School of Computer Science and Software Engineering
Monash University

Bachelor of Computer Science (Honours) (1608)
Project Report 2000

Developing support systems for an interactive video wall

by

Russell Clarke 12278769
`rwc@cs.monash.edu.au`

Supervisor: Prof. David Abramson
`davida@csse.monash.edu.au`

Declaration of Originality

I, Russell Clarke, declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given in the bibliography.

____________________________ (student signature)

____________________________ Date (day, month and year)

Abstract

Videoconferencing is being increasingly used for informal communication. An example of this is Monash University's School of Computer Science and Software Engineering. They operate a high-quality video ATM link between the staff common rooms in its Clayton and Caulfield campuses, to encourage communication and discussion between the two groups. The system consists of two FVC VANs, each with hardware MPEG-2 codecs, data projectors and cameras.

Response to the videoconferencing system has been mixed. While many people find the system useful, others are concerned with privacy issues. In addition, there exist several limitations such as the inability to communicate data through the system, remotely control the camera's field of view, or capture video to disk. By integrating these features into the existing system, this project aims to increase its usefulness and usability.

Reasons for adding each feature are given, together with strategies for implementation. A client/server application implementing video reciprocity is developed, using an infra-red motion detector as input. The client shuts down the ATM link and projector when there is movement in only one or neither of the two rooms, and starts up the link and projector when movement occurs in both rooms. A client/server application for tunnelling IrDA data through a TCP/IP connection is also implemented, using infra-red transceivers.

The reciprocity application is fully tested. Problems in upgrading the existing VAN software made it impossible for testing it *in situ*. The reciprocity application is designed for smooth installation once the upgrade is completed, and no code modification will be required. The infra-red tunnelling application requires implementation of a handshaking protocol to be fully-functional, and has been tested using other serial port devices. The remaining features, while unimplemented, have been researched and are discussed, and strategies for their implementation suggested.

List of Figures

Contents

1 Introduction

At Monash University's School of Computer Science and Software Engineering (CSSE), videoconferencing systems have been installed at the staff rooms in its Clayton and Caulfield campuses (see Figure 1). The "video wall", as it is known, is used for inter-campus seminars, departmental meetings, discussion between staff members, as well as informal chatting.

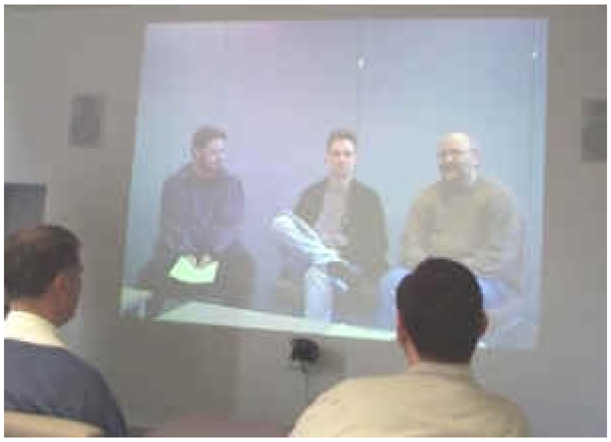


Figure 1: CSSE video wall (from Abramson 2000)

Response to the videoconferencing system has been mixed. While many people find the system useful, others are concerned with privacy issues. In addition, there exist several limitations such as the inability to control the camera's field of view or capture videoconferencing sessions to disk, which would be useful in a videoconferenced meeting.

The camera, which in the existing system is approximately 60 cm above the floor (as shown in Figure 1), could be positioned higher above the floor. The current low position is required so that the camera lens avoids the glare produced by the projector, but the current camera position requires that people sit a reasonable distance from the wall to be able to see a person's face well.

Opportunities exist for expansion of the video wall's use, such as infra-red communication between notebook or handheld computers on either side of the wall. A system also needs to be developed to switch between video links before a third video wall node can be installed.

This project aims to address privacy issues by introducing reciprocity and add improvements by implementing a user interface for remote camera control and repositioning the cameras at sitting eye-level together with an anti-glare system. It also aims to allow infra-red signals to be transmitted "across the wall", by means of infra-red transceivers connected to computers located beside the video wall. Automatic video wall switching can be implemented by using infra-red motion detectors to decide which rooms contain people, and linking those rooms together. The features are designed to be integrated into the existing system.

This thesis begins with an overview of videoconferencing systems and infra-red communication in Section 2. The aims of the project and the reasons for choosing to implement each feature are given in Section 3. Implementation and development of the systems are covered in Section 4, and the systems are then discussed and analysed in Section 5. Future work and conclusions are presented in Section 6.

2 Background Information

2.1 Uses for Videoconferencing

Videoconferencing allows users who are remote from each other to work and play cooperatively. The foundations of videoconferencing are in video telephony. The first video telephone call was made in 1927 (Kraut 1994), but since then, no video telephone product has been successful. Recent developments have had more success but the telephone is still the main form of remote communication. Videoconferencing systems need to be carefully adapted to human needs.

The technology for computer-based videoconferencing has only been available for about ten years (Fish 1994), and commercial systems have only been viable for approximately five years (PictureTel 2000, First Virtual 2000, VTEL 2000). The systems have grown rapidly from being used mainly in research to being off-the-shelf products.

Videoconferencing systems can be classified into two types: (1), narrow-angle systems, in which each person has their own camera, and (2), wide-angle systems, in which a number of people share a single camera. These systems will be referred to as "personal" and "group" videoconferencing systems respectively.

Personal systems may be used as "video telephones", or users may see a number of others simultaneously, usually in separate windows, although each user will have their own camera. Sound is usually monophonic. Sometimes these systems are referred to as desktop videoconferencing systems (Kraut 1994).

Group systems are most commonly found in corporate board rooms or conference rooms. The display is usually a large screen television or projector. Sound is usually stereo, and is audible throughout the entire room.

The VideoWindow (Fish 1990) and Cruiser (Fish 1993) systems are good examples of each of the two types of videoconferencing systems — group and personal, respectively. The group system is designed for conversations between two or more people, while the personal system is designed to be used on a user's personal computer. Each one has different problems associated with it, as will be discussed later.

Videoconferencing systems consist either of a view of a group of people or a single-person view, and, as mentioned above, each is suited to different situations. There are uses for both hardware- and software-based videoconferencing systems. Hardware systems are better suited to conferences between groups, or where the system will be running for long durations. Software systems are better suited for communication between two people, where conferences are short, or for streaming to a number of people in separate locations.

IP videoconferencing has some advantages over hardware-based videoconferencing, such as lower cost, use of existing network infrastructure, and the ability for users to use videoconferencing systems from their personal computers. However, hardware systems offer higher frame rates, better picture quality, and typically use a dedicated link so bandwidth isn't shared with LANs and Internet connections.

2.2 Human Factors in Videoconferencing Systems

In 1990, Fish *et al.* created the VideoWindow system (Fish 1990). This used a wide-screen projector and multiple cameras and microphones to transmit the video and audio between two locations.

Fish's team used two common rooms on two floors of a building. He studied the use of the VideoWindow system for three months, recording the transactions through the system.

He found that informal communication occurred much less through the VideoWindow than through face-to-face communication. Survey responses commonly reported that the limited camera field-of-view and the position and direction of the microphones created a situation where it was unnatural for informal communication to occur.

The idea that if a person A can see person B, then person B can see person A is not carried over to the VideoWindow system, and in group videoconferencing systems in general. Also, the position and direction of the microphones also caused people to speak unnaturally, in many cases speaking louder than normal.

The VideoWindow system also did not allow for people to carry out private conversations. If you want to say something to someone privately, you just move closer to the person and speak quietly. This is not possible in the VideoWindow system because of the microphone and speaker positions.

It is also much easier to avoid eye-contact through the VideoWindow system. In reality, if you wish to avoid conversation with someone, you can avoid eye-contact. The VideoWindow system makes it much easier to do this, just by getting out of the camera's field of view. Also, there is not the compulsion to speak to someone through the system.

With many videoconferencing systems, privacy is an important issue for most users. Two features commonly used to deal with privacy include reciprocity (node A can view node B if and only if node B can view node A), and the ability to disable the video system or to display a "busy" message. Some systems may also contain a list of all users active on the system (e.g. Cruiser). Privacy becomes less of an issue as the technology becomes more widespread (Kraut 1994, Fish 1992), and as users adjust to the system.

New users are often apprehensive about communicating to others through videoconferencing systems, because they feel that they may be interrupting other people (Kraut 1994). They tend to use it to communicate with people they already know well (for example, in their department), and then only at times that they think are "safe". As with privacy issues, with time, users adjust and intrusiveness becomes less of an issue. People develop a sense of how to use the system by observing how others use it.

Group-based videoconferencing systems are used more for informal conversations, so that two people working on the same project are less likely to discuss their work through the system than they would face-to-face (Cool 1992, Fish 1993). This is mainly due to the location of these systems and their impersonal nature. On the other hand, two groups (for example, business meetings) can be easily carried out over these systems. Privacy is also an issue here, since reciprocity does not necessarily apply with group systems (e.g. VideoWindow).

People can sit and watch a debate on television, or have a lengthy face-to-face conversation, but speaking to and watching people through a videoconferencing system seems different (Inoue 1995). Inoue *et al.* suggested that video and audio quality and resolution were not the only important factors in constructing a videoconferencing system, and that camera angles were also important.

They analysed several live debate/interview television programmes, and came up with a system that automatically changed the camera view depending on a simple transition table. A limited amount of feedback has been obtained so far, but users seem more comfortable with the system than with a static camera angle.

The solutions to various human interface concerns are mostly yet to find their way into commercial systems, but features such as AMX control are available for remotely controlling cameras (PictureTel 2000, First Virtual 2000), allowing users some camera interactivity.

2.3 Features of Videoconferencing Systems

Currently, only 10-15% of enterprise networks use video over IP (Perey 2000), but most who do use it for streaming video, not for videoconferencing. For interactive videoconferencing, a network bandwidth of 600 Kbit/sec is required, although 800 Kbit/sec or greater is ideal (Perey 2000).

ATM (Asynchronous Transfer Mode) combines the best of packet switching (like Ethernet) and circuit switching (like ISDN) and for this reason is generally considered well-suited for videoconferencing (Fish 1994), because it can handle sustained high-data rates in the presence of both voice traffic and data bursts.

Many commercial systems have now been developed. First Virtual Corporation (FVC) have a range of videoconferencing products, including the Video Access Node (VAN), and the VAN II, used by Monash University.

The VAN (First Virtual 2000) provides high-quality videoconferencing, using MPEG-2 codecs (compressor-decompressor). First Virtual claims that its video is comparable to that of VHS or laserdisk systems. The VAN uses the H.310 standard. The VAN implements encoding and transfer over ATM networks, including "long haul" ATM networks for wide area access. The VAN has a low latency (delay between transmission and reception). This is an important feature because it allows for use in business conferences, distance learning and medical applications. It supports both NTSC and PAL television formats. The VAN uses integrated software running under Microsoft Windows NT or IBM OS/2.

The VAN allows full-duplex audio, so that sound is received and transmitted simultaneously. This can cause audio feedback, so an echo canceller is required. The VAN includes an echo canceller. Echo cancellation is important for all videoconferencing systems (and also audioconferencing systems) that allow full-duplex audio, and this is easily handled by hardware systems (First Virtual 2000).

Comparable commercial systems to First Virtual's VAN include PictureTel's Concorde series and VTEL's room systems. These are both commercial systems which use televisions or other video output that accepts RGB or S-Video input, such as a projector. There are intended to be used as group systems.

There are many videoconferencing standards (Cool 1992). The most commonly used are the ITU's Series H standards H.320 (for ISDN), H.321 or H.310 (for ATM), and H.323 (for IP), as well as MPEG-2. H.323 is used mainly for software-based videoconferencing products, especially two-way conferencing on PCs. The other standards are mainly used for hardware-based products. The VAN uses H.310 for MPEG over ATM. Monash University's ATM network runs at 155 Mbps.

IP videoconferencing has some advantages over hardware-based videoconferencing, such as lower cost, use of existing network infrastructure, and the ability for users to use videoconferencing systems from their personal computers. However, hardware systems offer higher frame rates, better picture quality, and typically use a dedicated link so bandwidth isn't shared with LANs and Internet connections.

The VideoWindow system (Fish 1990) used a specially-developed camera that captured video images which were twice as wide as a standard NTSC video image (i.e. 720×2 or 1440 pixels). The resulting aspect ratio is 8:3, compared to the standard 4:3. Movie film and high-definition television use an aspect ratio of approximately 16:9. A very wide image produces a higher sense of realism than a standard television image. The VideoWindow's projected image three feet (91 cm) high by eight feet (244 cm) wide. The VideoWindow's images were approximately life-size. Coupled with the wide-screen image format, the resulting image is quite realistic.

2.4 Problems with Current Systems

Informal communication is frequent, interactive and expressive (Fish *et al.* , 1993). A later system developed by Fish *et al.* (1993) is the Cruiser system. This consisted of a number of nodes (computers) with video display, camera, speakers and microphone. Users could initiate a connection to any of the other nodes, or let the system choose a node at random. Random calls were not often answered. On the other hand, users found the system useful to call other people to discuss work, but usually this was only to schedule face-to-face meetings or for short discussions. When discussions became sufficiently involved, users met face-to-face.

The users were surveyed, and the main reason behind this was the inability to share work, for example computer screens or printouts, and the inability to work together — users often stopped a Cruiser session to use a blackboard or a computer together. Tang (1992) also found that shared drawing boards were an important part of videoconferencing.

Ishii *et al.* created a system which combined videoconferencing and a digital drawing board called ClearBoard (Ishii 1992). This used a mirrored LCD panel which displayed the image of the other person, and which also could be written on with whiteboard markers. The video image displayed on the LCD display was overlayed with the drawings from the other person, giving a shared whiteboard. The analogy used was that users were drawing on a sheet of glass separating them, with the difference that the drawing was shown mirrored correctly (i.e. the image was flipped horizontally when sent, so that a mirror image is not seen by the other user). The main problem users found with the system was that they could not erase their partner's drawings.

Videoconferencing is used for both formal and informal communication (Fish 1993). Formal communication includes scheduled meetings, conferences and seminars. Informal communication includes unscheduled meetings and personal conversations. Informal communication is useful in solving problems, generating ideas and helps people get to know each other. Informal communication is more difficult when the people communicating are on different locations (Sellen 1992). Videoconferencing is ideal in these circumstances.

Takao and Innami (1998) showed that desktop videoconferencing systems improved the quality of group decisions. They compared the results of solving a standard problem in a face-to-face meeting, a videoconferencing system where only the speaker is shown, and a videoconferencing system in which all members of the group are shown. There was little difference in the results for the speaker-only system compared with the face-to-face system, but a significant difference between the all-member system and the face-to-face system. Preliminary research ascribes the differences as due to reduced eye contact with each person, no distinction of status among the people in the group, and no recognition of physical difference, such as height, among participants.

Users generally think of using videoconferencing systems as a telephone call, rather than actually meeting them face to face (Fish 1993, Cool 1992). This may be due to the main areas of research conducted so far. Most of the studies are of videoconferencing among groups of people who are working in similar (or the same) environments, but have rarely, if ever, met in reality.

Videoconferencing offers many advantages over other forms of telecommunication. It is well-suited for communication between groups, and situations where distance prohibits two parties meeting face-to-face (Brookings 1999). Currently, videoconferencing systems are mainly used for group meetings, because no practical alternatives exist.

Brookings found that their videoconferencing systems encouraged group discussion, and strengthened ties in groups in separate locations. However, studies have shown that people can feel uncomfortable with videoconferencing systems (Fish *et al.* 1994).

Brookings also used their videoconferencing systems for presentations, which enabled speakers in other locations to give presentations without travelling. This enabled more experts to give presentations to the group.

2.5 The Future of Videoconferencing

The technology is now available to achieve high-quality, low-latency videoconferencing, as achieved by commercial systems such as the VAN II. Hardware MPEG-2 codecs, ATM networks, high-quality cameras and projectors are leading factors. Desktop videoconferencing is also growing in popularity, with realtime video codecs and technology such as gigabit ethernet now available on desktop PCs. As stated above, human factors need to be taken into consideration when implementing videoconferencing systems. Until very recently, videoconferencing wasn't taken seriously as a tool for education, business or play. Current systems still have many limitations, and much research has been conducted into features for videoconferencing systems. Shared workspaces, privacy issues and systems designed for informal communication are the main areas where work is being done. The present project aims to improve an existing videoconferencing system by addressing some of these limitations.

2.6 Infra-red Data Communication

Infra-red data transmission is another technology gaining popularity in recent years, due in part to acceptance of the IrDA (Infra-red Data Association) standard, and the increasing number of notebook computers and hand-held devices that have built-in infra-red ports. Nearly all notebook computers and PDAs (Personal Digital Assistants) have some kind of infra-red capability. Microsoft's Windows 2000 operating system (but not Windows NT) now has infra-red device support (Actisys 2000), and a number of infra-red transceivers are available for PCs that do not have infra-red ports. A number of CSSE staff have notebook computers and PDAs that have an infra-red communications capability.

The IrDA standard is by far the most common infra-red data protocol (IRDA 2000). However, there are many IrDA protocols, which are used under different circumstances. Specifications exist for wrist watches, dongles, mobile communications, photography, a minimum ("lite") set, control (standard) and others. In total there are 18 IrDA specifications (IRDA 2000).

Wired communications methods can send streams of information in both directions at once, because there are multiple wires (some to send data on, some to receive data from). With infra-red, there is the equivalent of only one wire (the infra-red path through the air), which has the following implications: IrDA protocols send packets one way at a time. If a device tried to send data and listen for data at the same time, it would "hear" itself and not the device it wants to communicate with. The way IrDA devices achieve two way communications is to take turns, also known as "turning the link around". This happens at least every 500 milliseconds, and can be made more frequent as necessary. This time is called "turnaround time". This latency makes it impossible to perfectly emulate wired communications — very timing sensitive operations are not suitable. Fortunately, many communication tasks are not so sensitive, and these can use IrDA.

All of the information carried on multiple wires must be carried on the single infra-red "wire". This is accomplished by subdividing the packets into data and control parts. In this way a logical data channel and control channel are created.

The standard IrDA protocol is complicated compared to other data transfer protocols. The protocol is "layered", including communication with IrLAP (Link Access Protocol) and IrLMP (Link Management Protocol), and each of these layers requires handshaking and transmission of separate data and control messages. Each of these protocol layers lies on top of the Physical Signalling Layer (PHY), which implements the physical IrDA data signalling. IrLAP provides device-to-device connection and ensures reliable, ordered transfer of data. IrLMP provides multiplexing of the IrLAP layer. Multiple channels of IrLMP can exist.

There are many infra-red data protocols. IrDA is now the most commonly used, but another protocol is Sharp ASK. In addition, there are several versions of the IrDA protocol. All protocols are byte-based, but have different parity bits, error check methods and range of transmission speeds. ASK ranges from 9600 bps to 38.4 Kbps, while IrDA ranges from 9600 bps to 115.2 Kbps, with some high-speed extensions allowing up to 4 Mbps. In the near future, speeds of up to 16 Mbps will be available (IRDA, 2000).

There are two types of infra-red data transfer: diffuse and directed. Diffuse infra-red allows many-to-many connections, does not require direct line of sight and can be uni- or bi-directional. Financial trading floors are an example of a use of diffuse infra-red. Direct infra-red is point-to-point, requires a direct line of sight and is a secure form of data transmission and reception. IrDA is an example of directed infra-red. The main reason why most IrDA connections are limited to one metre is that the possibility of interference is greater the further the two infra-red ports are, since a direct line of sight is required. This is especially noticeable with hand-held devices. Some devices can handle distances of greater than one metre, but IrDA-compliance only requires that communication can occur over distances of 0–1 metre.

The IrDA consortium claims that infra-red data transfer is very secure, and that using an infra-red connection to access a LAN is as secure as using a cable at any other access point on the network. IrDA also claims that infra-red data transfer is very reliable — often more reliable than wired connections, since it eliminates wear and tear. In addition, there is no possibility of bending pins or connecting plugs into the wrong socket.

Today, devices that contain infra-red ports include notebook and hand-held computers, printers, cameras and desktop transceivers. It is probable that, in the future, most photocopiers, fax machines, overhead projectors, telephones, bank ATMs, headsets and many consumer items will have infra-red ports. The present project aims to incorporate infra-red communication into a videoconferencing system, thus adding to this list.

3 Aims

The video wall is currently on 24 hours a day, so that people can walk up to the wall, see if anyone is in the room and start a conversation with anyone on the other side of the wall. This is arguably its best feature in terms of usability. It is almost a standard setup, the only modification being an infra-red motion detector that turns off the projector if no movement has occurred in the room for a number of minutes, and turns it on again as soon as motion is detected. This was installed to save on electricity costs.

A schematic diagram showing the setup of the current CSSE videoconferencing system is shown in Figure 2. Not included on this diagram are the infra-red motion detectors, which are attached to the projectors.

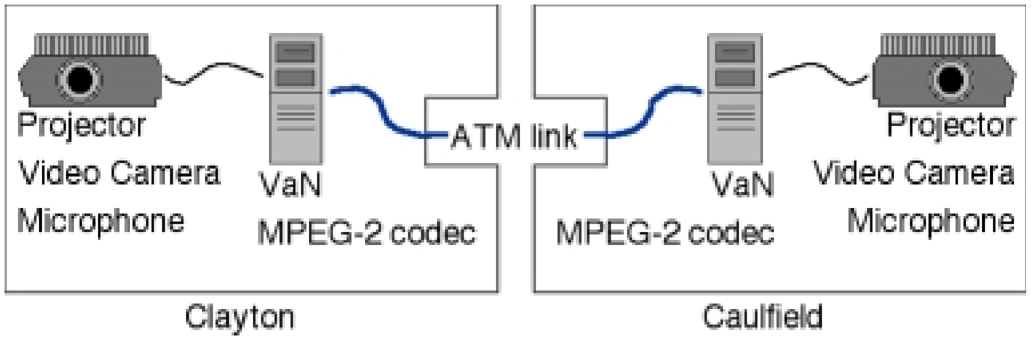


Figure 2: Diagram of current CSSE videoconferencing system

The current system can be made much more user friendly, as well as reducing costs further. The following improvements to the system would also make the video wall not just a simple video communication device, but a powerful tool for conversations, meetings and seminars.

- **Video reciprocity.** The current system whereby the projector is switched off when the room is empty can be extended so that both projectors are switched off. Such a system would also enable the ATM link to be shut down and start up when necessary. Currently, the ATM link is constantly up. Shutting down the link when its not in use would save on costs.

  This system also addresses the privacy concern of reciprocity. As mentioned in Section 2.2, previous research has shown that people are concerned by the prospect of being visible to someone without the reverse being true. For example, someone could be out of the camera's field of view, but still be viewing the video wall (see Figure 3). With the current camera and projector, it is impossible to eliminate the "viewer's field of view only" region, but it has been found that the infra-red motion detector can detect motion at a wide angle, which makes it suitable for the purposes of the proposed system.

  A proposed system is to design an application which would gather the state of both motion detectors (on or off), take the logical "and" of both states, and pass the result back to both VANs (see Figure 4). Each VAN would then decide whether to start up or shut down the ATM link, and also turn on or off the local projector.

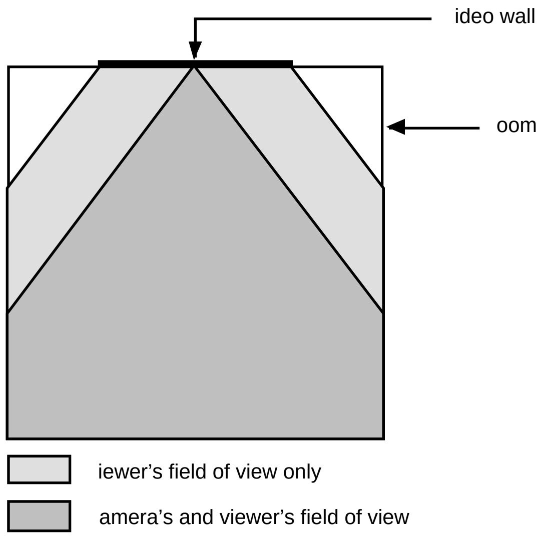

ideo wall
oom
iewer's field of view only
amera's and viewer's field of view

Figure 3: Field of view of camera and viewer

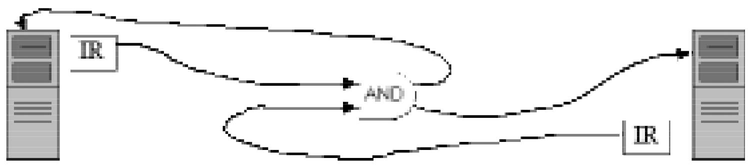


Figure 4: Diagram of proposed videoconferencing reciprocity system

- **Camera position.** The camera is currently about 60 cm above the floor, with a shield above the lens, to protect it from the light of the projected image. Unfortunately, the position of the camera produces an image which is a little unusual, and which some people find disconcerting. The position also prevents communication closer than about two metres from the wall. If a person stands any closer than this, their face will not be visible to the people in the other room. The lens shield also reduces the camera's field of view. Ideally, the camera would be placed approximately 1.2 metres above the floor, at eye-level for people sitting down. This would produce a more natural image, as well as allowing communication closer to the wall, without reducing the camera's field of view. Overall, it will improve the system's user-friendliness.

- **Data transfer.** The current system allows for audiovisual communication, but not data communication. Data communication could occur through ethernet connections, but this would allow only notebook computers to communicate, and would require some configuration. Another approach would be to use infra-red communication, since this would allow most notebook and handheld computers to communicate. An added advantage of using infra-red is that no configuration is involved — configuration is required only once; afterwards, it can be used immediately. Also, many CSSE staff have infra-red-capable computers. It therefore makes sense to incorporate infra-red communication into the system.

Adding infra-red transceivers together with software to transmit the data across the wall would allow people with notebook or handheld computers to transmit computer files, memos, business cards, and so on across the video wall (see Figure 5). Note that the IR boxes here refer to infra-red transceivers, not the infra-red motion detectors as in Figure 4.

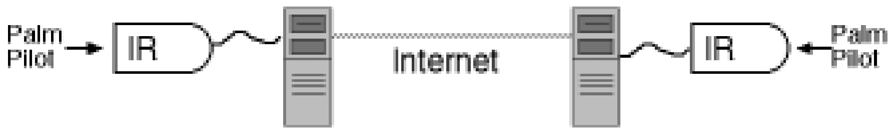


Figure 5: Diagram of proposed infra-red tunnelling system

The system would enhance the feeling of the "virtual room", since this would allow some non-audio-visual interaction to take place between the two rooms.

- **Automatic switching.** Currently, CSSE operates just two video nodes, one at Clayton and one at Caulfield. However, in the future, more systems may be installed, requiring a way to specify which video node to connect to. This could be done automatically if only two nodes are in use (as in the video reciprocity system), or when a reduced set of nodes exists (e.g. two video walls from the same campus are never linked).

  This idea could be taken further so that a list of possible connections can be specified, and these can be placed in order of preference. Additionally, a feature could be added to display a list of available nodes, and to let the user manually select which node to link to. Such a system would be absolutely necessary if two or mode video nodes were installed.

- **Camera control.** The camera is currently fixed, so that camera's field of view cannot be changed. If the camera could be controlled remotely, then panning and tilting and/or zooming would be possible, giving users more freedom in what they look at. Panning allows the camera to view to the left and right, tilting allows the camera to view up and down, and zooming allows the camera to get a close-up or wide-angle view.

  This would make communication easier, because the position where a person might want to view the wall is not necessarily the best position for the camera to capture them. For small groups (e.g. one or two people at each location), this could be important because of seating arrangements at either location. For example, chairs might be placed beside tables, not in front of the video wall. For large groups (e.g. six or more people at each location) this is even more important because it is unlikely that a single camera view will enable every person to be seen clearly.

  In the latest computer science seminars that have been broadcast over the video wall, from one campus to the other, a video camera on a tripod has been used, and has been controlled by hand, so that it followed the presenter as he or she walked around. This is a good example of how camera control could be used.

  Currently, if a person is out of the camera's field of view, they would need to move into the camera's field of view. If the camera could be remotely controlled, then this would be unnecessary. It would also allow people to view speakers at meetings, and to focus on different speakers as the speaker changed. Camera controls makes the system more useful in many circumstances, particularly organised meetings and discussions.

- **Video recording.** After the video is transmitted through the link, and displayed at the other location, it is lost. There are some circumstances, for example, meetings, where it would be useful to keep a record of the video conference.

  To address privacy concerns, people would need to be warned beforehand that video recording would be taking place. In addition, a visual indicator such as a flashing red dot could be displayed on the video wall to remind people that video recording is in progress. Video from the staff common rooms would not be recorded unless a meeting was taking place.

  Video recording would be especially useful for meetings and group discussions. The captured video could be used as a "backup" to minutes taken, or as a reference to participants in the discussion. For people unable to attend such meetings, a video of the proceedings would be invaluable. A number of meetings have taken place over the video wall recently, which would have benefited from having been recorded.

By improving the videoconferencing system, the project aims to improve its usability and usefulness, and to encourage greater use of the system. Individually, each of the proposed features seems minor, but together, they will contribute significantly to the entire system.

4 Implementation

4.1 Initial Development

The first, and most important part of the project was the "videoconferencing reciprocity" application. This comprised two separate programs: a client and a server.

The existing videoconferencing system ran under the IBM OS/2 operating system. The new version of the software ran under Windows NT, which has greater hardware and software support, so it was decided to move to this version. This involved ordering a new version of the videoconferencing software for the VAN from First Virtual.

The initial idea was to create applications using the standard BSD sockets interface, running under the Cygnus environment. Cygnus emulates a UNIX environment under Windows NT, thus allowing direct use of BSD sockets, the gcc (or g++) compiler, and other common UNIX tools such as "make". However, it was unclear how, or even if, Cygnus supported serial devices — the current version of Cygnus does not emulate the UNIX "/dev/" directory.

The project was originally written in standard ANSI C, using "make" and compiled with gcc. Initially an echo program was written to get a simple client and server running. The server application accepted connections on a single port, and sent any incoming messages to all other connections. Messages were in the form of typed text from the keyboard.

There were three main reasons for moving away from Cygnus — firstly, the serial support did not exist, secondly, it would have been cumbersome to execute the videoconferencing script files since it did not support direct execution of batch files, and thirdly, the reliance on Cygnus made it more prone to errors — at the time, Cygnus was still a beta version.

We decided to move to Microsoft Visual C++. This made serial support possible and made it simpler to execute the videoconferencing script files. It also allowed us to create a graphical user interface, and applications which were double-clickable from within Windows NT.

Originally, three programs were going to be developed: program 1 would read from the serial port and send it to program 2, which would do a logical “and” on the state of the connections, and send the result on to program 3. Program 3 would run the “up link” and “down link” script files. After some investigation, we found it was possible to combine programs 1 and 3, which became the “client”. Program 2 became the “server”.

The server would wait until two clients connected to it. It would then accept a message from each client, representing its state (on, off, or error). The server would update its state of the two clients, and do a logical “and” on the two states. The result would be sent back to the client, which would then run the appropriate script file, if necessary.

The second part of the project was the “infra-red (IR) tunnelling” application. As with the videoconferencing reciprocity application, this composed two separate applications: a client and a server. These tools were to be used in conjunction with the videoconferencing system, so that people on either side of the video wall could communicate not only audiovisually, but also by sending and receiving infra-red messages on notebook or handheld computers.

This application was intended to be developed in a similar way to videoconferencing reciprocity application. The server would wait until two clients connected to it. A client would then receive data from its serial port, and send it to the server. The server would then send the data on to the other client. This client would then write the data out to its serial port.

4.2 Development

The applications were developed in C++ using the Microsoft Foundation Class (MFC) framework, using the Winsock TCP/IP (internet protocol) sockets API (application programming interface).

Microsoft Visual C++ is part of the Microsoft Visual Studio package, which includes an IDE (integrated development environment) and compilers or interpreters for C/C++, Basic, Java, COM, and other programming languages. Visual C++ is a project-based application. Source code files are added to a “project”, which automatically handles all linking and tracks file modification dates. In turn, a project is part of a “workspace”, which can contain multiple projects.

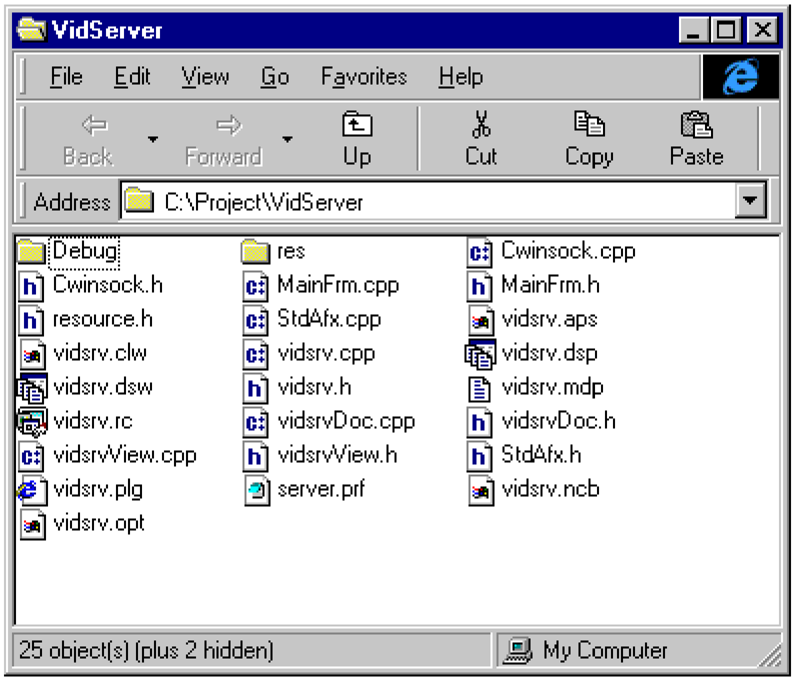
Visual C++ is bundled with a number of “template” workspaces, which are designed to help programmers get started by including commonly-used source code.

Both applications (the server and the client) were based on a template workspace called “MFC App” which is a simple MFC application that displays an empty window with the standard menus (File, Edit, etc). A few basic menu commands are implemented, for example “About” and “Quit”.

Both applications had to be written from scratch, since the Winsock API is significantly different from the BSD sockets API used under Cygnus. Windows resources had to be created for each window and its contents.

The videoconferencing reciprocity client and server were called VidClient and VidServer, and the infra-red tunnelling client and server were called IRClient and IRServer.

Figure 6 shows the files required to create VidServer, and Figure 7 shows the files required to create VidClient. The list of files used to create IRServer and IRClient looks similar.


VidServer
File Edit View Go Favorites Help
Back Forward Up Cut Copy Paste
Address C:\Project\VidServer
My Computer

```
Debug            res              Cwinsock.cpp
Cwinsock.h       MainFrm.cpp      MainFrm.h
resource.h       StdAfx.cpp       vidsrv.aps
vidsrv.clw       vidsrv.cpp       vidsrv.dsp
vidsrv.dsw       vidsrv.h         vidsrv.mdp
vidsrv.rc        vidsrvDoc.cpp    vidsrvDoc.h
vidsrvView.cpp   vidsrvView.h     StdAfx.h
vidsrv.plg       server.prf       vidsrv.ncb
vidsrv.opt
```

25 object(s) (plus 2 hidden)

Figure 6: VidServer source files

4.3 Videoconferencing Reciprocity Application

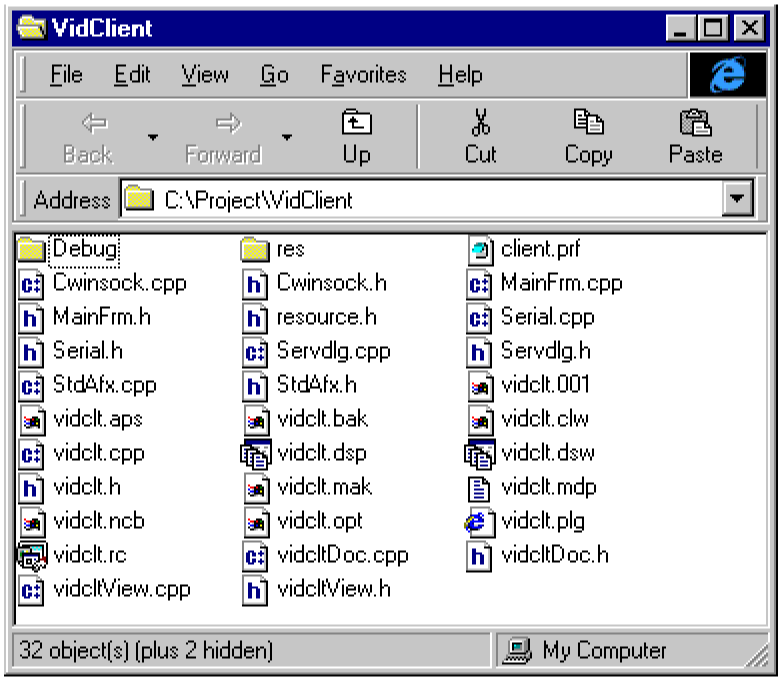
The server application, VidServer, waits for incoming calls on a pre-specified port (2000 by default). At most, two incoming calls are accepted. This type of application is also known as a daemon — it waits for connections and then handles them accordingly.

The client application, VidClient, connects to the server application at a IP address specified by the user. The host port that the server listens on is specified by the user in a text file. The client application has a similar text file, which contains the host port number to connect to, as well as the COM (serial communications) port to communicate on and the baud rate of the COM port.

The data that is received from the COM port is a projector message (the protocol is developed by Rutledge Engineering), usually sent to a data projector. Only two commands are recognised, "turn projector on" and "turn projector off". The projector messages consist of a string of five bytes — in order, they are: start, projector address, command, checksum and stop (Rutledge 1999).

After receiving a message, an acknowledgement message is sent in reply. This consists of six bytes — in order, they are: start, projector address, command[0], command[1], checksum and stop.

All message bytes are unsigned. The start and stop bytes indicate the beginning and end of a message. They are `FE` and `FF` respectively. The projector address is used to differentiate between multiple projectors. Since we are only communicating with one projector, the projector address is `01`. The checksum is the sum of all bytes between the start and stop bytes. A number of command bytes may be sent. For the purposes of the project, only one command needs to be sent at a time,



```
C:\Project\VidClient

Debug           res             client.prf
Cwinsock.cpp    Cwinsock.h      MainFrm.cpp
MainFrm.h       resource.h      Serial.cpp
Serial.h        Servdlg.cpp     Servdlg.h
StdAfx.cpp      StdAfx.h        vidclt.001
vidclt.aps      vidclt.bak      vidclt.clw
vidclt.cpp      vidclt.dsp      vidclt.dsw
vidclt.h        vidclt.mak      vidclt.mdp
vidclt.ncb      vidclt.opt      vidclt.plg
vidclt.rc       vidcltDoc.cpp   vidcltDoc.h
vidcltView.cpp  vidcltView.h

32 object(s) (plus 2 hidden)
```

Figure 7: VidClient source files

except for acknowledgement messages, which require the first command byte to contain `00`, and the following command bytes to contain the commands received.

The message received is also passed on to the projector, which can be connected on a separate COM port. This is necessary since the infra-red motion detector is now connected to the computer, rather than the projector, as was previously the case. After passing on the message to the projector, VidClient executes the relevant script file using the `ShellExecute()` MFC function.

A simple internet protocol was developed to transmit the "on" and "off" messages from each VidClient to VidServer, and to transmit the result of the logical "and" from VidServer to each VidClient. The protocol is ASCII-based, for easier debugging. VidClient sends one of `ONN`, `OFF` or `ERR`, depending on whether the state of the infra-red motion detector is on, off, or something else (this is unlikely, but implemented as part of the protocol as a precaution). VidServer sends back one of `ONN`, `OFF` or `MIX`, depending on whether the state of both clients was both on, both off, or a combination. A null byte follows each message, so it can be treated as a string when it is received. Each message is therefore four bytes.

TCP/IP communication is done using Windows sockets (Winsock). As with MFC, Winsock is message-based. For example, when a stream socket is created, a message ID number is assigned to it. A function (in this case, `CVidsrvView::OnStreamSrv()`, is associated with the ID number, using the MFC "message map". The MFC message map is a way to map message ID numbers to user functions. When a Winsock connection request is received, the `CVidsrvView::OnStreamSrv()` member function is automatically called.

This function handles connection requests, and decides whether to accept a connection or not. The algorithm used here is straightforward. Two pointers are previously assigned to nil, but are assigned to stream sockets if a connection already exists. If either of the two pointers is nil, then the connection is accepted, and assigned to the (previously nil) pointer. If both are assigned non-nil values, then the connection is refused.

The values of these two pointers are kept up to date. If a connection closes, or a connection error occurs, the offending socket is closed down and assigned nil. This way, connections can be closed and opened at any time, and the server will adjust to reflect the current connection state.

However, there is one exception. If a client process dies without its connection being closed (e.g. the client computer crashes, or the connection is physically broken and cannot be reconnected), then the server will remain "connected" to that client. The socket pointer will not be assigned nil, and the server will assume that the connection still exists.

This is a problem that exists with many IP applications. Unless a server application regularly polls its clients, this type of disconnection cannot be detected. However, there is a relatively simple solution if this occurs to VidServer — quitting and relaunching the server and then the client applications. This forces the old connections to be closed and reopened.

VidServer handles all its Winsock communication asynchronously. This means that it can send and receive data simultaneously. When VidServer sends data through a stream socket, it actually just initiates the send. The transfer takes place in the background. This leaves VidServer to continue its other tasks, such as receiving data from one of its clients.

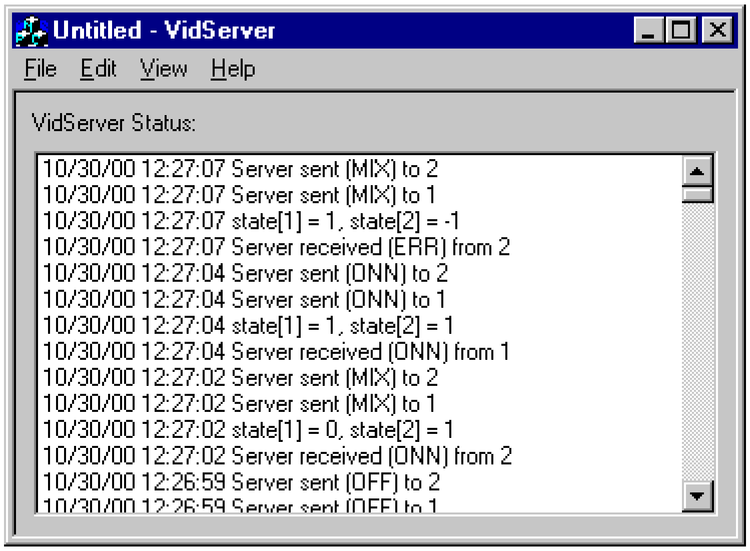
TCP/IP communication is achieved using the `CWinsock` class. This class was provided as part of the Visual C++ package. The serial port communication class (`CSerial`) was originally taken from public domain source code, but has been heavily modified to allow greater customisation required by VidClient. VidServer does no serial port communication.

VidServer and VidClient both read settings stored in external text files, named "server.prf" and "client.prf" respectively. They must be located in the same directory as the application itself. The ".prf" extension is a contraction of the word "preferences". Each preference file contains different settings. The files are text files, with one line for each setting. The "server.prf" file just contains settings for the listening TCP/IP port. The "client.prf" file contains settings for the server TCP/IP port, the COM port name and baud rate for the infra-red detector, the COM port name and baud rate for the projector connection, and the paths to the "up link" and "down link" script files. If a preference file cannot be found or opened, default settings are used, which are hardcoded into each application. However, the file must conform to the structure given, otherwise the settings will not be read correctly, causing unpredictable results.

Upon launching VidServer, the user is presented with a window (see Figure 8). The window contains a list displaying the current status of the server. Each line represents a separate event, and each event is timestamped. Events include Winsock initialisation, server port creation, client acceptation and connection, receiving and sending data, the current state of the two connections, and any errors that occur.

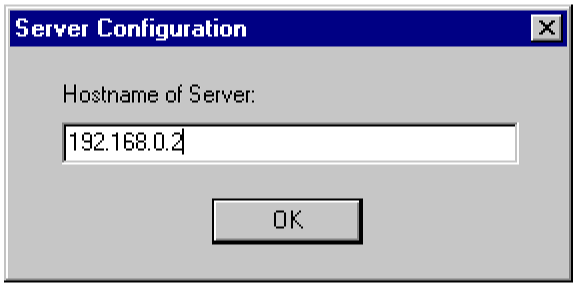
VidClient must be launched after launching VidServer, so that it can establish a connection with VidServer. Upon launching VidClient, the user is presented with a dialog box requesting the user to enter the host name or IP address of the server they wish to connect to (see Figure 9). After connecting to the server, a window is displayed, similar to the VidServer window (see Figure 10).

As with VidServer, this window contains a list containing a record of timestamped events. Events include Winsock initialisation, socket stream creation and connection to the server, receiving and



```
10/30/00 12:27:07 Server sent (MIX) to 2
10/30/00 12:27:07 Server sent (MIX) to 1
10/30/00 12:27:07 state[1] = 1, state[2] = -1
10/30/00 12:27:07 Server received (ERR) from 2
10/30/00 12:27:04 Server sent (ONN) to 2
10/30/00 12:27:04 Server sent (ONN) to 1
10/30/00 12:27:04 state[1] = 1, state[2] = 1
10/30/00 12:27:04 Server received (ONN) from 1
10/30/00 12:27:02 Server sent (MIX) to 2
10/30/00 12:27:02 Server sent (MIX) to 1
10/30/00 12:27:02 state[1] = 0, state[2] = 1
10/30/00 12:27:02 Server received (ONN) from 2
10/30/00 12:26:59 Server sent (OFF) to 2
10/30/00 12:26:59 Server sent (OFF) to 1
```

Figure 8: VidServer main window



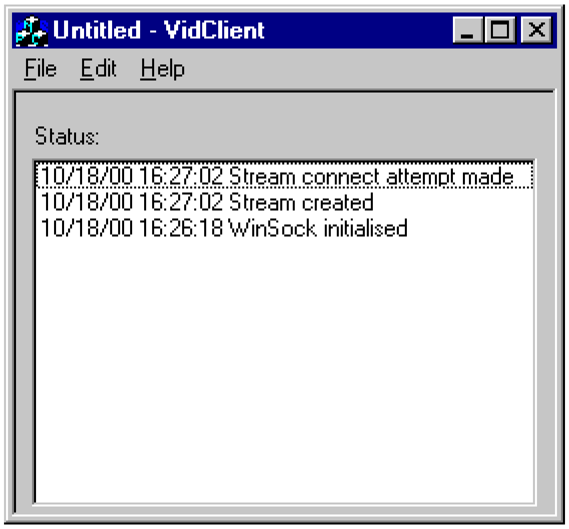
Hostname of Server: 192.168.0.2

Figure 9: VidClient configuration window

sending data, disconnection from the server, whether the link is starting up or shutting down, and any errors that occur.

For ease of testing, a test mode has been implemented (this can be turned on by recompiling the VidClient application with `TEST` defined). Figure 11 shows two VidClient applications with the test mode on. Test mode automatically generates “on”, “off” or “error” messages every five seconds, to simulate data coming from the infra-red motion detector on the serial port.

Currently, the only link between the two VAN computers is the ATM link, which carries the MPEG data. In the new system, the TCP/IP connection could be connected via the university’s standard Ethernet network, os possibly through the existing ATM link. This was previously unavailable with the VAN II 2.0 software (neither VAN is currently connected to the university network), but is possible with version 3.0.



```
10/18/00 16:27:02 Stream connect attempt made
10/18/00 16:27:02 Stream created
10/18/00 16:26:18 WinSock initialised
```

Figure 10: VidClient main window

4.4 Infra-red Tunnelling Application

The Visual C++ frameworks for the VidClient and VidServer applications were duplicated and used as a basis for the infra-red tunnelling client and server. They were renamed IRClient and IRServer.

The duplicated frameworks were modified so that IRClient reads data from the serial port, buffers it, and sends it to IRServer. IRServer in turn sends it to the other client.

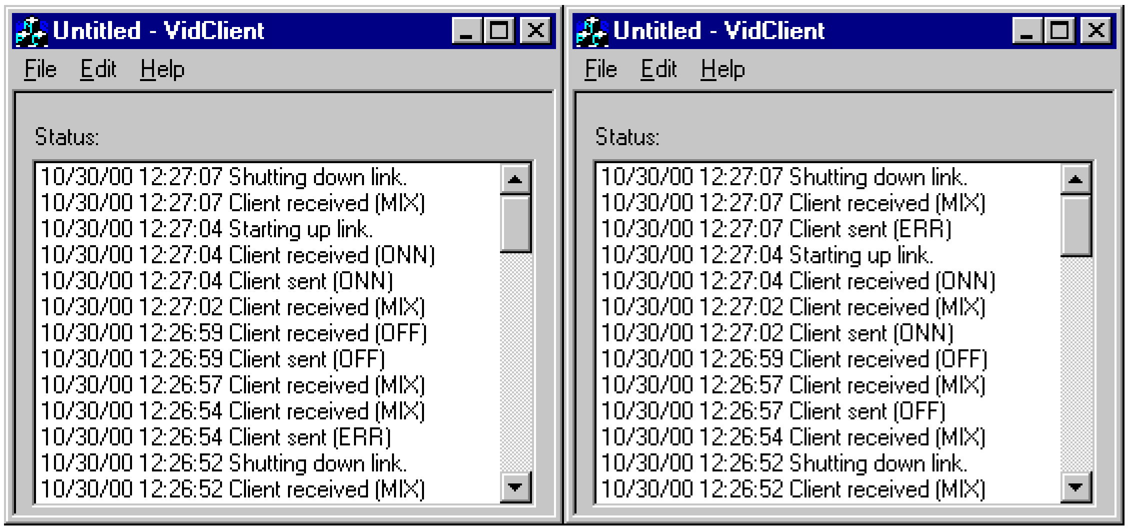
The infra-red tunnelling application could be used for other purposes. Any byte-based serial device that can withstand a certain amount of latency can be substituted for the infra-red transceivers. Such devices may include printers and some storage devices.

The serial device needs to be byte-based in order for it to work correctly. This means that it sends or receives data in packets of bytes, not bits. When application software reads from the serial port, the data received is a multiple of bytes. If a non-byte-based device sends one bit of data to a PC's serial port, the computer pads the data with zeros to fill one byte (IRDA 2000). It is up to the application to do any required data conversion.

Since neither IRClient nor IRServer do IrDA data translation, the data that IRClient sends may not be what is received if a non-byte-based serial device is used. If data translation were added, this would limit the use of the application, since it would now only work with a particular serial device. However, most devices are byte-based, since they are intended to be used directly with PCs.

The IRClient and IRServer applications are similar in design to their VidClient and VidServer counterparts. IRServer waits for incoming connections on port 2200 by default, so that it doesn't clash with VidServer's waiting port.

IRClient, like VidClient, initially displays a dialog asking the user to enter the host name of the IRServer they wish to connect to (VidClient's identical dialog box is shown in Figure 9). IRClient then connects to IRServer's listening port, and displays a window containing a list of timestamped events (see Figure 12). Events displayed mainly involve server connection and disconnection and any



```
10/30/00 12:27:07 Shutting down link.
10/30/00 12:27:07 Client received (MIX)
10/30/00 12:27:04 Starting up link.
10/30/00 12:27:04 Client received (ONN)
10/30/00 12:27:04 Client sent (ONN)
10/30/00 12:27:02 Client received (MIX)
10/30/00 12:26:59 Client received (OFF)
10/30/00 12:26:59 Client sent (OFF)
10/30/00 12:26:57 Client received (MIX)
10/30/00 12:26:54 Client received (MIX)
10/30/00 12:26:54 Client sent (ERR)
10/30/00 12:26:52 Shutting down link.
10/30/00 12:26:52 Client received (MIX)
```

```
10/30/00 12:27:07 Shutting down link.
10/30/00 12:27:07 Client received (MIX)
10/30/00 12:27:07 Client sent (ERR)
10/30/00 12:27:04 Starting up link.
10/30/00 12:27:04 Client received (ONN)
10/30/00 12:27:02 Client received (MIX)
10/30/00 12:27:02 Client sent (ONN)
10/30/00 12:26:59 Client received (OFF)
10/30/00 12:26:57 Client received (MIX)
10/30/00 12:26:57 Client sent (OFF)
10/30/00 12:26:54 Client received (MIX)
10/30/00 12:26:52 Shutting down link.
10/30/00 12:26:52 Client received (MIX)
```

Figure 11: Two VidClient windows

errors that occur. Individual data buffers are not shown in the event list, as they are in VidClient, since these are very frequent and are of little use to the user.

IRServer, like VidServer, opens up a port on which to accept connections. It displays a window containing a list of timestamped events (see Figure 13). Events displayed mainly involve client connection and disconnection and any errors that occur. As with IRClient, data buffers are not displayed in the event list.

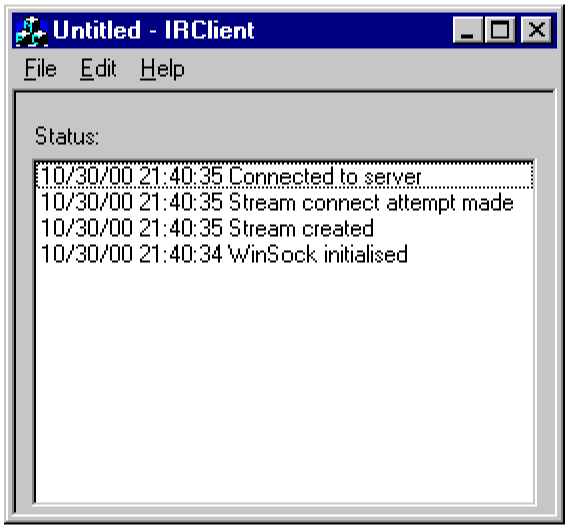
Each IRClient makes a connection to a server. It then opens a connection to a serial port. This serial port may be a direct COM port (such as COM1 or COM2), or may be a virtual COM port. The QuickBeam infra-red drivers create a virtual COM port, typically COM4, although this can be changed, and COM5 is commonly used if COM4 is a real port.

The QuickBeam software lets users specify the port using two names — a short name (e.g. “COM4”) and a long name (e.g. “COM4 (IrCOMM port)”). The short name is useful for standard communications programs which expect port names to be of the form COMx, where x is a number (also note that there is no space after “COM”). The long name is useful when a more descriptive name is required, such as in a list of communication ports shown to the user. IRServer uses the short name in its preference file by default, but either name can be used.

IRClient receives data from the serial port by reading data from the port every n milliseconds ($n = 500$ by default). The serial data is kept in a buffer until an explicit read call has been made.

If data was received from the read call, IRClient then sends the data to the IRServer application, via the TCP/IP connection. The IRServer application then receives the data. If two clients are connected, then it sends the data on to the second client. The second client receives the data and writes it to the serial port. Thus the infra-red data is converted to electronic data, transmitted across the internet using TCP/IP, and then converted back to infra-red signals at the other end.

As mentioned in Section 2.6, latency is not a problem with IrDA. IrDA is designed to withstand a certain amount of latency — it is important since it is easy to lose a connection if two handheld



```
10/30/00 21:40:35 Connected to server
10/30/00 21:40:35 Stream connect attempt made
10/30/00 21:40:35 Stream created
10/30/00 21:40:34 WinSock initialised
```

Figure 12: IRClient main window

devices are not pointing directly at each other, or if the connection is temporarily broken by someone walking in between the two devices. As also mentioned in Section 2.6, IrDA will also vary its baud rate to maintain a “good” connection (low error rate). Bright light, for example, can introduce errors. In any case, most files sent through infra-red are small text files — business cards, memos, etc — perhaps 4K at the most.

As with VidServer and VidClient, the settings are stored in preference files. The “server.prf” file just contains the TCP/IP port number on which to listen for connections. The “client.prf” file contains the TCP/IP port number on which to contact the server (this should match the port number in the “server.prf” file), and the COM port name and baud rate of the port to be used for tunnelling data. Either the long or short port names can be used. Physical port names (such as COM1 or COM2) can be substituted if a non-infra-red transceiver is to be used. If either preference file doesn not exist, standard settings are used.

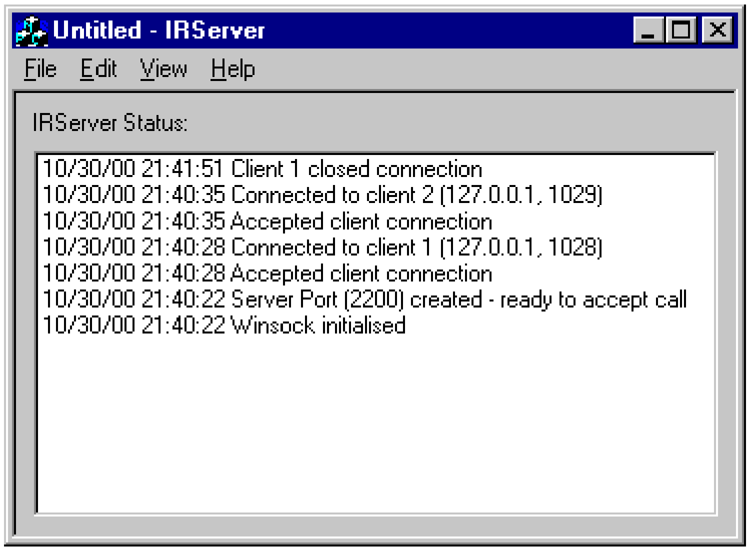
IRServer, like VidServer, does not poll its clients, so if the connection dies, both the server and clients need to be quit and relaunched to reestablish the connection.

4.5 Camera Position

As well as the two main improvements to the videoconferencing system (the videoconferencing reciprocity application and the infra-red tunnelling application), the camera position has been raised to approximately 1.2 metres above the floor, and is set in the wall, with the projected light being blocked around the camera lens.

This height is suitable for seated face-to-face conversation, and is a major improvement over the previous height of approximately 60 cm.

The author was not involved in the repositioning of the camera, but because it was part of the original design, it has been incorporated into the study.



```
10/30/00 21:41:51 Client 1 closed connection
10/30/00 21:40:35 Connected to client 2 (127.0.0.1, 1029)
10/30/00 21:40:35 Accepted client connection
10/30/00 21:40:28 Connected to client 1 (127.0.0.1, 1028)
10/30/00 21:40:28 Accepted client connection
10/30/00 21:40:22 Server Port (2200) created - ready to accept call
10/30/00 21:40:22 Winsock initialised
```

Figure 13: IRServer main window

4.6 Automatic Switching

The automatic switching application has been partially implemented, as part of the videoconferencing reciprocity application. If, in the future, three video walls exist (for example, one at the Caulfield staff room, one at the Clayton stall room, and new one at the Clayton meeting room), then there will be three VidClients trying to connect to the VidServer. If there are people in the Caulfield staff room and the Clayton staff room, then VidServer will link those two walls together, but if there are people in the meeting room and one of the staff rooms, then VidServer will link those walls together.

Most of the time, the video wall in the meeting room would be turned off, and would only be turned on when necessary, eliminating unnecessary connections if a Clayton-only meeting took place.

This scenario could also work if there are people in all three rooms. Initially the video link might be between the two staff rooms, but if a meeting is taking place, then as people leave the Clayton staff room and head to the meeting room, the link between the two staff rooms is shut down and the link between the Caulfield staff room and the Clayton meeting room starts up.

However, it is only a temporary solution. More control is needed, especially if two or more new video walls are installed. Since VidServer accepts a maximum of two connections, only one link is allowed simultaneously, but in the future, two concurrent links (e.g. between two common rooms and two meeting rooms) might be necessary. Also, there is little reason to have a video link between two rooms at the same campus. Additional, customisable software, or possibly modifications to VidServer and VidClient, would be required to implement these features.

If the software chooses appropriate video links automatically based on rules, such as "never link two VANs from the same campus together", then a method to identify each VAN needs to be implemented. This could involve numbering each node, or perhaps using descriptive terms, such as campus (e.g Clayton) and room type (e.g. meeting room) as a basis for rule making. The latter would require more work initially, but would allow new nodes to be set up more quickly.

4.7 Camera Control

Originally, the camera had been mounted on the wall, but since the camera repositioning, it is now set, recessed, in the wall. Some modification would be required to add the possibility of camera control, which is yet to be implemented.

There are three ways to control a camera: pan, tilt and zoom. Panning refers to swivelling the camera about its x-axis (left-right), tilting refers to swivelling the camera about its y-axis (up-down) and zooming refers to magnification. The modification would involve replacing the camera set in the wall with a camera on a small wall-mount with a motorised camera controller.

Panja (formerly AMX Corp.) is a company that manufactures camera controllers. These are small motorised Some are designed for computer control, for example, the Panja AXB-F117 Stealth 1 Camera Controller (Panja 2000). This camera controller connects to the proprietary AXlink data bus, or an RS-232 connection, which enables communication with PCs. The VAN is equipped with AMX control (First Virtual 2000), and thus can be used to directly control cameras.

Hardware interfaces (for example, the Panja AXB-CAM) can be used in conjunction with a camera controller to adjust the camera's pan, tilt and zoom. A software user interface based on these hardware interfaces would need to be designed to allow similar functionality, and a internet protocol would be needed to send the commands to the remote camera controller.

The implementation of a user interface would require some level of access to the VAN. Currently, the VAN is locked in a case, and can only be opened by administrators. Perhaps the administrators would also be responsible for allowing use of camera control, or a separate computer could be set up for this task, leaving the VAN to do its primary tasks.

4.8 Video Recording

Video can currently be captured by substituting video camera recorders for the cameras currently in place, and setting the video cameras to record to tape. However, this requires much human intervention. A better solution would be for software to intercept the video as it is being transmitted and to save it to disk as an MPEG file.

An application to record video and audio from a video link (and play it back) has yet to be implemented. This would be useful for capturing meetings, but would not be used for the staff common rooms. A method for synchronising the video from the two cameras (one at each location) would be required, and could be achieved using a client/server model as used by the videoconferencing reciprocity application, if each VAN captured their own video. If each (or just one) VAN captured video from both cameras, a synchronising application would not be required. Issues such as disk speed and real-time video encoding would also need to be addressed.

Some people may be wary of being captured, and would like to know when video recording is taking place, so some visual indication could be displayed on the video wall when recording is in progress. For example, a flashing red circle could be displayed in the top-right corner of the video wall, to simulate the behaviour of a consumer video camera when in recording mode. This recording indicator would have to be displayed on both video walls, even if only one side was recording the video. This information would need to be communicated to both VANs, and would require a simple internet protocol. Any visual feedback would not be captured to disk — it would just be overlayed on the live video projection.

5 Discussion

5.1 Videoconferencing Reciprocity Application

The videoconferencing reciprocity applications were the first part of the system to be developed. As mentioned in Section 4.1, the software was developed in Visual Studio only after it was discovered that Cygnus did not support serial port communications. Having no previous experience developing under Visual Studio, much of the first few months of development was spent learning how to use Visual Studio, MFC, the Winsock API and Windows resources.

As mentioned in Section 4.1, the current release version of the VAN II software, version 2.0, runs under OS/2. First Virtual's next version, 3.0, which is still in beta, runs under Windows NT. The VAN II software for Windows NT was first requested in March 2000, but was only sent, after many requests, in September. However, the package was incorrectly addressed, and took a number of weeks to arrive. VidServer and VidClient were developed independently of any VAN software.

The VAN II software received was version 3.0 beta 2, released in June 2000. Documentation for the software was available from First Virtual's web site. This included a manual for version 2.0, and an addendum of changes since version 2.0.

The existing VAN II 2.0 software is controlled using Audio Visual Services (AVS) command-line scripts. AVS is the software that controls the audio-visual functionality of the VAN. This software is also referred to as MOS H.310. In VAN II 3.0, the AVS commands are not available directly. A configuration program called "VAN Resource Manager" replaces the command-line scripts, and currently there is no way to replace command-line scripts for action such as "start AVS" and "stop AVS". However, further examination of the features of VAN II 3.0, or an update in a future version, may solve this problem. As previously mentioned, the author is not involved in upgrading the VANs.

In addition, the original VAN 2.0 software is being used while the software is upgraded on separate hard disk drives. The VAN computers have currently not been completely set up to run the VAN 3.0 beta software, which is unfortunately beyond the author's control. Since the new software has not been installed the application could not be tested *in situ*.

Despite this, all components of VidServer and VidClient have been tested — reading from and writing to the serial port, sending data to the server or clients, receiving data from the server or clients, and executing script files all work. When the VAN it set up with the new software, any changes to the application will be minimal, if any are required at all. Most settings can be changed by editing the preference files as described in Section 4.3.

As mentioned in Section 4.3, a test mode has been implemented in VidClient (VidServer does not require a test mode), which has been used to test all aspects of VidServer and VidClient except for serial port communication. Serial port communication has been tested separately.

VidClient and VidServer have been left running in test mode for approximately eight hours continuously. This is important, because it is likely that, in practice, these applications will be left running continuously for many days. In normal use, the infra-red motion detector is likely to send an "on" or "off" message about once every minute. Since VidClient's test mode sends messages to VidServer every five seconds, eight hours of test mode represents many days of actual use.

5.2 Infra-red Tunnelling Application

Development of the infra-red tunnelling applications has been difficult for a number of reasons. The infra-red transceiver used, the Actisys 200L, has limited documentation, and was initially unable to be used under Windows NT.

The Actisys transceiver came with driver software for Windows 95. The documentation (Actisys 2000) stated that it was unnecessary to install the driver under Windows 98 or Windows 2000 since these versions of Windows already had the driver built in. However, there was no mention of Windows NT. Information on the Actisys web site stated that separate Windows NT drivers were required, and that the drivers were available for purchase from Extended Systems (formerly Counterpoint Systems) (Actisys 2000). The drivers are part of the QuickBeam NT software package. A version for Windows 95, QuickBeam 95, was also available but was not purchased. QuickBeam NT was purchased online and installed, and on this basis it was decided that a second infra-red transceiver should be purchased.

However, when the transceiver was tested with the QuickBeam NT software, no data could be received from the test source, an Apple Newton MessagePad 130. The Actisys documentation states that the Actisys 200L transceiver works with Newton MessagePads. The process was tested out on three PCs, including a notebook, all running Windows NT, with the same results.

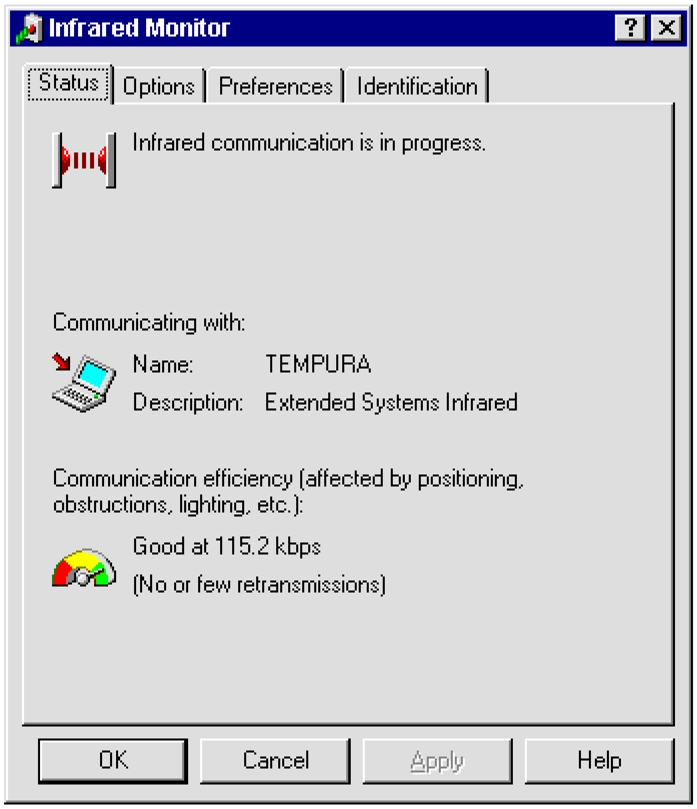
The Actisys infra-red transceivers and the QuickBeam software are both certified by IrDA to be IrDA-compliant. However, the Newton MessagePad uses Sharp's ASK infra-red technology, not IrDA, so that although the transceiver can receive signals sent by the Newton, the driver software cannot interpret them.

The driver was initially installed on a notebook computer running Windows 95. The software was able to be used without problems. The IRServer and IRClient applications are compatible with Windows 95, and because they have been built with MFC libraries linked in, they run on Windows 95 without any external libraries. However, since IRServer and IRClient were to be used in conjunction with the videoconferencing system, which runs on Windows NT, Windows 95 could not be used.

The infra-red driver opens the physical COM port that the infra-red transceiver is connected to, and send a message requesting the transceiver send data. Without the drivers installed, reading from the port returns no data, despite infra-red signals being sent to the transceiver. The protocol used by the transceivers is proprietary, and although it may be possible to decode the protocol, this hasn't been achieved. Since the driver opens the COM port, it can't be used by any other application (a COM port can be used by only one application at a time). The driver creates a virtual COM port, as described previously, which is used by applications to communicate with the infra-red device.

The Infrared Monitor control panel (see Figure 14) monitors data coming in through the transceiver, and displays the status of the infra-red connection — either a device is in range, no devices are in range, or, on Windows 95 only, "infra-red devices are operating nearby" (interference). When the Newton sends infra-red signals to the transceiver, the Infrared Monitor shows this as interference. Interference can also be caused by many remote controls (e.g. TV and VCRs). The Windows NT drivers do not show interference, instead displaying the "no devices in range" message (see Figure 15). Except for this difference, the Windows 95 and Windows NT infra-red drivers, and the "Infrared Monitor" control panel, look and behave identically.

In conjunction with QuickBeam NT, two Actisys transceivers were able to communicate with each other, and files could be sent using the QuickBeam software. The name of the PC being connected to shows up in the Infrared Monitor application where the name of the owner shows up in the case



Infrared communication is in progress.

Communicating with:

Name: TEMPURA

Description: Extended Systems Infrared

Communication efficiency (affected by positioning, obstructions, lighting, etc.):

Good at 115.2 kbps

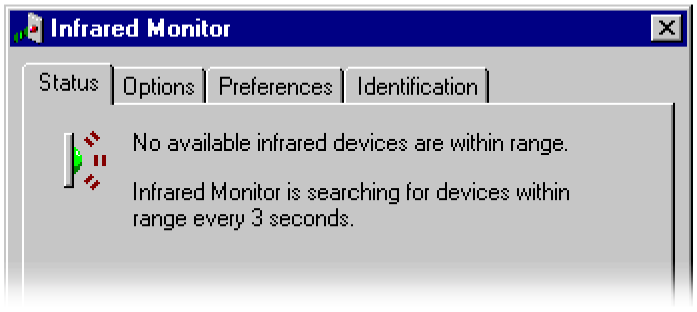
(No or few retransmissions)

Figure 14: Infrared Monitor control panel showing a connection to a device

of a PalmPilot (this is “Tempura” in Figure 14), and “Extended Systems Infrared” is displayed where the PalmPilot model would be shown. The QuickBeam NT software was also successfully used to transmit files from a PalmPilot. This proved that the combination of QuickBeam NT and the Actisys transceivers could be used.

However, when a terminal program such as HyperTerminal was tried, no data could be received from the virtual port. It was assumed that the QuickBeam NT software sends some handshaking commands to receive the data, and follows some protocol. A letter was sent to Extended Systems and Actisys, asking for advice. Extended Systems was unsure if it was possible, but suggested talking to the COM port directly, and also asking Actisys for suggestions. They also suggested using a low baud rate (9600 bps was being used for testing, which is the lowest baud rate supported by IrDA).

The Actisys transceivers use a proprietary protocol (Actisys 2000), which they license for a fee, and after signing a non-disclosure agreement. Extended Systems has used this protocol to create their QuickBeam NT drivers. It was not known what form the data transmitted on the QuickBeam virtual port took, so a request for information was made to Extended Systems, but Extended Systems was unable to provide more information. The protocol was initially thought to be standard IrDA, and standard IrDA handshaking commands were sent to the virtual port, without success, so it was assumed another protocol was being used.



No available infrared devices are within range.

Infrared Monitor is searching for devices within range every 3 seconds.

Figure 15: Infrared Monitor control panel showing no devices in range

However, IRServer and IRClient have been tested successfully with null modem connections and some other serial devices. In fact, it works with any serial device that initiates its own protocol (and can withstand some latency). A MIDI keyboard is such a device, and can be used with IRServer and IRClient, although this hasn't been tested with two MIDI keyboards. The infra-red transceivers, and the protocol used by the QuickBeam virtual port, on the other hand, require handshaking to be initiated by the application.

5.3 Using the Servers and Clients

The IRServer and VidServer application are intended to be launched and left to run in the background, and not interfered with unless some error occurs requiring manual intervention. This "no hands" approach is important because we are trying to create a virtual room where technology is not obvious, and people can use the system without having to use any equipment directly.

When the applications are first installed, they must be configured by editing their preferences files. The TCP/IP port number shouldn't require changing, but the COM port names (and possibly baud rates) will probably need to be changed. The preference files shouldn't require further modification, unless particular hardware modifications are later made to the system.

The VidServer and IRServer applications only need to be installed on one VAN (probably at Clayton), while VidClient and IRClient need to installed on each VAN. Each server application must be run before the clients are, otherwise the clients will be unable to contact the server. The order in which the disconnections occur (client first or server first) doesn't matter.

If required, use of the system can be monitored by analysing the timestamped log windows provided in each of the client and server applications. As mentioned in sections 5.1 and 5.2, these show when connections were made, when disconnections occurred, what data was sent and received, and in the case of VidClient and VidServer, the current state of the system (up or down).

As mentioned in Section 5.2, the IRServer and IRClient applications, in their current form, cannot be used for transmitting IrDA data from the Actisys transceivers. However, if other transceivers become available in the future which send data without handshaking, then the IR applications will automatically work with them.

5.4 Response

Since the partial upgrading of the videoconferencing system, more use has been made of it. Recently, the weekly computer science seminars have been displayed to the other campus via the video wall. A number of meetings have also taken place over the video wall.

The response to the videoconferenced seminars has generally been good. At least one minor problem has been identified when videoconferencing the seminars. The seminars at Clayton are not held at the opposite end of the room to the video wall. This means that there are no people at the video wall end of the room, and thus the motion detector sends commands to the projector to switch itself off. This leaves the presenter with no view of the audience at Caulfield. If the videoconferencing reciprocity software had been installed, it could have been disabled during this time, and those problems wouldn't have occurred.

The camera used for the videoconferenced seminars was a typical video camera sitting on a tripod, not the camera set in the video wall. This has been controlled by hand to follow the presenter, to zoom in on him/her when he/she is speaking, and to zoom out when he/she is referring to a slide. This is a simple instance of how the camera control could be used in a meeting or group discussion.

There have also been occasions where the infra-red tunnelling application would have been useful. In one case, a guest speaker was talking to a staff member over the video wall. The staff member remarked that if infra-red data could be transmitted through the video wall, then they could swap contact details using their PalmPilots.

Informal conversation among various members of staff and students regarding the new features has been positive. Students in particular are interested in the new features, which have generated interest in the video wall. Some students have also taken part in discussions over the video wall. Several suggestions have been made, some of which have been incorporated in Section 6.2 below.

6 Conclusions

6.1 Review

This project has discussed the benefits of integrating new features to the existing videoconferencing system, and its aims, namely, to improve the usability and usefulness of the system. The idea has been to add new features “behind-the-scenes”, so that the new features do not cause any disruption to the current system and do not require manual human intervention.

So far, the videoconferencing reciprocity server and client applications have been developed. These applications address the first issue outlined in the aim (Section 3). Unfortunately, due to problems beyond the author’s control, the VAN system could not be configured with the new VAN II 3.0 software required by the project, and thus the applications could not be tested *in situ*. However, each application has been tested as much as possible, and all components work as required.

The infra-red tunnelling server and client applications have also been developed. Again, this could not be tested *in situ*, for a number of reasons, including those given for the videoconferencing reciprocity application. However, the main reason it could not be tested is that data could not be read directly from the Actisys infra-red transceivers, with or without the infra-red driver software. The reason for this is that a protocol must be followed, which is not known and may be proprietary. Using the QuickBeam NT software from Extended Systems, files could be sent from one transceiver to another, or between a PalmPilot and a transceiver. However, it has been tested using a standard serial connection between two computers, and the applications work successfully in this case. In its current form, the software can be used in instances where a protocol between the application and the serial device is not required.

The remaining features have been investigated and discussed. The author was not directly involved with the camera repositioning, but it has been completed. The other remaining features have not been developed. The VidServer and VidClient applications could easily be extended to incorporate the automatic link switching application. By purchasing camera control hardware, and implementing the relevant portions of the AXLink protocol, camera control could be developed, using a client/server approach similar to that used by the videoconferencing reciprocity and infra-red tunnelling applications. Video recording requires more research done in order to capture the MPEG data from the video stream and write it to an MPEG file. However, it too could use a client/server model to synchronise video recording and to notify video nodes that recording is in progress.

Response to the developed features and on the proposed features has generally been positive, and comments have been taken into consideration for future work.

6.2 Future Work

The additions made to the videoconferencing system so far are only part of the whole system.

Both the videoconferencing reciprocity and infra-red tunnelling applications can be extended and improved. There are a few minor interface changes that could be made. For example, many of the menu items in all of the applications are not implemented – in fact, most don’t need to be implemented, but should probably be removed (e.g Copy, Paste, Save and Print). There are left unimplemented because they were included in the template MFC application that was used to create the client and server applications.

VidServer and VidClient could be used as the basis for a central hub for controlling video wall links. VidServer and VidClient could include a menu item and/or a button to force close a connection or reinitiate a connection, which would be useful if there were connection problems. VidServer could poll its connections to ensure that the connection was still up, and automatically disconnect any clients that did not respond within a certain time. This would solve the problem of having to manually close and relaunch VidServer if the TCP/IP connection is lost.

Another minor change would be to improve the format of the preference files. Currently, if the preference files do not conform to the correct format, settings will be set incorrectly. A better format would be to use something like `PORT = 2000` instead of just `2000`, which would allow settings to be written in any order. Another improvement would be to add a Preference dialog to the VidClient and VidServer applications, allowing administrators to adjust settings without having to quit the application or edit text files.

Another possible addition would be to not only log the events in a window, but also write the events to a file. This log file could then be archived and would be useful for statistical purposes. For example, it would be possible to calculate when the system is being used most, and for how long it was used in a given time period.

The automatic link switching application could be incorporated into VidServer, since it is already partially implemented. In addition, VidClient could show a list of possible video links, and users could arrange the links in order of preference, or even to manually make a connection to another video link, even if that link was already connected to another node. This would be useful for a situation such as a meeting, since the two nodes to be connected are known in advance. As mentioned in Section 5.1, VidServer would need to be modified to allow more than two clients connecting to it.

Some parts of the video recording application could be built into VidServer or VidClient, such as a "record" button. However, a separate application would be more appropriate for playback tasks. Such an application could feature a VCR-like interface, with buttons for record, play, stop, fast forward and rewind, as well as a time display. However, more research needs to be undertaken in order to access the MPEG video data and write it to a file so that it can be played back.

An application to remotely control the camera could be built into VidServer and VidClient, but again, this might be better implemented as a separate application. A camera control device (such as the Panja F117 Stealth 1) first needs to be purchased for each camera to be controlled, and modifications need to be made to the cameras so that they can be attached to the controller. The controllers use the AXLink protocol, which can be obtained from Panja.

The application would need to be split into client and server components, and would behave similar to VidClient and VidServer. The "camera controller" client would need a user interface — perhaps a horizontal slider for "pan", a vertical slider for "tilt" and another horizontal slider for "zoom'. Each client would send a numerical value to the server whenever a slider was changed, which would in turn send it on to the other client. That client would then generate the appropriate AXLink commands, and send them, via the serial port, to the camera control device.

More research needs to be done on the protocol used by the QuickBeam infra-red drivers before the infra-red tunnelling application can be completed. If the protocol were implemented, this would open up the possibility of using infra-red devices as remote controls. For example, an application could be written for PalmPilots that uses IrDA messages to remotely control the camera.

As mentioned in Section 4.7, a separate computer could be installed beside the VAN to handle camera control tasks, and allow people to access the camera control software. This computer could also run the infra-red tunnelling application, providing easy access to the infra-red transceiver.

7 Appendices

7.1 Appendix A. Source Files

The complete source files for this project are available at:

`http://www.csse.monash.edu.au/~rwc/honsproject.zip`

7.2 Appendix B. VidServer User's Guide

```
Before launching VidServer, you may need to configure the "server.prf" file.

The format of the "server.prf" file is as follows (default value in brackets):

line 1:  TCP/IP port number to wait for VidClient connections on (2000)

The file must be called "server.prf" and located in the same directory as VidServer.

After launching VidServer, you will be presented with a window containing a list.  Each
line in the list represents a different event.  Recent events are shown at the top of
the list.  At this point, VidServer is able to send, receive and handle connections from
VidClient.

VidServer will wait until two clients are connected to it before any transfer of data
takes place.  If only one client is connected, it will ignore all data it receives from
that client.

When you have finished using VidServer, choose "Exit" from the "File" menu.  This
will quit VidServer and close any open TCP/IP connections.
```

7.3 Appendix C. VidClient User's Guide

Before launching VidClient, you may need to configure the "client.prf" file.

The format of the "client.prf" file is as follows (default value in brackets):

```
line 1:  COM port name for communication with infra-red motion detector (COM1)
line 2:  COM port baud rate for infra-red motion detector (9600)
line 3:  COM port name for communication with projector (COM2)
line 4:  COM port baud rate for projector (9600)
line 5:  TCP/IP port number to connect to VidServer on (2000)
line 6:  absolute or application-relative path to "up link" script (upscript.bat)
line 7:  absolute or application-relative path to "down link" script (downscript.bat)
```

The file must be called "client.prf" and located in the same directory as VidClient.

After launching VidClient, you will be presented with a dialog asking you to enter the host name of the server you wish to connect to. This could be "localhost" if VidServer and VidClient are running on the same machine. After entering the host name, press the OK button.

You will then be presented with a window containing a list. Each line in the list represents a different event. Recent events are shown at the top of the list. At this point, VidClient is able to send, receive and handle connections from VidServer.

VidClient will display events in its list notifying you when the "up script" and "down script" are being run.

When you have finished using VidClient, choose "Exit" from the "File" menu. This will quit VidClient and close any open TCP/IP connections.

7.4 Appendix D. IRServer User's Guide

Before launching IRServer, you may need to configure the "server.prf" file.

The format of the "server.prf" file is as follows (default value in brackets):

```
line 1:  TCP/IP port number to wait for IRClient connections on (2200)
```

The file must be called "server.prf" and located in the same directory as IRServer.

After launching IRServer, you will be presented with a window containing a list. Each line in the list represents a different event. Recent events are shown at the top of the list. At this point, IRServer is able to send, receive and handle connections from IRClient.

IRServer will wait until two clients are connected to it before any transfer of data takes place. If only one client is connected, it will ignore all data it receives from that client.

When you have finished using IRServer, choose "Exit" from the "File" menu. This will quit IRServer and close any open TCP/IP connections.

7.5 Appendix E. IRClient User's Guide

```
Before launching IRClient, you may need to configure the "client.prf" file.

The format of the "client.prf" file is as follows (default value in brackets):

line 1:  COM port name for communication with infra-red transceiver (COM1)
line 2:  COM port baud rate for infra-red transceiver (9600)
line 3:  TCP/IP port number to connect to IRServer on (2200)

The file must be called "client.prf" and located in the same directory as IRClient.

After launching IRClient, you will be presented with a dialog asking you to enter the
host name of the server you wish to connect to.  This could be "localhost" if IRServer
and IRClient are running on the same machine.  After entering the host name, press the
OK button.

You will then be presented with a window containing a list.  Each line in the list
represents a different event.  Recent events are shown at the top of the list.  At this
point, IRClient is able to send, receive and handle connections from IRServer.

When you have finished using IRClient, choose "Exit" from the "File" menu.  This
will quit IRClient and close any open TCP/IP connections.
```

8 References

(Abramson 2000) `http://www.csse.monash.edu.au/~davida/VirtualTeaRoom/`

(Actisys 2000) `http://www.actisys.com/`

(Brookings 1999) Brookings Institution, "Cyber scholars build virtual bridges", *Communications News*, February 1999, vol. **36**, issue 2, p. 80.

(Cool 1992) Cool, C., Fish, R. S., Kraut, R. E. and Lowery, C. M., "Iterative design of video communication systems", *Conference Proceedings on Computer-supported Cooperative Work*, November 1–4, 1992, Toronto Canada, pp. 25–32.

(First Virtual 1999) First Virtual Corp., "Video Access Node Technical Supplement", November 1999

(First Virtual 2000) `http://www.fvc.com/`

(First Virtual 2000a) First Virtual Corp., "Video Access Node II (VAN II) User's Guide, Release 3.0 Revision C", August 2000

(Fish 1990) Fish, Robert S., Kraut, Robert E. and Chalfonte, Barbara L., "The VideoWindow system in informal communication", *Proceedings of the Conference on Computer-supported Cooperative Work*, October 7–10, 1990, Los Angeles, CA USA, pp. 1–11.

(Fish 1993) Fish, Robert S., Kraut, Robert E. and Root, Robert W, "Video as a technology for informal communication", *Communications of the ACM*, vol. **36**, issue 1 (1993), pp. 48–61.

(Fish 1994) Fish, Robert S. and Kraut, Robert E., "Networking for Collaboration: Video Telephony and Media Conferencing", *Proceedings of the CHI '94 Conference Companion on Human Factors in Computing Systems*, April 24–28, 1994, Boston United States, pp. 375–376.

(Giles 1999a) Giles, S., and Abramson, D., "The Virtual Tea Room: Integrating Video into Everyday Life", *International Wireless and Telecommunications Symposium*, Malaysia, May 17–21, 1999.

(Giles 1999b) Giles, S., and Abramson, D., "The Video Wall Project - Video Technology at the Leading Edge in Education", *Learning Technologies '99*, Noosa, Qld, October 20–23, 1999.

(IRDA 2000) `http://www.irda.org/`

(Inoue 1995) Inoue, Tornoo, Okada, Ken-ichi and Matsushita, Yutaka, "Learning from TV programs: Application of TV presentation to a videoconferencing system", *UIST*, November 14–17 1995, Pittsburgh PA USA, pp. 147–154.

(Ishii 1992) Ishii, Hiroshi and Kobayashi, Minoru, "ClearBoard: a seamless medium for shared drawing and conversation with eye contact", *Conference Proceedings on Human Factors in Computing Systems*, May 3–7, 1992, Monterey, CA USA, pp. 525–532.

(Kraut 1994) Kraut, Robert E., Rice, Ronald E., Cool, Coleen and Fish, Robert E., "Life and Death of New Technology: Task, Utility and Social Influences on the Use of a Communication Medium", *Proceedings of the Conference on Computer-supported Cooperative Work*, October 1994, Chapel Hill, NC, USA pp. 13–21.

(Panja 2000) `http://www.panja.com/`

(Perey 2000) Perey, Christine, "But implementation is still fuzzy; Topology, bandwidth and QoS questions remain open", *Network World*, March 13, 2000, p. 54.

(PictureTel 2000) `http://www.picturetel.com/`

(Rutledge 1999) User's Manual R5975236, pp. 2-23–2-24.

(Sellen 1992) Sellen, Abigail J., "Speech patterns in video-mediated conversations", *Conference Proceedings on Human Factors in Computing Systems*, May 1992, p. 49.

(Takao 1998) Takao, Shinji and Innami, Ichiro, "The effects of the two modes of video-conferencing on the quality of group decisions", *Proceedings of the 1998 Conference on Computer Personnel Research*, 1998, pp. 156–158.

(Tang 1994) Tang, John C. and Rua, Monica, "Montage: Providing Teleproximity for Distributed Groups", *Proceedings of the CHI '94 Conference on Human Factors in Computing Systems*, April 24–28, 1994, Boston United States, pp. 37–43.

(VTEL 2000) `http://www.vtel.com/`

(Walker 2000) Walker, Richard W., "Use of video and web apps drive LAN upgrades", *Government Computer News*, Feb 7 2000, vol. **19**, issue 3, p. 20.

9 Glossary of Terms

Actisys Company who manufactures infra-red transceivers.
ATM Asynchronous Transfer Mode, a low-latency, fast computer network.
CSSE Monash University School of Computer Science and Software Engineering.
Extended Systems Company who produces QuickBeam.
First Virtual Company who manufactures the VAN.
FVC First Virtual Corporation.
IR Infra-red.
IrDA Infra-red Data Association.
QuickBeam Software (drivers and application) used to transfer data between IrDA devices and computers using infra-red transceivers.
VAN Visual Access Node, the videoconferencing system used by Monash University's CSSE.
VAN II Newer version of the VAN.