User-Centred Design of an MBone Videoconference Polling Tool

Andrew Patrick, Ph.D.

Communications Research Centre, Ottawa, CANADA

E-Mail: Andrew.Patrick@CRC.doc.ca

WWW: http://debra.dgbt.doc.ca/~andrew

Version 3.3

March 11, 1998

Contents

Abstract

The "MBone" is the portion of the Internet that has implemented multicasting and videoconferencing is the most common MBone application. The MERCI research project includes an analysis of user needs and opinions related to MBone videoconferencing. To facilitate this work, a new MBone polling tool (MPoll) has been developed according to user-centred design techniques. First, design specifications were developed based on experience with the MBone applications and an evaluation of user needs. Next, a proposed user-interface (UI) design was sketched on paper. This UI design was implemented in a non-functioning prototype (using Perl/Tk) and MBone users evaluated it for familiarity and ease of use. Suggestions for improvements were used to revise the design, which was then implemented in Tcl/Tk and C. The UI component was combined with a multicast networking component to form an operational program. The program was subjected to repeated Alpha and Beta tests where MBone users provided comments and the program was improved. MPoll is currently being used to collect real-time ratings and opinions from MBone videoconference users. MPoll is also useful as a decision support tool during remote meetings, and as a general polling tool.

Introduction

MBone Videoconferencing (MVC)

Multicasting is a new computer network protocol that allows for the efficient distribution of network traffic to multiple users simultaneously (Kumar, 1996; Macedonia & Brutzman, 1994). The "MBone" is the portion of the Internet that has implemented multicasting. Videoconferencing is the most common MBone application in use today. Typical videoconferencing applications digitise images and sounds at one site, encode them into a standard format, and transmit the resulting data over a network connection. At the receiving site the data is collected, decoded, and presented to the user. The multimedia data produced during videoconferencing is very bandwidth demanding. Audio encoding tools (such as VAT and RAT; see Hardman et al., 1995) typically require 64 Kbps, while low quality video tools (such as VIC; see McCanne & Jacobson, 1995) require 128 Kbps for displays of 2-5 frames per second. If multiple copies of these media streams were required to service each member of a videoconference then most Internet connections would become congested. Multicasting provides a distinct advantage for videoconferencing because of its ability to efficiently distribute network data to multiple users without redundancy on the network connections.

The MERCI Project

MERCI (for Multimedia European Research Conferencing Integration; see Kirstein & Bennett, 1997) was a large research project who's goal was to further the development of multimedia conferencing tools. The MERCI project consisted of a number of work packages, one of which was concerned with "usability and assessment". This package was responsible for ensuring that user needs are fully understood and that the MBone videoconferencing (MVC) tools were usable. A number of techniques were used to evaluate usability, including heuristic analyses, questionnaires, user observations, and experimental trials. One assessment tool that was missing, however, was a real-time rating tool that could be used during a MVC session. Such a tool could be used to rate the quality of the tools or the multimedia data being received. Further, the ratings could be updated as network conditions change or different conference components come into use. Ratings could also be made about the quality of the sessions themselves, in terms of the effectiveness of the communication (e.g., did the session meet its business or educational goal). Without such a tool, users have been using the shared white board and text editor tools to record comments about the session and collect anecdotal reports. A more systematic approach was needed.

Towards a Network Polling Tool

A review of the literature revealed some previous research on gathering timely opinions from users during videoconferencing sessions. Isaacs et al. (1994, 1995) described a remote-lecturing tool called "Forum" that allowed a speaker to make a presentation to a group of remote listeners. Forum sends audio, video, and slide data to all the recipients using the multicasting protocol, in a manner similar to the MVC tools. The remote listeners use a similar interface to view the presentation. One component unique to Forum was a polling function where the speaker could ask multiple-choice questions of the audience and they could give their responses. The responses were displayed in a shared histogram that was updated whenever a new response was given. Isaacs et al. found that the polling tool was valuable for the speaker to maintain a sense of contact with the audience during the remote presentation. It was also useful for determining the interests and background of the audience.

Our own research on remote meetings has also found a need for decision support tools. Having a tool to present an issue, list the alternative solutions, and collect the opinions is very valuable for conducting effective remote meetings. Thus, it appears that a general-purpose opinion-collecting tool would be a useful addition to the MVC tool kit.

Design Specifications

With this history in mind, some design specifications for a new polling tool were developed. The goal was to develop a tool that would collect ratings and opinions from MVC participants. The tool should support both quantitative and qualitative data in a systematic fashion. At least three types of questions should be supported: multiple-choice with a single response, multiple-choice with multiple responses (a checklist), and short-answer text responses. The tool should support distributed question generation and data collection so that each user could create new questions and maintain their own tally of results.

Another design specification was to collect and distribute the results in real-time, or at least as quickly as possible. In addition, responses to poll questions could be changed and updated in a real-time fashion. This is necessary if opinions change during the course of a MVC session or questions are only appropriate for certain periods of the session.

In terms of the user interface, an easy, intuitive interface was desired. In addition, a simple graphical display of the results is needed to support quick decision-making during a MVC session. The interface should appear similar to the current MVC software and behave in a familiar fashion. Further, the tool should be started from the session directory (SDR) like the audio and video tools so it can be added as a standard MVC component.

A network protocol that could handle multicast network outages and packet loss was needed because these are common occurrences on the MBone. Finally, a practical requirement was to develop the tool quickly so that it was available to the MERCI partners within the period of the project. Further, since the MERCI group used a number of computer platforms, the tool had to be portable to a variety of systems. In addition, the tool should not use any proprietary software since the author's institution makes new technologies available for license by companies who can develop commercial products.

User-Centred Design

Pencil and Paper Interface Sketch

The first step in the design process was to develop pencil and paper sketches of the user interface. Using the design specifications developed previously, a graphical user interface was sketched. This interface looked similar to the interfaces found in the session directory (SDR) and audio (VAT) tools. A list of poll questions was sketched on the left side of the tool and clicking on an item in this list would cause a second panel to appear with the details about this question. In addition, a results panel was sketched where the users' opinions were summarised in a histogram display.

Prototyping

The second step in the design process was to develop a non-functional prototype of the intended interface that could be shown to potential users for comments. The Perl/Tk scripting language was used to build this prototype because it supports the Tk graphical objects (e.g., buttons and sliders) that are found in the audio and video MVC tools, while still being relatively simple and easy to script. This prototype required scripting by hand so this can be considered a form of "slow" prototyping, instead of the more common "rapid" prototyping that is done with graphical drag-and-drop interface design tools.

The main interface screen of the prototype is shown in Figure 1. It contains a scrolling list of question

topics on the left side of the screen and control buttons on the right. The functionality of the buttons was simulated by displaying an appropriate sample screen when a button was pressed. For example, selecting any question topic in the list and then pressing the "View Question" button resulted in the display of a sample Question Details screen, as is shown in Figure 2. The full question is presented on this new screen along with buttons for the user to choose when making their response. If the "View Results" button was pressed on this question screen then a sample Results Screen was displayed, as is shown in Figure 3. This Results Screen shows a graphical representation of possible results. Incoming results were simulated by periodically adding to the data using a random number generator. The entire prototype required about 600 lines of Perl/Tk code.

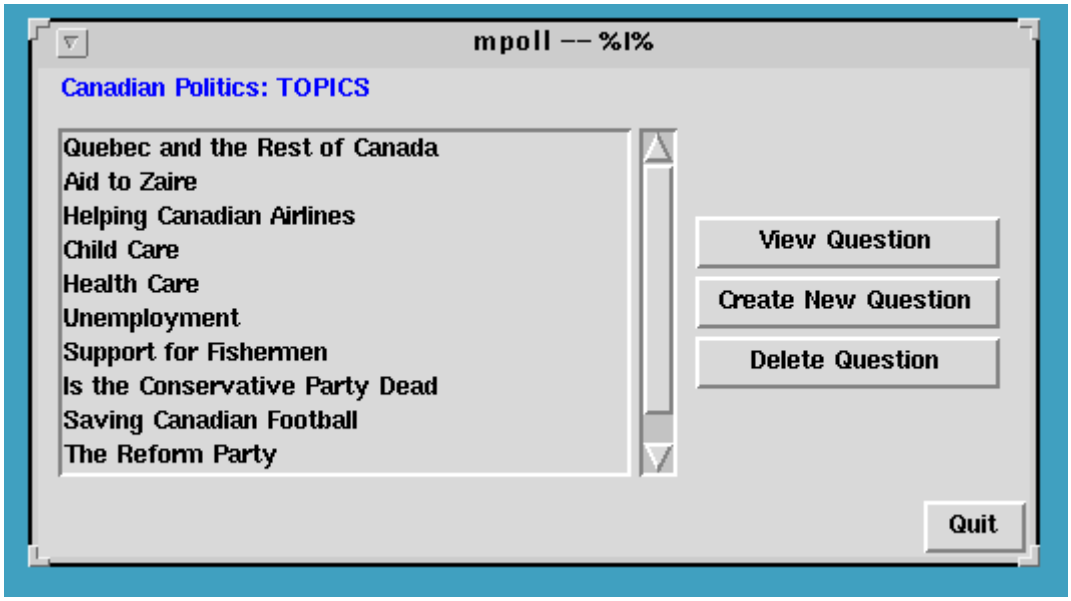

Figure 1: The main interface screen of the prototype.

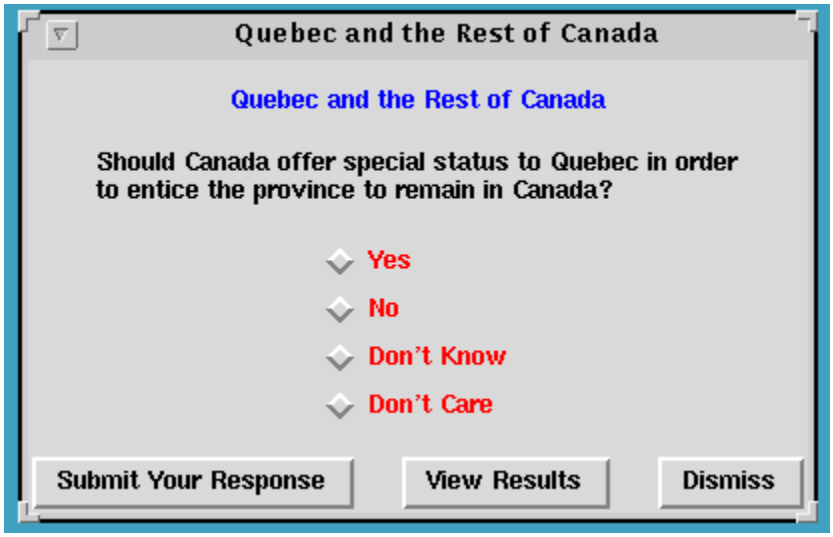

Figure 2: The question details screen of the prototype.

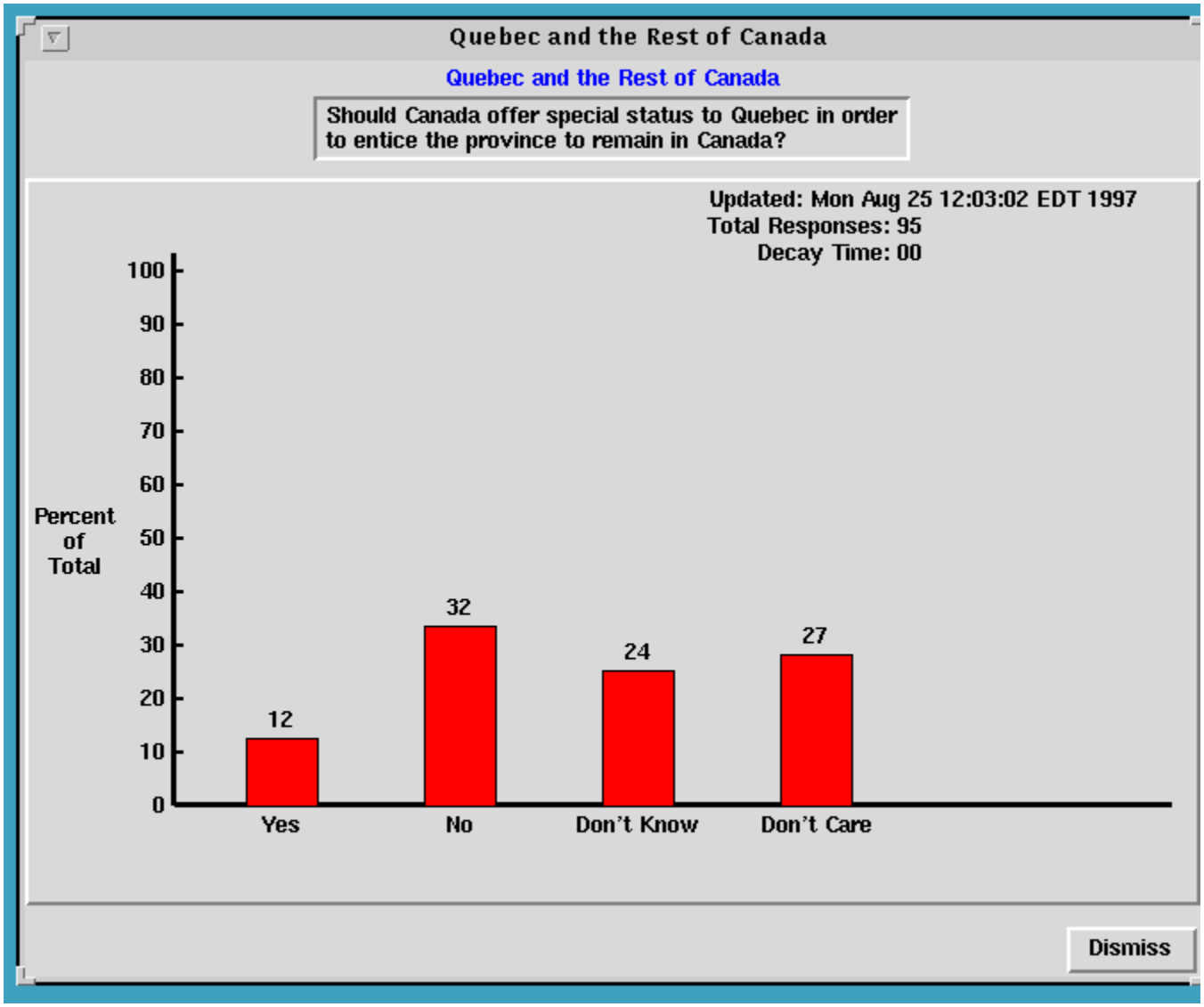

Figure 3: The Results Screen of the prototype.

Informal Interface Testing

The next step was to test the prototype interface for usability. This was done informally by showing four current MVC users the prototype and recording their reactions. A simple testing protocol was used to ensure that each user examined all the possible features of the prototype. The main result of the testing was that users did find the interface intuitive and familiar. They felt that the tool behaved similar to the other MVC tools they were using, and that each button and screen appeared as they expected. Users also reported that the results screen (Figure 3) was easy to interpret and useful.

The users did have some suggestions for improvement. For example, a "Delete Question" function was originally omitted and users pointed out that this was a necessary feature. Also, the "View", "Create" and "Delete" buttons contained on the main interface page (Figure 1) were originally different sizes and

not aligned and the users preferred that they all be a uniform size.

The users also reported that help information would be a good feature so internal help messages were added to the design specifications. The users also expected to see a logo on the main screen of the tool as had been done for the SDR and RAT tools. Finally, the users reported that the tool was going to need a short name similar to the names given to the current MVC tools (i.e., VAT, RAT, VIC). The name "mpoll" was chosen and the capitalisation was later changed to "MPoll" to be consistent with the common capitalisation of "MBone".

Implementation and Iterative Design

Following the success of the prototyping and testing phases, it was decided to implement the design ideas in an operational program. The user interface portion of MPoll was implemented using the Tcl/Tk language. This is the interface language used for most of the MVC tools, including SDR and RAT, so by using Tcl/Tk, MPoll would have a similar look and feel. Some of the code developed for the Perl/Tk prototype was translated into Tcl/Tk and other code was added to give full functionality and more features.

A networking component was also need for MPoll. The C programming language was used because it contains many routines for Internet networking. This component of MPoll had to establish TCP sockets for sending and receiving data and handle the details of the multicasting protocol. A new polling protocol was also developed to define the nature of the MPoll packets and their contents. Packet formats were defined for questions, question deletion instructions, and responses. This polling protocol was based on the Session Description Protocol (SDP) that has been developed by the IETF and implemented in the SDR program.

The Embedded Tk library was used to integrate the Tcl/Tk and C components. This public-domain library allows Tcl/Tk scripts to be included in C programs. It also allows the two languages to share data variables and pass values back and forth.

Alpha Testing

Once a functional version of MPoll was completed, the next step was preliminary user testing and refinements. The MPoll program was tested on a variety of computers in our laboratory and at various sites on the Internet. The first refinement was a local cache of questions and responses. Without a cache it was necessary to re-enter all questions and responses whenever the program crashed, and crashes was frequent in the initial Alpha versions. Adding a local cache meant that MPoll could be restarted without losing any of the previous work.

Early MPoll users were also interested in understanding the methods used for network communications. In addition, a convenient means of watching incoming and outgoing packets also makes program debugging easier. These comments resulted in a network traffic monitor and later this was refined to include a "verbose" mode where the contents of the packets could be examined.

The initial testing also revealed a need for a membership monitor that would display all the current MPoll users in a session. This type of monitor is already included in the MVC audio and video tools and adding it makes MPoll similar to the other tools and aids in debugging. A new membership packet format was defined and a membership list panel was added to the interface. Later, a "keep all" option was added to the membership monitor so all users who have run MPoll during a session are shown, in

addition to highlighting of the currently active users. This allows users to track MPoll usage during long testing periods.

Some refinements were also made to the question creation interface. It was necessary to restrict the length of the question and response strings to ensure that they were of reasonable length and fit into the results display. It was also necessary to check that topic and questions strings were actually entered when a question was created because any empty strings created problems when they were displayed in the main list of question topics. During the course of testing it was also necessary to increase the number of possible multiple-choice options from 5 to 10 to accommodate some larger questions users wanted to pose. The program was also expanded at this time to handle text responses, in addition to multiple-choice responses.

Another function that was added as a result of Alpha testing was the ability to save the results in a summary file. This function creates a simple text file that contains a summary of each question and the collected responses. Without this save function users had to capture a screen image of the histogram display or transcribe the results shown there.

Finally, the Alpha testing revealed a number of programming and logic errors that had to be repaired. The most common errors involved socket programming for the multicast protocol, memory leaks when the program was run for long periods, and coding that was not portable to other platforms.

Beta Testing

The Beta testing phase involved publishing MPoll more widely and encouraging its use for MERCI meetings and other MVC sessions. This Beta phase led to two further refinements. One was to add a "demo" mode to the program so that incoming responses are simulated, as was done in the initial prototype. This allows quick demonstrations of the program in the laboratory to visitors and potential customers.

The second refinement was based on a suggestion from the users about how to tabulate the responses for multi-response multiple-choice questions. Initially, MPoll calculated a percentage based on the total number of responses given and the users indicated that a more meaningful calculation for this type of question is to base the percentage on the number of responders.

Current Status and Conclusions

Currently, a Beta version of MPoll has been released for world-wide circulation (see http://debra.dgbt.doc.ca/mbone/mpoll/mpoll.html for the latest information). The main screen from this latest version is shown in Figure 4. This is very similar to the prototype version except for the addition of new buttons that are used to access new features.

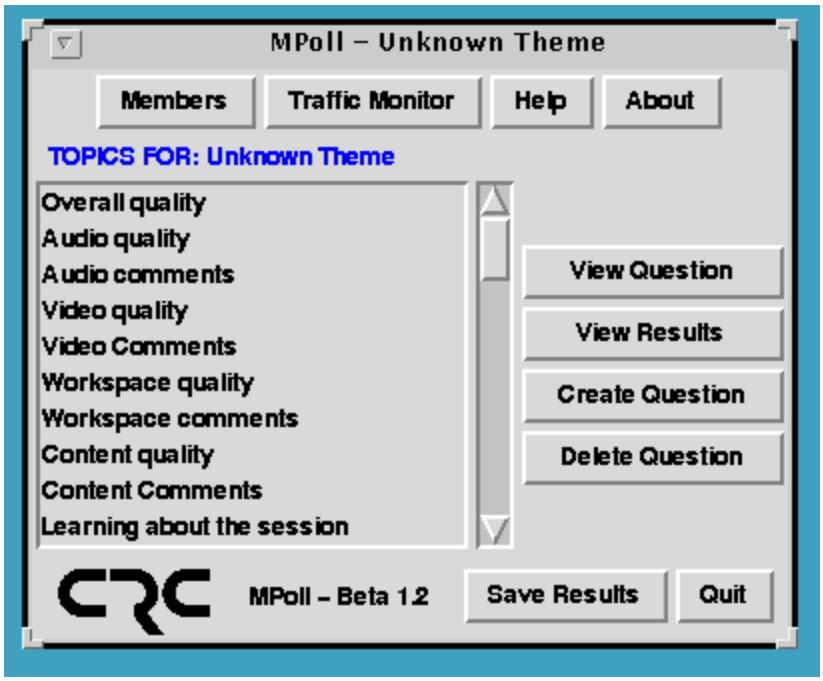

Figure 4: The main interface screen of the current version.

The response screen of the current version is shown in Figure 5. Users are shown the complete question and given instructions on how to respond. Figure 6 shows the results screen for the current version of MPoll which is very similar to the initial prototype.

Figure 5: The question details screen of the current version.

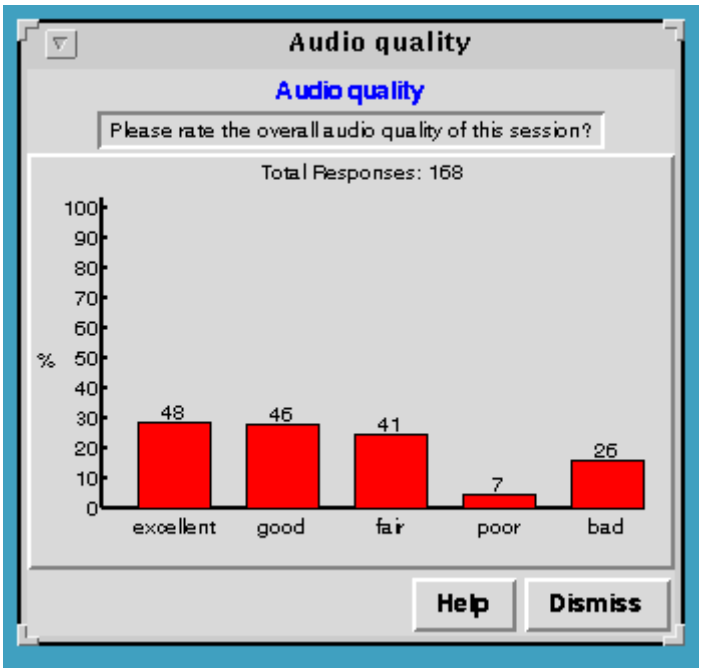

Figure 6: The results screen of the current version.

The MPoll program consists of 6,100 lines of C and Tcl/Tk code. It currently compiles on the Sun UNIX (SunOS and Solaris 2), Silicon Graphics IRIX, and Win32 computer platforms and should be easily portable to other similar operating systems. The program has been tested in the laboratory and at a number of sites on the Internet. User suggestions and reactions have been collected throughout the design process and the program has been refined and improved accordingly.

MPoll has been used for the bi-weekly MERCI management meetings alongside audio, video, and shared text editor tools, and it will be used for the follow-up MECCANO project. MPoll has proven useful for collecting quality ratings during these sessions and as a decision-support tool. MPoll has also been used to collect media and content quality ratings during one remote seminar with successful results.

Finally, MPoll is now available for license by companies who can use the technology to develop commercial applications.

References

Hardman, V., Sasse, A., Handley, M. & Watson, A. (1995). Reliable audio for use over the Internet. *Proceedings of INET'95*, June, Honolulu, Hawaii.

Isaacs, E.A., Morris, T. & Rodriguez, T.K. (1994). A Forum for Supporting Interactive Presentations to Distribute Audiences, *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW)* Chapel Hill, NC: ACM Press, 405-416. http://www.sun.com/tech/projects/coco/papers/forum-CSCW94.html

Isaacs, E.A., Morris, T., Rodriguez, T.K., & Tang, J.C. (1995). A comparison of face-to-face and distributed presentations. *Proceedings of CHI 95*. http://www.acm.org/sigchi/chi95/proceedings/papers/ei_bdy.htm

Kirstein, P.T., & Bennett, R. (1997). Recent activities in the MERCI conferencing project. *Proceedings of JENC8*, Edinburgh, May, pages 923-1 - 923-11 Postscript

Kumar, V. (1996). *MBone: Interactive multimedia on the Internet.* Indianapolis: New Riders.

Macedonia, M.R., & Brutzman, D.P. (1994). MBone provides audio and video across the Internet. *IEEE Computer,* 30-36.

McCanne, S., & Jacobson, V. (1995). VIC: A flexible framework for packet video. ACM Multimedia '95, San Francisco, CA, pp. 511-522.