Pandora: An Experiment in Distributed Multimedia

Tony King

ORL 92-5

To be published in the Proceedings of Eurographics '92, September 1992

Pandora: An Experiment in Distributed Multimedia

Tony King

Olivetti Research Limited,
24a Trumpington St, Cambridge, CB2 1QA, United Kingdom

Abstract

An experimental workstation is described which supports digital video and audio in a distributed environment, and which presents this functionality to the user through the medium of a video-extended implementation of the X Window System. The Pandora Workstation is built out of two quite separate parts. A standard UNIX machine (the Pandora Host) brings standard hardware and software computing resources to the system; a highly-specialised processing engine (Pandora's Box) handles the high-bandwidth, time-critical, and device-dependent processing. A 50 Mbit/second ATM network provides for real-time data communication within the system. Nineteen Pandora Workstations have been deployed within Olivetti Research Ltd and the Cambridge University Computer Laboratory, and are used routinely to run distributed applications including video mail, video conferencing, and real-time media delivery services.

1. Background

Since its establishment in 1986, a major theme of the work carried out at Olivetti Research Ltd (ORL) has been the design and use of high-performance networks. Many of the personnel come from a networking background and, in particular, some have been closely involved with the development of the Cambridge Ring [Wilkes79], and the Cambridge Fast Ring (CFR) [Hopper88] at the Cambridge University Computer Laboratory.

It was well-known that the slotted-ring structure of the CFR has particularly desirable properties for the communication of continuous media such as video and audio streams [McAuley90]. Fine-grain sharing of bandwidth and predictable, end-to-end delays are a consequence of the small packet size and asynchronous channel access method used by the CFR. The CFR, in fact, has always exhibited the properties of the Asynchronous Transfer Mode (ATM) mode of operation [Vorstermans88], which is a recent development in the networking community, aimed at providing support for integrated services including video, voice, image and text.

By the early part of 1988, ORL researchers had completed some rudimentary experiments in sending digital video and audio over the CFR [Wilson89] and it was clear that the idea worked at some level, but many questions remained unanswered: how would the network perform under the loading produced by a real user community, and what practical benefits does such a service bring to the user anyway? It became clear that a large-scale experiment would enable us to push the limits of our networking technology, and indicate future directions for our research; it would also provide experience of, and useful insights into, our target application area [Hopper91]. We resolved to design, build, and deploy a large (for us) number of multi-media workstations.

Our manpower was limited, so the project goals had to focus on the network and the applications, and allow the intervening levels of hardware and software to be designed for economy of effort, rather than architectural elegance. Our approach was to split the design into two parts. Firstly, an off-the-shelf UNIX machine provided all the facilities we needed to make the system usable, which we otherwise would have had to have developed for ourselves. Secondly, a video peripheral encapsulated the real-time processing required to deliver media to the user and, incidentally, provided the project with it's name - sufficiently scary things go on inside Pandora's Box that we recommend nobody look inside!

2. Pandora Architecture

We made three major architectural decisions to help speed the design effort:

Firstly, to provide the visual effect of integrated video and graphics by overlaying video windows onto workstation graphics using analogue-domain mixing, rather than attempting a fully integrated digital hardware solution.

Secondly, to run two separate networks to each system, an Ethernet for traditional traffic and a CFR for video and audio. This was not because the CFR was incapable of carrying both video and Ethernet traffic (this has been done), but because of the considerable amount of work which would have been required in order to provide the necessary infrastructure.

Thirdly, to use general-purpose programmable components rather than specialised logic. The resulting system was rather bulky but this was a small price to pay for a flexibility which allowed us to solve problems as they occurred by writing software. Only towards the end of the project did we understand the issues well enough to think about designing specialised hardware.

The distributed system architecture is shown in figure 1. We wanted a geographically distributed user community in order to provide a real test of the communication facilities Pandora would provide. ORL is about half a mile distant from the University, and both have house CFR and Ethernet systems which are bridged across a fibre link. A video repository of 3 GBytes capacity is located at ORL, and this gives both sites access to storage for about 6 hours of video and audio.

Figure 2 shows the hardware architecture of a Pandora Workstation. Two connections feed into the Box from the Host, a Transputer Link for control and status, and an analogue cable (normally connected to the monitor) which delivers composite video into the mixing logic in the Box. The Box is logically and physically split into five modules: Capture, Mixer, Audio, Server and Network, each implemented as an independent processing unit comprising a Transputer CPU, memory, high performance ports to the other modules, and the appropriate electronics for input/output. A brief description of each follows.

The Network Module implements a high-performance CFR interface. Two Transputer CPUs are used here; one implements the network protocols, the other functions as a device driver for the CFR interface chip.

The Capture Module hardware digitises a standard monochrome composite video input to 8-bit resolution, subsamples (crudely, by dropping samples) to reduce the horizontal spatial

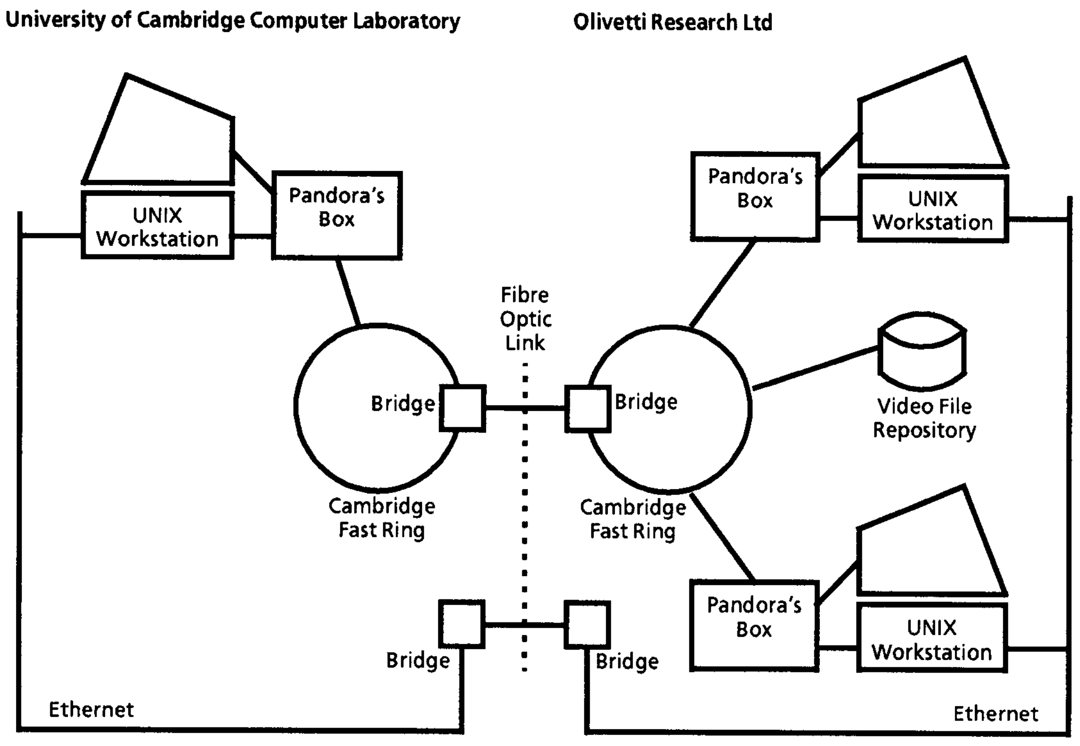

Figure 1. *Pandora Distributed Architecture*

resolution by a factor of 2 or 4, and stores the resulting image in memory for subsequent processing by the CPU. Under program control, lines of video are de-interlaced, and written to the output port for use by other modules.

The Mixer module CPU reads digital image data from it's input port, and writes it into an 8-bit deep framestore. The CPU also manages a 1-bit deep framestore (the Mask Plane) whose bits map one-to-one on to the pixels in the main framestore. These bits determine, for each pixel in the framestore, whether or not that pixel will be delivered to the screen, in preference to the corresponding pixel from the Host. The bit stream from the Mask Plane connects to the control input of an analogue switch which diverts either Pandora or Host video to the screen. In addition to providing an "exclusivity" mechanism - video windows obscure text and graphics, they do not mutually interfere - the Mask bits clip rectangles of video to arbitrary shapes, so that text and graphics windows can appear to obscure video. System timing is obtained by phase-locking on to the Host's video synchronisation pulses.

One subtle consequence of the scheme as described would be perceived by a user as an error - the cursor disappears as it moves over the video. A hardware keying mechanism is therefore provided such that a voltage level corresponding to anything other than "black" on the feed from the Host overrides the switch control and forces screen output to be taken from the Host. If the Host paints a black background wherever video is to be seen, then the cursor (if not black itself) will throw the switch and cause itself to be seen (this, in fact, duplicates the "exclusivity" function of the Mask Plane, but cannot perform the clipping function).

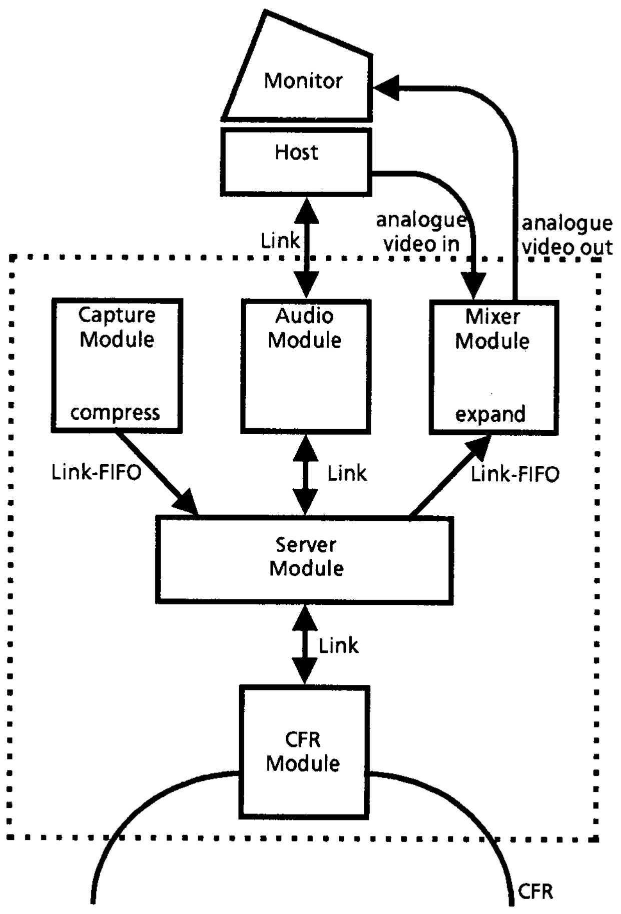

Figure 2. *Pandora's Box Architecture*

Two pipelined processing stages sit on the output port of the Capture Module, and on the input port of the Mixer Module, and perform video compression and expansion, respectively. They are provided primarily to allow a larger number of networked video streams than would otherwise be possible. The scheme uses horizontal DPCM with error propagation [Clarke89] to achieve a data reduction of 2 to 1. The expansion unit also provides horizontal and vertical interpolation which allows the displayed image to be increased in size by a factor of two. These two units are complex systems in their own right, and shall not be considered in further detail here. From the point of view of the main modules they are treated as part of the FIFO data path between the modules.

The Audio module comprises, at it's heart, a CODEC which converts the analogue signals provided by standard telephone handset equipment to a CCITT 64 kBit/sec digital data stream. This is buffered in an 8-bit wide FIFO for CPU access.

The Server Module may be viewed as an intelligent crossbar switch which handles the transfer of streams of data between the components described above. Although the hardware (a Transputer, fast memory and bus interface logic) is very simple, the functionality of this module is very high, and the performance requirements stringent, as described in detail in section 4.

3. Pandora's Box - Low-Level Processing

This section examines in detail the critical hardware-software subsystems which underlie the real-time performance of the system. Most important, in this respect, is the CPU, so we begin by discussing the Transputer's properties, as they affected us [Inmos89].

Transputer

For reasons of performance and modularity, the design model is that of multiple, communicating, processing subsystems. The Transputer offers support for multiprocessing such that the designer can concentrate on applying the technology, rather than inventing his own support environment from scratch.

Processes running on Transputers wishing to communicate perform input and output on named channels. Messages are unbuffered; if either process is not ready to communicate the other process blocks until both processes are ready. Channel communication thus provides the means of process synchronisation. The chip provides four 10 Mbit/sec communication links, each driven by a dedicated dma engine. If the channel name is associated with one of these hardware links then the communication occurs between processors, otherwise it occurs between processes running on the same chip. Process scheduling is performed in hardware, and therefore is quite fast, of the order of microseconds.

Links are convenient for inter-module communications (although FIFOs augment the links, as described below). The hardware scheduling is an important factor in Pandora's real-time performance, providing a performance and functionality which would be difficult to achieve purely by software means. One very useful feature is the on-chip timing facility which allows processes to measure intervals to microsecond resolution, or to be scheduled to run at specified times. Timers were used extensively during low-level software development to gain insight into the complex interactions of real-time processes, and so to tune the code for maximum performance.

Link/FIFO Ports

The ports which connect Capture to Server, and Server to Mixer, are implemented using a hybrid scheme which combines the raw speed of a FIFO with the convenient synchronisation properties of a link. A link, on it's own, is not fast enough to handle the many streams of video which we require to flow between the video modules, however, it is entirely suitable for the Server-Audio Module interface, and here links alone are used.

The port combines a 32-bit wide, 512-word deep FIFO, offering a bandwidth of 12 MByte/sec, with a pair of links which connect the processors (one bidirectional communication channel requires two unidirectional hardware links). Two procedures are defined for using the port: "segment.to.fifo" runs on the sending module and "segment.from.fifo" runs on the receiver. The former takes a block of data from memory, splits it into a number of small blocks, called Slices - each a fraction of the length of the FIFO - and writes each into the FIFO accompanied by a descriptive message which is sent down the link. The latter, complementary, procedure, runs upon receipt of the link messages, reads the Slices from the FIFO, and assembles them into a block of data in memory. A link buffer process maintains a queue of the link messages and models the ability of the FIFO to hold several Slices. This mechanism increases the degree of decoupling of the sender and receiver, ensuring that neither will become idle waiting for I/O.

Commands and Reports

As the processes which manipulate the real-time data execute, it is necessary to monitor their progress, stop, reconfigure, and restart them. Monitoring is important as it gives the higher levels of software opportunity to respond to overload or error conditions. Two channels are defined - Command and Report - which thread through each of the processors in the system, using the spare hardware links (those not used for FIFO synchronisation). The channel originates and terminates at the Host, and uses the Host-Pandora Transputer link as the physical communication path.

Display Processing

The traditional way to avoid the display of partially-updated frames in a moving sequence is to double-buffer - to update images off-screen and to display complete frames only. In the case of Pandora, we desire to display many streams which are mutually asynchronous with respect to their frame timing; there is rarely a moment, when all the incoming frames simultaneously are complete and ready for display. A display having, logically, many independent framestores, each to handle just one stream, is the ideal solution, and has been demonstrated [Wilkes84], but is too complex for our application.

Our approach is to take account of the position of the raster-scan whenever a video update is scheduled (a complete input frame is buffered), and to take care that writes and reads do not overlap. Segments of video coming in to the Mixer are buffered in off-screen memory and the update is started upon receipt of a start-of-scan interrupt. At the memory speed used in the Mixer - 300 nS per 32-bit word access - the Transputer can move data into the display space at approximately half the speed it is scanned out. However, at a scan rate of 60 Hz, and an input frame rate of 12.5 Hz, it is rare for more than one frame to be ready at the start of a scan; also, the rectangle of video is narrow compared to the scan, so the update usually can stay ahead of the scan.

The scheme as described (and implemented) does not cope properly with the visual priority of overlapping video windows; the input frames are seen randomly to interleave with one another. A solution is always to keep the last complete frame for each stream buffered in memory. Now, after a video window is updated, any other windows which obscure it are re-painted, so maintaining the correct visual effect. This has not so far been implemented due to space constraints in memory; however, the first revision of the hardware will have enough additional memory to allow this technique to be implemented.

Audio Processing

Pandora audio is sampled at 8 kHz and mu-law encoded to 8-bits, producing standard telephone quality audio. The samples are buffered in a FIFO and handled by the software in 16-sample (2 mS) quantities. 2 mS is a convenient quantity to work with since it's occasional loss is known to be tolerable for voice [Want88]. One of the main goals of the audio driver software is to control end-to-end delay. For a two-way conversation this must be kept below 10 mS; any longer and communication becomes difficult.

The main work of keeping the audio delay to acceptable levels is performed by buffer management code running on the Audio Module. Data is delivered from the network at a rate which varies slightly with time; this variability is called jitter and is a consequence of the asynchronous nature of the network. Typically, there are several incoming streams to be digitally mixed prior to delivery to the audio CODEC. The different streams all possess different amounts of jitter and all must be buffered prior to mixing, in order that the mixer process can deliver a steady flow of data to the CODEC. A queue of 2 mS buffers is maintained for each stream. The queue expands and contracts to soak up the jitter, but maintains a mean value of delay of under 10 mS. Occasionally, however, perhaps due to an abnormal condition on the network such as a reframe, the instantaneous jitter may become very large. The queue grows to accomodate this, and then settles down once more, but now with an unacceptably large mean value of delay. In order to recover, a claw-back strategy is adopted. Every 8 seconds, one 2 mS audio segment is dropped, and the behaviour of the queue examined; the process is repeated until the mean delay is back down to an acceptable value.

4. Pandora Intermediate-Level Software

In this section we look at the three levels of software which lie between the hardware interfaces, and the application: a video-extended implementation of X, an object-based interface to the Box, and data-path processing.

Video-Extended Implementation of X

X [Scheifler86] is the means by which various elements of the project are bound together into a coherent whole. At the most basic level X paints the black background against which the windows of video are shown. At a higher level the extension mechanism allows specialised functions for controlling video and audio hardware to be implemented and to exhibit the same performance as the core routines. At the system level the Client-Server communication protocol is the means of setting up connections between Boxes.

An X Server translates characteristics of display hardware into abstractions which can be manipulated by client library routines (Xlib); similarly, the Olivetti video extension to the X Server provides for abstractions of our video and audio hardware to be manipulated by a set of specialised video library routines (Xv). The abstractions for the Pandora hardware are called Objects, and are typed: types include VideoIn, VideoOut, AudioIn, AudioOut and PandorasBox. An Object has Controls, the manipulation of which allows control of the underlying hardware resources.

The Xv library provides the client interface to Objects and their Controls. The routines fall into four functional groups: those that query the status of Objects and Controls, those that set the values of Controls, those that make and destroy Objects, and those that make and destroy connections between Objects. As a concrete example consider setting up a video connection between two systems. Both Servers have, by default, a "PandorasBox" Object initialised. Other Objects are created by setting Controls on this Object; so calls to Xv procedures

XvMakeVideoOut() on one Server, and XvMakeVideoIn() on the other, create video source and sink Objects, respectively. Finally, a call to XvMakeVideoConnection() causes CFR network addresses to be exchanged between the two Servers, and a video connection to start up.

It is worth noting here a consequence of extending the Client-Server architecture for video, which took some of us slightly by surprise. The X Server is so called because it exposes it's functionality to any client on the network. Any machine can access, and draw graphics upon, the display at which a user is working. A video-extended X Server, however, also allows any client to start a video stream out of the local camera at any time. When using a Pandora Workstation, judicious use of a lens-cap is encouraged!

The Interface to Pandora's Box

The style of interface just described is mirrored here, at the next level down, by the interface between Pandora's Host and Pandora's Box. The Interface Software [Wray89] has some characteristics similar to the X Client-Server model. A set of library routines are provided for use by the device-dependent part of the X-Server's video extension, to control the Box-side of the interface. Request-reply and asynchronous event channels are implemented over the Host-Box Transputer Link. A system of abstractions of Box resources (called Objects) is maintained.

However, whereas standard X models hardware as having a fixed connectivity between devices (display, keyboard, mouse), the Pandora Interface aims to control a far richer set of resources (video and audio sources and sinks), and to enable the hardware connectivity dynamically to be re-configured as streams are created and destroyed.

The Objects (these are quite different entities to the X abstractions called Objects) are components of an idealised machine for manipulating streams of video and audio. They are created and destroyed by sending requests to a pre-existing "Factory Object" and connect via ports to other objects. An example is the "Audio Source Object" which takes audio segments from a microphone and delivers them to an output port. The ports are typed and the type system is used to prevent nonsensical connections. Because the Objects represent real resources, the Factory is entitled to refuse to honour a request to make an Object until sufficient resources are released.

All Objects can send a standard set of events representing common error conditions, back to the Host. In addition, particular event types are defined for particular error conditions in particular Objects. The "event.input.too.fast" of a "Network Sink" Object complains about a network connection refusing to accept data at the rate at which it is offered, presumably because the receiver is having trouble keeping up and data packets are returning to the sender marked "not accepted".

This situation illustrates an important principle of design of the low-level software - referred to by the designers as "data is blown, not sucked". In other words, no flow-control mechanism is imposed on network streams. The transmitter sends data as fast as it wishes; if the receiver starts to lag then "back-pressure" is detected by the sender and communicated via the event mechanism to the Host, and thence up to the higher levels of software, which may wish to take corrective action.

Pandora Data-Path Software

The Data-Path software [Jones90] provides data transport on the Server Module between sources and sinks of video and audio data within the box. The quality of the images and sound which are delivered to the user depends on the performance of this software. We require that, as the load increases, or error conditions develop, the system copes as best it can, degrading gracefully, reporting it's status to the higher levels, and recovering when possible.

Routing of the streams through the Server is table-driven. Each stream has an ID and each table has an entry corresponding to this ID which contains routing information. This routing information is provided by the Interface Software described above. A stream is set up by writing the tables in a back-to-front fashion, from the sink towards the source, and data flows when the path is complete. Data is held in queues of buffers, each of which is tagged with a stream ID. Whenever a buffer is filled, it's intended route is looked up using the ID, and buffer references are passed to the appropriate output processes. One copy moves data from a source Port into a buffer; one copy is made from the buffer to each destination Port.

The result is that the task of moving a quantity of data from input to output is broken down into short, self-contained steps; each step is designed to require as few resources as possible, and to be decoupled from the activities of other processes. Fine-grain interleaving of processes, good use of CPU cycles, and freedom from idle states and deadlock result from using this design principle.

An additional subtlety is to maintain high and low priority buffer queues for data coming off the network (the usual bottleneck) for audio and video, respectively. In adverse conditions, audio data is read in preference to video, causing video segments to be thrown away at the sender (and causing the picture to break up), but allowing the sound to survive - important in the case of person-to-person communication. The need for this facility was made abundantly clear in the early days of commissioning the system, when cards with written messages such as "I can't hear you, can you see me?" were an indispensable development tool.

5. Application Support

X provides a library of routines (Xt - the X toolkit) for constructing self-contained components of user interfaces, called "Widgets". Widgets interact with application code in a well-defined way and implement common user interface functions such as labels, buttons and scrollbars.

Using elements both from Xt, and Xv, the video library, Video Widgets have been devised which encapsulate most of the Pandora-specific work needed by a video application. As well as providing a "look and feel" which is coherent between video applications, the Video Widget manages connection set-up, creation of video windows, allows the control of characteristics such as frame rate, resolution, etc, and can check for obscured video windows - in which case the stream can be shut down, and the resources re-allocated.

6. Applications

There are three modes of use of the Pandora system. We refer to these as "Stored", "Soft Real-Time", and "Hard Real-Time", each of which has had a demonstration application written.

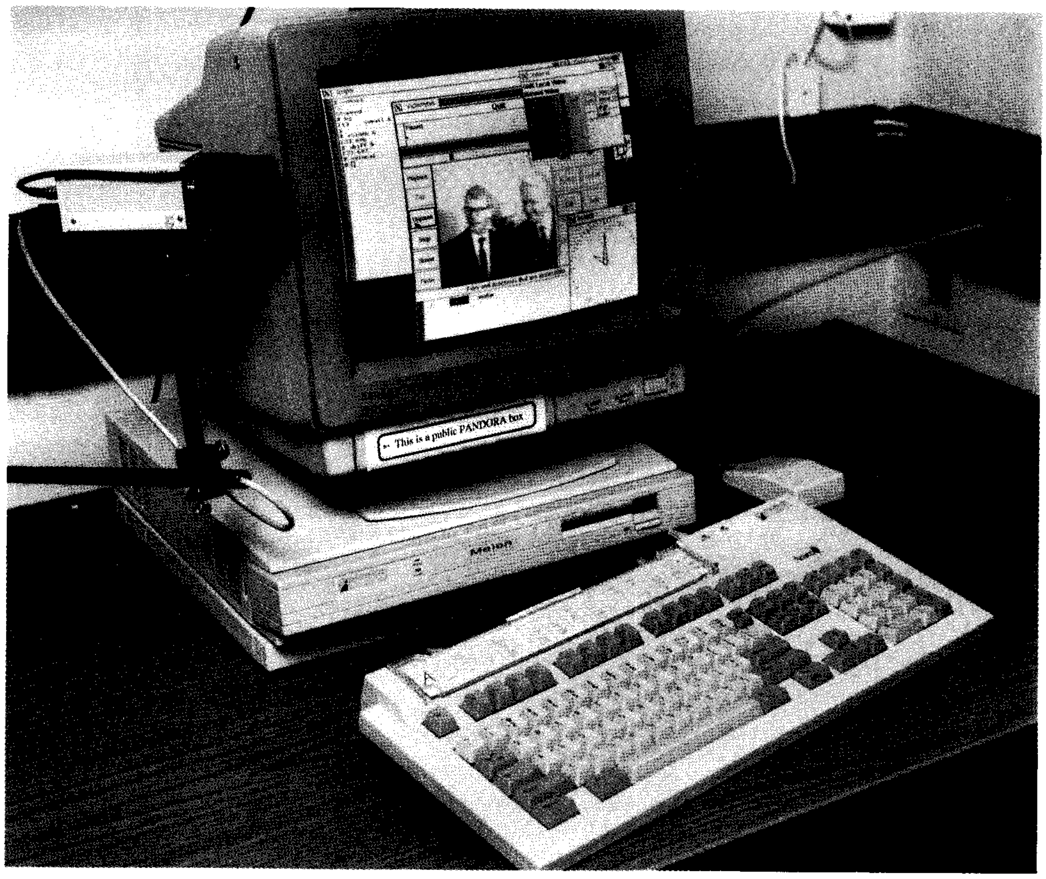

Figure 3. *XvMail application replaying a recorded news item*

"Stored" refers to the recording and playback of digital video to and from a file repository. The demonstration application is a video mail system, "XvMail", which has facilities to record, edit, send to a recipient, replay and delete video clips. The facility is also used to maintain a gallery of short clips of people introducing themselves - useful both as a way of keeping a record of visitors to the laboratory, and of introducing ourselves to visitors. Another facility is the automatic recording of recent news and weather broadcasts from the Media Server (described below). Figure 3 shows a Pandora Workstation running an XvMail application and replaying a recorded news item.

"Soft Real-Time" describes the unidirectional delivery of real-time media to a user. The demonstration application here is called "XvMedia". One Pandora's Box is configured as a "Media Server" with attached TV receiver, radio and CD player. A standard unit on a serial line from a networked computer provides for control and selection. The application provides an interface both to the video and audio, and to the control unit.

"Hard Real-Time" is the most demanding type of application, requiring bidirectional real-time streams to be maintained between a number of users. This is best demonstrated by the "XvConfer" application which provides a video conferencing facility for up to four participants. The application is designed to encourage unstructured, spontaneous use. Two users engaged in a video conversation can invite others to join in at any stage, without prior arrangement; similarly, participants can leave at will.

7. Performance and Evaluation

We display 64x64, 128x128, or 256x256 resolution images and use the DPCM compression to reduce to 4-bits/pixel for network transport. We never transmit greater than 128x128 images - the 256x256 display size is obtained by interpolation at the display end - so the maximum network bandwidth consumed by a stream of video is thus of the order of 800 kbit/sec, or 1Mbit/sec including audio. The maximum throughput of a CFR station chip is of the order of 3-4Mbit/sec or 3 to 4 streams of video and audio. This corresponds neatly with the available space on the display screen, and the saturation of other resources in the system - notably the Mixer CPU. We observe that the machine is remarkably balanced as regards it's performance. The pictures - though far from broadcast TV quality - are usable, and appropriate to the telephone-quality audio.

An upgraded version of the hardware has been developed, and we hope to expand our user community shortly. The system is regularly used for co-operative working and social contact between users located at the different sites, and for circulating "video memos" within ORL. Little in the way of a database of video material has been generated yet, however, mainly because of the lack of suitable authoring tools, and limited video repository capacity.

We are in a position routinely to use, and to make up our minds about, multimedia. Developments in workstation architecture, however, are beginning to make our scheme look a little cumbersome. In particular, very fast I/O busses such as TURBOchannel [DEC91] have performance sufficient to bring real-time data into the heart of a workstation for direct manipulation by application software. We believe this "hands on" approach (manipulate the pixels and sound samples directly as they fly past the processor) will supplant our rather inflexible "hands off" approach (manipulate the media only through rigid interfaces) in the near future, as busses and processors increase in speed.

8. Acknowledgements

The work described is the end result of the efforts of a large number of people. Andy Hopper and Ian Wilson helped define the project goals and system architecture. Oliver Mason, Dave Clarke, Roy Want and Andy Jackson worked on the hardware; Alan Jones, Stuart Wray, Tim Glauert and Mark Chopping wrote the intermediate-level software; Tim Glauert, Geoff Wiginton and Damian Gilmurray worked on the applications. Ian Wilson implemented the video file repository.

9. References

[Clarke89] D.J Clarke, Spec for Compressor/Expander System for Pandora, Internal Report, Olivetti Research Limited, July 1989.

[DEC91] Digital Equipment Corporation, TURBOchannel Developer's Kit, April 1991.

[Hopper88] A. Hopper and R.M. Needham, The Cambridge Fast Ring Networking System, IEEE Trans. Comp., Vol. 37, No. 10, Oct 1988.

[Hopper90] A. Hopper, Pandora - An Experimental System for Multimedia Applications, ACM Operating Systems Review, Vol. 24, No. 2, April 1990.

[Inmos89] The Transputer Data Book, Inmos Limited, 1989. [Jones90] A.H. Jones, Pandora Low-Level Software, Internal Report, Olivetti Research Limited, April 1990.

[McAuley90] D.R. McAuley, Protocol Design For High Speed Networks, Univ. Cambridge Tech. Report No. 186, Jan 1990.

[Scheifler86] R. Scheifler and J. Gettys, The X Window System, ACM Trans. Graphics, Vol. 5, No. 2, pp. 79-109, April 1986.

[Vorstermans88] Layered ATM Systems and Architectural Concepts for Subscribers' Premises Networks, J.P. Vorstermans and A. De Vleeschouwer, IEEE Jour. Sel. Areas in Comm., Vol. 8, No. 9, Dec 1988.

[Want88] R. Want, The Reliable Management of Voice in a Distributed System, Tech. Report No. 141, University of Cambridge Computer Computer Laboratory, 1988.

[Wilkes79] M.V. Wilkes and D.J. Wheeler, The Cambridge Digital Communication Ring, Proc. Local-Area Communications Network Symposium, May 1979.

[Wilkes84] A.J. Wilkes, D.W. Singer, J.J. Gibbons, T.R. King, P. Robinson and N.E. Wiseman, The Rainbow Workstation, The Computer Journal 27 (1984), 112-120.

[Wilson89] Experiments with Digital Video for Workstations, I.D. Wilson, D.R. Milway and A. Hopper, Tech. Report 89/2, Olivetti Research Limited, Mar. 1989.

[Wray89] S. Wray, The Interface to Pandora's Box, Tech. Report 89/4, Olivetti Research Limited, 1989.

Pandora: An Experiment in Distributed Multimedia

Tony King

ORL 92-5

To be published in the Proceedings of Eurographics '92, September 1992