A Multimodal Instant Messaging System using XML-Based Protocols

John Lewis and Edwin Blake
Collaborative Visual Computing Laboratory, Department of Computer Science
University of Cape Town
Private Bag Rondebosch 7700
Ph: (021) 650 2670 Fax: (021) 689 9465
Email: {jlewis, edwin}@cs.uct.ac.za

Abstract— **Instant Messaging is traditionally a text only affair. However, there are instances when it would be useful to bridge to other types of media, like speech. The SoftBridge is an application framework that enables this kind of communications bridging using instant messages. Its use of protocols like the Jabber Instant Messaging Protocol and the Simple Object Access Protocol makes it simple, open and extensible. It also allows bridging to non IP communications infrastructure, like the telephone network. We describe the design and architecture of the system, protocol and extensibility mechanism. Finally we describe our experimental methodology and discuss the results of two sets of experiments: those involving a Deaf User, and those involving a set of Simulated Users.**

Keywords—**instant messaging, bridging, XML, SOAP, Jabber, web services, speech synthesis, speech recognition, multimedia**

I. Introduction

Instant Messaging (IM) is becoming the primary means of communication for many people. From the moment they arrive at their office in the morning until when they go home, they are connected. Not only do they use IM to communicate with their friends and colleagues, they also use it to access online services: weather reports, news headlines, stock and currency quotes. Currently, the majority of these systems are text based. Their architecture is server–centric. One participant sends a text message to a server, which then forwards that message to the recipient.

However, there is a growing demand for more than the transfer of text. In some cases, people may want to send simple graphics and images. In other cases, users may simply not be able to handle textual data. Furthermore, there may be limitations on equipment and interfaces that makes text transmission either impossible or undesirable. This where the concept of the SoftBridge comes in [1].

The SoftBridge allows one user or endpoint to send a message to another in the sender's preferred form and format. The SoftBridge, acting like the text relaying server mentioned earlier, converts the message into a form that the recipient can handle. This process may involve one or more *media transformations* that are carried out on the media. Additionally, the SoftBridge can handle n–way conversations, in which many users, using heterogenous media types, communicate in a single chat room.

The system is built upon an open Instant Messaging platform called Jabber[2]. All Jabber messages are passed as XML, and there is a well defined method for extending the protocol. We make use of this facility in the SoftBridge. We also use XML SOAP (Simple Object Access Protocol) [3], the up and coming web based RPC protocol, for implementing our own extensibility framework. The framework incorporates a caching mechanism, as well as a load balancing capability to enable a high degree of scalability.

II. Background

The SoftBridge is an example of a synchronous *mixed media space*. IMPROMPTU, Tattle Trail and Thunderwire are examples of *audio only* media spaces. Telgo323 and TelgoSIP, on the other hand, are limited mixed media spaces. JabCast is not a media space as such — it is more of a universal messaging system — but it makes use of the same core protocol as the SoftBridge.

A. IMPROMPTU

IMPROMPTU [4], [5] is a distributed system that allows audio to be streamed to a handheld device, on request. It also provides an interface on the device that allows the user to correctly select the audio stream they are interested in. They provide *text–to–speech* and *speech recognition* services that allow voice prompts to be streamed to the device, and captured voice commands to be interpreted. The project addresses some important issues regarding the location of the various components of a media handling system. Finally, they address the issue of media transport, and this is where there were some problems. They implemented part of the system in Java, and part of it in C++. In the Java domain, they used Java Remote Method Invocation (JRMI) [6] to signal, and Java Media Framework (JMF) [7] RTP streams to transport the audio. In the C++ domain, they were forced to use sockets and text commands for signalling and for transporting audio. This lead to separate Java and C++ API's for the framework, which were not entirely compatible. They implemented various applications on top of the framework, including a MP3 audio streamer, a baby monitor, a headlines reader, and an interesting telephone that scrambles speech.

B. Tattle Trail

Tattle Trail [8] is "an archiving audio chat application for mobile users over IP." It is built on top of the IMPROMPTU framework, and is intended to showcase the facilities provided by that platform. The idea is that users enter and leave the chat room at will, and when they return, they are updated on the current context of the conversation. They do this by *catching up*, which means reviewing the recent history of the chat session. They can do this at high speed, and much of the project is devoted to the best interaction methodologies for achieving this. There is also a facility to issue *alerts*, which allow users to be offline, but still receive periodic updates of the conversation.

C. Thunderwire

Thunderwire [9] is a shared, audio only media space. It involves broadcasting high quality audio to handheld wireless devices, enabling a fully shared audio environment. This project was one of the most comprehensive, because they conducted a two month long field trial. They also investigated the social implications of the media space.

D. Telgo323/TelgoSIP

South African telecommunications utility Telkom provides a device called the *Teldem* that allows Deaf users to communicate with one another using a standard telephone line. It is essentially a teletype terminal with a small LCD screen, a keyboard and a built in modem. When Deaf user A wants to communicate with Deaf user B, A instructs the Teldem to dial user B's telephone number. User B's Teldem then answers and connects. The conversation is performed by each user typing messages, which then appear on the screen of the other user. Once the conversation is complete, they disconnect. The Teldem only provides for one-to-one conversations between Deaf people.

Telgo323 [10] was developed to allow a deaf user with a Teldem to communicate with a hearing user with a normal telephone. The text from the Teldem is passed to a speech synthesis system, and the resultant audio is fed to the hearing user's telephone. In the reverse, the audio captured from the hearing user is passed to a speech recognition system, and the recognised text is sent to the Teldem. This system was designed to allow a one-to-one conversation between a deaf and hearing user. Unfortunately, due to performance problems with the speech recognition system, only the deaf-to-hearing side of the system was fully implemented. The initial prototype, Telgo323 used the H323 telephony protocol to handle the telephony signalling, whereas a subsequent port, TelgoSIP, uses the Session Initiation Protocol.

E. JabCast Secure Realtime Communications

According to their web site, "The JabCast Secure Realtime Communication System is the first technology that allows for real time interactive text, file and document exchange in a completely secure environment"[11]. However, after close examination of the specification, and the white papers for the implementation, it is apparent that JabCast only allows real time communication using short messages, but transfers other media asynchronously. No content adaption or conversion is attempted by this system.

F. The Jabber XML Instant Messaging Protocol

In an attempt to rationalise and consolidate the Instant Messaging field, the Jabber Foundation [2] was founded to develop an open, XML based instant messaging protocol. Due to it's simplicity and extensibility, the protocol is gaining popularity. It is soon to become an IETF standard named XMPP (eXtensible Messaging and Presence Protocol), which should cause its support to surge. It has a server-centric architectural design, in that messages are passed between clients via a central server.

III. Architecture and Protocol

The inspiration for the SoftBridge came from the partial success of the Telgo* projects: allowing multimodal bridging.

A. Aims

The aims of the SoftBridge system are as follows, in order of priority:
1. Provide an application framework for multimodal bridging.
2. Allow multiuser, multimodal conversation sessions.

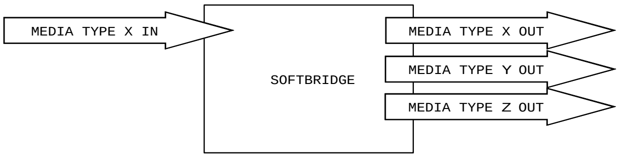

Fig. 1. High level concept

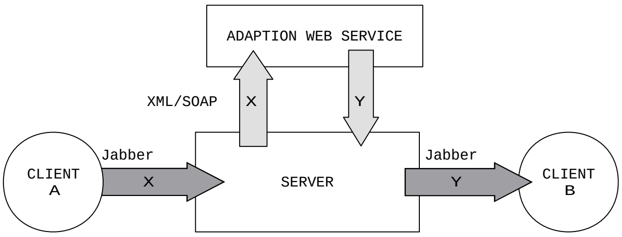

Fig. 2. SoftBridge Data Flow

3. Be open and extensible, allowing the addition of new media conversion services.

Hence we divide the design section into three parts: *Application Framework*, in which we discuss the foundations and basic platform; *Multiple User Support*, in which we describe the multiuser facility; and *Extension Mechanisms*, in which we explain how the system may be extended.

B. Application Framework

The operation of the system, in a highly simplified form, is illustrated in Figure 1

In Telgo323 and TelgoSIP, text data went in from the Teldem, and synthesised audio data came out, eventually finding its way to the hearing user. In the reverse, audio data from a single hearing/speaking user went in, and the recognised text came out, on its way to the deaf user. In the SoftBridge we are attempting to generalise and pluralise this approach. We *generalise* it in the sense that the input and output may be any media type, not just text or speech. We *pluralise* it in that the user relationship is now a one–to–many broadcast, instead of a one–to–one unicast.

Our system is based on the Jabber architecture [12] and consists of the following components:

- Set of Users
- Set of Media Types
- Set of Media Adapters
- The Main Server
- Breakout servers

This architecture is illustrated in Figure 3.

B.1 Users

A User has the following characteristics: *Name*, *Input Media* and *Output Media*. *Input Media* is the type of media that the client can capture from the outside world. *Output Media* is the type of media that the client can render. For instance, for a cellphone, “inputmedia” would be SPEECH, and “outputmedia” would be SPEECH. For a text based chat system, “inputmedia” would be TEXT, and “outputmedia” would be TEXT. It is the responsibility of the client to capture or render the media data.

B.2 Media Types

A Media Type is a symbolic name for a class of media. Common media types are TEXT, SPEECH and VIDEO. The “inputmedia” and “outputmedia” attributes of each user must be one of the registered media types.

B.3 Media Adapters

In a bridging scenario, User A with media type X has sent a message (containing media data of type X) to User B that can only render media of type Y. Hence the SoftBridge must convert the media from type X to type Y. This requires a adapter that can perform conversion XY. The system has a registry of media adaptions, in the form (inputmedia, outputmedia, webservice url).

The media adapters are implemented as XML SOAP web services that must expose a standard interface. This enables the system to access any adapter, query it for media conversion capabilities, and perform media adaptions, simple by knowing its URL.

B.4 The Main Server

The server ties the rest of the components together. The core algorithm is as follows:

```
1. M = Message from User A to User B
2. X = Output media type of User A
3. Y = Input media type of User B
4. if X is not equal to Y
 (a) T = media adapter for XY
 (b) if T exists, perform media Adaption
 (c) Substitute new media into M
5. Forward M to B
6. Go To 1
```

This flow of data is illustrated in Figure 2.

Hence, if a user can handle the media that another user is sending, no adaption takes place. Otherwise, an adaption web service is found and called. The server also takes care of ancillary issues like presence (available, away etc.) and notifications.

Load balancing in achieved by measuring the time it takes for a particular web service to process a request. Based on this measurement, we select the service with the shortest average processing time to handle the request.

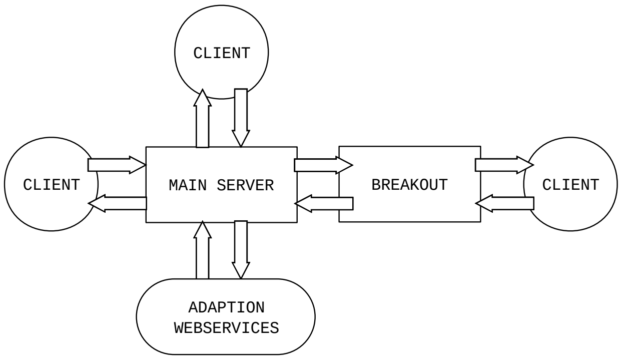

Fig. 3. Overall System Architecture

B.5 Breakout Servers

In some cases, it is not possible for a client to send XML messages straight to the SoftBridge server, mainly because it cannot handle Internet protocol. An example of this is telephony support: telephones cannot handle IP, and cannot send XML formatted strings. Instead, we added a *Telephony Breakout Server*, that acts as a gateway between the Public Switched Telephone Network (PSTN), and the IP/XML world of the SoftBridge.

A telephone subscriber connects to the Telephone breakout by dialling into it. The breakout, in turn, is connected to the SoftBridge. The telephone user is categorised as a SPEECH IN/SPEECH OUT SoftBridge user, so the bridge captures audio and feeds it into the SoftBridge. Messages destined for the telephone based user will get automatically converted into audio by the bridge, and the breakout then streams it down the line. The Telephony breakout can also be configured to perform speech recognition and speech synthesis locally, making it into a TEXT IN/TEXT OUT SoftBridge user. This functionality is used to give instructions and feedback to the user in both modes. In turn the user can control the bridge using either speech commands, or DTMF tones.

C. Multiple User Support

The Jabber foundation has defined a standard way of providing multiple user support [13]. Users join a chat room by sending it an “Available” presence message. The chat room then relays their availability to the rest of the participants. When a user sends a message to the chat room, the message gets relayed to all the users in the room, including the sender.

The SoftBridge chat room component operates in the same way. Incoming messages destined for the chat room do not undergo any media conversion — the chat room is only a relay. The chat room then forwards the message to each of the participants, performing media conversion when necessary.

Unfortunately, this naive approach can cause inefficiency, as several participants may have the same media requirements, and multiple conversions may be performed. In order to prevent this occurring, we implemented a *message cache*, that caches the output messages. When a message M of output media type O is to be sent, the cache is checked. If message(M,O) is found in the cache, the cached message is sent. If it is not found, message(M,O) is inserted into the cache.

D. Extension Mechanisms

The system can be extended by registering additional media conversion web services. These web services must implement the following interface:

```
interface MediaAdapter {
        String[] getInputMediaTypes();
        String[] getOutputMediaTypes();
        String adapt(String inputMedia,
          String outputMedia, String inputXML);
}
```

The actual messages are extended using the method prescribed by the Jabber protocol: special “X” elements are inserted in the XML message, and the media data in encapsulated inside those. A standard, text only Jabber message has the following format:

```
<message from=‘userA@server’
 to=‘userB@server’>
<body>Message Text</body>
</message>
```

An extended Jabber message containing audio, for instance, would be as follows:

```
<message from=‘userA@server’
to=‘userB@server’>
<body></body>
<x xmlns=‘jabber:x:softbridge’>
<audio bits=‘16’ freq=‘22500’
 encoding=‘base64binary’>
  BASE_64_ENCODED_AUDIO_DATA
</audio></x></message>
```

The SoftBridge extracts and sends the entire contents (in this example, the `audio` element) of the X element to the webservice. Depending on the type of media returned by the service, the server then deletes the entire X tag, and inserts the XML data returned from the service.

The webservice URL is registered with the system using the Administration webservice.

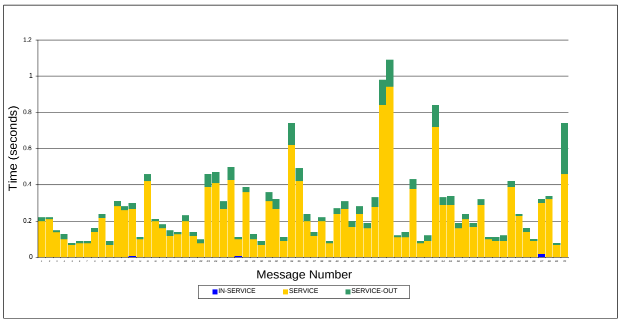

Fig. 4. Time Division for Text to Speech Bridging

IV. Evaluation

We are in the process of evaluating the system in terms of performance and robustness. We are currently carrying out two categories of experiments: "real user" experiments, and "fake user" (or simulated user) experiments.

A. *Experimental Methodology*

A.1 Real User Experiments

In the "real user" experiment, a deaf user has an online conversation with a hearing user. The deaf user is classified as a text in/text out SoftBridge user. During the experiment, the hearing user was either text in/text out, text in/speech out or speech in/speech out. Hence, we were able to test the system in the following situations:

1. text in/text out → text in/text out (no bridging)
2. text in/text out → text in/speech out (text to speech bridging)
3. text in/text out → speech in/speech out (text to speech and speech to text bridging)

The system logs all messages and media conversion operations for measurement purposes.

A.2 Fake (simulated) User Experiments

We have also developed a SoftBridge client that allows chat logs (IRC or otherwise) to be played back into the system. It is possible to specify the media input and output types of the simulated users. It is possible to create logs that test various boundary conditions, such as simulating overload and non-realistic usage. This enables us to get a more complete picture of the system's performance.

B. *Experimental Results*

In this section, we present the results from the single "real user" experiment we have carried out so far. The details are as follows:

- *Number of participants*: 2
  – 1 deaf user.
  – 1 hearing user.
- *Messages passed*: 188
  – *text-text*: 71
  – *text-speech*: 112
  – *speech-text*: 5
- *Average text length*: 32 chars
- *Avg. processing time (all)*: 0.215 sec.
- *Avg. processing time (bridged)*: 0.514 sec.
  – *Text-Speech Avg*: 0.271 sec.
  – *Speech-Text Avg*: 3.873 sec.
- *Avg. Infrastructure Time*:
  – *Text To Speech*: 14%
  – *Speech To Text*: 1%
- *Correlations*:
  – *Total Processing Time/Message Length*: 0.923
  – *TTS Service Time/Message Length*: 0.932

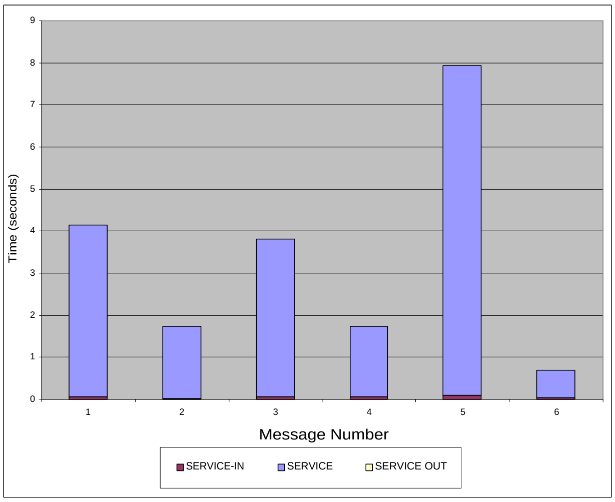

Fig. 5. Time Division for Speech To Text Bridging

C. *Discussion of Results*

The results show that the performance of the system itself (known as the "infrastructure") is adequate for simple one-to-one conversations. Considering the time it takes to either compose or dictate a message, the sub second processing times (avg. 0.215 sec) for text to speech adaption are acceptable, and doesn't cause any appreciable delay. This comment also applies to the processing time for the speech to text adaptions (avg. 3.873 sec).

Figure 4 shows the time breakdown for all messages that were bridged from text to speech: "in-service" is the time between receiving the message and passing it to the conversion service, "service" is the time between passing the message to the service and receiving it back, and "service-out" is the delay between receipt from the service, and transmission to the recipient user. Figure 5 presents the same information for messages that were bridged from speech to text.

It may be observed that for text to speech bridging, the larger infrastructure delay is in the “service-out” component. This is due to the handling of the audio data returned from the conversion service. The same observation is true for the speech to text bridging: the audio data is received from the sender and passed to the service, so the “in-service” component is larger.

It may be also be observed that the largest time component in both bridging cases is the media adaption itself. Unfortunately, this delay is caused by the performance of the speech synthesis and speech recognition systems respectively, and is unlikely to improve in the near future. However, due to the extensible nature of the SoftBridge, it is trivial to add new services with higher performance.

Finally, Figure 6 shows the relationship between the length of a message and the total processing time, which turns out to have a very high correlation (0.923), as expected.

V. Conclusion

The SoftBridge provides an extensible, open platform for developing multimodal bridging applications, being based on XML protocols like Jabber and SOAP.

The system not only allows bridging between different media modalities, like text and speech, but also between different media infrastructures, like the Internet and the telephone network, through the use of Breakout servers.

We have performed some initial experiments, using “real user” data, and the results show that the performance of the system is acceptable, especially after taking the performance of the underlying services (speech synthesis, recognition) into account.

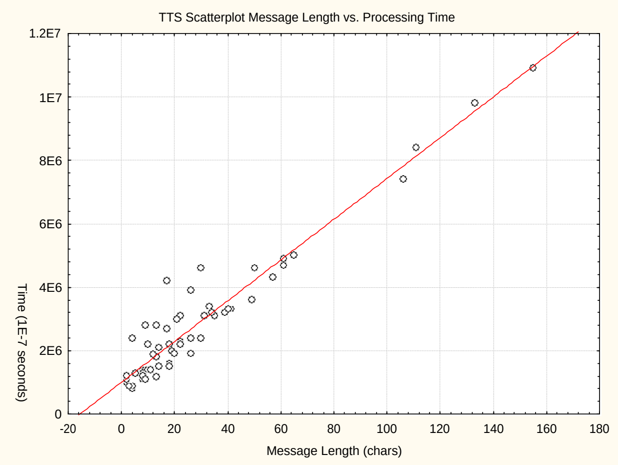

Fig. 6. Comparison of Total Processing Time with Message Length

References

[1] J A Lewis, W D Tucker and E H Blake, “SoftBridge: An Architecture for Bridging the Digital Divide,” in *Proceedings of the South African Telecommunications and Applications Conference (SATNAC 2002), Champagne Sports Resort, South Africa, Sep 4-9 2002 (CDROM Publication)*, 2002.

[2] Jabber Software Foundation, “What is Jabber?,” *http://www.jabber.org/about/overview.html*, 2003.

[3] “The SOAP Forum,” *http://www.soapforum.org*, 2003.

[4] K. Lee, “IMPROMPTU: Audio applications for mobile IP,” *Master’s thesis, Massachusetts Institute of Technology*, 2001.

[5] C. Schmandt and J. Kim and K. Lee and G. Vallejo and M. Ackerman, “Mediated Voice Communication via Mobile IP,” in *Proceedings of the UIST International Conference, pp 141-150*, 2001.

[6] “Java Remote Method Invocation,” *http://java.sun.com/products/jdk/rmi/*, 2003.

[7] “The Java Media Framework,” *http://java.sun.com/products/java-media/jmf/*, 2003.

[8] Jang Soo Kim, “TattleTrail: An Archiving Voice Chat System for Mobile Users Over Internet Protocol,” 2002.

[9] Debby Hindus, Mark S. Ackerman, Scott D. Mainwaring, and Brian Starr, “Thunderwire: A Field Study of an Audio-Only Media Space,” in *Computer Supported Cooperative Work*, 1996, pp. 238–247.

[10] M Glaser and W D Tucker, “Web-based Telephony Bridges for the Deaf,” in *Proceedings of the South African Telecommunications and Applications Conference (SATNAC 2001), Wild Coast Sun, South Africa, Sep 3-5, 2001 (CDROM Publication)*, 2001.

[11] “The JabCast SRC System,” *http://www.jabcast.com.*

[12] Jabber Software Foundation, “Jabber Technology Overview,” *http://www.jabber.org/about/techover.html*, 2002.

[13] Peter Saint-Andre, “JEP-0045: Multi-User Chat,” *http://www.jabber.org/jeps/jep-0045.html*, 2003.