HSTP : Hyperspeech Transfer Protocol

Sheetal K. Agarwal, Dipanjan Chakraborty, Arun Kumar,
Amit Anil Nanavati, Nitendra Rajput
IBM India Research Laboratory
4, Block C, Institutional Area
Vasant Kunj, New Delhi, India
{sheetaga, cdipanjan, kkarun, namit, rnitendra}@in.ibm.com

ABSTRACT

HTTP provides a mechanism to connect web sites. Almost all sites have a large amount of *hypertext content* that provides connection to other sites in the World Wide Web. The success of the WWW can be partly attributed to the seamlessly browsable *web* that is formed through this connectivity. However, navigation of *hypermedia content* through non-visual interfaces has not received as much attention. Specifically, telephony voice applications offer immense usability and penetration benefits and can act as alternate information access and delivery mechanisms. Connectivity across voice applications poses interesting and novel challenges. In this paper, we define HYPERSPEECH as a voice fragment in a voice application that is a hyperlink to a voice fragment in another voice application. Further, we present HYPERSPEECH TRANSFER PROTOCOL (HSTP) – a protocol to seamlessly connect telephony voice applications. HSTP enables the users to *browse* across voice applications by navigating the HYPERSPEECH content in a voice application. HSTP can also be used for developing cross-enterprise applications that allow a user to *transact* across two or more voice applications.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia

General Terms

Human Factors

Keywords

Hypermedia, telephony applications, HTTP, call transfer

1. INTRODUCTION

There has been an explosive growth in the number of telephony voice applications in the recent past. The first generation of IVR (Interactive Voice Response) applications that capture a user's response through dialing DTMF digits on a telephone keypad are still used widely in banking and travel industries. Enhancements in speech recognition and natural language understanding technologies over the past decade has seen the evolution of these IVR applications to conversational systems where a user can speak to the system and the voice is recognized to perform the desired user action. Enterprises use these conversational systems as a cost-effective way to automate their call centers. Telephony access to enterprise applications thus provide an alternate channel to reach the customers. With increasing penetration of mobile phones, telephony voice applications are more easily accessible than those on the Internet/PC.

Telephony voice applications so far have been successful in providing access to data and services in the context of a single business. Services requiring integration across businesses such as linking payment gateways with a tele-shop, require mechanisms to interconnect the two applications. In the Internet world, a large part of the success of the World Wide Web is due to the interconnection across different websites that is provided through the hypertext/hypermedia links. As an example, it is possible to seamlessly purchase items at a website and then make payment at the bank's website. Such cross-enterprise connection of applications in a standardized manner provides the Internet users with a rich set of applications. Such an interconnection is missing between telephony voice applications.

One possible mechanism to provide the interconnection is through underlying programmatic service interfaces. Here, the tele-shop voice application accepts the credit card information from the buyer. It then uses existing web based online channels to connect and supply this information to a payment gateway that authorizes the payment with the backend credit card service. This solution requires the buyer to share her credit card information with the tele-shop's voice application that may not be really trustworthy, and thus is unsafe. The other option to provide integration involves a human link typically provided through a call center.

Interconnection in the speech domain has been addressed by Barry Arons in [2]. He defined Hyperspeech as an application for presenting "speech as data". The purpose of his system was to allow a user to navigate *within* a database of recorded speech without any visual cues. In his application, the speech data was manually recorded and created different types of (relational) links to organize that data into a speech-only hypermedia structure. However this technique cannot be directly extended for telephony voice applications because protocol level issues related to telephone call transfer

HT'07, September 10–12, 2007, Manchester, United Kingdom.

need to be addressed across applications. Seamless browsing and interconnection *across* telephony voice application continues to be an unaddressed problem.

Paper Contribution: We have developed a protocol called HYPERSPEECH TRANSFER PROTOCOL (HSTP), that provides a mechanism to connect telephony voice applications with each other. HSTP enables voice-driven transactions that can span multiple cross-enterprise voice applications. HSTP allows easy interconnection across voice applications and this can be used for providing a seamless browsing experience to the telephony user. We also present an API that voice applications can use for adding HYPERSPEECH content.

HSTP impacts several application categories as well as enables new applications. The implication of HSTP is deeper for developing regions where there is a huge market of businesses (small-medium and micro-businesses), with a need for telephony voice applications [14]. Apart from enabling secure cross-enterprise transactions, HSTP also allows navigation across voice applications, potentially hosted on different enterprises. It enables the concept of *links* between voice applications and provides the user with the ability to browse forward and backward across voice applications.

The rest of the paper is organized as follows: Section 2 presents a motivating scenario. Section 3 describes and formulates the technical problem addressed this paper. Section 4 provides the high level architectural details of the HSTP. We provide the detailed description of the HSTP components and the session management in Section 5. In Section 6, we present the use of HYPERSPEECH in voice applications and describe the details of the proof-of-concept applications. Section 7 provides the related work in this area and conclusions are presented in Section 8.

2. SCENARIO

We explain the concepts of HYPERSPEECH through the following use case scenario:
Jonathan is a busy salesman who travels frequently. His work typically requires him to stay in a place for a few days. Once he is in a new place, he has to go around looking for grocery stores in his locality for his daily needs. He prefers taking phone numbers of the identified stores and places orders on the phone subsequently. Home delivery services deliver the goods to his home. However, often the home delivery boys don't accept credit cards and even if some do, Jonathan tries paying by cash since he doesn't want to share his credit card information with untrusted home delivery agents. This often causes problems since he often runs out of cash.
During his travel, he visits a city and finds out that there is a yellow pages service in the city that he can call up to receive phone numbers of several businesses. He promptly calls up the service and uses the telephony voice application to browse through the grocery stores in the vicinity of his hotel. On Jonathan's prompt, the call gets transferred to a grocery store and goes to the voice application of the store. Jonathan easily specifies the items he needs to buy from the cataloger. The order is placed and a delivery guarantee is made within half-hour. To his surprise, the grocery store's voice application also accepts credit cards securely over phone. Jonathan selects the option and his call gets transferred to yet another voice application of a secure payment gateway. The secure payment gateway already knows about the amount of money the grocery store wants to charge to Jonathan, and securely authorizes the payment by taking in Jonathan's credit card details and transacting with the credit card company's authorization system. The delivery boy comes within half-hour and delivers the goods to Jonathan.
We observe a few salient features that HYPERSPEECH enables in the above scenario: (1) Telephony voice applications across enterprises are seamlessly integrated with one another(2) For transactional operations, voice applications transfer context (e.g. amount to be billed on a credit card) to other voice applications; (3) Users can employ purely spoken language interactions to complete a transaction.

3. PROBLEM DESCRIPTION AND STATEMENT

Today many organizations have their own telephony voice applications that provide an access channel to their customers. Using technologies like [14], individual subscribers can also create and host their own voice applications. To facilitate browsing of a such a network of telephony voice applications and to support the transactional scenario outlined in the previous section, a much refined system than what exist today is requirded for navigation over the phone. Navigation of an interconnected web of such voice applications could be done in free form browsing mode or in a transactional mode.

Hyperlinking two cross-enterprise voice applications to support such navigational capabilities requires addressing of some interesting issues. Telephony voice applications today are primarily accessed through phone calls – a legacy service supported by telecom networks. A *phone number* is associated with a particular voice application, which is equivalent to the *URL* of a web application. Legacy telephony channels offer voice call transfer but are inflexible to allow application context transfer. *Hyperlinks* in the Internet allow navigation as well as context transfer through HTTP. However, navigation and cross-voice application transactions require call transfer as well as application context transfer. Telephony voice applications (deployed in the IT infrastructure and hence accessible over IP channels) could be linked together using hyperlinks. However, in order to support an immersive telephony speech-based navigation and transactional browsing experience, a synchronization of the voice call with the application context transfer is necessary.

The core technical problem that we solve in this paper is *to develop a mechanism that allows a session at a telephony voice application to be transferred to another telephony voice application so as to enable a workflow in which these participate.* As a solution, we present the HYPERSPEECH TRANSFER PROTOCOL (HSTP) that enables navigation of a network of HYPERSPEECH enabled telephony voice applications. We define HYPERSPEECH as a voice fragment in a voice application that is a hyperlink to a voice fragment in another voice application.

4. HYPERSPEECH ARCHITECTURE

In this section, we describe our solution approach and corresponding architecture for enabling user-driven and system-driven navigation of HYPERSPEECH links. User driven navigation enables voice based browsing of a network of interconnected voice applications. System driven navigation enables transactional operations spanning across multiple voice applications that may reside in different organizations.

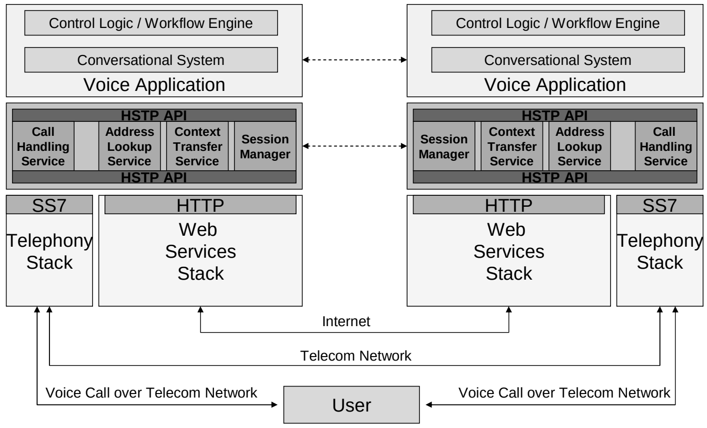

Figure 1: HYPERSPEECH **Protocol Stack**

4.1 Solution Approach

Our goal is to enable a telephonic conversation between the user and a voice application that can get transferred to and from another voice application without losing the relevant context. In the Internet world, HTTP/HTTPS take care of transferring the relevant context along with the user request for visiting a web page. However, the telephone network is different.

The telephone network is a widely distributed system of intelligent switching nodes that provide real-time information paths between two or more parties. Nodes in the telephone network communicate to establish and tear down calls so that two or more parties can communicate via terminal equipment (such as a phone acting as an endpoint) and the network. This process is referred to as *signaling*. Dual Tone Multi-Frequency (DTMF) protocol is a well known example for terminal equipment-to-network signaling in which the terminal equipment generates simultaneous pairs of tones to represent each dialed digit. Signaling on circuit-switched telephone networks can take place *in-band*, that is, on the same channel with voice communications (known as the *talk path*), or *out-of-band*, that is, through some communications channel other than the talk path. Out-of-band signaling creates a direct, message-based digital information link between the telephone switch and the computer-based application. To enable HYPERSPEECH based browsing and transactions, call control information needs to be enriched with application specific information. However, the call control information that can flow in current telephony signaling protocols is limited and restricted. This is primarily because they were designed mainly to enable voice call control and do not easily accommodate the context of the application that may be behind the call. Here we are considering the existing Public Switched Telephone Network (PSTN) and GSM/CDMA networks rather than SIP based IMS [6] networks. A discussion on the latter appears in section 8 of the paper.

There are two solutions to this problem. The existing signaling protocols such as SS7 [15] can be modified to incorporate application specific information but this would require replacing existing deployments which is a very costly and inefficient proposition. The alternate solution is to keep the existing telephone signaling protocols untouched and build the extended signaling mechanism further out-of-band, to support application based signaling in a channel out of current telephony network. We adopt this solution and present the overall approach next.

4.2 Architecture

Figure 1 depicts different layers of the HYPERSPEECH stack. We next describe the functionality of each of the layers and the components therein.

- **Voice Application** is topmost layer that comprises the voice driven business application. It consists of the telephony conversational system that the caller interacts with. A workflow engine or some other application logic in the voice applications determines the behavior of the conversation system. The workflow for a transaction that spans across multiple sites would get split across voice applications requiring them to coordinate and cooperate with each other. In case these voice applications are not autonomous, a centralized workflow engine could be used to orchestrate the interaction. Free form browsing is driven by the user commands within a voice application as well as across voice application boundaries.

- **HSTP** is our proposed protocol that implements out-of-band signaling for HYPERSPEECH session control. Apart from providing the capability of transferring a HYPERSPEECH session, it also introduces four commands that callers can use for navigation and browsing. They are: *next, previous, browse_forward (n), browse_back (n)*. These commands rely on the functionality provided by the HSTP layer underneath. For instance, a *browse_forward* command from the user is received by the voice application and results into invocation of a call transfer operation exported by the HSTP API. This involves not just a simple telephone call transfer but requires transfer of appropriate context to be sent out-of-band requiring proper handling of the associated synchronization issues. A session needs to be created and managed because unlike in the Internet, the browser for voice is a server-side component. HSTP sits on top of two stacks - the telecom signaling stack such as SS7 and the Web Services stack. It consists of the following sub-components:
  - *HSTP API* presents a programming interface to the voice application for submitting user requests for call transfer to a hyperlinked voice applica-

tion. The lower layers use this API to hand over incoming calls or returning calls (i.e. those that were transferred out earlier).

  - *Call Handling Service* interfaces with the telecom stack to receive or transfer phone calls. For each incoming phone call this service needs to determine whether it is a fresh call from a caller, a returning call, a call transferred from another voice application or a reconnect attempt from a caller whose session may have terminated abruptly. All are treated differently. It interacts with the Session Manager to obtain the required information and forwards the call to the voice application appropriately.
  - *Session Manager* component is responsible for establishing, maintaining and terminating HSTP sessions. In addition to Call Handling Service, it interacts with the Context Transfer Service for establishing and maintaining an HSTP session. It also takes care of purging/maintaining the appropriate data structures to handle sessions in the event of call drops or re-connections.
  - *Context Transfer Service* allows telephony voice applications to share context information to establish an HSTP session across applications.
  - *Address Lookup Service* translates a phone number to the address of the Context Transfer Service of the target voice application. This creates a loose coupling between voice applications where details of the target voice application need not be known apriori.

- **Telecom Stack** consists of the basic call switching functionality available in current telecom (PSTN) and cellular networks (CDMA/GSM).
- **Web Services Stack** incorporates the web services functionality which is becoming the dominant technology for distributed systems integration. The Context Transfer Service in the HSTP layer of the source site uses Web Services to interact with its peer in the target site. All the voice application level HYPERSPEECH session control signals are passed through this mechanism which is out-of-band for the telecom network and resides in the Internet.
- **User** interacts with the voice application through the telecom stack using ordinary phones. She gets charged for the voice calls made to original voice application or to the ones visited through HYPERSPEECH links.

We note that web services stack can be replaced with other protocols with similar capabilities to achieve out-of-band signaling for HSTP session.

5. HSTP: HYPERSPEECH TRANSFER PROTOCOL

In this section, we present details of the HSTP protocol for session based communication between telephony voice applications. As described in Section 4, the HSTP layer consists of five components. Design of each of those is presented next along with the HSTP message format.

5.1 HSTP Message Format

An HSTP message is exchanged between the HSTP layers of the telephony voice applications involved. Following are the contents of the message:

- **CALLER_NO:** This is the phone number of the person who calls the voice application.
- **SESSION_ID:** This is a globally unique identifier that is used to establish and maintain a session across multiple voice applications.
- **VA1_NO:** This is the phone number of the source voice application.
- **VA2_NO:** This is the phone number of the target voice application.
- **HOP_NO:** This represents the number of steps that the user session has to be transferred back or forward while browsing interconnected voice applications.
- **APP_CTXT:** This is the application payload that is delivered by HSTP. It can be used by the voice application to transfer context information across to other voice applications. In addition to application specific information, APP_CTXT consists of two protocol prescribed fields namely SRC_ANCHOR and TGT_ANCHOR. TGT_ANCHOR represents application specific information regarding the point in the target application where the call has to be transferred. Similarly, SRC_ANCHOR contains information regarding the point in the source voice application from where the call got transferred.
- **TYPE:** If the TYPE value is 0, the application is treated to be of browsing type. If this is 1, it is treated to be of transactional type. For reconnect attempts, the TYPE is set to 2.

In addition to this message, an ACK is transmitted to confirm receipt of a correct message.

5.2 HSTP API

The HSTP API consists of two parts. First one provides an interface to voice applications to connect to other applications for enabling HYPERSPEECH links. The second part is utilized by lower layers of the HYPERSPEECH protocol stack for purposes such as handing over a received phone call or an incoming request for a new HSTP session. These APIs are described below:

- *transferHSTPSession(VA2_NO, APP_CTXT, TYPE)* This method is invoked by the voice application to transfer an HSTP session to a hyperlinked voice application or in the case of transactional operations, transfer it back to the source voice application.
- *processIncomingCall(dialledNo, callerNo)*

  This method is invoked by the Computer Telephony Integration (CTI) interface on receipt of each incoming phone call from the underlying telecom stack.
- *createNewHSTPSession(HSTPMessage)*

  This method is invoked through a SOAP message supplied by the underlying layer in the Web Services stack. The SOAP message consists a new HSTP session request and is received from another voice application.

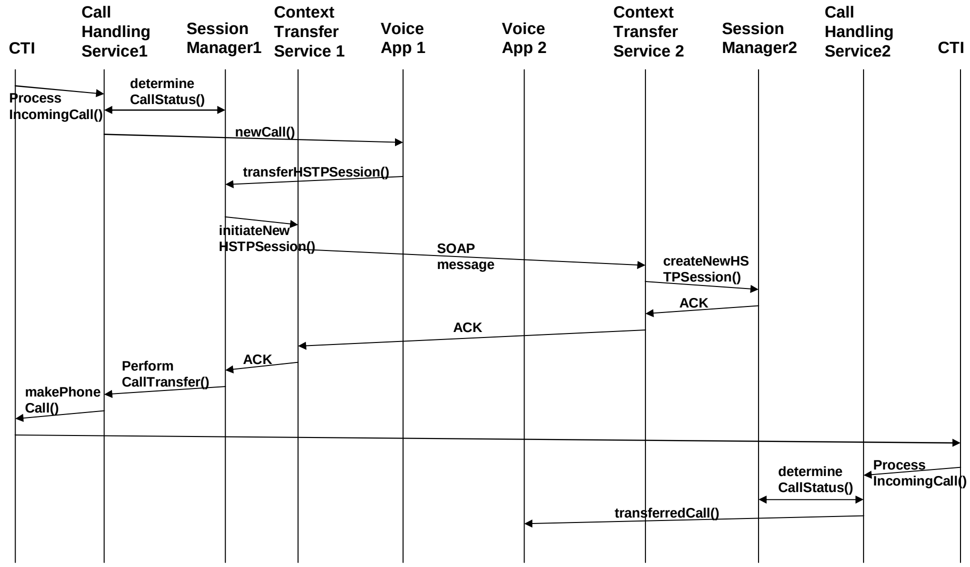

Figure 2: HSTP **based call-transfer.**

The *transferHSTPSession()* method allows the voice application to offer *browse_forward (n)*, *browse_back (n)* operations in its API exposed to the user.

5.3 Call Handling Service

Call Handling Service implements the *processIncomingCall()* method of the external API as well as *performCallTransfer()* method which is an internal operation invoked by the Session Manager during session establishment.

The CTI receives the phone call from the telecom stack and invokes *processIncomingCall()* method of the Call Handling Service, which then invokes *determineCallStatus()* method of Session Manager for determining the type of the call. Four cases arise. In each case, it delivers the call to the voice application registered against the *dialledNo* along with the information received from the Session Manager.

- *Fresh Call:* If the Session Manager determines the call to be fresh, then it registers the call and the Call Handling Service simply delivers it to the voice application registered against the *dialledNo*.
- *Transferred Call:* If the call has been transferred from another voice application to this voice application through a hyperlink then it is a context based call. If the call's context has HOP_NO set to zero then it is meant for this voice application. The Session Manager reports this and makes available the (1) call TYPE, (2) the APP_CTXT information that it must have already received from the source voice application. The call is delivered to the voice application registered against the *dialledNo* with this information.

  The call could be a part of a user browsing session or part of a transaction which is specified by the Type.
- *Call InTransit:* If the new call is a context based call but its HOP_NO is not 0 then the call is not meant for this voice application and is traversing back the links visited earlier, i.e. it needs to be transferred further. The Session Manager returns a new HSTP message which is then used to invoke *transferHSTPSession()*.
- *Reconnect Attempt:* If the new call is a reconnect attempt then it could be due to call drop from the current voice application itself or from a voice application reached by following hyperlinks in the current voice application. In the former case, call is delivered to the local voice application and its timers are reset by the Session Manager. In latter, it is forwarded to the voice application to which it had been transferred earlier. Following the same process it would eventually reach the voice application where it dropped off.

performCallTransfer(callHandle) is an internal method implemented by Call Handling Service. Session Manager invokes it (as part of *transferHSTPSession()* method) when it has successfully transferred the context through the Context Transfer service and is ready to transfer the user phone call. Call Handling Service interfaces with the CTI to perform the call transfer which interfaces with the telecom stack.

5.4 Session Manager

Session Manager is responsible for establishing, maintaining and terminating an HSTP session between two HYPERSPEECH enabled voice applications.

5.4.1 New Call Registration

The Call Handling Service informs Session Manager about the arrival of new call through *determineCallStatus(dialledNo, callerNo)* method. Session Manager checks its INCOMING-CALL-TABLE (refer Fig. 3(b)) to determine whether a corresponding entry exists. If not then it is a fresh call and a new session id is generated for this call which is a globally unique identifier. The new call's parameters are registered in the INCOMING-CALL-TABLE. The call being a fresh one, VA1_NO is kept same as the CALLER_NO, HOP_NO is set to 0, APP_CTXT is null, TYPE is set to 0 and CALL_RCVD is set to 1.

5.4.2 Session Establishment

This is responsible for establishing an HSTP session between two telephony voice applications. It first transfers the context of the voice application and then facilitates the actual phone call transfer from one voice application to another, through the telephony channel.

1. *Context Transfer Initiation at Source*

   The session establishment module implements *transferHSTPSession()* method to allow a HYPERSPEECH application session to be transferred. It takes four parameters described below:

   (a) The VA2_NO is the phone number at which the voice application to be transferred to is hosted.

   (b) The HOP_NO parameter is used when the user wishes to browse to a voice application that is more than one hop away from the current voice application. This can happen while issuing a *browse_back* command, for instance or to close a transaction that spanned across multiple voice applications. HOP_NO specifies the number of hops that need to be performed while transferring the HSTP session.

   (c) APP_CTXT contains the application specific context information required for performing the HSTP session transfer in the context of the application semantics. SRC_ANCHOR and SRC_ANCHOR form part of APP_CTXT in addition to other application specific information.

   (d) TYPE parameter specifies whether the HSTP session transfer is in the context of a transaction or as part of free form browsing by the user. This helps in setting the appropriate values of session parameters such as session timeouts or could be utilized by the voice application.

   For a transactional operation the HOP_NO is set to 1, TYPE is set to 1 and APP_CTXT is populated with application context information (including the source and target anchors). For a browsing operation the HOP_NO is set to greater than or equal to 1, TYPE is set to 0 and APP_CTXT is populated with application context information (including the source and target anchors).

   When the voice application issues a session transfer request to HSTP, an HSTP message is composed and sent to the target voice application. The details of the message parameters are described in Section 5.1. This HSTP message is handed over to the Context Transfer Service for transmission to the target voice application. The session transfer information is recorded in the OUTGOING-CALL-TABLE (refer Fig. 3(a)) of the source site. The table contains the phone number (VA2_NO) of the application to which the call will be transferred.

   The *transferHSTPSession()* method may also be used transfer a HYPERSPEECH session not to a new voice application but to the source voice application which it had come from.

2. *Context Transfer Request Handling at Target*

   The session establishment module also implements the *createNewHSTPSession()* method that is used to receive and process HSTP session context transfer requests received from other voice applications.

SESSION_ID	CALLER_NO	VA2_NO	APP_CTXT	TYPE	STATUS
4938	<no1>	<no5>	--	0	0
...	...	...	...	...	...
...	...	...	...	...	...
3039	<no3>	<no8>	<msg>	1	2

(a) The OUTGOING-CALL-TABLE

SESSION_ID	CALLER_NO	VA1_NO	HOP_NO	APP_CTXT	CALL_RCVD	TYPE
4938	<no1>	<no2>	4	--	1	1
...	...	...	...	...	...	...
...	...	...	...	...	...	...
3039	<no3>	<no4>	1	<msg>	0	1

(b) The INCOMING-CALL-TABLE

Figure 3: Incoming and Outgoing call tables

   If the HSTP message is successfully received by the Context Transfer service at the target site, it creates a new entry and stores relevant information from the message to its INCOMING-CALL-TABLE (refer Fig. 3(b)). As seen in the figure, the information stored is a copy of the message it receives from the source voice application. The only additional column is the CALL_RCVD flag. This flag specifies if a telephone call corresponding to this message has arrived or not. If the call has not arrived yet, then the value of this flag is 0 for the specific message. Once the table is updated, the Session Manager at the target site sends an ACK back to the source.

3. *Context Transfer Completion*

   When the ACK from the target site is received by the source voice application, the STATUS in OUTGOING-CALL-TABLE is updated to 1 to indicate that the call is yet to be transferred. This field is used to distinguish a reconnecting call from a fresh call in case of call drops as explained later in this section. HSTP then requests the CTI to transfer the telephone call to the phone number of the target voice application. When the call has been transferred, the STATUS is updated to 2 to signal the completion of session transfer.

4. *Context Transfer Failure Management*

   Two timers are used to manage the context transfer session. T_{ct} is the call-transfer timeout parameter and is the time for which a transferred context is stored in the INCOMING-CALL-TABLE at the *target* site starting from the time when the context transfer request has been received. If the telephone call does not arrive within T_{ct} units of the arrival of the message, then the entry is removed from the INCOMING-CALL-TABLE. If any call from the CALLER_NO arrives after T_{ct}, then it will be treated as a fresh call. This time is calculated as follows:

$$T_{ct} = N_r \Delta T + T_{ack} + T_{source-app} \tag{1}$$

   where,

   ΔT is the average time it takes for the telephone call to be transferred in a telecommunication network,

N_r is the number of allowed phone transfer retires,

T_{ack} is the average transmission time for the ACK to be transferred from target to source site, and,

$T_{source-app}$ is the execution time for firing a telephone call transfer request at the source site.

Similarly, T_{cs} represents the time for which the *source* site will save the state of the call, starting from the time when it issues the $newHSTPSession()$ call. This timeout is calculated as follows:

$$T_{cs} = T_{message} + T_{target-app} + T_{ack} + N_r \Delta T + T_{source-app} \quad (2)$$

where,

$T_{message}$ is the average time it takes for the message to be transferred in the IP network,

T_{ack} is the average time it takes for the ACK to be transferred in the IP network,

$T_{target-app}$ is the execution time for parsing the message by the target site and then issuing the ACK.

Thus, if the source site receives a call from the same CALLER_NO after T_{cs}, then the call is treated as a fresh call. However if the call transfer was not successful then the caller can still call within T_{cs} and will be able to resume the session from the call-transfer point in the particular application.

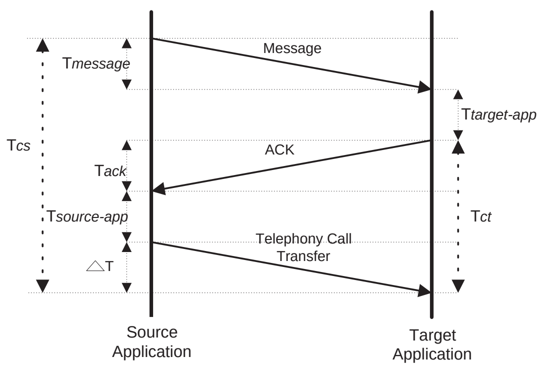

Figure 4: Timeout Diagram of T_{ct} and T_{cs} with N_r=1

5. *Phone Call Transfer*

   Once the context transfer succeeds and HSTP at the source site initiates transfer of the call, the telecom signaling stack takes care of the actual transfer. The transferred phone call is received by the CTI at the at the target voice application and delivered to the Call Handling Service. As explained earlier, the *processIncomingCall()* method of the Call Handling Service processes the call, determines it as a transferred call and hands it over to the target voice application with the information returned by the Session Manager at the target site. This completes the HSTP session establishment process.

   If the call has to be transferred back to the source voice application, then the same procedure is followed that is triggered by invoking transferHSTPSession() method.

5.4.3 Session Maintenance

Once an HSTP session has been established, Session Maintenance module is responsible for keeping track of that session until it expires.

Session Life Time: For browsing operations, a timer T_l is used to represent the average time for which the session information should remain alive at a particular site and thus represents the maximum *life* of a session. If a call is successfully transferred from the source site, then it will not know when the session ended at the target site since explicit session closure is not prescribed for now, primarily to avoid message exchange overhead. Therefore, the source site has its own timeout parameter T_l that defines the time for which the entries for the session will be maintained in the INCOMING-CALL-TABLE and the OUTGOING-CALL-TABLE. This is a high value when compared with T_{ct} and T_{cs} and is a configurable parameter that could be set based upon statistical analysis of the environment being deployed in. For instance, assuming that no caller in a region talks on phone for more than x hours at a stretch, this timeout value can be set to that. Thus, if the caller calls at a time t after her earlier call, then then the call will be treated as a fresh call if $t \geq T_l$ and a new SESSION_ID will be generated for this CALLER_NO. Figure 4 shows the various timeout parameters that should be met for a successful call transfer.

For transactional operations, the timer T_l is configured to have a small value for safety reasons. In normal cases, the session tear down mechanism takes care of cleaning up the information for transactional operations and the timer is helpful in abnormal terminations.

Session Failure Management: The call can drop due to one of the following reasons

1. The call transfer took more time than one of the timeout parameters T_{ct} or T_{cs}.
2. The user disconnected the call.
3. The voice application disconnected the call.

In any of the cases, the user can reconnect to any of the voice applications that it was browsing. If the duration after which the reconnection is made is greater than T_l, then the call will be treated as a fresh call. However, if the call is made within T_l, and there is an entry in the OUTGOING-CALL-TABLE with STATUS equal to 1, it indicates that the context was transferred but not the call so this is most likely a reconnect attempt. For confirmation, the call is delivered to a system voice application, rather than the intended one, that informs the caller of possible reconnection attempt and prompts her to confirm. Based upon the response, the call is either delivered to the intended local voice application as a fresh call, after creating a new session or the session is transferred to the same voice application where it was transferred before disconnect. In the latter case, the TYPE field of the HSTP message is set as *reconnect*. Thus HSTP handles the reconnection of any calls that fail in the call transfer process.

5.4.4 Session Tear-down

For transactional operations, the Session Manager removes the session information from the INCOMING-CALL-TABLE and the OUTGOING-CALL-TABLE if a session has completed at the local site, and has been successfully transferred back to the site from where it came from or the current site was the source site. The session completion at local site includes successful return of the call from any sites to which it may have been forwarded to. Therefore, the session information keeps getting cleared as the call starts moving back towards its source. This also means that in the context of a transaction, the user cannot issue *browse_forward* or *browse_back* command.

For browsing operations, once HSTP realizes that a session has ended (through the timeouts mentioned above, or through a specification by the application), it deletes the corresponding entries from the INCOMING-CALL-TABLE and the OUTGOING-CALL-TABLE. However, no information is passed to the other sites from/to which the session would have been transferred to/from this site. This mechanism reduces the network traffic overhead. The entries from the other sites are automatically deleted based upon the timers set in the HSTP layer at those locations.

5.5 Context Transfer Service

Context Transfer Service in the HSTP layer is both a Web Service client and a Web Service itself. The Session Manager uses it while establishing an HSTP session with the target voice application. Session Manager passes an HSTP message to it through the *initiateNewHSTPSession()* method. Context Transfer service at the source site takes the target voice application's phone no. (VA2_NO) and the Address Lookup service to determine the web service URL of the Context Transfer Web service of the target site. It then makes web service call over an HTTP session and invokes a *createNewHSTPSession()* method on the target Context Transfer Web Service. The HSTP message is supplied as payload. An ACK is returned to the source Context Transfer Service once the transfer completes successfully.

A scalable design of Address Lookup Service is out of the scope of this paper. One possibility is that it could be modeled as the Domain Name System (DNS) that has been very successful in the Internet. In addition, the Context Transfer service would need to use a dynamic invocation framework such as Web Service Invocation Framework (WSIF) [9] to be able to make web service calls on arbitrary web services without first having to implement corresponding service clients.

5.6 HSTP Application Characteristics

HSTP applications could consist of free form user driven browsing as well as transactional operations.

Browsing: When a fresh call is received at an Hyperspeech enabled source voice application the Call Handling Service of the HSTP layer registers the call and creates a new HSTP session. When the voice application initiates a session transfer to a second Hyperspeech enabled voice application, the session establishment procedure (as explained in Section 4) is used to transfer the call. Same procedure is applied when the second voice application initiates another session transfer to a third voice application. At this point, the user may issue a request to go back to the second voice application (by issuing a *previous* or *browse_back (n)* command) or, if the third voice application supports it, two steps back to the source voice application. For latter case, the HSTP layer at the third site receives an HSTP session transfer request with the value of HOP_NO set to 2 but the VA2_NO (i.e. target application address) set to the second voice application's phone number which is what it knows about. The HSTP layer sends an HSTP message to the second site with the value of HOP_NO set to 1. This is interpreted by the HSTP layer at the second site and realizing the value of HOP_NO to be non zero, it initiates an HSTP session transfer request to the first site and sets the value of HOP_NO to be 0 in its HSTP message. The call is thus transferred from the target site to "2 hops back" to the source site.

Once the user is back at the first site, she can issue a request to *browse_forward (n)* one or two steps (with n=1 or 2). The command *next* is used to simply browse to the next hop. HSTP will find the details of the site to forward to from its OUTGOING-CALL-TABLE and transfer the session. Thus, multiple hops of forward and back can be supported by HSTP using this mechanism.

Transactions: The same scenario as above could be enacted in the context of a transactional operation spanning across three Hyperspeech enabled voice applications. In such a case, the basic interaction remains the same with only two changes. First, the decision regarding when to transfer the HSTP session is taken by the workflow control logic in the voice application rather than by the user and in most cases the HSTP session would come back to the source voice application. Second, when the session is returned back the APP_CTXT field of the HSTP message contains the results of the operation performed by the application.

6. HYPERSPEECH ENABLED VOICE APPLICATIONS

Hyperspeech content can be incorporated in a telephony voice application to provide a connection to other voice applications. Creating a Hyperspeech link from one voice application to another requires using of the *transferHSTPSession()* function with the correct parameters. Hyperspeech content can also be provided in voice applications for supporting *browse_back* and *browse_forward* commands. Thus the features of HSTP can be used by any voice application through the provided API. Authoring an application with Hyperspeech API ensures that low level programming integration is not required to develop a cross-enterprise voice application. In this section, we describe the implementation of two prototype applications that use Hyperspeech to connect to each other and hence create a new integrated cross-organization voice application. The application is in line with the scenario presented in the paper.

6.1 Implementation and Platform Details

The voice applications were authored in Java and VoiceXML-JSP using IBM Rational 6.0 IDE as the development platform. We use the IBM Voice Toolkit for testing the applications. These applications are deployed in the Apache Tomcat 5.0 application server as shown in Figure 5. This server is connected to the PSTN network through a Dialogic card that forms the Computer Telephony Interface (CTI). A Genesys Voice Browser is used for interpreting the VoiceXML and for interacting with the CTI. Genesys utilizes Websphere Voice Server to enable speech recognition and text-to-speech synthesis when required. The voice applications are represented by a phone number. When a user calls this number, the application is rendered by the Genesys Voice Browser and presented to the user.

6.2 Prototype Applications

We have developed two Hyperspeech enabled voice applications that interact with each other (through HSTP) in order to complete an online transaction. The first application is a tele-grocery store. Users can call the tele-grocery

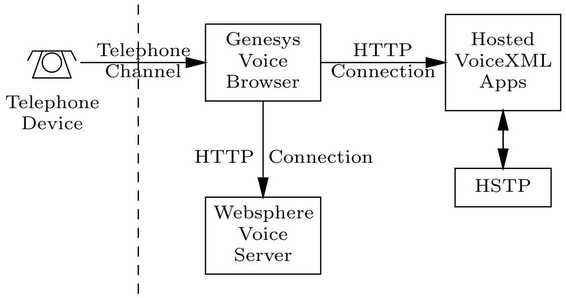

Figure 5: Voice application deployment platform.

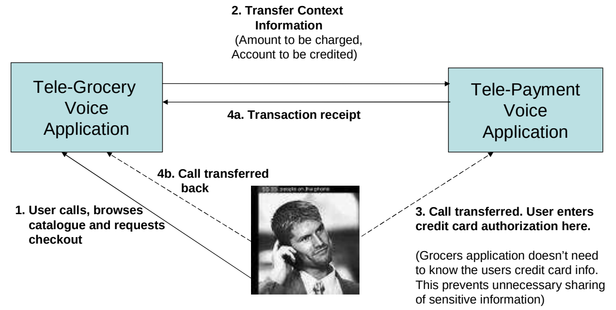

Figure 6: Prototype to demonstrate HYPERSPEECH enabled voice applications

store and order the items of interest. The second application, linked to the first one through a HYPERSPEECH link, is a tele-payment gateway that accepts the user credit card information and charges the user. Figure 6 depicts our prototype setup. Following are the steps that describe the use of HYPERSPEECH in connecting these applications:

1. The user starts by making a call to the tele-grocery store. The call is received by the Genesys Voice Browser and forwarded to the tele-grocery store voice application. The HSTP layer should actually reside between the Voice Browser and the voice application and could actually be bundled with the browser itself. Since Genesys is an external product, in our prototype, it forwards the call to the voice application which in turn interacts with the HSTP module.
2. The HSTP module establishes a new session for the current call. The user can now browse the catalog and add the products of interest to her shopping cart. Once done, she requests the system by choosing the appropriate IVR menu option, to checkout and make a payment. This results in invocation of the HYPERSPEECH link to the tele-payment voice application. As part of this invocation, the voice application supplies payment related information and requests for a session transfer by invoking *transferHSTPSession()* API method of HSTP module.
3. As described in section 5.5, the context transfer web service in HSTP, then looks up the address of the peer web service at the target site. In the current prototype, whenever a *VoiceSite* is created, it is registered in an address lookup service implemented as a registry of web services. After receiving the context information, the HSTP module at the target site returns back an ACK. The HSTP module at source site then transfers the user call to the target site.
4. At the target site, the payment application authenticates the user, processes the payment and then invokes a HYPERSPEECH link to transfer the session back to the tele-grocery store. It sends the receipt of the payment as part of the session context transfer.
5. On receiving the call and the positive ACK, the tele-grocery application stores the payment receipt for its records and confirms the order delivery to the user.

For making web service calls, we use the WSIF [9] framework that allows clients to invoke web services dynamically without having to implement the corresponding web service client stubs. Thus a single dynamic invocation client, implemented in Java, was used.

The implementation of this scenario hints at the numerous advantages of connecting telephony voice applications through HYPERSPEECH. The present implementation shows a transactional call-transfer across voice applications. However the same API can be used to enable seamless browsing of these applications. As an example, the tele-grocery voice application can provide HYPERSPEECH links to a friendly stationary tele-store voice applications so that customers who need stationary items can *browse* to this tele-store once they are done with grocery.

6.3 Implementation Limitations

In the prototype, HSTP functionality is implemented along with the application. To implement HSTP as a layer that intercepts incoming calls and forwards them to applications after appropriate session management, current Voice Browsers need to be upgraded to include HSTP. The telephone call transfer is not automatic in our prototype due to inadequate telephony platform configuration available to us. We plan to address these limitations in the near future.

7. RELATED WORK

In the 1990s, many researchers worked in the area of providing access to the WWW through a voice interface. In [5], the authors propose transcoding the HTML content to make it conversational. The transcoding method is made sufficiently tolerant to the different structure and format of the HTML documents. A user study conducted in [4] reveals that for browsing the WWW, the choice for an interaction modality is governed more by the usability attribute of user satisfaction than by efficiency of use. Most users prefer to use speech over mouse. In [3], skimming has been used as a means to browse the audio data. However, WWW content was not primarily designed to be accessed through voice. Moreover, such systems are typically specific to a particular website and do not address cross-enterprise transactions.

Later, multimodal browsers were also proposed as an easier means to browse the WWW through speech as an alternative input mechanism. In [12], the authors present interface actions on a particular browser (HTML/VoiceXML), convert to events, and distribute them to the other browsers in a multi-modal framework. They define a synchronization protocol, which distributes such events. In [1], the authors present an architecture for a multi-modal Web browser for accessing a patients record system through a phone. A method to generate the multimodal user-interface pages by using XSLT stylesheet has been presented in [7] and [16].

However, most multimodal applications have been delivered successfully over Internet. In the telephony world, the speech interface (over telephony channel) needs to be synchronised with the graphical interface (over a data channel), which is a non-trivial task.

Enterprise voice portals are increasingly being adopted by organisations to provide voice self-service solutions [8]. Voice Portals such as 1-800-555-TELL provide information ranging from airline status to sports update. Airline companies such as United Airline provide a telephony interface to perform ticketing transaction operations. Other examples of transactional IVR is the City of Yellowknife where residents can make tax payment through an IVR [10]. In such IVR system, the merchant's system accepts the information and passes it onto payment gateway over regular IT channels. HSTP provides a safer alternative by providing credit card information directly to the payment gateway IVR.

The Hypertext community has also worked in extending the hypertext to other media. Barry Arons introduced a Hyperspeech [2] application for organizing, sorting, filtering *audio notes*. Since audio is sequential, this application allows browsing to specific nodes in the database. However this application is restricted to browsing within the context of a particular audio database. As an analogy, this work can be considered as a hyperlink *within* a text document, whereas the presented work in this paper is equivalent to system that enables hyperlinks *across* such text/HTML documents. Attempts to present the WWW documents through voice have been made in [11]. The authors present a method to render HTML documents through content and navigational cues to the user. The user can choose any *active link* and thus browse to different positions in the HTML document. However this approach still does not handle browsing across different sites and so is not as *hyper* as the hypertext.

8. CONCLUSION

In this paper, we defined HYPERSPEECH — a mechanism to connect telephony voice applications that are deployed at different sites. We introduced an underlying protocol, HSTP, that provides synchronization of the telephony call with the application logic at the time of call transfer from one site to another. In the design of HSTP, we ensured that no change is required to existing communication standards, both in the IP world and in the circuit-switched PSTN network. Similarly, HYPERSPEECH is also supported without enforcing any changes to current standard voice programming languages such as VoiceXML and SALT. Increased connectivity across the different voice applications is likely to lead to a World Wide Telecom Web [13].

In the future, we plan to investigate into the role of HSTP in a purely Voice-over-IP (VoIP) environment. It is likely that HSTP can be made more efficient by exploiting some of the session management capabilities [17] offered by VoIP, when the last mile connectivity to the user is also over IP. Based on the adoption of HSTP, future voice application specification languages such as VoiceXML and SALT should be enhanced to provide tags specific to handle Hyperspeech content. The Voice Browsers should also be made to support the HSTP protocol.

9. REFERENCES

[1] C. Armaroli, I. Azzini, L. Ferrario, T. Giorgino, L. Nardelli, M. Orlandi, and C. Rognoni. An architecture for a multi-modal web browser. In *International Conference on Spoken Language Processing*, pages 2553–2556, 2002.

[2] B. Arons. Hyperspeech: Navigating in speech-only hypermedia. In *ACM Hypertext '91*, 1991.

[3] B. Arons. SpeechSkimmer: A system for interactively skimming recorded speech. *ACM Transactions on Computer-Human Interaction*, 4:3–28, March 1997.

[4] J. A. Borges and J. J. N. J. Rodriquez. Speech browsing the World Wide Web. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, pages 80–86, 1999.

[5] M. K. Brown, S. C. Glinski, and B. C. Schmult. Web page analysis for voice browsing. In *Proceedings of Intl. Workshop on Web Document Analysis*, 2001.

[6] G. Camarillo, M. Angel, and G. Martin. The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds. John Wiley Publishers.

[7] S. E. Chang. The design of a secure and pervasive multimodal Web system. In *International Conference on Advanced Information Networking and Applications*, volume 2, pages 683–688, 2005.

[8] S. Cramoysan, B. Elliot, and D. Kraus. Magic Quadrant for IVR Systems and Enterprise Voice Portals. *Gartner Research Note G00144047*, Dec 2006.

[9] M. Duftler, N. Mukhi, A. Slominski, and S. Weerawarana. Web Services Invocation Framework (WSIF). In *OOPSLA '01*, 2001.

[10] Finance Department. City of Yellowknife: Payment Options. *http://www.yellowknife.ca/OnlineServices/Payment-Options.html*.

[11] S. Goose, M. Wynblatt, and H. Mollenhauer. 1-800-Hypertext: browsing hypertext with a telephone. In *Proceedings of ACM conference on Hypertext and hypermedia*, pages 287–288, 1998.

[12] J. Kleindienst, L. Seredi, P. Kapanen, and J. Bergman. CATCH-2004 multi-modal browser: overview description with usability analysis. In *Proceedings of IEEE International Conference on Multimodal Interfaces*, pages 442–447, 2002.

[13] A. Kumar, N. Rajput, D. Chakraborty, S. Agarwal, and A. Nanavati. WWTW: The World Wide Telecom Web. In *SIGCOMM Workshop on Networked Systems for Developing Regions*, Japan, Nov 2007.

[14] A. Kumar, N. Rajput, D. Chakraborty, S. Agarwal, and A. A. Nanavati. VOISERV: Creation and Delivery of Converged Services through Voice for Emerging Economies. In *WoWMoM'07 Proceedings of the 2007 International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Finland, June 2007.

[15] K. Meier-Hellstern and E. Alonso. The use of SS7 and GSM to support high density personal communications. In *Proc. ICC*, 1992.

[16] I. V. Ramakrishnan, A. Stent, and G. Yang. Hearsay: enabling audio browsing on hypertext content. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, NY, USA, 2004.

[17] J. Rosenberg, , H. Schulzrinne, and P. Kyzivat. User Agent Capabilities in the Session Initiation Protocol (SIP). RFC 3840, Aug 2004.