The Desk Area Network

D.R.McAuley*and I.M.Leslie†

September 11, 1996

Abstract

The paper describes experience with a novel architecture, the "Desk Area Network" for use within an computing end-system; the architecture extended several concepts used in modern high speed networks into computer system design, with the goal of building a platform able to supply "Quality of Service" in support of multimedia applications.

1 Introduction

The paper presents the work on the "Desk Area Network" (DAN), a computer system architecture to address the problems raised in supplying high bandwdith channels between the components of a modern desktop computer when there are constraints on the required bandwidth and timeliness of the individual channels. This problem arises in the context of supplying end-to-end "Quality of Service" (QoS) to support networked realtime multimedia applications.

The first section presents the QoS problem in the networking context and then focuses in on the problem as faced by end-systems attached to an ATM (Asychronous Transfer Mode) network[1]. The solution described is based on the use of an ATM *switch fabric* for processor, memory and device interconnect inside computer systems in place of more traditional bus architectures.

This work had a key role in shaping the architecture of the Nemesis operating system [?].

2 Background

A goal of one thread of recent and ongoing research in network, computer architectures and operating systems is to engineer systems which can support "Quality of Service"; that is the desire to provide systems where access to shared resources is provided in terms of lower bounds on the amount of resource and upper bounds on timeliness of access.

As an example, consider a simple video monitoring application in which video is digitised and compressed by a device attached to the network, shipped across the network into a computer, where it is both displayed in real time and processed to perform motion detection, then, if motion is detected, recorded to hard disc. Clearly the demand on the disc is variable, and indeed if the compression is based on a variable bitrate algorithm (e.g. to achieve constant quality), the demands placed on the network, computer bus, and processor are likewise variable.

There are two key questions to be asked at this point:

1. Is the system to be built solely for this purpose or do we wish to use more general purpose network and computer systems in which there are other tasks competing for resources?

2. Must guarantees be provided *all of the time* or just *most of the time*?

*University of Glasgow

†University of Cambridge

[1] See "Relevant bits of ATM".

Increasingly the answer to the first question is that a generic infrastructure is desired. Sharing and resusing resources is often more cost effective, while it is desirible that the system can adapt to new services, protocols and applications as they arise.

On the second point, if the application requires cast-iron guarantees, then we must perform "peak-rate allocation"; that is reserve an amount of the resource equal to the peak that the application *might* demand. In a shared environment this means admitting applications as long as the sum of the peak demands is less than the amount of resource.

For our example video monitoring application, consider the network element; let us assume some bandwidth properties of the video encoder and define the capacity of some link shared by several such streams of data – say the encoder has a peak rate of 10Mbps, a mean of 1Mbps, and the shared link a capacity of 20Mbps. If we insist on cast-iron guarantees we would only admit two such streams onto the link.

On the other hand if we allow the possibility of "overbooking" the bandwidth, we might admit many more channels and hence increase average utilization at the risk of delay or loss; in networking parlance this is "statistical multiplexing". Traditionally data networks (e.g. X.25, Ethernet, Internet, ...) have engaged in statistical multiplexing without any attempt to provide a quantified (non-zero!) minimum service level for individual channels; indeed the main goal of the design work in Media Access Control protocols like token passing and the CSMA/CD algorithm of Ethernet, is to aim for a fair share of the bandwidth resource to all active participants.

However, the desire to move real-time traffic (e.g. video and audio) through networks will often require some minimum guarantee: consumers will demand that video-on-demand is at least something approaching "VHS quality"; likewise audiophiles listening to concerts over the net; there may be legal issues – digital security video may have minimum requirements to be admissible in court.

So the "Quality of Service" problem is to provide a channel with constraints on the delay, jitter (i.e variation in delay) and loss chacteristics in the face of a variable demand about which we may only have the simplest statistics like peak and mean rates, while at the same time trying to maximise the utilization.

Both the Internet and ATM communities are working hard to provide network QoS; indeed much of the theoretical work on characterising traffic, defining effective scheduling and queuing algorithms in switches, and call acceptance, is applicable to both domains.

The discussion so far has concentrated on network QoS to illustrate the issues; however, from the application's point of view, achieving a suitable QoS in the network is only part of the problem – we must also achieve QoS within the computer itself. That is we must ensure the computer system is ready and able to receive the data, ensures timely dispatch of the application code, then supplies sufficient processor and disc capacity for the application to do it's job; again this must be achieved in the face of simultaneous demands from other tasks.

3 The Desk Area Network

The DAN machine was built to explore the provision of QoS in a computer interconnect. We selected to use a ATM cell based switch fabric as the primary processor, device and memory interconnect – a particular goal was to interconnect continuous media devices (e.g. audio and video devices) attached to different DAN systems over an ATM LAN, in order to provide the underlying channels for distributed real-time multimedia applications.

One element of the work focussed on the design of devices capable of acting as sources or sinks for streams of media data injected into or received from the ATM network directly. Data would be only be by the host processor when the data actually required processing. Furthermore it was desired to ensure that, in as far as it is possible, synchronous real-time control operations to the device should be avoided.

Another element of the work addressed the operating system and protocol issues raised by the desire to enable applications running on the host processor to control the communications between the various devices. The processor is in effect performing third party channel setup and is not normally involved in the actual data flow.

3.1 The DAN Fabric

The experimental DAN system was built using a space division switch fabric designed and built by the Fairisle project [?]. This provided a 16 port switch fabric with each port 8 bit wide and clocked at 20MHz; while this is a fairly modest per port bandwidth, the 2.56Gbps maximum system bandwidth is quite respectable (the fabric layout is shown schematically in figure **??**).

One relevant property of the Fairisle fabric is that is *blocking* – should two or more cells contend for the same output, or an internal link within the fabric, one will succeed in getting through and the others will remain at the relevant input until the next cycle.

All communications between attached devices is based around the exchange of sequences of cells. Some view the chopping up of arbitrary sized messages between devices into fixed sized chunks with some distaste. However, such techniques are already widely used in currently fashionable busses such as SUN SBus and PCI, and for the same reason – to enforce rearbitration of the system interconnect at frequent intervals so that urgent small messages can preempt long transfers. In current workstations and PCs this facility is widely used to ensure that a processor cache line fetch or flush can preempt long disc and network DMA transfers.

Given the desire to choose some small sized transfer into which to dice messages we selected to use the now standardised ATM cell format (48 bytes of payload and 4 bytes of header[2]). As our view was that the primary network to which the DAN system would attach was an ATM LAN this choice made the network interface card extremely simple.

A key issue in ensuring the provision of QoS within the DAN fabric is the ability to control the rate at which devices inject cells into the fabric on different channels (and hence to different output ports); the devices described below can all be rate limited by controlling software running one of the processor nodes of the DAN.

While we chose to use our particular fabric, the work on the devices discussed below is in fact independent of the exact realisation of the ATM fabric – for example any TDM system whether bus, dual bus, folded bus or ring could just as easily be used if a rate controlled access is implemented in the arbitration.

3.2 The DAN Devices

In all four devices were produced for the DAN machine:

- video capture (the so called "ATM Camera")
- video display (the "DAN framestore")
- audio input / output and DSP
- processor and cache node

while three functional units were implemented by modification of the Fairisle Port Controller:

- main memory node
- ATM interface
- Ethernet interface

[2]For the pedantic who know the standard says 53 bytes per cell, we ignore the HEC field inside the DAN as it performs no useful function here.

3.2.1 The Fairisle Port Controller

The use of Fairisle ATM switch fabric as the DAN fabric has already been mentioned; the Fairisle Port Controller (FPC) was the second element of the Fairisle switch which we were able to reuse for the DAN machine by modication of the embedded software.

In the Fairisle switch, for each bidirectional link[3] attached to the switch there is an FPC card. The card includes the transmission system realising the ATM link, and buffer memory arranged as a number of queues. A FIFO queue at the output suffices, as it's role is simply to deal with the mismatch in clock frequencies between the output of the switch fabric and the transmission line. However, the blocking properties of the Fairisle fabric dictate that when the fabric is heavily loaded queues will build up at the input; as part of the Fairisle work was to investigate various queuing strategies, a RISC processor was used to manage the queues – in the final FPC v.3 card, an ARM600 processor.

Given this functionality it then becomes clear that an unmodifed FPC can be used as the "network interface" for the DAN machine, it was simply required to do the same job as in a switch – transfer cells between the LAN link and the fabric.

Finally, there was an IO connector enabling the card to be fitted with Ethernet and with a different software configuration the ATM interface could also be used as an Ethernet interface.

3.2.2 ATM Camera

The final version[4] of camera has the ability to generate digital video streams from up to 3 sources simultaneously (if they are genlocked) in a number of formats (24 bit RGB, 15 bit RGB, 8 bit RGB, 8 bit monochrome, 16 bit YUV) as sequences of 8x8 pixel tiles packed within an AAL5 frame[5].

Optionally JPEG compression is applied to the tiles inside the AAL5 frame. To provide resilience to cell (and hence packet) loss, we chose to compress each AAL5 frame independently. This ensures that if an AAL5 unit carrying a set of tiles is lost, only the tiles in that AAL5 unit[6] are affected. This technique is conformant with standard JPEG coding; any JPEG device should be able to decompress and display the image. We simply use reset markers to delineate the frames, trading a small increase in bandwidth against greater resilience to cell loss. Raw and compressed video can be produced simultaneously subject to the 160Mbps limit associated with the ATM fabric to which the Camera is attached.

A microcontroller on the board is used to select source, resolution, format, compression option and destination VCI per video frame. This selection is based on a cyclic schedule loaded by the camera manager in response to requests from applications. The main processor need only interact with the device whenever the schedule needs to be changed.

3.2.3 Frame store

Early work with the DAN used off the shelf components for the framestore; in particular a DS5000/25 workstation attached to the switch fabric. This allowed testing of the ATM camera and experiments with various versions of the tiled video format.

Further, the goal of investigating the required functionality of a DAN frame store was performed using this emulated set up. At one extreme of functionality, the frame store ran a complete X-server with video extensions and a process which provided a software emulation of a Pandora box[**?**]; at the other, the frame store contained only interrupt routines which rendered 8x8 pixel tiles.

Later, a frame store was purpose built for the DAN. The high capacity path into the frame store allowed multiple streams to be displayed simultaneously. Control and data paths are conceptually

[3]We use 100Mbps 4B5B coded links – in fact the FDDI physical layer.

[4]This refers to the final *research* version. Nemesys Research Limited has produced a production version with considerably more functionality.

[5]For a introduction to AAL5 see "Relevant bits of ATM".

[6]The number of tiles in an AAL5 unit is configurable.

separate. Control operations allow a window manager to create, destroy, move and resize windows and for clients to be connected to windows. For data transfer the framestore provides only a function to render a pixel tile at an offset within a particular window. A nearly fully functional X-server was developed to use the frame store over the DAN.

The frame store incorporates novel protection features which prevents incoming streams from writing to pixels belonging to another window. This is now in the process of being patented.

3.2.4 Audio Input/Output

A general purpose DSP processing and audio CODEC node was built using the Analog Devices' ADSP-2111 DSP chip and associated "DAT quality" audio CODECs. The memory system for this processor was constructed to optimize operations on stream data which was destined to be received from the ATM switch fabric and, after processing, retransmitted to another DAN node.

This board (see photograph **??**) illustrates that the interface to the ATM switch fabric need not be particularly complex; indeed in terms of gate count, the DAN interface is considerably simpler than many popular busses.

3.2.5 DAN processor node

A processing node was developed which attaches directly to the switch fabric through which it communicates with its main DRAM based memory. The node is composed of a general purpose RISC processor, the ARM600, with first level cache on chip, and a large external second level cache.

This manner of attachment allows the direct insertion of high bandwidth media streams into the processor cache for applications requiring access to the media data. Results from this work are published in [**?**].

3.3 Protocols

The DAN mechanism of device attachment differs from approaches based on "Networked Devices" (e.g. [**?**]) in that the components are attached by means of a reliable and secure communication channel to a processing node that is responsible for their management. In the same manner as a bus based system, this allows the control protocol to be comparatively simple – it is not necessary to implement protocol modules to deal with communications errors and encryption modules for security. Devices present a heterogeneous set of control interfaces as appropriate to their function – the control of a device can be as simple as a register read / write interface as in the prototype ATM camera, or more involved, such as in the case of the DSP node where control is achieved by downloading appropriate code.

The separation of control and data paths is a key to the DAN architecture. Direct plugging of streams from one device into another, with a processing element being involved only in the set up phase relies on this separation. Indeed it can be argued that many of the interesting features of the DAN are to do with its connection oriented nature rather than the use of ATM *per se*.

Another use of the separation of control and data is the provision of synchronisation information. A node, e.g. a general purpose processor, synchronising two video streams needs only to look at time stamp information derived from the video, not at the actually video streams themselves. For this reason we have defined a common interface and protocol for synchronization events; these form a low rate cell stream (e.g. one cell per video field) to the appropriate manager. These individual synchronization streams are then available to a synchronization service within the DAN for onward propagation or local adaptive control as appropriate.

Unless it is desired to process the media data, the main processor of the system only deals with the synchronization and control messages, while the data transfer operations are performed independently and directly between the attached devices (including the network interface device). This requires that the data stream generated by the device contains all the in-band information required by the receiver (e.g. multiplexing, sequencing and synchronization information) Requiring the stream to pass through the processor to have this information added would defeat one of the main objectives of the DAN approach.

As an example, we can consider the encoding of video; video is transmitted in (AAL5) frames of composed of a number of 8x8 pixel tiles, with each frame containing in-band control information, such as the X/Y offset of the tile within the frame, and the frame number. However, to simplify the in-band protocol, where possible, advantage is taken of circuit setup and renegotiation to communicate parameters which change on a long time-scale; for example, frame size, frame rate, and compression options.

This general approach has been presented in [**?**].

3.4 Software

The first runtime system used within the ARM processor based nodes of the DAN was the Wanda micro-kernel, a locally developed operating system used by a number of projects. This provided a familiar environment in which to implement the device management functions. Indeed both the software emulations for the experimental frame buffers (the X-server with Pandora video extensions [**?**] together with the Pandora box emulator, and the very much simpler video tile blitting code) were implemented over Wanda.

Later, in conjunction with the ESPRIT Pegasus project [**?**] a version of the Nemesis operating system was ported to the port controller processors. Nemesis is concerned with providing QoS guarantees to applications; in some respects it is the operating system analog of the the DAN. Nemesis follows the philosophy that the operating system should not be involved in the data transfer between two applications, or between an application and a device. The operating system should limit itself to the control functions, for example in establishing connections between an application and a device.

The novel features of the DAN provided a good environment for Nemesis and many features in Nemesis can be traced to the requirement that it run on the DAN. In particular the DAN is a multiprocessor without shared memory; this meant that the natural synchronisation paradigm was event counts and sequencers; hence event counts and sequencers became the native synchronisation paradigm for Nemesis, even when run on uniprocessor workstations. Other synchronisation primitives (semaphores, POSIX threads) were implemented over event counts and sequencers.

The IO architecture of the DAN has also influenced the IO architecture of Nemesis, again even when run on uniprocessor workstations. The DAN enforced consideration of how to seperate the control and data path of IO transactions while maintaing efficiency. The same view when applied to a uniprocessor machine has strained some of the underlying hardware but has also indicated how little extra functionality can be required to increase performance significantly. For example the frame buffer driver on the DEC Alpha workstation implementation of Nemesis emulates the DAN framestore. This enables clients of the framestore to write directly to the frame buffer device (using the same pixel tile primitive), bypassing the operating system and window system, *but still maintaining protection* – one application cannot draw on another applicaition's pixels.

The mainstream Nemesis work has also demomstrated the same approach applied to the provision of QoS in disc access and continues with higher level funcitons suich

4 Conclusions

Our DAN based approach to a multimedia workstation can be contrasted with two approaches seen in the industry today: either throw more processor at the problem; or supply dedicated paths for audio and video.

The first is in one sense simple to implement and indeed provides the greatest functionality – move everything via the memory system and get the processor to do the tedious task of routing the data between devices. It also usually suffers badly from artifacts of the IO subsystem due to the simplistic schemes by which the IO capacity is allocated – it is easy to obtain unacceptable performance for video and audio if there are other tasks performing IO even when the system is lightly loaded.

The second approach simplifies the hardware and guarantees the data gets where it is going in time by dedicating resource. However if the data paths restrict the allowable formats and the programability is limited, the cost of the reduced generality is that new applications are restricted.

The DAN work has taken a middle ground and demonstrated two things:

1. it is possible to build an IO system within a computer which is both general purpose and delivers QoS to real-time media streams,

2. with careful consideration of the control and data path seperation, such an IO system can be realised with hardware of similar complexity to traditional busses.

The DAN used ATM as one way to ensure the find grained sharing of bandwidth within the system while simplifing the attachment of the system to an ATM LAN; the latter was viewed as important in our scenarios where a significant proportion of the load on the interconnect was between the network interface and the other devices on the DAN.

Whether one would choose to use ATM style interconnect may be down to a matter of taste; however, the key leasons we have learnt :

1. implement the bandwidth sharing to ensure the requisite real time granularity is achieved for the media streams,

2. implement a scheduling mechanism for the interconnect that provides software controlled sharing of the proportions allocated to the individual channels,

3. ensure the behaviour of the scheduling scheme for some channels is defined and meets their real time requirements even when the applied load is in excess of capacity.

4. separate control and data paths cleanly so that device to device data transfer can be established and controlled by a third party.

The Relevant Bits of ATM

Asychronous Transfer Mode (ATM) is a multiplexing and switching technique. Transmission is divided into as series fixed sized *cells*. Each cell contains a label, usually called a *header*, describing the channel to which it belongs and a body or *payload* which contains the information to be sent.

Communication using ATM is connection oriented. A connection is established, for example, through the use of a signalling protocol (just as in the telephone network). Cells belonging to a particular connection follow the same path through the network and are routed according to their label.

ATM provides both a fine grain sharing of, and demand driven access to, the underlying communication capacity. Some have described ATM as a compromise between circuit switching, with its fixed partitioning of capacity, and packet switching, with its completely flexible partitioning. Indeed ATM is used to carry both fixed capacity circuits and variable size packets.

In the latter case, above the ATM layer, a protocol stack will contain an ATM Adaptation Layer (AAL). One such adaptation layer, the so called AAL5, is used to carry variable sized packets. AAL5 is a protocol for grouping a number of cells which belong to the same channel (by virtue of the information in their labels) into a packet, and for checking that all the cells of that packet are present and in the correct order by use of a packet length field and CRC32.

The CCITT (as was) defined the term ATM alongside STM (Synchronous Transfer Mode – meaning circuit switching) and PTM (Packet Transfer Mode – for packet switching). Since then the term ATM has come to be associated with the particular standards for cell size and header functions, which is a mildly irritating, but widely accepted, use of the generic term for the particular.

However, worse still many now use the term to imply this and all standards standards (some slightly daft) from the ITU-T and ATM Forum for higher layer protocols which the underlying cell transmission and switching systems are expected to support.