Experiences of building ATM switches for the Local Area

Richard Black, Ian Leslie, Derek McAuley
University of Cambridge Computer Laboratory

28th January 1994

Additional Project Team
Mark Hayter, Shaw Chaung, Simon Crosby,
Matthew Doar, Richard Earnshaw, Daniel Gordon,
Eoin Hyden, Ian Pratt

Abstract

The Fairisle project was concerned with ATM in the local area. An earlier paper [**Leslie91**] described the preliminary work and plans for the project. Here we present the experiences we have had with the Fairisle network, describing how implementation has changed over the life of the project, the lessons learned, and some conclusions about the work so far.

1 Introduction

The Fairisle[1] project was a three year effort at the Computer Laboratory begun in October 1989, to design and build an ATM local area network, and to investigate the architecture and management algorithms appropriate to the local area.

The project included the construction of ATM switches, host interfaces, device drivers, and management software. Within the Computer Laboratory, other research projects such as multimedia, operating systems, workstation architecture and distributed systems are now using the bandwidth provided by the Fairisle network, and providing the network with real data.

This paper presents a report of the the final outcome of the construction phase of the project. Section 2 presents an overview of the switch design and other associated hardware components; section 3 describes the associated low-level software and firmware; section 4 describes the switch services; section 5 presents the raw performance figures for a variety of typical uses.

The network is now in use as a platform in support of further ATM switch and network research, and as a testbed for Quality of Service and Call Acceptance Control experiments funded by British Telecom.

2 Components

2.1 The Switching Fabric

The Fairisle switch fabric is composed of 4 by 4 crossbar elements implemented on a single 6400 gate equivalent Xilinx FPGA [**Xilinx92**]. The largest switch so far constructed is a 16 by 16, the fabric built as a two stage delta. More commonly, two stage 8 by 8 switches are used for experimentation.

[1] This project was supported by the UK Science and Engineering Research Council and HP Labs, Bristol.

The fabric has an 8 bit data path and is clocked at 20MHz (nominally) for a raw bandwidth per port of 160Mbit/sec. The fabric is also cell-synchronous; besides the 20MHz clock signal, a further *frame start* signal is distributed to all port controllers, and any pending cells at the inputs are injected a defined number of clock ticks after the frame pulse.

In common with many other designs the fabric elements are self routeing, that is each cell has routeing tags prepended to it which indicate the requested output (at each stage) for that cell; two bits of routeing information and one priority bit are used. The arbitration units in the switch elements implement round robin like service – the winning input in each frame is remembered in the arbitration unit and if contention is experienced for the next cell, the "next" requesting input wins ("next" in the obvious modulo 4 manner). The format of the routeing tag is given in figure 1.

Bit	Name	Use
0	Active	Always set for a cell
1	Priority	Set if cell is a priority cell
2-3	Route	Output requested for this cell

Figure 1: Switching fabric routeing tag per stage

As the fabric or output port controller may reject a cell due to contention or finding a full output buffer, an acknowledgement signal is provided to the input. At a defined point during the cell frame, this signal can be observed and indicates whether or not the cell has succeeded in traversing the fabric and is being accepted by the output – in the case of failure, it is the responsibility of the input port to decide what to do with the cell.

Since the switching fabric used in the Fairisle switch is both internally and output blocking, the switch is required to be input buffered; to minimise the effect of the blocking on throughput, the fabric is run faster than the line rate (160Mbit/sec v. 100Mbit/sec), which also has the side effect of requiring some modest amount of output buffering.

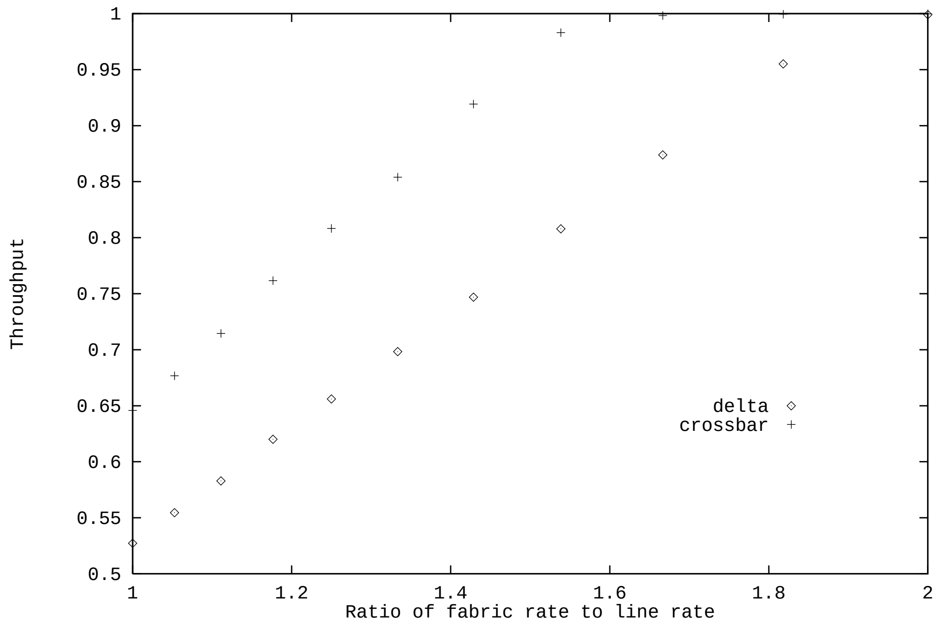

Figure 2: Throughput against fabric, line rate ratio

Many comparisons of input and output buffered switches [**Karol87**] assume a internal switching rate equal to the line rate. Given the relative complexity of input and output buffering this seems rather restrictive for the input buffered case. Indeed our current fabric is used well below it's maximum rate; the 20MHz rate currently used is dictated by the design and implementation of the clock distribution circuitry.

Figure 2 demonstrates the effect on throughput of this speed-up for uniform random traffic; the graphs are for a 16 by 16 crossbar (where only output blocking is experienced) and for the Fairisle two stage delta; in both these examples, the worst case of fifo queueing at the input is used. Hence for the 160:100 ratio present in the Fairisle switch, while the *fabric port* utilisation is the expected 53%, the *line* utilization achievable (i.e. throughput) is approximately 87% (for this oft used hypothetical traffic).

The work of the AN2 [**Anderson93**] designers has led to a reconsideration of the use of the priority bit within the Fairisle fabric. AN2 uses an iterative bipartite matching algorithm to compute a switch schedule each cell time. A key stage is the first iteration in which reserved slots are removed from the scheduling if any data for the relevant circuit has arrived at the input buffer; this provides the mechanism to provide reserved bandwidth for certain channels, while being able to use such capacity for other channels when not required. While within Fairisle we use a self routeing switch fabric, rather than separate routeing and scheduling mechanisms as in AN2, the same ability to provided reserved slots is being implemented using the priority bit. Non-reserved slots, and reserved slots for which no reserved traffic arrives are allocated on the strictly round robin basis described above.

Finally, as part of an ongoing SERC funded research project to apply formal techniques to networking problems, the fabric elements and fabric itself has been formally verified [**Curzon94**] using the HOL system.

2.2 The Port Controller

Each Fairisle Port Controller provides an input port and the corresponding output port for the switching fabric. The port controller also interfaces to the transmission system. A detailed description of the hardware may be found in [**Orange93**][2]. Three versions of the hardware were produced:

Version	Processor	DMA Xilinx	Number	Comments
FPC1	ARM3	3042	5	Initial prototypes
FPC2	ARM3	3064	20	Fairisle build
FPC3	ARM610	3090	60	British Telecom build

FPC1s have been retired; the FPC2s are in use to provide a service network; the FPC3s are used for the majority of the ongoing research work. The versions are due to changes in processor availability, cost savings based on volume, and increased low level functionality.

The port controller consists of three major sections: the queue manager based around an ARM RISC processor; network buffer memory and DMA engine; and transmission system. The transmission system is discussed further below. Figure 3 shows an overview of the port controller, the lower half forming the processor section, and the upper the buffer section.

The input buffer on the card consists of 128k bytes of (35ns) static RAM arranged as 2048 cell buffers of 64 bytes each – room for the cell payload, header, fabric routeing, VP/VC mapping and "next" pointer. This buffer memory resides in the address space space of the processor to enable the processor to both manipulate the control information and to send and receive cells from locally executing tasks (e.g. signalling cells). Cells can be injected into the input buffer from both the transmission line and, via a loop-back fifo, from the output section of the fabric - this enables ports on the same switch to easily communicate with each other.

[2][**Orange93**]is used throughout to indicate that further detailed information is available in the ATM document collection, available by anonymous ftp from ftp.cl.cam.ac.uk as /public/reports/ATM/docs-93-2

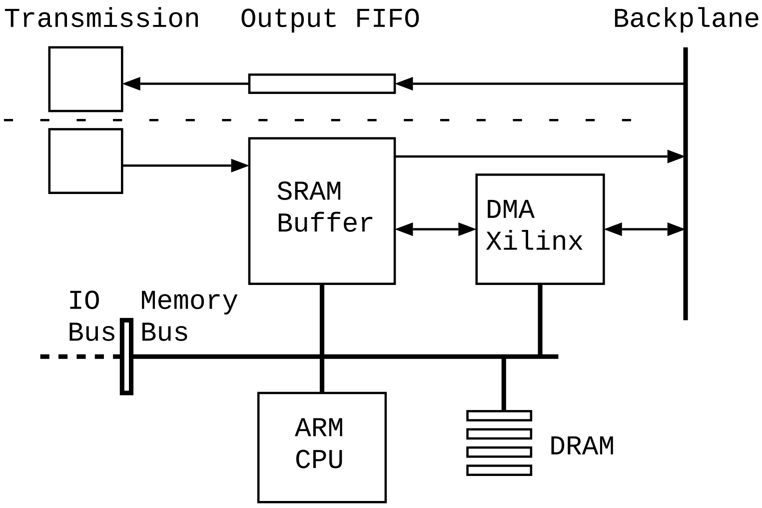

Figure 3: Port controller schematic

The processing section of the port controller has an ARM processor and runs the Wanda kernel [**Dixon92**] to provide an environment in which to implement services such as switch and network management. The provision of the standard IO bus for the ARM chip-set also enables the provision of other services; for example, a port controller with an Ethernet interface has been used as an Ethernet / ATM IP router.

The software responsible for cell queue management is performed at a high interrupt priority (the so called FIQ) and effectively runs asynchronously with respect to the Wanda kernel. The software interacts with the static RAM DMA engine (also implemented using a Xilinx device) to both enqueue cells arriving from the transmission system and dequeue cells for injection into the switch fabric.

Each port controller is a double height extended depth euro-card; a backplane connects port controllers to their associated input and output ports on the switch fabric and provides the clock and frame sync distribution from a master clock. A complete switch of 16 ports, composed of a 16 by 16 fabric, master clock board and Ethernet interface fits into a standard 19" 6U subrack (just).

2.2.1 Transmission System

The transmission system is based on the ATM Forum *de-facto* standard using AMD TAXI components at 100Mbit/sec. The majority of the physical media in use is 50Ω coax – fibre is used only for long runs greater than several hundred metres, as the coax has proved very reliable for short runs. A Xilinx chip performs the framing and HEC calculation, and small fifos are used to decouple the transmission derived clocks from the internal clock domain.

The transmission system was initially implemented as a plug in card on the first two prototype versions of the port controller – we intended to move to SONET which was always going to be available “real soon now” – issue three port controllers include the TAXI transmission systems on the main board.

The transmission system can be configured to interleave data symbols with various numbers of idle symbols on the line. This permits a range of line speeds (512 different values) to be emulated, from 0.4 Mbit/sec up to 100Mbit/sec. This has been provided to enable experiments into the effects on the network of links of various speeds. It has a practical use in that when using transmission converters, such as the TAXI to 34Mbit/sec G.703 used for early Super-JANET experiments, the switch output rate can be matched to the line.

2.3 Host Interfaces

A VME interface developed early in the project was shelved in favour of the Olivetti Research YES-v2 [**Orange93**][3] interfaces when the project acquired DEC Turbo-channel based systems. The switch performance results presented later have been obtained using such interfaces.

2.4 Indirect Developments

The modular nature of the Fairisle switch has lead to related developments which now form part of the ATM environment in which the networking experiments are being performed.

The Desk Area Network (DAN) reuses the port controllers and switch fabric of the Fairisle switch to implement a workstation in which the fabric is used as the main bus of the system. Multimedia devices (video and audio, capture and display) and processor cache/memory systems all using the Fairisle fabric have been built and are described elsewhere [**Hayter91**, **McAuley93**, **Hayter93**].

Of more direct relevance to the network research are the Null Port Controller and Multicast fabric.

2.4.1 Null Port Controller

The Null Port Controller is a fifo queueing port controller for the Fairisle switch. The device is extremely basic; besides the standard transmission daughter board, the port controller is composed of three components: a Xilinx 3042, a 256K by 8 bit “triple ported” VRAM, and a fifo memory.

The main buffering function is performed in the fifo. The Xilinx is again used for all control functions.

The VRAM is used to perform the header remapping and to prepend the fabric routeing tags. Arriving cell data is shifted into one of the serial access memory (SAM) ports of the VRAM at a tap point defined by some bits of the VP/VC. After complete reception of a cell, the payload and parts of the header are written into the main DRAM array at a row address defined by further VP/VC bits. This area of memory has been initialised during call setup with the appropriate new VCI and fabric routeing information[4], so that when the cell and new header are read back into the other SAM, the new cell is ready for injection into the switch fabric. 4K translations can be supported.

Hence the VRAM is used to provide both the header mapping, high speed double buffering and the ability to retransmit a cell. This compares with the initial (and more obvious) designs which use an SRAM translation table and either, a fifo with resettable read pointer, or two fifos. VRAMs are cheaper.

Such a port controller clearly has minimal abilities to provide quality of service to streams (different priorities and retry count on blocking). However, our main motivation for building the NPC was

[3] Thanks are due to Olivetti Research Ltd. for permission to replicate the design.

[4] The hardware pedant will realise that this requires the use of the appropriate bit masked write when copying from SAM to DRAM.

based on the observation that many ATM switches will effectively only be acting as low to high rate multiplexors without overload and our desire was to investigate how simple (cheap) this allowed port controllers to become. In particular our aim was to attach 10 Ethernets to an ATM link into an ATM backbone – the *maximum* cell buffer occupancy at each Ethernet input port is one[5] – anything more complex than fifo in this circumstance is gratuitous.

The Sapphire switch [**Prudence93**] is an experimental ATM switch built by HP Labs based on the Null Port Controllers and the 8 by 8 Fairisle fabric. Port controllers are attached to six of the fabric ports, the seventh is used for connection of a switch management processor and the eighth port is socketed for the addition of an optional transmission port.

2.4.2 Multicast Copy Fabric

The multicast copy fabric was an experimental addition to the Fairisle Switch. This was a hardware device for replicating multicast cells on the way into the switch by presenting them across multiple inputs. The design and performance of the system is described in [**Doar93**].

3 Port Controller Firmware

Xilinx components have been extensively used in the project as their reprogrammability provided the ability to migrate functions from software to hardware in the light of experience. Furthermore as both *de jure* and *de facto* standards have been accepted, it has been possible to modify the firmware to track these changes – this enables us continue to use the Fairisle equipment for switch and network experiments while integrating it with commercially available equipment.

The firmware underwent several redesigns to add functionality; the relevant features of "Xi2", "Xi3" and "Xi5" designs (i.e. those that have seen service) are discussed.

3.1 Initial Design: "Xi2"

Initially the Xilinx chip did little more than to read cells into and out of the SRAM into buffer slots allocated by software and passed to the hardware via two shared 8 entry circular buffers, one each for reception and transmission. On reception the hardware would copy the cell to the first reception buffer and interrupt the processor. The interrupt routine would check the received VCI, perform the VCI translation, add the relevant routeing information, and append to a queue. The service process of these queues, invoked on a successful transmission, would remove entries and pass to the transmission hardware via the circular buffer.

At an early stage, VCI mapping was added to the hardware, as it was observed that the synchronisation cost of the external memory cycles required to read and write the header was having an adverse effect on the processor performance. The VCI mapping and routeing information are stored in the SRAM buffer memory and the mapping function can be invoked on a per cell basis by use of a control bit in the transmit descriptor; transmission without mapping is used for control cells which originate from the local processor.

The first traffic on the network was that generated by various packet based sources (e.g. switches booting and managing themselves!) and it was observed that, even at comparatively low mean utilisations, the occurrence of NACK events from the fabric was sufficiently frequent that the cost of handling such events in software (which often involved reloading the transmit queue) was unacceptable. This cost was decreased by the addition of a 1 bit retry count to the transmit descriptor.

[5] Assuming the sensible cut-through design in which Ethernet frames are segmented into cells and forwarded as soon as the danger of Ethernet collision is past.

After these modifications the Xilinx design was known as “Xi2”. This design is discussed in detail in [**Orange93**].

3.2 Later Changes: “Xi3”

Performance evaluation of the cell forwarding software on the port controller revealed that much of the time was spent manipulating the short circular buffers and the free queue; furthermore the off-processor operations required to perform these were expensive. While it was desirable to continue to have the processor active in implementing the cell forwarding queues (this is one of the research areas the equipment is designed to investigate), the free queue management and the interface between software and hardware were redesigned.

The cell buffers fall into four classes: waiting collection by the reception FIQ, on the free queue, on the transmit queue, and being used by software. The cell buffers in the first three classes are linked together by cell buffer numbers in their link field. These form a single list with pointers in the Xilinx chip marking the boundaries as shown in figure 4.

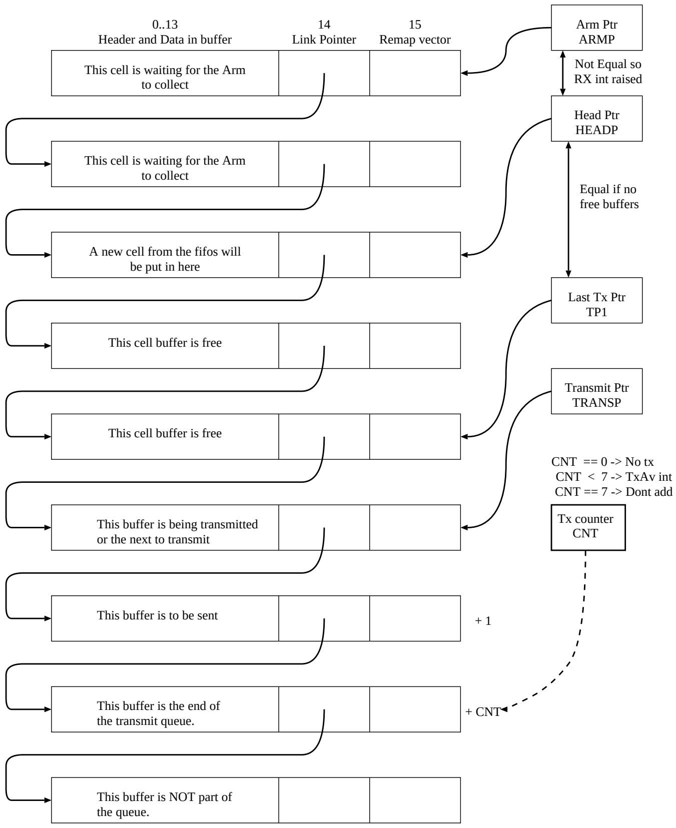

Figure 4: Xi3 queue structure.

The queue must be set up by the processor before starting the Xilinx. The queue is automatically maintained because the processor adds buffers for transmission to the tail. Once these are sent they form part of the free queue before being used for reception. The length of the transmission section is limited to a maximum of eight in a trade off between the desires to pipeline the transmission function and provide software with a reasonably timely indication when congestion is experienced – without this limit and indication the most cunning queueing software invented would degenerate at times of congestion[6] to fifo behaviour. The same requirement for a short feedback loop exists in the design of a host interface (and associated device driver) which aims to implement per channel priority and rate control.

Buffers not in the queue are assumed in use by software, and must be entirely managed by software. This is straightforward while the processor is not a net source or sink of cells (e.g. signalling cells); if it is, both dummy transmissions and receptions are implemented to enable software to correct any imbalance.

The link field in each buffer contains not only the next buffer in the list, but also the number of retries (between zero and seven inclusive). Sensible use of retry counts actually reduces the demand on the processor during contention in the fabric, so that at these critical times, cycles are available for more complex queueing algorithms.

The interrupt interface was also modified so that in a single operation it is possible for the software to read the interrupt condition, the buffer number at the head of the list, and if a cell arrival has occurred cause the pointer to the head of the list to reload as a side effect. Such interfaces carefully constructed to match the commonly required operations of the software considerably speeded the interaction. With this design, known as "Xi3" [**Orange93**] almost the full line rate could be handled (see section 5).

3.3 Instrumentation: Xi5

The most recent iteration of the Port Controller Hardware, done in the context of a British Telecom funded follow-on project, includes a larger Xilinx device (3090). The design for this chip, known as "Xi5" [**Orange93**], is from the point of view of cell management the same as the "Xi3" design; the additional space has been used to implement telemetry support for more detailed measurements and experimental purposes.

The requirements of the theoreticians and modellers were for traces of cell arrivals and departures from switches; for each cell handled it was desired to record the channel identifier of each cell and these two times. This presents some problems, as each 100Mbit/sec link is generating about 10Mbit/sec of trace, and not satisfied with one port's observations, we are required to do it for whole switches.

The solution adopted is based on the observation that when performing such experiments, the important aspect of some streams is simply their temporal behaviour, rather than contents (e.g consider an open loop variable rate coded video stream). Hence the Xilinx component now contains a free running counter (which ticks at the rate of the switch fabric framing signal) and may be configured *per VCI* to stamp the value of this counter into the *data* portion of the cell when a cell arrives and leaves the system. The position of the stamp is determined by some bits in the VCI, so different switches can be configured to place the information at different points in the cell.

3.4 FIQ software

The ARM fast interrupt (FIQ) is reserved for use by the cell forwarding hardware. When handling a FIQ, the ARM processor remaps part of its register set to provide private registers for the FIQ handler – this enables state to be maintained between FIQ invocations without having to save and restore state to memory, greatly reducing the interrupt entry and exit overhead.

The structure of the list data structure shared between the processor and hardware is shown in figure 4. An interrupt is generated when a cell arrives, if there are less than eight buffers on the transmit section of the queue, or if a NACK occurs. On entering the FIQ handler, the interrupt status is obtained and the used to dispatch to the correct handler routine; this process takes a maximum of 7 instructions of which one is an non-cached data read.

[6] I.e. at the worst possible time.

On reception, the incoming VCI is read from the received buffer, and, after range checking, used to index into the VCI table. Having obtained the queue pointer for this VCI from the table, the buffer number is appended to the queue. Worst case for this path is 24 instructions of which one is a non-cached data read.

On transmission, the default configuration serves three queues in strict priority – one of these queues is used for all forwarded cells. In this configuration, the instruction count for transmission of forwarded cells is 16, with an additional 5 instructions for each extra queue that must be checked. More complex strategies with multiple queues and fair service disciplines are also available for specific experiments.

4 Software

4.1 Port Controller

The use of Wanda as the software platform on the switch has already been mentioned. As well as the low level FIQ handler for cell forwarding, each port controller contains three major services:

- a gateway process responsible for implementing the MSNL [**McAuley90**] signalling functions,
- the MSNL "rarpd" service, a process to supply systems desirous of booting, with their own address and that of the bootserver,
- "minetd"; a kernel implemented service which provides echo and `traceroute`-like functions.

In fact, the boot ROM on each port controller contains a complete copy of Wanda together with signalling code, and a boot client. The boot client performs an MSNL "rarp" and establishes a connection across the ATM network to its bootserver. Different versions of both Wanda and the Xilinx configuration can then be booted into the system.

In service use, one port acts as a switch master processor, providing certain per switch services, e.g.:

- Ethernet gateway,
- aggregated console output; the console output from each attached port controller is forwarded to an appropriate X11 program for display,
- local boot service; the master processor can be configured act as a bootserver, and reboot another port from it's own memory – such booting takes a fraction of the time taken to boot from the UNIX based bootserver[7].

Besides being used for forwarding cells, the cell buffer memory and Xilinx are used by Wanda to provide an ATM network interface to user level tasks, enabling these various services to use the network for communication. Performance of this interface is comparatively slow, but is sufficient for these services.

To enable the simple interchange of lines card between 4, 8, and 16 port switches the first order of business on booting after a basic self test, is to establish the "shape" of the switch fabric. Investigative cells are transmitted into the fabric, with the cell headers and contents carefully constructed so that in any configuration at least one of the cells will be received by the transmitting port – the received cell can then be used to establish the size and location of the port on the fabric. The fact that 16 ports may all be doing this simultaneously (e.g. at power on) adds entertainment.

[7]Software currently being implemented will also enable the port being booted to be initialised with the VC entries that were in service at the time of a crash.

4.1.1 Signalling and Management

Port controllers are capable of either performing independent signalling or acting in concert as a switch. Early in the project when there were a small number of port controllers being used by different people, using ports independently was the norm. In this mode the individual port controllers implement signalling even to other ports on the same switch, but can be individually rebooted and configured.

More recently, as complete switches have been available, it has been possible to move to using management code which controls all the ports on the switch as a coherent entity. This is in line with the initial expectations discussed in [**Leslie91**].

The signalling system also allows applications to perform third-party call setup on behalf of dumb ATM entities which cannot signal on their own behalf, for example the ATM Camera. The ability to perform third party setup is critical to the simple design of such systems.

For a set of switches under the same administration in the local area we believe that knowledge of the complete topology at each switch is a reasonable assumption; to achieve this we implemented automatic mechanisms for topology determination based on those used in the Autonet [**Schroeder90**] [**Rodeheffer91**]. However, the current experimental work programme would be hindered by automatic reconfiguration and so the system has not been deployed (or fully tested).

4.2 Unix

Hosts are currently connected to network using a mixture of ORL Yes and Fore interfaces. The hosts have access both a subnet interface for IP over ATM and raw ATM channels. A key feature of this implementation is that only the data paths are implemented within the kernel; all the signalling, socket control, and management is implemented in a generic and portable user space daemon. Details of this work have been previously presented in [**Black94**].

5 Performance

The performance measurements presented in this section were aimed at verifying that our switch design (which included, at some considerable complexity, the ability to easily reprogram and experiment with its major components) provides a realistic platform on which to perform further ATM switch and network experiments.

The first results represent the raw throughput of the port controller for ATM traffic. In the first iterations of the switch, throughput was found to be a problem and the firmware was redesigned to lower the number of non-cached memory reads and the processor upgraded to one with a write buffer to reduce the impact of writes (the ARM cache is write through so this was relevant for all memory writes). Full line rates are now achievable with sufficient processor power left to enable the implementation of more complex queueing algorithms.

The second two experiments demonstrate the performance of the port controller when acting as a gateway from Fairisle to both Ethernet and the Cambridge Fast Ring (CFR). Both are found to be limited by the appropriate network interface, rather than the forwarding process to Fairisle.

5.1 Port controller throughput

Initial throughput measurements for the Fairisle port controller (version 2) were reported in [**Orange93**], from which the FPC2 data is taken. The data rates are presented in terms of the data portion of the cells only – the 100Mbit/sec transmission line rate is equivalent to a maximum data rate of 87.3 Mbit/sec.

Figure 5 shows the performance for various burst sizes. The first plot indicates the sustainable data rate at which the processor spent all its time in the FIQ routine, the second the absolute maximum rate achievable.

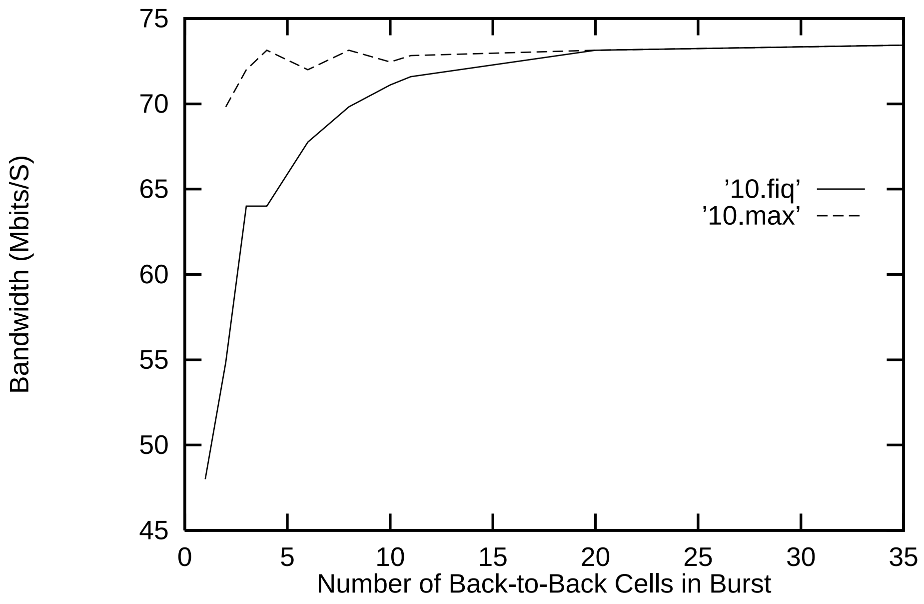

Figure 5: FPC2 throughput performance

As expected, for small bursts the overhead of entering and leaving the interrupt routine reduces the rate at which the CPU first becomes always busy. For larger blocks the interrupt overhead becomes amortised and the full rate is obtained. For small blocks at the maximum rate, the artifacts are caused by beating between the arriving cells, the cell-synchronous switching fabric and the arbitration of accesses to the cell buffer memory.

The measurements show that the early port controllers (using the ARM 3 processor) were capable of a sustained throughput of 73.4 Mbit/sec. One of these port controllers, which was equipped with an alpha-silicon ARM 600 processor (8 entry write buffer), was observed to be limited in throughput at 82.5 Mbit/sec by a firmware restriction in "Xi3", the CPU having cycles to spare.

The issue three port controllers (ARM 610 processor) are capable of forwarding at the full line rate (87.3 Mbit/sec). The information presented in figure 6 demonstrates for various loadings and burst sizes the percentage of time the CPU is idle; the idle time is defined to be that which is available for Wanda threads, once interrupt dispatching and other kernel activities have been taken into consideration.

Depending on the burst size, the CPU starts to be entirely consumed by the FIQ routines in the range 40 to 50 Mbit/sec. Hence, although the port controller is capable of handling the entire line

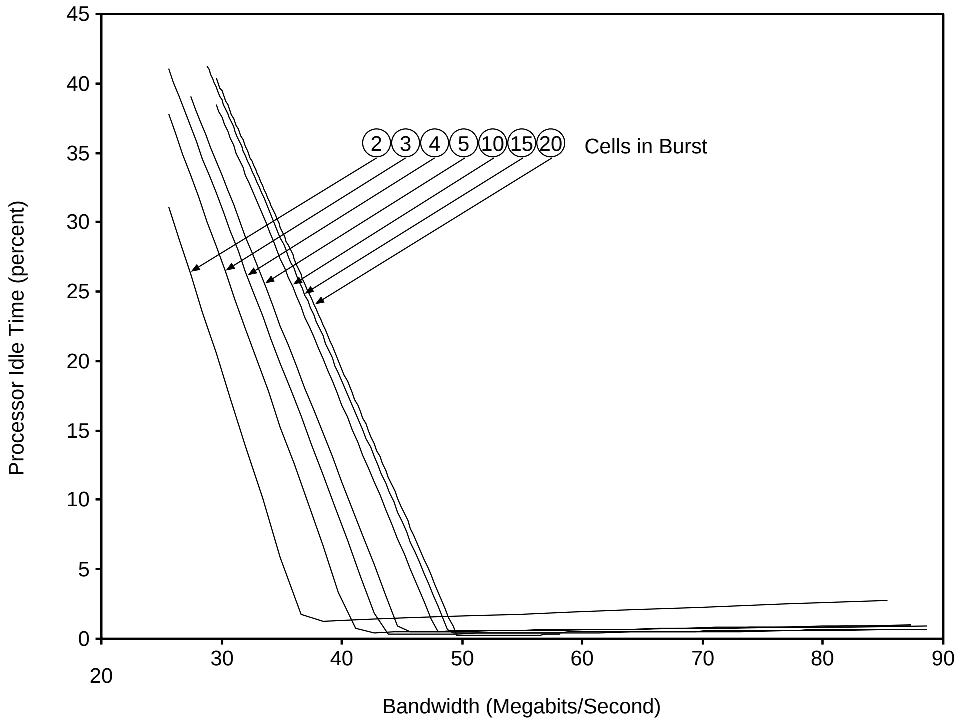

Figure 6: FPC3 processor usage

rate, above the 40 to 50Mbit/sec rate, the interrupt dispatching overhead prevents any time from being made available to Wanda processes.

5.2 Routeing to other networks

Using the standard I/O bus for the ARM chip set it is possible to interface to other networks; in particular we have used Ethernet and the CFR.

5.2.1 Ethernet

Forwarding between the Ethernet and Fairisle is performed by a user level Wanda process dealing in packets. All data movements between the user level buffers and the Ethernet or ATM interface is performed in software. In the case of packets to and from the ATM interface, the segmentation and reassembly functions are also performed by software.

The sustainable throughput achievable is approximately 6.7 Mbit/sec; previous experiments indicate that the limiting factor is the bandwidth available for the copy between memory and the Ethernet card which only provides a 16bit wide interface.

5.2.2 Cambridge Fast Ring

As part of the Super-JANET demonstration phase, a Pandora system situated at University College London was connected to the CFR based Pandora infrastructure at Cambridge via a pair of Fairisle switches and an intermediate 34Mbit/sec PDH circuit carrying ATM cells. The experimental setup is shown in figure 7.

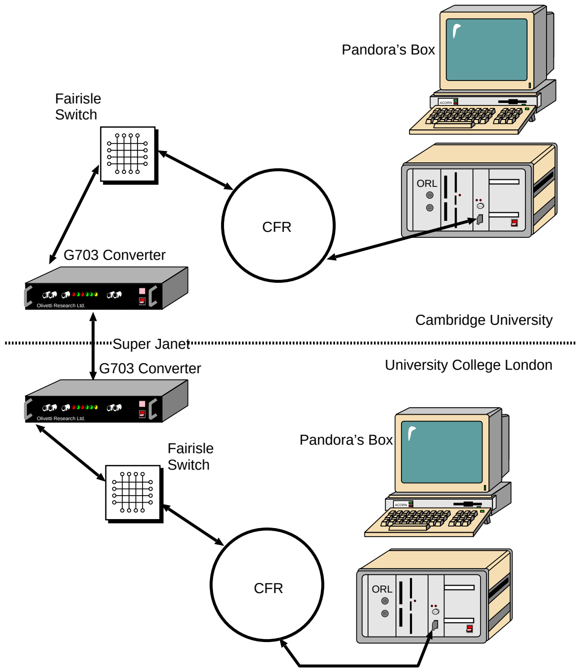

Figure 7: Pandora over Super-JANET

Again the performance was limited by the I/O throughput of the network interface, at about 3 Mbit/sec. However this was sufficient to run the Pandora video applications between the sites.

While the CFR is also an ATM network with identical concepts of virtual channels and signalling to those of Fairisle, the CFR predates B-ISDN standardisation and uses a 38 byte cell (32 payload, 4 header, 2 CRC). In this case forwarding is performed by mapping the payloads of 3 CFR cells to that of 2 Fairisle cells in the obvious manner. To be implemented effectively this requires some form of "push" indication – *irrespective of adaptation layer*. For this we used the AAL5 user-user indication for all adaptation layers.

Hence, *the bit* was found to be useful in implementing a buffering strategy, but at a point in the network which has (and should have) no comprehension of adaptation layers. The bit was simply used as a generic indication of the "recovery unit"; in this sense, it can be used both as an aid to implementing timely buffering strategies, and more importantly to implement more effective discard algorithms in switches [**Ramanathan93**].

6 Conclusions

6.1 The Hardware Software Divide in the Port Controller

The port controller design in Fairisle was aimed at producing an experimental platform rather than a product. This has led to a very flexible implementation in which functionality, once understood, can be placed in hardware (or more accurately, Xilinx firmware). VCI lookup, free queue management, retry strategy and telemetry have all migrated from the processor to hardware. We expect this trend to continue and in particular believe that low level queue management in hardware will follow.

The question then arises as to the need for the processor. We see the processor engaged in longer time scale activities, possibly altering the parameters of the low level queue management as conditions within the switch change. Thus, the processor need not be any faster than it is now for the line speeds we are dealing with, and as functionality migrates out we can see it being able to cope with higher transmission rates.

As mentioned above, maintaining a tight feedback loop between the contention point and queueing point within the port controller is important. This is a key part of the motivation for migrating functionality into the hardware. This may have direct parallels in host interface design – the response time of the host software may require that certain protocol functions be implemented in the host interface (e.g. any link based flow control).

The flexible implementation of the port controller has also allowed us to use the port controllers as traffic sources and monitoring devices whose in band functionality is completely configurable.

6.2 Input Buffering

We were originally attracted to input buffering because of its inherent simplicity. We were well aware of its performance limitations, but were not overly concerned since we were considering the local area where utilisation is not a key factor. Oft cited head of line blocking was also not a concern since we were building port controllers that did not have to perform fifo queueing and a fabric that could run faster than the line rate.

This decision was clearly partly born of necessity; it allowed us, with our rather modest resources, to build a switch. There is no custom silicon in the design and although there are Xilinx devices, none is over 10,000 gates. However, our own experience and the example of AN2 has led us to believe that input buffered switches can achieve output buffered performance; this is achieved without the excessive hardware requirements of output buffered switches but rather by the application of more effective arbitration and/or the fabric speedups possible due to simplicity.

6.3 Reliable Cross Fabric Communication

Something that is often overlooked in the discussion of fabrics is what happens to cells that must be discarded. The most effective discard policy would be implemented at a point in the switch where information derived from relevant higher level Quality of Service is available. Hence "Hail Mary" fabrics which can discard cells internally and are dimensioned on calculations that predict that such events are rare based on traffic assumptions, are unlikely to be universally valid.

Fairisle, because it allows blocking fabrics and blocking outputs must deal with cells that do not get through; the input port controller is always aware. The decision to actually throw a cell away can be made in the processor if required. This fact is exploited in the Desk Area Network; the connection between cache and memory is hardly going to be successful if one in 10^9 cache lines are thrown away!

6.4 The Bit

A number of years ago a campaign was mounted to gain the acceptance of a new adaptation layer by CCITT, now known as AAL5. Part of this campaign was to acquire a bit (or more accurately code points) in the ATM cell header for communication of the frame boundaries.

Within Fairisle we have taken the approach of defining the *semantics* of the bit such that it is consistent with the current AAL5 usage and makes sense in the AAL1/2 and AAL3/4 cases. This has proved useful in dealing with congestion conditions and our experience has shown that it is straightforward to implement in switches.

6.5 Future Work

The British Telecom sponsored extension to the Fairisle Project will be using the flexibility of the implementation to run a number of experiments with artificial and real traffic. We will be able to measure quantities such as cell loss, cell variation extremely accurately on a switch by switch basis.

The flexible manner in which processing functions are distributed will also allow us to experiment with new signalling architectures, in particular those based on ODP. We are particular interested in the relationship between signalling, security and third party setup. This is also the forward looking direction being taken by the ITU-T for release 2 and release 3 B-ISDN signalling. We are contributing to this effort in the hope that we never have to implement Q.93B / SSCOP / ...

References

[**Anderson93**] Thomas Anderson, Susan Owicki, James Saxe, and Charles Thacker. *High Speed Switch Scheduling for Local Area Networks.* Technical Report 99, DEC Systems Research Center, April 1993. (p 9–3)

[**Black94**] R. Black and S. Crosby. *Experience and Results from Implementation of an ATM Socket Family.* In Usenix Winter 1994 Conference. USENIX, January 1994. (p 9–10)

[**Curzon94**] Paul Curzon. *The Formal Verification of the Fairisle ATM Switching Element.* Technical Report, University of Cambridge Computer Laboratory, 1994. Awaiting confirmation of acceptance to conference. (p 9–3)

[**Dixon92**] M. J. Dixon. *System Support for Multi-Service Traffic.* Technical Report 245, University of Cambridge Computer Laboratory, January 1992. Ph.D. dissertation. (p 9–4)

[**Doar93**] J. M. S. Doar. *Multicast in the Asynchronous Transfer Mode Environment.* Technical Report 298, University of Cambridge Computer Laboratory, April 1993. Ph.D. dissertation. (p 9–6)

[**Hayter91**] M. Hayter and D. McAuley. *The Desk Area Network.* ACM SIGOPS Operating Systems Review, 25(4), October 1991. Also University of Cambridge Computer Laboratory Technical Report 228, May 1991. (p 9–5)

[**Hayter93**] M. D. Hayter. *A Workstation Architecture to Support Multimedia.* Technical Report 319, University of Cambridge Computer Laboratory, 1993. Ph.D. dissertation. (p 9–5)

[**Karol87**] M. J. Karol, M. G. Hluchyj, and S. P. Morgan. *Input versus Output Queuueing on a Space Division Packet Switch.* IEEE Transactions on Communications, 35(12), December 1987. (p 9–2)

[**Leslie91**] I. Leslie and D. McAuley. *Fairisle: An ATM Network for the Local Area.* In Communications Architectures & Protocols, volume 21(4) of *Computer Communications Review*, pages 327–336. ACM Press, September 1991. (pp 9–1, 9–10)

[**McAuley90**] D. McAuley. *Protocol Design For High Speed Networks.* Technical Report 186, Cambridge University Computer Laboratory, 1990. Ph.D. dissertation. (p 9–9)

[**McAuley93**] D. McAuley. *Operating System support for the Deak Area Network.* In Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster University, November 1993. (p 9–5)

[**Prudence93**] Mike Prudence. *Sapphire ATM Hub ERS.* Dawn Project Memo, HP Laboratories, Bristol, UK, April 1993. Version 1.0. (p 9–6)

[**Ramanathan93**] S. Ramanathan, P.V. Rangan, and H.M. Vin. *Frame-Induced Packet Discarding: An Efficient QoS Management Strategy for Video Networking.* In Fourth International Workshop on Network and Operating System Support for Digital Audio and Video, Lancaster University, November 1993. (p 9–13)

[**Rodeheffer91**] T. Rodeheffer and M. Schroeder. *Automatic reconfiguration in Autonet.* Technical Report 77, DEC Systems Research Center, September 1991. (p 9–10)

[**Schroeder90**] M Schroeder, Birrell, Burrows, Murray, Needham, Rodeheffer, Sattertwait, and Thacker. *Autonet: a High speed self-configuring Local Area Network with point-to-point links.* Technical Report 59, DEC Systems Research Center, April 1990. (p 9–10)

[**Xilinx92**] *The Programmable Gate Array Data Book.* Xilinx Inc., 1992. (p 9–1)