The Design and Implementation of a RAID-3 Multimedia File Server

Alan J. Chaney, Ian D. Wilson and Andrew Hopper*
Olivetti Research Laboratory
24a, Trumpington Street
Cambridge CB2 1QA
United Kingdom

Tel: +44-1223-343000
Fax: +44-1223-313542

Email: {iwilson,achaney,ahopper}@cam-orl.co.uk

Abstract

The Olivetti Research Laboratory has developed an experimental system based on intelligent peripherals connected directly to an ATM network. As well as multimedia modules (e.g. audio and video) the system also includes a directly connected RAID-3 storage server called the "Disc Brick". This paper describes the architecture of the Disc Brick, and discusses some of the hardware and software issues raised by its design. It also presents measurements taken from a Disc Brick in operation, and discusses how the observations relate to the original design objectives. Finally, the paper attempts to evaluate the Disc Brick as part of ORL's family of directly connected peripherals.

Introduction

An experimental system has been constructed at the Olivetti Research Laboratory which aims to provide a rich multimedia environment for a variety of users [1]. It utilises ATM communications technology as the basic interconnect both for computers and multimedia peripheral modules. Each module is made up of a number of standard component parts: the hardware consists of an ARM CPU, up to 32 Mbytes of memory, and a 100Mbit/s ATM TAXI interface; the low-level software consists of a microkernel called ATMos [2] which provides a mechanism for scheduling processes and controlling low level hardware. In addition there is provision for interoperation with other systems by using protocols such as TCP/IP and XTP, or distributed platforms based on CORBA.

The ATMos framework is used to implement both switches (4x4 and 8x8) and end-point modules (or *direct peripherals*). The modules include ATM video, ATM

---

* University of Cambridge and Olivetti Research Laboratory

audio, ATM LCD tile, ATM TV, ATM frame store, ATM processor farm and the Disc Brick, as well as workstations and PCs (see Figure 1).

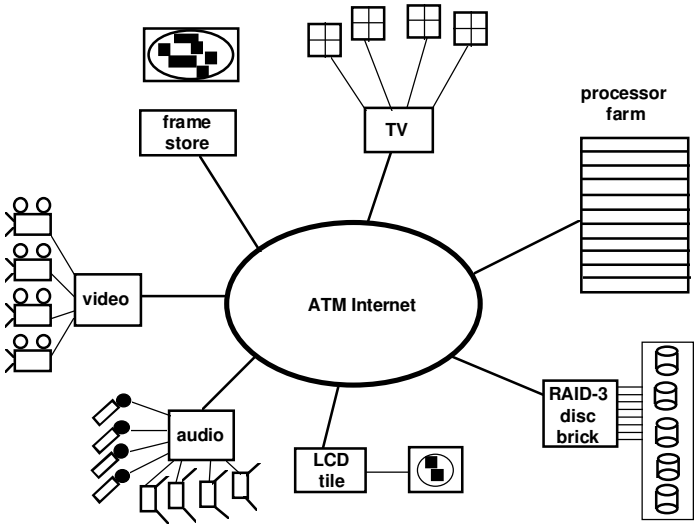

Figure 1

This approach has similarities with the Desk Area Network [3] but also differs from it in important ways. Rather than exploding the workstation, ORL's approach is to treat each direct peripheral module as a first class ATM object; furthermore, most of the traffic from a module will typically flow to points elsewhere in the system and not to a nearby PC or workstation. However, as with the DAN, the approach uses a network as the peripheral interconnect, and hence is scalable.

The system has been deployed in the laboratory and has been in use for about 18 months. Some two hundred modules and switches are available for experimentation. A typical office has four cameras, four speakers, four microphones, a display tile and a workstation as its multimedia infrastructure. In addition, there are five Disc Bricks available on the network for use as multimedia fileservers. Applications in use include a video-mail system, which takes advantage of the Disc Brick's performance by recording several video and audio channels simultaneously. Other applications include a multi-way video-phone which uses four video streams and an audio stream between the corresponding parties.

The rest of this paper concentrates on one particular directly connected ATM peripheral: the RAID-3 Disc Brick.

The Disc Brick Filing System

The Olivetti Research Laboratory has been working in the multimedia area for many years. One of its earlier systems, Pandora [4], relied on a fileserver for the storage of audio and video objects. It is the filing system which was originally designed for the

Pandora Video Repository which now runs on the Disc Brick, since the overall requirements for both systems are virtually identical.

The main aims of the filing system design were:

- simple, hierarchical directory structure;
- efficient use of intelligent disc drives;
- efficient handling of both large and small filing system objects; and
- optimisation for sequential, as opposed to random, access.

The use of a conventional filing system for multimedia data differs from the approach taken by other people. For example, Berkeley's Filing System for Continuous Media [5] makes no attempt to lay the disc out, and relies on pre-allocating contiguous areas for data storage. Lancaster's Continuous Media Storage Server [6] is more sophisticated, in that it keeps a separate directory disc to hold meta-data, but still uses fixed contiguous areas on its data discs. The approach described in this paper has the advantage that all data types can be stored using the same filing system, without sacrificing the performance benefits of the simpler architectures.

Disc Layout Issues

There are two important factors when it comes to making efficient use of intelligent disc drives. Firstly, there is no point in attempting to optimise accesses based on a drive's physical geometry. Indeed, it is often not possible to find out the real drive geometry: track and sector sparing, automatic sector reassignment etc. all conspire to make the drive's physical characteristics irrelevant. Secondly, the actual disc layout (taking into account sector interleave and track skew) is completely hidden from the user, and optimised for large sequential accesses. Most drives nowadays have large data buffers and perform read-ahead and write-behind. This, combined with "read on arrival" which reduces the effect of rotational latency, means that the best way to get optimum performance from them is to *stream* large amounts of sequential data in very much the same way as handling a tape. The repercussions on the filing system design are:

- to lay the disc out so that objects are in large, sequential chunks;
- to allocate I/O buffers so that transactions are as large as possible; and
- to minimise the amount of disc seeking.

The algorithm used is straightforward: assume all new files are going to be large, and reserve disc space accordingly. In other words, when allocating space for a file, reserve a large contiguous chunk at the beginning, and then use up the reservation as necessary. As a file grows extra contiguous chunks are reserved as required, and when the file is closed the unused space is freed. This scheme has a tendency to fragment the space available, but this is not a problem since the spare blocks are mopped up as a side-effect of the filing system's housekeeping operations.

In order to minimise the amount of disc seeking, there is an attempt to store all the blocks which comprise an object as close together on the disc as possible. For this to work in practice, when a new object is created, it must be positioned on the disc in an area where there is a reasonable amount of contiguous free space. Obviously, there is a trade-off between having to seek a *small* amount while handling an object, and having to seek a *large* amount in order to get to it in the first place. To find an appropriate place for a new object, the filing system searches outward from a known disc position looking for the area with the largest amount of contiguous free space, but weighting the results of the search using the inverse square of the seek distance. In this manner, disc fragmentation is kept low, as are seek distances between related filing system objects. Also, head scheduling (either "elevator" or "least seek") is performed by the disc device driver, reducing the effects of seeking still further.

File Structure

In order to handle large and small objects in an efficient manner, a modified version of the UNIX *inode* structure is used. Each object is represented by a *header* block which contains its length, protection mode, last modification date and so on. Only a small portion of the header block is used for this information, and the remainder is used to hold pointers to the object contents. As with the inode structure, there are data blocks which are directly accessible, and those accessible via one, two or three levels of indirection. To all intents and purposes, this allows objects to grow arbitrarily, with a maximum of three levels of indirection before reaching the object contents.

Optimisation For Sequential Access

Many studies have been made in the past about the nature of general purpose filing systems and how they are used in practice. Two results are of particular significance:

- the majority of accesses are read operations; and
- the majority of accesses are sequential.

It was an essential part of the filing system design that it should be optimised for this type of use, even though this can result in a performance degradation when random accesses are performed. To this end, the filing system performs *read-ahead* and *write-behind* at the object level, so that disc operations can, hopefully, be overlapped with user data manipulation. The pipelining effect obtained by this technique means that, assuming the object is reasonably contiguous in its disc layout, the required streaming effect can be achieved.

Cache Strategy

One of the more radical decisions taken when the filing system was being designed was to cache header/indirection blocks and directory contents, but *not* to cache object contents. The justification is that the disc performance is worst when the ratio of seeking to data transfer is very high, and best when performing large sequential accesses. The former mode of operation is typical of general filing system

housekeeping (for example, when picking up a set of header blocks during a directory listing), and is something which is relatively easy to optimise using a cache. The latter, however, is typical of accesses to object contents—the layout of which has been arranged to be as contiguous as possible. Using buffer memory for read-ahead and write-behind rather than as a data cache allows the discs to operate in their most efficient manner, with the added bonus that it is not necessary to perform a cache search or invalidation for each data transfer—not an insignificant overhead.

The final vindication of the "everything but data" cache strategy is seen when the filing system is used with multimedia objects. Video files, in particular, are often tens or hundreds of megabytes in length, and are usually accessed sequentially. Such usage is guaranteed to flush any kind of data cache, and maintaining a cache can only add overhead to what is, essentially, a disc streaming operation. Performance figures for the filing system running on the Disc Brick are given later in this paper.

Which RAID Version?

RAID is an acronym for Redundant Array of Inexpensive Disks and was first presented in [7]. The methods of organising the data on a RAID array are of interest. The aggregate capacity of several drives was desired, and the maximum transfer rate possible would be achieved by accessing all drives concurrently. The two RAID configurations considered applicable were:

> *RAID-3*: Byte striping, where words are read/written in parallel to several discs, each byte going to a separate disc; and
>
> *RAID-5*: Sector striping, where each drive has one logical sector.

Previous work [8] has indicated that the optimum performance should be given by a RAID-5 system. However, this is only applicable:

- for wide arrays (say 9 or more drives); and
- for small transfers (one or two sectors).

In our case it was felt that the comparative simplicity of RAID-3, coupled with our need to transfer very large files, made the RAID-3 architecture the most suitable.

Design Criteria for the Disc Brick Hardware

The main design criterion was to balance the available disc drive transfer performance with that provided by the network interface. At the time work started, a typical high performance disc drive was capable of sustaining transfers of 2.7Mbyte/s. Four such drives in parallel would give an overall transfer rate of around 10.8 Mbyte/s (approximately 90Mbit/s). The ATM network interface had a raw transfer rate of 100Mbit/s. There was a good match between these two figures, although it was realised that other factors (such as the overhead of managing the

discs and the network protocol) would make it impossible to achieve this maximum. It was also a design criterion that there should be no performance loss when operating with one failed drive.

Hardware Architecture

The Disc Brick has the following major sub-systems:

- ARM 610 RISC processor with a 32MHz clock and 32 MB of memory;
- 5 x SCSI interface controllers, each with an 8 bit SCSI-2 interface;
- 5 x SCSI disc drives;
- parity generation/decoding logic;
- a DMA controller;
- a 2k x 32 bit bi-directional FIFO;
- an ATM network interface.

A physically compact design was produced utilising a number of components already developed for other ORL direct peripherals. Five 3.5" SCSI disc drives are used for the storage of data in the Disc Brick—typically these are Seagate ST12550N Barracuda drives, each of 2GB capacity.

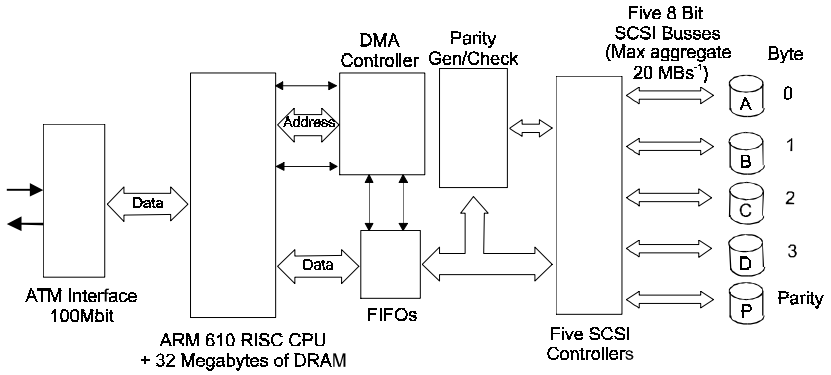

Figure 2

The array is 32 bits wide and requires five separate SCSI ports. Four of the drives hold user data (with the fifth holding parity information) giving an effective capacity of 8GB with 2GB discs. The four data drives are shown as A,B,C and D in Figure 2. Hardware was designed to generate parity information and reconstruct drive data in the event of failure.

Parity is generated during write operations with simple *exclusive OR* logic. If the data from one of the drives A to D is to be regenerated, then the data from the remaining three available drives is passed through the regeneration logic together

with the data from the parity drive. This “reconstructs” the data for the unavailable drive.

A FIFO is used to buffer data to and from the SCSI ports. There are two possible data transfer methods: polled I/O transfers using the block transfer mode of the ARM; and DMA. In order to reduce the interference between the ATM network and disc I/O, a fine grain bus sharing scheme is used where block operations are split into a number of small transfers by the logic of the DMA controller.

This technique relies on the utilisation of the memory bus by the ARM CPU being much less than 100% (due to the presence of its on-chip cache). Observations indicated that typically only 30% to 50% of the available memory bandwidth was being used. The DMA controller was designed to perform a 32 bit word transfer in 125ns. In the Disc Brick design, the ARM has a memory transfer clock (MCLK) of 16MHz. The DMA controller is allowed access to the bus for 1µs (16 MCLK ticks) and then the transfer is suspended for a further 1µs. Allowing for arbitration delays, this gives an effective sustained DMA transfer rate of about 20 Mbyte/s—a good match for the maximum SCSI bus transfer rate. In the Disc Brick each SCSI port has a maximum transfer rate of 4 Mbyte/s, which gives a theoretical aggregate of 16 Mbyte/s for 32 bit data words. The data for the parity drive is never transferred to or from the ARM memory, and thus has no effect on overall performance.

Disc Brick Performance

To evaluate the performance of the Disc Brick, it is necessary to do more than just measure disc and network throughput. Each subsystem is controlled directly by the ARM, and hence carries a CPU and memory bandwidth overhead. Similarly, because of the nature of the devices, there are possible interference effects (interrupt latency, for example) which determine the behaviour of the system as a whole. To be able to understand these interference effects, it is first necessary to understand how each of the subsystems performs in isolation.

Each ATMos system has an idle process which is executed whenever there is nothing else which can be scheduled. The idle process is written in such a way that, by noting the value of a single memory location at the beginning and end of a performance measurement, it is possible to infer the average “idleness” of the processor and memory during that time. All the measured timings that follow are also accompanied by an indication of the system idleness, shown as a percentage.

Raw RAID Array Performance

Figure 3 shows the performance of the “raw” RAID array. Measurements were taken of the time to read/write 50, 100 and 1000 Mbytes of contiguous disc, and the results averaged. Each set of timings were made for *polled* access mode (where the ARM is responsible for the FIFO data transfer) and *DMA* mode (where transfers are handled by an external DMA controller). In order to illustrate that there is no performance penalty in data reconstruction, the timings were repeated for a system in which drive A had been disconnected.

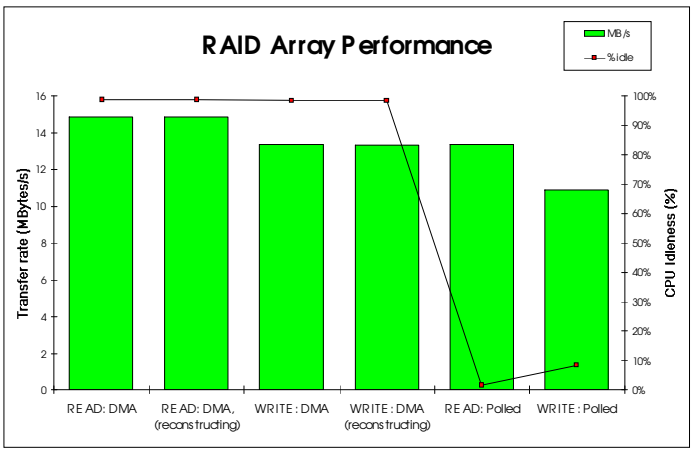

Figure 3

- The *read* transfers are entirely SCSI bus limited, since the theoretical maximum data rate from each drive is only 4Mbyte/s.
- The fact that *write* operations are slower than *read* operations is an artefact of the disc drives themselves—not of the RAID system.
- The effective transfer rates from both *polled* and *DMA* modes are very similar, but the variation in CPU loading is dramatic.

Raw Network Performance

Figure 4 shows the effect on overall system performance of various amounts of ATM traffic, in the absence of disc activity. Measurements were taken for approximately 10, 20, 30 and 40 Mbit/s of continuous network activity, user process to user process.

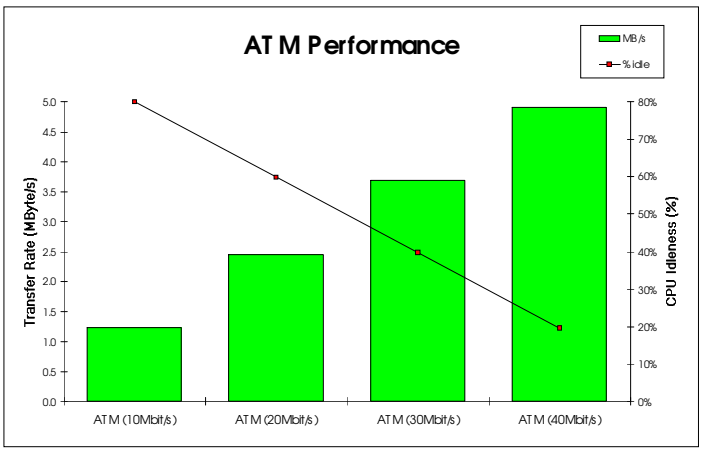

Figure 4

Actual bandwidth requirements at the ATM physical layer are some 10% larger than these figures, primarily because of the 5 byte header overhead for every 48 bytes of ATM cell payload. As can be seen, CPU saturation would be reached significantly before the theoretical 100Mbit/s limit. This is due to the fact that the prototype ATM interface in the Disc Brick generates an interrupt for every cell received or transmitted, and that the adaptation layer processing is performed exclusively in software.

Filing System Performance

To evaluate the performance of the filing system, measurements were taken of the time to read/write files of 5, 10, 50 and 100 Mbytes, and the results averaged. Because creating a file carries with it a significant block allocation overhead, measurements were also taken of the time to *re-write* (i.e. write in place) the same files. Timings were also taken of the filing system performance when an extra data copy is involved, for example by using the ANSI C functions *fread* and *fwrite* through a "buffered" interface. The results are shown in Figure 5.

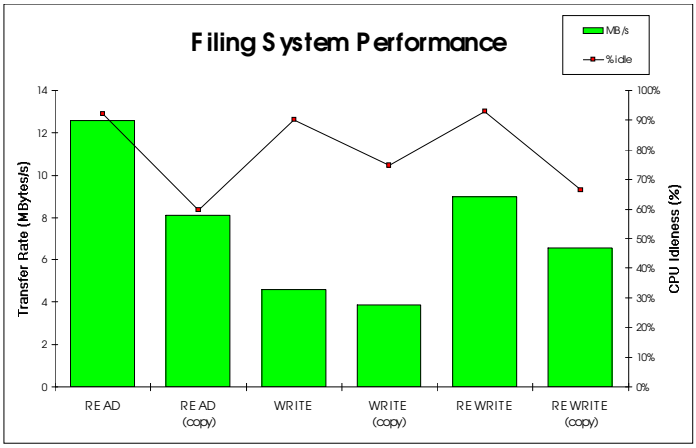

Figure 5

- The CPU overhead of using the filing system rather than the raw disc is not large—especially for *read* operations.
- Comparison of the write and re-write timings gives some idea of the overhead involved in block allocation. The filing system always performs block allocation synchronously to preserve disc integrity, and this has an obvious knock-on effect.
- The overhead of the extra data copy in each of the operations is highly significant. Not only is the CPU less idle, but the overall filing system performance is much lower.

Combined Network And Filing System Performance

The measurements in Figure 6 are identical to the *read* timings described above, but with the added overhead of 10, 20, 30 and 40 Mbits of network traffic occurring in parallel. These values reflect the conditions which would be found in a real multimedia file server, and indicate the extent to which disc and network operations interfere with each other.

As can be seen, the filing system performance degrades gradually as the network load is increased. This is not unexpected, but it is worth observing the difference between the raw measurements and those which involve an extra data copy: here the performance has become significantly worse, indicating that both the network traffic *and* the data copy are interfering with disc activity.

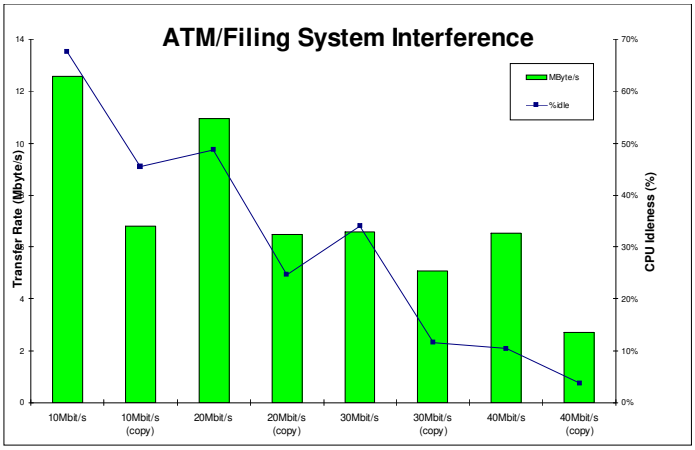

Figure 6

Other Applications

The Disc Brick is currently being exploited commercially in a collaborative agreement between *Conner Peripherals Inc.* and *ATM Ltd.* A range of RAID based ATM direct peripherals is envisaged. It is also part of a "Video on Demand" field trial in the Cambridge area [9]. Connectivity is being provided by *Cambridge Cable*, TV set-top decoder boxes are being provided by *Online Media*, programme material is being provided by *Anglia Television*, and networking infrastructure is being provided by *ATM Ltd.* Disc Bricks are also being used in a collaborative project with the University of Lancaster [10] which is investigating the issues of storage server scalability.

Further Developments

Since the conception of the Disc Brick, the media transfer rate from 3.5" disc drives has increased due to new coding techniques and greater rotational speed. Newer, faster, higher capacity drives are regularly becoming available, and it is essential that the design of the Disc Brick hardware and software should keep pace with these developments. Similarly, network technology is changing sufficiently rapidly that it may soon be necessary to re-evaluate the current 4+1 RAID configuration.

It will be possible both to reduce the cost and improve performance of future systems. Cost reduction can be achieved by integration: the ARM CPU has a very small core which can be integrated together with other components on a single chip. Thus ATM modules, including the Disc Brick, can be much reduced in cost. At the same time a StrongARM is under development which is based on the DEC Alpha CMOS process. This will make available a component with a clock speed of about 200MHz and will provide a performance upgrade path. Similarly, chipsets are becoming available to reduce the overhead of handling the ATM network—the interface to which is the main bottleneck in the current system.

Conclusion

This paper set out to show that a combined network and storage object could be produced which provided a good balance between the performance of the storage and the network. The prototype system was built to demonstrate this—the results show that the disc sub-system works well, and the ARM processor makes an excellent RAID storage controller. However, there is a performance bottleneck in the ATM network interface. New technology will shortly be available to improve the ATM performance.

The Disc Brick fits well into the ORL family of distributed ATM peripherals. With 8GB of capacity, it provides storage for a reasonable amount of multimedia data (e.g. approximately 12 hours of MPEG-1 encoded video). The network interface can support about 40-50Mbit/s of continuous traffic concurrently with high performance disc accesses.

The redundancy provided by the RAID-3 approach has proved important. The usage of multimedia data in the office environment is such that most objects are large, but have a short life. It is expensive and very time consuming to implement a comprehensive backup strategy for such data—the added reliability and availability of the RAID system is helpful.

In conclusion, the ORL Disc Brick provides an efficient and cost effective storage solution in an ATM networked multimedia environment.

Acknowledgments

The work described is the end result of the efforts of a large number of people. In particular, Philip Elwell provided invaluable assistance with the design of both the hardware and software of the Disc Brick. Caroline Bardelay wrote a test suite for the Disc Brick; Ian Crowe, Ernie Wisner and Sharon Grant helped to produce the prototype systems.

References

1. Wray, S.C., Glauert, T. H. and Hopper, A., *Networked Multimedia: The Medusa Environment*, IEEE Multimedia, Winter 1994.

2. French, L. J., Wilson, I. D. and Girling, C. G., *The ATMos Reference Manual*, Olivetti Research Laboratory, 1993-1995.

3. MacAuley, D. R., *Operating System Support for the Desk Area Network,* Proceedings of the 4th International Workshop on Networking and Operating Systems Support for Digital Audio and Video, 1993.

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

5. Anderson, D. P., Osawa, Y. and Gorindan, R., *A File System for Continuous Media,* ACM Transactions on Computer Systems 10(4), November 1992.

6. Lougher, P., and Shepherd, D., *The Design of a Storage Server for Continuous Media,* The Computer Journal (special issue on multimedia), February 1993.

7. Patterson, D. A., Gibson, G., Katz, R. H., *A Case for Redundant Arrays of Inexpensive Discs (RAID),* Proceedings of ACM SIGMOD, June 1988.

8. Lee, E. K. and Katz, R. H., *Performance Consequences of Parity Placement in Disc Arrays,* Proceeedings of ASPLOS, April 1991.

9. Online Media Ltd., *The Cambridge Interactive Television Trial,* Press Release, September 1994; Daily Telegraph, 11th October 1994.

10. Lougher, P., Pegler, D., and Shepherd, D. , *Scalable Storage Servers for Digital Audio and Video,* Proceedings of IEE International Conference on Storage and Recording Systems, April 1994.