g., rtspd) can be used to stream the actual voice message directly to the caller phone using RTP. We also provide additional options like saving or deleting the message.

B. Email by phone

Email is one of the most convenient forms of communication. However, convenience is limited by the necessity of an Internet connected computer. Our "Email By Phone" system solves this problem by allowing people to check and send their emails over any telephone. The server side application logic is implemented using Java Server Pages and Java Servlet. These programs generate VoiceXML pages based on the mail box content and user input. More information can be found at [5].

C. Event notification and scheduling

Asynchronous event notification is useful when polling for the event is inefficient. For example, the email-by-phone system can be modified to notify the user of any important email by calling user's cell phone. Text-to-speech is used to play out the email content on the phone. Alternatively, the user can go to a web page and schedule a birthday reminder or wakeup call by recording his own audio announcement or a text message. The system notifies the user by phone at the scheduled time. These simple systems do not need VoiceXML. However, a VoiceXML browser is needed to allow the user to schedule events from phone, or to "snooze" and notify again after a short while. We are extending sipvxml to allow initiating a new call for notification.

D. Audio volume level for conference

Multi-party audio conferencing among heterogeneous clients with different audio devices causes annoying distortion of audio. Some participants are heard very loud and some are not heard at all. Ideally, the conference server should balance the input audio level from the participants before mixing. However this imposes additional processing requirement on the server for every audio packet. Another approach is to tell the participant to adjust his volume level for both microphone and speaker. The participant connects to a "audio level feedback" system before joining the conference and speaks into it. The systems announces if the user's microphone volume is acceptable, too high or too low. The system also plays back a pre-recorded audio file and allows the user to adjust his speaker volume. This processing is built in a server side CGI script that is accessible via a VoiceXML browser.

E. Advanced conference control

Our current conference server implementation provides a web interface for floor control by the moderator and participant list display. We can extend it such that conference control can be done using the same telephone that the participant or moderator is using for the conference.

F. Integrating speech recognition

Our current implementation accepts user input only via DTMF digits. VoiceXML is designed for spoken audio input as well as DTMF. Allowing both mechanisms will improve the user experience.

G. Security

Every Internet system should deal with security. Our architecture has three places where security needs to be considered: telephony gateway, SIP signaling and RTP media transport, and HTTP/RTSP access to the backend servers. Telephony gateway and SIP security are explained in the CINEMA technical report [11]. HTTP and RTSP can use simple shared secret (Digest authentication [2]) or more secure Transport Layer Security (TLS).

In the bridged transfer case, sipvxml authenticates the caller using PIN and provides its own credentials to the conference server in SIP authnetication. In the blind transfer case, if the caller is using a regular phone connected via a gateway, the SIP authentication will contain the credentials of the gateway. To prevent malicious users connecting to a restricted conference, we pass the credentials from sipvxml to the caller's gateway in the REFER message, which in turn is used by the caller's gateway to call the conference server. The credentials can be that of the browser or the caller. For example, Refer-To header may contain `sip:staffmeet-<timestamp:hash>@conference.com`, where `<timestamp:hash>` is base-64 encoded and hash is MD5 hash of "browser-host:timestamp:shared-secret". The scheme works only when the conference server can interpret this URI. The server should reject the call if the timestamp is old, to prevent replay attack.

The browser should use HTTP POST method, instead of GET, to avoid storing CGI input in the web server log.

V. Related work

The Voice Browser working group of world wide web consortium (W3C) is improving the VoiceXML [1] specification. VoiceXML applications for interactive voice response are developed by many commercial organizations. Plum Voice Portal Technology[6] can present existing websites or intranet applications to a phone user. It can also deliver follow-up information via email or fax. Open VXI[7] is an open source VoiceXML interpreter. IBM's WebSphere provides HTML-to-VoiceXML transcoding that can be converted to speech by a VoiceXML browser. Talking E-Mail[8] allows users to access emails from various interfaces including voice, i-mode, Web and WAP. None of these applications use SIP for call control.

Rosenberg et al. [6] describe a SIP interface to VoiceXML dialog servers including SIP URI for VoiceXML service and message flows for blind and bridged call transfer. Tellme studio [9] provided the first SIP based VoiceXML development platform that allows users to test custom VoiceXML pages or scripts. We used this for initial testing of our Email by phone system. Our work describes the design of a SIP based VoiceXML browser and its application in our IP telephony test bed. At the time of writing, ours is the only known implementation that associates the VoiceXML transfer tag with the SIP REFER message for a conferencing application. Moreover, sipvxml can be used as a third-party voice application server like Tellme or an integrated component in the CINEMA architecture [3], [4], [11] for campus or enterprise VoIP services.

[6]http://www.voicexml-ivr.com/solutions/intro.html
[7]http://www.speechworks.com/products/voicexml/openvxi.cfm
[8]http://www.voice3g.com/appblocks.htm
[9]http://www.tellme.com

VI. Conclusions

VoiceXML is a powerful technology that allows telephone users to access services that are typically available to web users. SIP interface to a VoiceXML browser allows such services from a IP telephone as well as a regular telephone, via a gateway, using the call transfer feature in an interoperable manner. We have implemented a simple SIP based VoiceXML browser and used it to allow telephone users to connect to our conferencing server. This along with other services that we have implemented will help users to join a conference, check mails and stay informed from anywhere thus enabling ubiquitous availability. We are extending our IP telephony test bed to incorporate various voice based services like accessing unified messages, event notification and scheduling, email notification and conference control via a telephone.

VII. Acknowledgements

Daniel Liu and Naho Ogasawara implemented Email by phone using Tellme studio. Sean Mandel and Visda Vokhshoori implemented parts of sipvxml. The work is supported by grants from SIP Communications, Inc.

References

[1] VoiceXML specification. http://www.w3c.org/voice/.
[2] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP authentication: Basic and digest access authentication. RFC 2617, Internet Engineering Task Force, June 1999.
[3] W. Jiang, J. Lennox, S. Narayanan, H. Schulzrinne, K. Singh, and X. Wu. Integrating internet telephony services. *IEEE Internet Computing*, 6(3):64–72, May 2002.
[4] W. Jiang, J. Lennox, H. Schulzrinne, and K. Singh. 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.
[5] D. Liu and N. Ogasawara. Project: Email by phone using voicexml, May 2001. http://www.cs.columbia.edu/~kns10/projects/spring2001/emailbyphone/EmailByPhone.html.
[6] J. Rosenberg. A SIP interface to voiceXML dialog servers. Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.
[7] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: session initiation protocol. RFC 3261, Internet Engineering Task Force, June 2002.
[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. RFC 1889, Internet Engineering Task Force, Jan. 1996.
[9] H. Schulzrinne and S. Petrack. RTP payload for DTMF digits, telephony tones and telephony signals. RFC 2833, Internet Engineering Task Force, May 2000.
[10] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (RTSP). RFC 2326, Internet Engineering Task Force, Apr. 1998.
[11] K. Singh, W. Jiang, J. Lennox, S. Narayanan, and H. Schulzrinne. Cinema: Columbia internet extensible multimedia architecture. technical report CUCS-011-02, Department of Computer Science, Columbia University, New York, May 2002.
[12] K. Singh, G. Nair, and H. Schulzrinne. Centralized conferencing using SIP. In *Internet Telephony Workshop 2001*, New York, Apr. 2001.
[13] K. Singh and H. Schulzrinne. Unified messaging using SIP and RTSP. In *IP Telecom Services Workshop*, pages 31–37, Atlanta, Georgia, Sept. 2000.
[14] R. Sparks. SIP call control - transfer. Internet Draft, Internet Engineering Task Force, July 2001. Work in progress.