Browser-based Sign Language Communication

Yuanyuan Wang, William D. Tucker
Department of Computer Science
University of the Western Cape, Private Bag X17, Bellville 7535
Tel: +27 21 9592461, Fax: +27 21 9593006
email: {**2918818**, btucker}@ uwc.ac.za

Abstract-This paper describes the design and evaluation of two browser-based video communication prototypes that support sign language communication between Deaf people. The research explores combinations of technologies, protocols and architectures with the hope to eventually provide a mobile video system that Deaf people would want to use enough to pay for. Technology products, and in particular mobile and web-based video communication systems, are designed for the majority of people in general. These are not necessarily suitable for Deaf people who have very different physiological and cultural needs. We focus on browser-based video transmission because end-users need not struggle with application installation. Web-browsers are also common on mobile phones. This paper compares two prototypes built with Adobe Flex and HTML5, H.264 and H.263 video codecs, and PC and mobile phone implementations. The paper describes the motivation, related work, methods, prototype design and finally analyses results of user experiments conducted with Deaf users.

Index Terms— Network services, web services, mobile services, video codecs and protocols

I. INTRODUCTION

This paper describes the implementation of two browser-based video communication prototypes with different video codecs, and compares and evaluates the video quality of the two prototypes. The target community for the use of the technology is the Deaf Community of Cape Town (DCCT), an NGO (non-governmental organization) that supports disadvantaged Deaf people. Deaf with a capital ‘D’ denotes people who use sign language as their mother language. The distinctions between the terms "deaf", "Deaf", and "hard of hearing" are based principally on the individual's preferred language (spoken or signed) rather than on the actual degree of hearing loss.

According to census statistics, there are roughly 4 million people with hearing impairment in South Africa [3]. Of these, 10% are profoundly Deaf, and they use South African Sign Language (SASL) as the primary means of communication. SASL has a totally different grammar and structure from English. In South Africa, and with the community that DCCT serves in particular, most Deaf people are under-educated and under-employed due to a combination of physiological and socio-economic factors [4]. Without text and computer literacy, and unable to speak or hear, Deaf people find text communication difficult. That said, Deaf users frequently use SMS (Short Message System) with both Deaf and hearing users. However, their awareness of poor grammar and spelling in English embarrasses them and inhibits them from using text to communicate with hearing people they know are more literate than they are. Thus, Deaf people prefer to communicate in sign language.

The two browser-based video communication prototypes can provide a sign language communication service between Deaf users. The research question is to explore how to design and evaluate browser-based video communication systems such that Deaf people will actually want to use them, and pay for the service if they deem it good enough. In order to find out what is 'good enough' for Deaf users, we designed, tested and compared two browser-based prototypes that provide semi-synchronous and/or asynchronous, as opposed to synchronous, video for Deaf users. This project is inspired by a prior semi-synchronous Deaf communication project, tested with DCCT members that adapted the synchronous x264 codec for asynchronous video in a standalone application [1]. Our focus is on a browser-based system because it is always on-line and can be used at anytime, anywhere, on any device. Browser-based systems already transmit video and audio data over the Internet. However, a significant problem is that the existing browser-based systems are not suitable for the specific requirements of Deaf people that wish to communicate in sign language. First of all, some of the solutions are not open source. Some have low quality video. All of them include voice because the video conferencing systems are actually meant for hearing users.

This paper describes two open source browser-based video systems implemented specifically for sign language communication. We choose Adobe Flex and HTML5 to construct the two prototypes. Both Adobe Flex and HTML5 are well known browser-based development technologies. Adobe Flash has both synchronous and asynchronous capabilities. HTML5 has asynchronous capability. This research aims to help Deaf users to access advanced network technology easily within a browser. Our methods therefore focus on both technological and social factors. In our opinion, Deaf culture and user behavior has an influential effect on the types of technologies we should use. Therefore, user inclusion is performed during the course of the project.

The paper is organized as follows: Section II describes work related to currently available open source generic browser-based video systems, and also some dedicated standalone Deaf video systems. Section III presents research methods including user requirements, their analysis, and evaluation procedure and experimental design. Section IV describes prototype implementation. Section IV details the results achieved, and Section V concludes the paper and suggests avenues for future work.

II. RELATED WORK

Work related to our browser-based Deaf video prototypes can roughly be divided into three categories: technologies that can be used to build such prototypes, reference implementations for browser-based video although built for hearing, and not Deaf, users and finally, video systems explicitly tailored to support sign language communication between Deaf people.

A. Technologies

Adobe Flash is a common, yet proprietary, way for users to exchange video and audio data over the Internet. Many browsers support Adobe Flash with plug-ins. There are two ways to transmit media data between a server and a client using Adobe Flash [5]. Firstly, video media can be transferred asynchronously as a download with Hypertext Transfer Protocol (HTTP). This method practically guarantees a high standard of video quality that is primarily dependent on the host machine's processing capability at the expense of the delay incurred to wait for the media to download. Secondly, video can be streamed with the Real Time Media Protocol (RTMP) [6]. In this way, the bandwidth availability determines the quality and speed of the video playback. Streaming can be real-time, or continuously start and stop thus providing a mixture of real-time and asynchronous transfer, making it semi-synchronous in nature. Video quality can also be artificially degraded to improve the streaming speed.

Another technology is HTML5 [7], the newest version of Hypertext Markup Language (HTML), the core markup language of the Internet web pages. HTML5 is a revision of HTML4. HTML5 adds new tags and new Application Programming Interfaces (APIs), and incorporates web forms 2.0. HTML5 supports live audio and video in a web page. This new character of HTML5 makes it a possible alternative to Adobe Flash when building browser-based media services.

XML [8] is a general-purpose language that is used to create a set of markup languages for individual responses. A markup language is a computer language with a logical structure beside the data. XML is classified as an *extensible* language, for it can be used to define specific tags for a given user application. Each XML tag is used to mark each part inside an XML document. Tags always appear in pairs. An XML document can be handled by using the Document Object Model (DOM) [9]. An XML file can be taken as a tree structure, and each node in this tree structure has its own type, name, value and attributes. DOM is used to set and get these nodes, and adjust the positions of nodes. The most important aim of XML is to build a bridge between two different information systems. The documents and data of the two systems can be shared and exchanged easily by using XML [10]. For example, NewsML [11] is an extended XML format. It is used in Japan as a standard format in the group of Japan Newspaper Publishers & Editors Association. Japanese newspaper agencies are able to get big headline news from the major newspaper companies easily through NewsML transmission.

Before introducing video codecs [12], the difference between the media file and the codec should be clarified. A media file is a container to store video and audio data, often with some scripting. The algorithm used to compress video and audio data is the codec. A video codec is a technology, often embedded in a device, to compress or decompress digital video data. A video codec represents a fundamental analogue dataset in a digital way. A typical video codec model includes the following steps: decoding and sampling, input processing, output processing and encoding. In a video communication system, the size of the video frame and sequences are determined by the codec. Video codecs have a significant impact on video quality [13]. H.264 [14] is a standard for video compression. This standard is also called H.264/MPEG-4 AVC or AVC. H.264 has a number of new features that make it particularly more efficient than the previous codec standards in a variety of network environments. The key new features include multi-picture inter-picture prediction features, lossless macro-block coding features, and flexible interlaced-scan video coding features and so on. The aim of H.264 is to present a better video quality at low bit rates than the previous standards such as H.263 and MPEG-2.

B. Reference applications and projects

Browser-based video communication systems are commonly available. We are interested in open source solutions because we can examine the architecture. Tokbox (www.tokbox.com) is a browser-based communication system that supports live video with Adobe Flash. Tokbox users need not install or download specialty plug-ins in the client. Tokbox is like a web version of Skype (www.skype.com) without PSTN (public switched telephone network) breakout. Dimdim (www.dimdim.com) is another browser-based system based on Adobe Flash. It is open source and supports multi-user conference. However, Dimdim users need to install custom plug-ins to use advanced features such as desktop sharing. Vmukti (www.vmukti.com) is another browser-based open source system for conferencing. Vmukti is built on the .NET framework, and is therefore much different from Tokbox and Dimdim. This means that Vmukti users need to install .NET in order to use it, and this system only runs on the Windows operating systems.

The frame of view for all of these systems leans toward the 'floating' head to support (and not replace) audio communication, and the video frame rate and resolution appear suboptimal in order to prioritize voice traffic for hearing users.

C. Video for sign language

The best example of mobile Deaf video research is MobileASL (mobileasl.cs.washington.edu). To balance the video quality and bandwidth issues, this project uses skin detection algorithms to find important areas in the video, called regions of interest, and focuses on the movement within these areas only. These are areas of the body that are most used to communicate in sign language and are outside the 'floating' head frame of view from the neck up, and include the torso and areas peripheral to the chest. The sign language in MobileASL is ASL (American Sign Language). The real-time video codec used by MobileASL is H.264.

In 2008, a research project on Deaf video communication was implemented based on a high quality asynchronous video service. The project developed a semi-synchronous video communication standalone application with high video quality and minimal latency [13]. To evaluate the QoS (Quality of Service) of the application to see if it satisfied

Deaf users, an objective video quality measurement tool called MSU (Moscow State University) video quality measurement tool was used to gather objective data such as frames per second. In addition, user observation and interviews with Deaf users were used to collect subjective data. Triangulated objective and subjective results showed that H.264 could be adapted to provide quality asynchronous video communication to support sign language communication.

III. Methods

We wish to combine the features covered in the previous section with the end goal of browser-based video on a mobile phone. In order to move in that direction, we employed an iterative mixed qualitative and quantitative method. Firstly, we intentionally involved Deaf users from DCCT. Secondly, we leverage quantitative methods to objectively measure and analyze prototype performance. The result from each iterative cycle guides the research effort of the design and evaluation of browser-based video prototypes. Each iteration is intended to affect some change in the prototypes to meet the requirements of Deaf users gathered from the previous iteration, similar to a user-orientated spiral model in the software development life cycle. An iteration starts with planning and moves through development, evaluation and analysis, spirals up and re-enters the planning stage.

We developed multiple prototypes in order to perform evaluation and analysis. A simple prototype is built and evaluated quickly. Then the prototype is intensified based on analysis of user feedback. Various prototypes, say A and B, are not necessarily developed at the same time, as versions of A and B appear throughout the spiral of software life cycle. In each cycle from the planning to the analysis, a traditional waterfall model is used. Each phase of the waterfall transforms an outcome of the previous step into the income of the current step, and produces a new outcome as output.

The technical system development methodology in this paper is a prototyping approach that is a vector triangle with three axes: human-centered qualitative research methods, quantitative methods to collect metrics and iterative software engineering methods. User involvement produces valid user requirements and evaluation that are supported by quantitative data analysis. The iterative software engineering method adjusts the direction and produces a series of prototypes.

With human-centered research, prototype design is driven by the collection of user requirements, analysis of those requirements and the user interface. We also need to measure video quality and require an overall experimental design to combine these activities. This section describes each of these issues in turn. Prototype implementation issues are presented in the next section.

A. User Requirements

The target group for this research is Deaf South Africans. We have the opportunity to work with a representative sample in the form of the staff and social workers of DCCT, located at the Bastion of the Deaf in Newlands, a suburb of Cape Town. The DCCT staff and social workers act as research participants. In the planning phase, we also got help and ideas from another Deaf NGO, SLED (Sign Language Education & Development) staff, who taught us South African sign language for six months.

DCCT has supported a community of nearly one thousand Deaf people in the Cape Town area since year 1987 [15]. Many DCCT members have poor levels of spoken, written and reading literacy in any of the eleven official South African languages. They use SASL as their primary language for the daily communication [15, 16]. DCCT exemplifies Deaf cultural pride along with the illiteracy, physiological impairment and underemployment of many Deaf South Africans, particularly those that are historically disadvantaged [4]. Therefore, DCCT provides the local Deaf community with a wide range of benefit programs, such as group work and community development.

The Deaf people in this community have two particular characteristics relevant to the technology research. The first is that they use sign language as the primary language to communicate with other Deaf people. The second is that they have limited computer literacy. Only a few communication applications are used by them, such as Skype and Camfrog (www.camfrog.com), and from observation and interviews we know that they do not use these very often simply because most Deaf people in this community do not have PCs at home or advanced cell phones. In fact, the only real Internet access they have is at the Bastion, and they also battle to physically get to the Bastion because of problems with public transport, especially its cost.

The issue of sign language means that these people have specific requirements that are fundamentally different from the majority of Internet communication users. The issue of textual and computer (not sign language) illiteracy means that they are unable to grapple with commonplace Internet-based communication software. We must endeavor to address these issues in our prototypes.

We collect user behavior data in three ways to understand and analyze user requirements. Firstly, we record computer usage and gross bandwidth consumption from 2007 in the computer lab at the Bastion. We analyzed this data and saw what Deaf users actually do when they use those computers (see Table 1). Secondly, we visit with Deaf participants at the Bastion once a week since the beginning of 2009. We communicate with Deaf users face-to-face, using SASL ourselves and/or with a sign language interpreter. Thirdly, we explain the project to Deaf users and used a questionnaire to collect data about technology usage. We analyze both quantitative and qualitative data, and can therefore build informed prototypes on both PC and mobile platforms. We focus on the user interface and video quality to support sign language communication, like similar projects, e.g. [17], [18].

As mentioned before, the following table shows how DCCT members use computer and network. The total user number has increased from 2007 to 2009 while the total login times has increased and then dropped down. Deaf people use mail as their major communication application. In the meanwhile the number of mail usage has decreased from 2007 to 2009. We are considering it might because mailbox could not fit their communication requirements. It is easy to see that the usage of video chat software does not increase rapidly. The usage of both Instant Messaging (IM) and video chat are up and down in the three years. It seems like Deaf people tried to use IM applications and video chat applications, but they gave up finally.

TABLE 1

Year	Login Times	User Number	Internet Item		
			Instant Messaging (e.g. MSN)	Video Chat (e.g. Skype)	Mail (e.g. Gmail)
2007	929	92	8	4	661
2008	1025	129	51	11	245
2009	677	157	11	8	112

This data describes what Deaf users did with the Internet at the Bastion from 2007-2009.

B. *Requirement analysis*

Due to exposure to technologies to support Deaf communication since 2000 [4], and the introduction of a computer lab to the Bastion in 2004, Deaf users associated with DCCT have attained varying degrees of computer literacy. Table 1 show that many Deaf users are familiar with email. We therefore built the prototype-Flash and prototype-HTML in the style of an email client.

C. *Measurement of service quality*

This project gathers both objective and subjective data to evaluate service quality. We record the usage of server resource and evaluate the performance of the system. The objective data is analyzed with linear graph. Subjective data is collected with user observation, interviews and questionnaires. We visit DCCT weekly to perform user observation and gather participants' feedback. These visits, combined with the study of South African sign language, and our relationship with DCCT as an organization, provide opportunities to comprehend the Deaf community deeply. As we involve ourselves with the target community, we come to appreciate Deaf culture and user behavior. The understanding enriches our thoughts and helps us consider system design from alternative viewpoints, for example as hearing mobile phone users we might not see that the high resolution camera is always on the wrong side of the phone for high quality enough video to support sign language communication, or that a Deaf person must put the phone down in order to sign with both hands. Overall, such subjective understandings combined with objective quality comparisons triangulate to yield informed prototype design and evaluation.

D. *Experimental design*

The experimentation consists of three phases. The first phase is to 'system test' prototypes in the laboratory [19]. The second phase examines prototypes with a few participants in a laboratory environment, e.g. two Computer Science students who took a six month sign language course and two DCCT staff. These participants have experience with computer software. A questionnaire is prepared concerning the prototype and the experienced volunteers answer it and give feedback. The third phase tests prototypes in a real world environment at the Bastion, as in [20]. Five DCCT members use the prototypes and provide feedback. A questionnaire is prepared for, and answered by, the five participants. Video data of the Deaf users are recorded automatically by the prototypes with their consent. We ask Deaf participants questions about the prototypes using a sign language interpreter, such as: what functions confuse them, what they like in a given prototype and why they like it. The test data is collected and analyzed at the end of each phase. Evaluation and the next round of design are based on this data and its analysis.

IV. IMPLEMENTATION

This part shows how technologies and protocols were combined to build the two browser-based prototypes: prototype-Flash and prototype-HTML. Both prototypes run only on a PC at this time. The Flash prototype is a real-time tool and the HTML5 prototype is asynchronous.

A. *Prototype-Flash*

Prototype-Flash: Figure 1 shows the steps to start a video chat in prototype-Flash. When both client A and client B decide to start a chat, an Adobe Flash setting dialog is shown on each client. The Flash server stores the video stream from client A temporarily after client A agrees to open his webcam from the setting dialog. To obtain faster streaming, the voice is ignored because it is not needed. The Flash server manages all video streams by mapping each stream to a unique username of each client. The video is published and any client can get the video stream if she knows the unique username of the publisher. In this prototype the username is the login name. In prototype-Flash, many common features of communication software are provided such as text chat and user profile modification. The Flash server handles an online user list to store information about online users. This list is a shared object [21] that is shown in client. The administrator can add a new user, delete a user or modify user data, while a guest cannot control other users' data. The text chat data is also stored in a shared object.

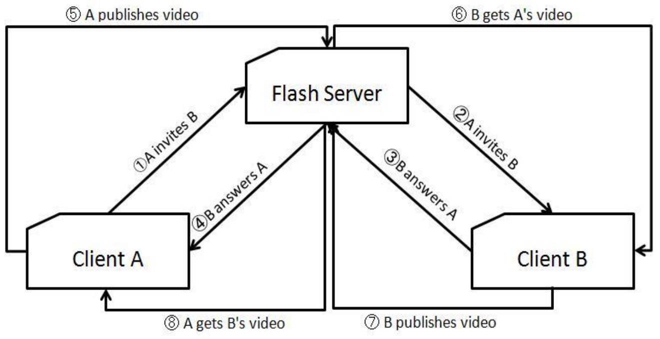

Figure 1. The figure shows how a two-way video streaming starts using Adobe Flash technology.

B. *Prototype-HTML*

HTML5 supports online audio and video playback in a web browser via HTTP. This prototype provides one-way video streaming from server to client. Neither HTML5 nor JavaScript provides for local video capture. Thus the user must capture a video manually, and data is sent asynchronously. We enact the HTTP connection with the Document Object Model (DOM) Application Programming Interface (API) called XMLHttpRequest [22]. This API is used in web browser scripting languages, like JavaScript. XMLHttpRequest can create a connection with the web server, sending HTTP requests directly to the server and handling the responses from it. We use XMLHttpRequest to get a video message and user profile data, while using form submit to post a video file to the server. There are five HTTP connection types in prototype-HTML when a client connects to the server via XMLHttpRequest: login, logout, new video message check, message delete and user profile change.

When a user logs in, the server prepares a work folder for the user, and reads the user profile into the online user list. The contact list of the user and video messages for him are sent to the client. A video file information list is presented when a user logs in and new video messages are listed out. All video files in the list are sent from other users to this client. The video files are stored and managed on the server; only the link addresses and file information data are sent to the client through a XML format.

The client gets the XML data and shows the data in the web browser. The user can playback each video message via the link address that is stored in the attribute "url". The user can modify his profile and delete all video messages by sending requisite requests. A modified profile is written into a XML file when user logs out.

Apparently due to security concerns, JavaScript does not provide an API to upload a local file onto the web server. We therefore use an HTML form submit to upload a video. The header information and video data are combined and sent together. The server splits the request data into header and video stream, and saves the stream as a video file in a work folder. The file information is saved into an XML file in the same folder.

V. Results

In this paper we are going to detail the result of phase one and phase two only. In the first phase of experimentation, the performance quantitative data is gathered via system test. The memory and bandwidth usage data was monitored and logged on the server side. The performance data help us to analyze if prototype-Flash and prototype-HTML could run continually and stably. The result of analysis will be referred in the next step of development also. We used a virtual machine with Windows XP OS as the server. Windows performance monitor [23] and Performance Analysis of Logs (PAL) are used to record usage data and evaluate the data. Figure 2.1 and figure 2.2 show the CPU and memory usage history of prototype-Flash and prototype-HTML. The blue line gives us an idea about the percentage of processor to handle user process. In this case user process means the conversation between client and server. The red line illustrates how many memory bytes are available. It is easy to see that prototype-Flash spends all CPU resources from the start even when there is no video chat starts. Prototype-HTML expends CPU resources only when a communication starts. In both figures the red lines are almost straight. It shows prototype-Flash and prototype-HTML do not ask a lot of memory during conversations.

About network workload, we monitored transferred bytes throughout the network. The pink line draws the total bandwidth in real-time. The aqua-blue line specifies how many bytes of data the server is received. Figure 2.1 points out that in prototype-Flash client and server do not exchange data if there is no two-way communication. Although the server spends a lot processor resources, it releases part of the resources during a video chat starts. The highest percentage of bandwidth expended on conversion is about 50%. Figure 2.2 indicates that prototype- always consumes some bandwidth even when there is no conversion. The basic percentage of bandwidth expended by our prototypes is about 60%. The server uses a little more network resources when a communication starts.

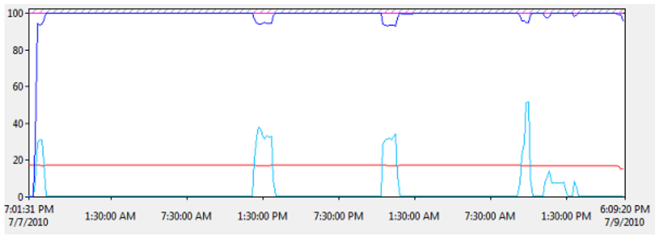

Figure 2.1. This figure shows two days performance log on prototype-Flash server.

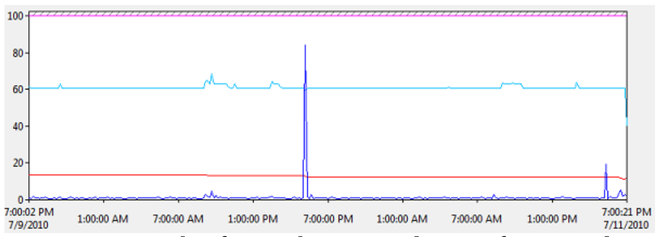

Figure 2.2. This figure shows two days performance log on prototype-HTML server.

In the second phase, the qualitative data about user feedback is collected throughout questionnaire. The four test subjects, two Deaf and two hearing, gave both prototypes a positive evaluation, and considered the video quality to be acceptable. We were informed that both Deaf users would like to use the prototypes in actual life. Table 2 presents an overview of some of the feedback. Both the computer science students and the Deaf users gave prototype-HTML higher evaluation about video quality and user interface. One of the subject said he thought prototype-HTML included more interactive elements and gave him a better interaction experience. However, the overall impression of prototype-Flash and prototype-HTML are the same. Subjects considered both of the two prototypes could be good communication tools, yet their quality can still be improved. Subjects were more satisfied with the QoS of asynchronous video. They were not concerned about real-time communication very much.

TABLE 2

	Average Point (0–100)		
	Overall Impression	Video Quality	User Interface
Prototype-Flash	75	75	75
Prototype-HTML	75	90	95

Average points of prototype-Flash and prototype-HTML are shown.

VI. Conclusions and Future Work

We believe that browser-based sign language communication to be promising technology on both PC and mobile platform for Deaf users. Section I introduced the motivation and the background about this research. The aim of the project is to build and test out two browser-based sign language communication systems. Section II presented the related work which is the reference of this paper. Section III addressed the research methods. The qualitative method and quantitative method and software engineer method work together and point to our research direction. Section IV detailed the implementation. Section V described the result we got. From the data gathered, it seems that the prototype-HTML is more popular with our audience. Prototype-HTML server also spends less computer resources than prototype-Flash. However, prototype-Flash uses bandwidth cleverly.

We have not yet tested the two prototypes with Deaf people with more limited computer skills. Furthermore, neither prototype can run video communication on both PC and mobile phone. In the final third phase, we will attempt to port the best prototype, according to data triangulation, to a mobile phone. The mobile version should be similar to cellular video conferencing and/or Short Message Services (SMS), depending on the temporal modality, real-time or asynchronous. Some of the physical problems associated with mobile devices we are unable to fix, such as having the high quality video camera next to the display and having wide angle camera to view the torso of a signing user instead of the 'floating' head.

ACKNOWLEDGMENT

The authors thank the staff and members of Deaf Community of Cape Town (DCCT) for their participation in the project. We also thank Telkom, Cisco and THRIP for financial support via the Telkom Centre of Excellence (CoE) programme. THRIP funding is managed by the National Research Foundation (NRF). Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and therefore the NRF does not accept any liability in regard thereto.

REFERENCES

[1] A. Cavender, R.E. Ladner, and E.A. Riskin, "MobileASL:: intelligibility of sign language video as constrained by mobile phone technology," in *Proc. 8th ACM SIGACCESS Conf. Computers and Accessibility*, 2006, p. 78.

[2] P. Ladd, *Understanding deaf culture: In search of deafhood*, Multilingual Matters Ltd, 2003.

[3] D. Miller, K. Gyllstrom, D. Stotts, and J. Culp, "Semi-transparent video interfaces to assist deaf persons in meetings," *in Proc. 45th Annual Southeast Regional*, 2007, p. 506.

[4] M. Glaser and W.D. Tucker, "Telecommunications bridging between Deaf and hearing users in South Africa," in *Proc. CVHI 2004*, 2004.

[5] J. Gay and S. Allen, "Macromedia Flash Communication Server MX: Use cases and Feature Overview for rich Media," *Messaging and Collaboration*, 2002.

[6] A. Fecheyr-Lippens, "A Review of HTTP Live Streaming," 2010.

[7] I. Hickson and D. Hyatt, "HTML 5," W3C Working Draft. http://www. w3. org/TR/html5/, vol. 25, 2008.

[8] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML) 1.0," W3C recommendation, vol. 6, 2000.

[9] A. Le Hors, P. Le Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion, and S. Byrne, "Document object model (DOM) level 3 core specification," W3C Recommendation, 2004.

[10] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML) 1.0," *W3C recommendation*, vol. 6, 2000.

[11] M. Kodama, T. Ozono, and T. Shintani, "An Implementation of a NewsML Management System using Meta Data," in *Proc. the Annual Conference on JSAI*, 2006, pp. 3B3–3.

[12] I.E. Richardson, *Video codec design*, Wiley, 2004.

[13] Z.Y. Ma and W.D. Tucker, "Adapting x264 to asynchronous video telephony for the Deaf," in *Proc. South African Telecommunications Networks and Applications Conference 2008*, 2008.

[14] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H. 264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, 2007, pp. 1103–1120.

[15] W.D. Tucker. "Softbridge: a socially aware framework for communication bridges over digital divides," Ph.D. thesis, Dept. Computer Science, UWC, Bellville, SA 2009.

[16] R.D. Aarons and P. Akach, "6 South African Sign Language: one language or many?" in *Language in South Africa*, Cambridge Univ. Pr, page 127, 2002.

[17] R. Campbell, "Categorical perception of face actions: their role in sign language and in communicative facial displays," *The Quarterly Journal of Experimental Psychology*, vol. 52, 1999, pp. 67–95.

[18] L.J. Muir and I.E. Richardson, "Video telephony for the deaf: Analysis and development of an optimised video compression product," in *Proc. tenth ACM International*, Multimedia, 2002, p. 652.

[19] T. Kallio and A. Kaikkonen, "Usability testing of mobile applications: A comparison between laboratory and field testing," *Journal of Usability Studies*, vol. 1, 2005, pp. 4–16.

[20] I. Bongartz, A.R. Conn, N. Gould, and P.L. Toint, "CUTE: Constrained and unconstrained testing environment," *ACM Transactions on Mathematical Software*, vol. 21, 1995, pp. 123–160

[21] P. Tandler, T. Prante, C. Müller-Tomfelde, N. Streitz, and R. Steinmetz, "Connectables: dynamic coupling of displays for the flexible creation of shared workspaces," in *Proc. 14th Annual ACM Symposium on User Interface Software and Technology*, 2001, p. 20.

[22] L.D. Paulson, "Building rich web applications with Ajax," *Computer*, vol. 38, 2005, pp. 14–17.

[23] M. Knop, J. Schopf, and P. Dinda, "Windows performance monitoring and data reduction using watchtower," in *Proc. 11th IEEE Symposium on High-Performance Distributed Computing*, 2002.

Yuanyuan Wang is currently studying for a Masters degree at the University of the Western Cape (UWC) with the Bridging Applications and Networks Group (BANG). Her research interests include browser-based application, mobile phone application development and Deaf community.

William D Tucker is a Senior Lecturer in Computer Science at UWC and leads BANG research there. His main research interests include ICT4D project and network technology.