Protocol and interface for ATM LANs.

David J Greaves* Derek McAuley† Leslie J French* Eoin Hyden†

March 10, 1994

Abstract.

This paper advocates local area networks using the Asynchronous Transfer Mode where data is carried in the payloads of fixed size cells. We describe the design and performance of a simple ATM host interface for a RISC workstation together with the MSNA protocol architecture. A feature of the MSNA architecture is that it can provide ATM virtual circuits directly to applications by demultiplexing to a degree normally associated with the transport layer of a protocol stack. We discuss implementations of the ATM adaptation layer of varying complexity and discuss the appropriate place for the adaptation layer to be terminated in a multi-media workstation.

1 Introduction.

Recently, the Asynchronous Transfer Mode (ATM) has been advocated as an important technology for the wide area interconnection of heterogeneous network types, including traditional packet and synchronous switched networks. We consider that the advantages of ATM for the wide area also extend into the local area. The interconnection of these local and wide area networks to provide an end-to-end ATM service is an important component in providing a flexible service to applications.

Computer manufacturers are beginning to promote the use of ATM techniques in privately owned and operated digital networks. Many networking companies are coming forward offering private ATM solutions with a promise to migrate to upcoming standards (e.g. from CCITT) and emerging *de facto* standards (e.g. from the ATM Forum).

There has been a trend for certain workers in the field to associate the term 'ATM' with the full set of CCITT recommendations for Broadband ISDN. We take our definition of ATM from the the broader meaning of the term actually defined by CCITT in recommendation I.113; that is:

> *the use of a fixed-length cell as the primary means of information transfer where the periodicity of cells is not known by the receiver in advance, but is indicated by a label in the cell header.*

For an ATM system to inter-work with other ATM systems there must be an agreed set of functions and associated semantics which each of the systems supports. Taking interoperation with B-ISDN as a pre-requisite, this requires primarily the support of a subset of the user plane semantics of the CCITT definition of the B-ISDN ATM layer. This gives:

*Olivetti Research Ltd., Trumpington Street, Cambridge CB2 1QA, United Kingdom

†Cambridge University Computer Laboratory, Corn Exchange Street, Cambridge CB2 3QG, United Kingdom

- an end to end virtual circuit with a known correspondence between the VCI/VPI values encountered at the two ends of the circuit,
- ordered delivery of cell payloads (in this case to be able to inter-work with B-ISDN, 48 octets) by the circuit,
- an extra bit (in the header) carried transparently with each cell.

Collaborative research work to investigate this approach is being carried out at the University of Cambridge Computer Laboratory and at Olivetti Research Ltd. We are developing ATM host interfaces for standard workstations and have implemented the MSNA protocol suite in UNIX and in an experimental kernel called Wanda [4]. In this paper we present an overview of MSNA and relevant adaptation issues together with the design and performance results of an ATM interface for the DEC Turbochannel.

2 ATM Motivation.

The motivation for the use of ATM techniques in the local area stems from four cohesive points of view; ATM is:

- a technique which lends itself to the implementation of cost effective high performance switches,
- a flexible local area communication mechanism for the support of multi-service traffic,
- a means to deliver wide-area ATM services (especially B-ISDN) to desktop end systems,
- a means to decouple various matters of policy and implementation for example in resource management, charging, authentication, etc[1].

The technological and performance arguments for networks based on a general topology interconnection of ATM switches have been, and continue to be, fully represented in the literature. Switches with 160 Mbit/s line rates and aggregate capacities of several Gbit/s are now feasible with current technology, offering a packet switched service at rates previously only associated with STM networks. Small scale switches for private and LAN use are now becoming available; in our environment we use switches varying in size from 4 by 4 to 16 by 16 developed within the Fairisle project [14]. Designs for larger switches such as required for public networks present interesting scaling problems.

However, our consideration of ATM based networks in general (rather than simply B-ISDN), also leads to the consideration of networks based on different cost, technology and management tradeoffs, which serve to extend the spectrum of solutions offered by ATM networks. Examples include DQDB extended to support virtual circuits, the ATM ring recently proposed as a CPN [13], and the 500 Mbit/s CBN [6] (a network in everyday use between our two laboratories). These ATM technologies encompass a variety of physical layer and cell header formats, but are able to inter-work at the ATM layer.

The flexibility of ATM comes from the fine grain of multiplexing present in the ATM layer, which allows a large delay-insensitive packet to be pre-empted by higher priority or time-sensitive traffic. While it is true that this allows ATM to provide an integrated mechanism to carry traditional STM and PTM services, the ATM multiplexing can also offer a flexible interfacing mechanism to end-systems and hence a much richer range of possible services can be provided to applications. It is this latter aspect which we have developed at Cambridge.

[1] Such matters are described in more detail in a previous article in IEEE Network [15].

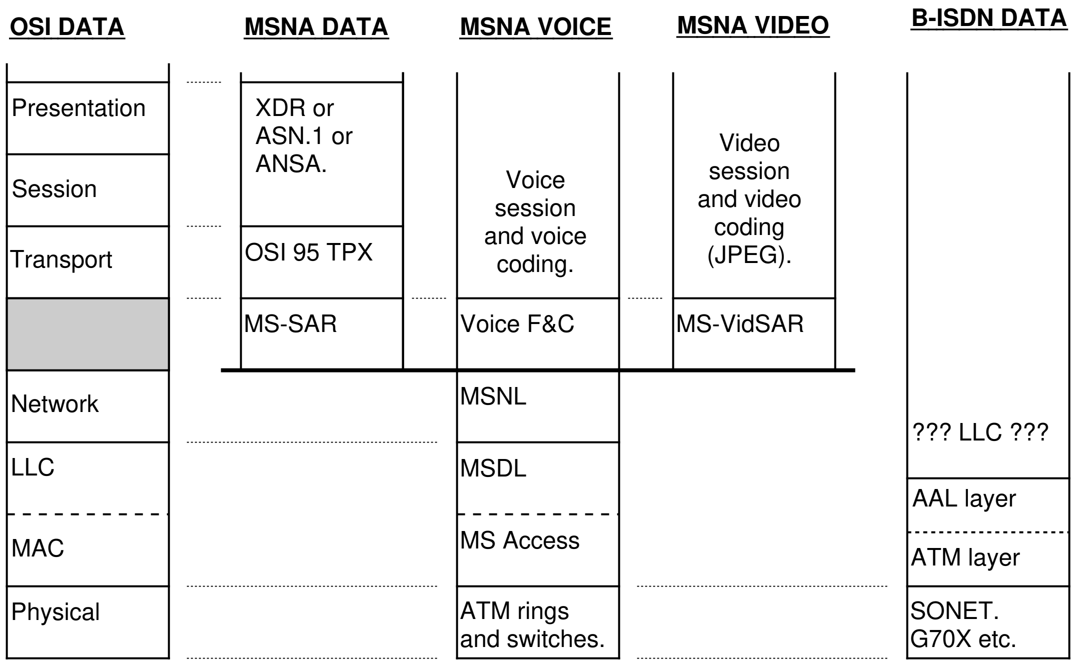

Figure 1: Functional mapping of MSNA to the OSI reference model and CCITT B-ISDN reference model.

3 MSNA.

The Multi-Service Network Architecture (MSNA) was developed to address both the naming and signalling required for the establishment of end-to-end ATM cell streams, and the integration of the in-band processing of cells with the systems present at the end points of communication links [16].

This protocol architecture takes an internetworking approach as we believe that different varieties of ATM networks will appear, depending on the functionality and data rate required, and the geographic area covered. An operational MSNA 'internet' exists in Cambridge UK and spans the sites of Olivetti Research Ltd., the University Computer Laboratory and University Engineering Department. The physical network components consist of Fairisle ATM switches [14], slotted rings [11] and [6], and, in the near future, radio based ATM systems.

The Multi-Service Network Layer (MSNL) is the basic ATM service, giving an end-to-end stream of cells. It defines the MSNL address and circuit establishment techniques, including the mechanism for requesting various qualities of service (we are currently working on call acceptance and control algorithms to enforce the QoS guarantees).

Different ATM networks must present a common interface (MSDL) to the MSNL layer. During MSNL circuit establishment, routing is performed at each MSNL entity to select the appropriate MSDL instance for the next hop. The MSDL implementation for each network type then maps the MSNL circuit establishment mechanisms to those appropriate for its underlying network type; this can involve both address mapping and, in some cases, protocol mapping. For example, the MSDL instance for B-ISDN would map the MSNL mechanisms to Q.93B to establish that portion of the virtual circuit which traverses the B-ISDN network.

Once the circuit is established, the routing of cells from one MSDL instance to another is normally performed by hardware. Where the two MSDL instances are identical, this is the normal VCI mapping function seen in ATM switches. Where they are different, the hardware may also be required to translate the other fields of the header.

A particular design goal in MSNA is to eradicate the requirement to perform layered multiplexing whcih is often found in communications systems. For reasons of performance and quality of service, we often wish to identify the end-system application entity with the minimum of overhead and invoke the normal resource allocation mechanisms of the end system (e.g. processor scheduling and buffer management) before engaging in higher-layer protocol processing. Hence the MSNL address contains a component which identifies the end-system and a component which identifies an entity within the end-system, even to the extent of identifying an application level entity; this is similar to the common IP/UDP, or IP/TCP, host/port pair.

Where possible, and certainly when application specific quality of service is required, we aim to map a single application association to an MSNL virtual circuit so that when communication is establishedq, it is possible to identify the application entity within the end system directly from the VCI of a received cell. Then, although we may build adaptation, error control, transport, session and presentation layers on top of MSNL for particular services, these provide no further multiplexing. This also fits naturally with some services, such as video, which implement the majority of the in-band processing in hardware.

Such an approach of course does not insist that the identified entities be applications, and where appropriate (examples in use include IP over ATM and Ethernet bridging), the entity can be a particular protocol stack which implements further in-band multiplexing.

4 Adaptation layer protocols

Our aim in producing an ATM 'internet', with ATM delivered into the end system, means that our consideration of adaptation is concerned primarily with efficient implementation in such end systems rather than as a means of interconnection of current packet or circuit switched traffic types. However, by providing end-to-end service for cell payloads, MSNL can support any adaptation layer, including those recommended by CCITT.

Within MSNA, and in keeping with the desire to eradicate unnecessary multiplexing, we have implemented only adaptation layers which do not perform cell level interleaving of different segmented blocks over a single virtual circuit. When cell level interleaving is required, distinct VCIs are used; that is, we use multiple MSNL streams and allow the VCI to perform its intended role of distinguishing them. In B-ISDN terms, this is equivalent to using a virtual path service where the end-to-end VCI bits distinguish cell interleaved blocks, rather than using a single VCI and sorting cells into blocks by yet another layer of multiplexing, for example by use of the MID in AAL-3/4.

In the MSNA architecture, the block level service is known as MSSAR (multi-service segmentation and reassembly) and is built on top of MSNL. We have been using a particular adaptation protocol for a number of years [9][2], while investigating issues in adaptation. We investigated the minimal information required to implement sufficiently secure yet efficient and simple to implement adaptation layers, and have now arrived, together with other researchers, at a class of protocols which require that the ATM cell header include a logical end-to-end user data bit[3]. The bit can be used to implement a range of different adaptation layers, but in this paper, we concentrate on adaptation layer protocols for data services.

[2]This adaptation layer, called UDL, places 8 bits of sequencing information in the header of each cell, and is the one used in the performance example presented in Section 6.2.

[3]The December 1991 CCITT draft I.361 extended the payload type to three bits and defined this end-to-end user data bit. It is termed the 'ATM layer user-user indication'.

For data services, as a starting point we may take the definition:

> *The fundamental role of the data adaptation layer is, at the transmitter, to take a PDU and provide segmentation into cells and, at the receiver, correctly reassemble the cells.*

The requirement for *correct* reassembly is generally necessary if the adaptation layer is to fulfil its nominal role of adapting ATM cells to the variable-length PDUs of conventional protocol stacks, but we question the requirement for accurate reassembly more closely in Section 5.1. The issue is a hardware versus performance tradeoff predicated on the desirable complexity of an ATM interface which performs or helps with adaptation in hardware.

A suitable data adaptation layer is the ATM adaptation layer type 5, being standardised within CCITT [2]. AAL-5 uses the bit to indicate last cell of a block, where the last cell includes a length field (or cell count) and a 32-bit CRC over all the cells forming the block. The length field is used by the adaptation layer to determine that the correct number of cells have been received. If, necessary, a further length field, indicating the valid data section of the PDU, etc., can be placed elsewhere in the PDU by higher level transport or RPC protocols.

The CRC detects the usual bit and burst errors, as well as most cell reordering while the length is required to detect certain cell drop outs where the content and position of the cell contrive to provide no contribution to the CRC. The CRC polynomial used in an ATM adaptation layer needs to be chosen with care, since if the CRC of $x^{i*384}+1$ is zero, swapping two cells distance i apart will not be detected. The polynomial selected for AAL type 5 (the standard CCITT CRC32) will detect such pair swaps up to the maximum size of the AAL5 PDU, 64K bytes.

We have measured the rate at which a workstation can calculate a CRC in software. Calculation of a 32 bit CRC on blocks of data on a 25MHz DECStation 5000/200 can be done at only 13 Mbit/s. This is using an 8 bit at a time method and two 256 by 32 look up tables. (Using 16 bits at a time requires two 64K by 32 look up tables which have undesirable effects on cache performance.) This suggests that assistance with the calculation of AAL CRCs is required in the interface.

5 Host and workstation ATM interface design.

A workstation interface for an ATM LAN should not require a host processor interrupt for every cell transmitted or received. Such interfaces thus differ from conventional LAN controllers (e.g. Ethernet) in that they must provide buffering for multiple received PDUs (cells). Simple FIFO buffering of the received and transmitted cell streams is sufficient to enable the host to operate asynchronously with respect to the network and also to reduce the interrupt rate below the one-per-cell level. This allows host interactions to occur at the 'packet' rate – although at high load, this is still a rate of interrupts twenty[4] times that experienced from a fully utilised Ethernet. An interface which restricts itself to this minimum hardware complexity is the current Olivetti Research Turbochannel ATM interface, described in Section 6.2.

Two ATM-specific functions may be readily incorporated into a more sophisticated ATM workstation interface. On the receive side, these are:

1. *sorting*, or *fragmented reassembly*, of received cells according to their incoming VCI/VPI – and for protocol stacks such as AAL-3/4, sorting on fields within the payload (e.g. the MID[5]), and;

[4] Our links are currently 100 Mbit/s and of course full duplex.

[5] In MSSAR (and AAL-5), MIDs are not used. Further in MSNL there is no *a priori* division of the header into

2. the extraction and checking of per-cell adaptation layer check functions and protocol headers[6].

The inverse of these functions applies to the transmit side of the interface.

The following question arises: what applications require interfaces which should have, or need to have, this level of complexity? A high performance system may wish to have these functions implemented in hardware to relieve load on the main processor. On the other hand, for a low power mobile ATM station, several megabits-per-second can easily be handled by the CPU, providing a trade off between hardware complexity and processor cycles.

The advent of a low power, high performance ATM interface ASIC would probably be suitable for a range of interfaces, but restricted in the supported adaptation layer protocols. Equally, in some instances, the bus architecture of a machine may lead to network interface solutions in which no adaptation is done in the interface, rather cells are routed to various dedicated processing units, for example, the framestore, compression engine, disk controller or main store. This latter approach preserves the fine granularity of ATM over the internal workstation bus, and the network attachment has then become merely an ATM multiplexer/demultiplexor. An extension of this approach is to implement the internal interconnections within the end-system as an ATM switch. This is being investigated within the 'Desk Area Network' project at Cambridge [10].

5.1 Cost and role of end-to-end check functions.

Returning to data applications, a necessary protocol function is guaranteed end-to-end integrity of PDUs. In OSI and TCP/IP, this function is implemented in the transport layer using the TP4 and TCP check functions respectively. The cost of computing the check function in software has been recently widely debated.

One view is that data generally has to be copied at least once by the processor, whether by the kernel or user-space code, and whether from a shared IPC buffer pool or from dedicated interface buffers. It is then of negligible cost to introduce into the copying code a simple checksum or cyclic exclusive-or function (as used in XTP [12]). In addition, through suitable implementation, data which is never processed, need not be checked.

An alternative view is that a combination of current operating system environments and protocol stacks leads to the inefficiency of a software copy. For example, many operating systems lack mechanisms to allow network DMA into user address spaces (e.g. user processes cannot lock down pages), while at the point in the network interface where DMA could be simply implemented, a protocol stack can require a sufficiently complex demultiplexing operation, that the interface has insufficient information and intelligence to identify the target user address space.

An further extreme view is that PDU integrity might be ensured at the network level; this may be available in an ATM environment as a virtual circuit attribute in the style of QoS parameters. The network level can combine knowledge of the error performance of each individual data-link along the route and so may be able to offer a *nothing-corrupt-if-delivered* QoS.

Several observations are easily made:

1. Within a reliable end-system, it is pointless to implement a transport level integrity check of the same strength (or weaker strength) than an already implemented end-to-end check at

separate VCI and VPI fields (c.f. Section 4).

[6] Adaptation protocols based on a header flag, such as the one described in Section 4 do not require adaptation layer headers in every cell payload.

a lower protocol level[7].

2. A CRC algorithm is the best type of data check, given that we are considering heterogeneous ATM networks, including radio and optical mobiles, where the error rates will typically be higher than in B-ISDN.
3. CRC calculation in software is too slow and CPU intensive for most applications (which is why weaker checksums have normally been used).
4. The network interface is a potentially sensible place to put CRC hardware.
5. If hardware in the interface is helping with the adaptation layer requirement that PDUs which are wrong owing to cell removal, repositioning[8] or replication are not (eventually) discarded, it must perform a check on the cells. This is either with per-cell sequence numbers or whole PDU check functions (e.g. CRC).

These observations lead to a number of possible solutions:

1. Do not check cells at the adaptation layer. Instead, pass up to the transport layer all cells received on a virtual circuit until the end-of-PDU indication is set, then check only at the transport layer. This essentially results in a transport protocol which is able to operate directly on concatenated ATM cell payloads. There is no reason why it should not also be able to operate and inter-work with instances of itself on non-ATM data-links.
2. Implement a specified level of checking at the adaptation layer, then pass up the partially checked PDU to the transport layer software, where the remainder of the checking is done. This may in fact be a sensible approach if, as mentioned, for efficiency the transport entity can only do weak checking, such as checksum (checksum does not spot out-of-order cells.)
3. Fully reassemble and check each received PDU at the adaptation layer and then pass up only correct PDUs with the implicit semantic that they are checked and correct.
4. Calculate the CRC (or other check) of each cell payload in the interface and pass up this partial CRC along with the payload. The benefit of this is that higher level software is able to combine the partial CRCs to obtain a full CRC for the PDU at twelve[9] times the rate it would achieve by performing the whole CRC in software.

Similar issues where hardware implementation and protocol design cannot easily be decoupled can arise when attempting to place other higher-layer functions in the physical layer interface; encryption and presentation engines are significant examples. We do not discuss these issues here.

6 Implementing host and workstation ATM Interfaces.

Existing hardware implementations of ATM interfaces are now presented in Sections 6.1, 6.2 while enhancements are presented 6.3.

[7]To avoid this duplication in practice, the transport protocol must be able to accept a 'data-ok' indication from the lower layers.

[8]Although CCITT has defined that B-ISDN should not reorder cells, we are considering all types of ATM system.

[9]Assuming a 32 bit CRC, there are twelve 32 bit words in each cell payload.

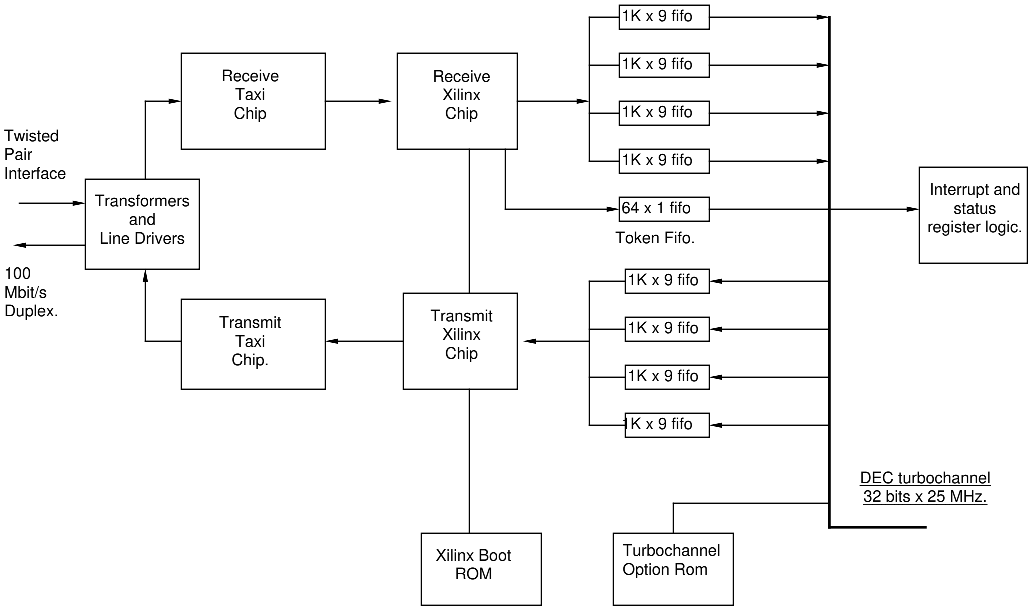

Figure 2: Block diagram of the 'Yes V2' option module

6.1 Previous ATM interfaces for workstations.

ATM interfaces for Unix workstations have been reported by Davie [3] and Traw [18]. These are both high functionality interfaces. Traw's controller adapts ATM cells on a 155 Mbit/s SONET STS-3 carrier to an IBM RS 6000 Microchannel and performs sorting of received cells in microcoded hardware. However it does not implement reassembly check functions. Davie's controller is for the DEC Turbochannel and is equipped with two Intel 960 processors to offer a variety of processing options. The controller will connect to four STS-3 carriers or one STS-12 carrier.

Simpler controllers have been built by Olivetti Research and Fore Systems for Turbochannel and S-bus respectively. These do not perform protocol processing, except for the ATM header check (HEC-8), and in the case of the Fore Systems units, the payload AAL-4 CRC-10. Neither interface maintains state information at a granularity larger than a cell.

Design and performance of the ORL interface is now presented.

6.2 Olivetti Research simple Turbochannel ATM interface.

The ORL controller, known locally as the 'Yes V2' interface, employs a physical layer using the AMD TAXI transmission devices at 100 Mbit/s. The physical layer cell format accords with a *de facto* standard drawn up by Sun and Xerox and now adopted by the ATM Forum [1]. Hence taking the header and inter-cell symbols into account this affords 94.5 Mbit/s of payload throughput.

The controller has separate receive and transmit FIFOs, each able to hold 76 cells. A further 'token' FIFO is used merely as an up-down counter to keep track of the number of cells completely received. Host data transfer is either through programmed IO or by DMA. DMA transfers of up to 39 cells are supported by the current Turbochannel host system, the DECStation 5000. DMA transfers may either be used to copy a block of memory which has already been formatted by the processor with the appropriate ATM cell header inserted every 13 words, or in conjunction with programmed IO, where the processor reads or writes the ATM header word directly to/from the FIFO and then initiates a DMA transfer of the cell payload.

A receive interrupt can be generated on each cell received, or alternatively, on the arrival of cells with the header indication flag set. Using this latter mode, a PDU may accumulate in the receive-side FIFO until its last cell has been received, and then the host interrupted. Of course, the host may find other cells in the FIFO from incompletely received PDUs on other associations, in which case it must sort and store these as usual. However, these cases will be rare for many data applications [17] and, in all cases, the overall frequency of interrupts may be reduced. On the

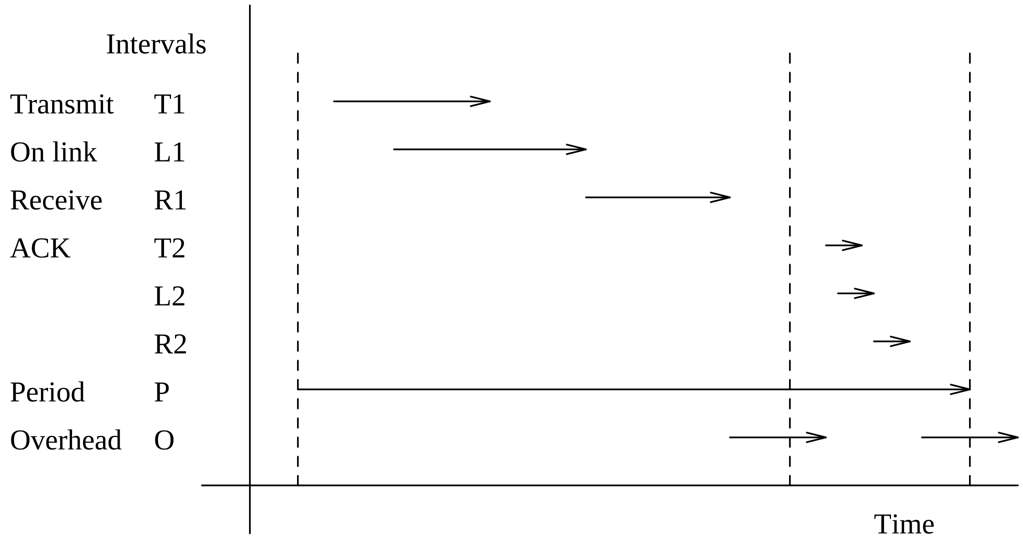

Figure 3: Stages of the loop-back test

transmitter side, interrupts are available on TX FIFO less than half full and on end of DMA.

Transmit side interrupts are not yet required in the Unix device driver, since, as is explained shortly, we have found that sustainable transmit rates are less than the 100 Mbit/s available and so the transmit side does not go busy.

Experiments where performed using MSNA under both Ultrix and our local experimental kernel Wanda, on DECStation models 5000/200 and 5000/25 respectively[10]. The experimental system consisted of a single workstation with the MSNL protocol in its kernel provided to user processes as an MSNL *socket* and a Turbochannel controller, with the output from the TAXI transmitter looped back to the TAXI receiver using a short length of cable.

Two processes were involved, a sender and receiver process. The sender process sends a PDU of 70 cells through its MSNL socket and out through the controller. Since the controller cable is looped back, the receiving side of the interface directly receives the PDU. The receiver passes the cells up through another MSNL socket to the receiver process. The receiver process then sends a single cell ACK message in the reverse direction. The process then repeats; this repetition gives a stable display of various signals on an oscilloscope for taking measurements in a non-intrusive manner. All values for throughput represent ATM cell payload throughput.

Figure 3 represents one cycle of this repeating process and labels the various time intervals of interest in the test:

T1: copy from buffer to transmit FIFO, including fragmentation; for Ultrix this buffer is the kernel space buffer,

L1: time of PDU on the wire,

R1: copy from receive FIFO to buffer, and locating the receive control block and the relevant buffer on the first cell,

P: total time for one periodic cycle,

O: overhead including context switch, socket and user code, and for Ultrix user/kernel copies.

[10]The major between the two machines is that the /200 has an 800 Mbit/s Turbochannel while the /25 has a 400 Mbit/s version.

T2, **L2**, and **R2**, are the appropriate intervals for the acknowledge. During the experiment, the processor was 100% utilized; this was the case even when "interrupt on end of block" was set, as the rate at which data was inserted into the transmit FIFO was less than the 100 Mbit/s line rate in most of the experiments, so that the time between the end of the transmit phase and the start of the receive phase was only 4μs.

The measured values of interest are presented in Table 1. Other experiments have shown that it is reasonable to allocate the operating system and user overhead component equally amongst the transmit and receive paths. Hence we can deduce that sustained transmission rates of 28 Mbit/s and 46 Mbit/s, and reception rates of 16 Mbit/s and 19 Mbit/s can be achieved with Ultrix and Wanda respectively. The througput rates shown over the whole period, of 9 and 13 Mbit/s, show the performance possible to a simple RPC type of communication.

		Ultrix		Wanda	
	Interval	μs	Mbit/s	μs	Mbit/s
Transmit	T1	792	34	540	50
Receive	R1	1485	18	1385	20
Period	P	3125	9	2150	13
Overhead	O	420	-	80	-

Table 1: Interval durations and equivalent throughput for programmed IO

Two observations can be made from the results. The first is the difference between the transmission and reception rates; this is explained by the cost of per cell checks on reception, including VCI checks and, with the SAR protocol being used, sequencing checks. The second is the difference between the Ultrix and Wanda performance; this is mainly due to the lack of a data copy in the case of Wanda where all data is moved between kernel and user via a pool of interprocess communication buffers, with the optimization of not changing their virtual address on transfer of ownership having been taken to avoid performance penalties due to cache flushing.

Considering the hardware simplicity of an interface which is only required to support programmed IO, the performance is still respectable. The 'Yes V2' also has support for DMA and we have been able to achieve some preliminary results using DMA, presented in Table 2. However, while comparing programmed IO on Ultrix and Wanda was straightforward, in supporting DMA, significant differences are experienced in the implementation route taken for the two operating systems. Of major importance is that with the machines used, while the processor has a write through cache, it does not snoop on the memory bus and hence explicit action must be taken to avoid inconsistencies between the cache and the memory system due to DMA actions. Hence while tranmission DMA is straightforward, DMA on receive is more complex.

	Ultrix		Wanda	
Interval	μs	Mbit/s	μs	Mbit/s
T1	210	128	215	125
R1	-	-	401	67

Table 2: Interval durations and equivalent throughput for DMA

With Ultrix, advantage can be taken of the need to copy from user to kernel mode; during the copy, a valid header is inserted in front of every 48 octets of user data copied, so that on transmission, multiple cells are moved in a single DMA transfer. Using this mechanism the sustainable transmit throughput can be increased to 61 Mbit/s, with the DMA transfer itself happening at 128 Mbit/s. This "multiple cells at a time" mechanism is not suitable for use on reception (without resorting to another copy) as each cell header must be interrogated to obtain the VCI. In fact within Ultrix, we have so far been unable to deal with the cache/memory consistency problem so that results for receive DMA are not available.

With Wanda, we were able to implement DMA on both receive and transmit. As there is no user/kernel copy operation within Wanda, we chose to implement a DMA transfer per cell.. Even so, the overall transfer rate from memory to transmission FIFO is still a respectable 125 Mbit/s, and together with Wanda's lower per packet overhead this represents a sustainable transmission rate of 99 Mbit/s (hence greater than the supported 94.5 Mbit/s). The manner in which the interprocess communications buffers are used within Wanda also allow us to overcome the cache/memory consistency problem for reception so that the FIFO to memory transfer achieved is 67 Mbit/s resulting in a sustainable throughput of 57 Mbit/s.

6.3 Worthwhile functionality in the ATM interface.

We have shown that an interface which merely reduces the host interaction rate to below once per cell is only sufficient for half-duplex operation of a 100 Mbit/s ATM interface, when using the full processing power of current RISC workstations. Using DMA for each cell increases the performance of the interface over programmed IO, but fails to release the processor to enable it to handle other activities. This is contrary to the typical case when using DMA with lower speed and non-ATM interfaces.

The approach of advance formatting a user message into cells enables longer DMA transfers to take place. Per-cell processing is still required to generate the cell structure, but this processing is decoupled in time from actual cell transmission. In addition, consecutive cells need not be for the same virtual circuit, hence enabling more flexible rate control and smoothing out bulk arrivals, which can be of benefit to downstream switches.

An alternative transmit structure of not much greater complexity puts the complete fragmentation sub-system in the interface. The host writes a prototype 32 bit cell header to a header register in the interface. This header, along with its HEC, are automatically inserted in the outgoing stream after every 12 words (48 bytes) of data are transferred to the interface. (Data may be moved either by programmed IO or by DMA.) In addition, the interface may insert other convergence sub-layer and AAL fields, including any per-cell (AAL-3/4) or final (AAL-5) CRC and length checks. This interface removes all per-cell processing from the host, but has lost the ability to intermix cells from different virtual channels at a fine granularity in the outgoing stream.

For reception, the presence of interleaved cells from different virtual circuits must be handled, but one can implement a relatively straightforward interface where the hardware has only the capability to handle adjacent cells on the same virtual circuit and software is used otherwise.

When a cell is received which is on a different virtual circuit from the previous cell a 'context swap' of the interface is required. The processor reads out the partially-calculated running state, such as total AAL-PDU length and overall CRC or checksum, and any other fixed values, such as block identifiers, VCI, VPI and MIDs, and stores them in an internal association table. The identity of the next virtual circuit is available by reading just the cell header out of the buffer store in the interface. There are then two methods for context restoration aspect. The processor may either re-load the interface with the running values of parameters which contain inter-cell state, or it may reset these registers to a known starting value and afterwards combine the separate results in software.

An interface for a modern RISC processor using this approach should easily be able to transmit at the 100 Mbit/s rate. Reception will only be significantly slower than transmission if cells from separate associations are highly interleaved.

In general, the incorporation of CRC generating and checking hardware is vital for high performance interfaces. Since we intend to use the new adaptation layers which employ CRCs, such hardware must be deployed in all of our future interface designs.

6.4 A Sorting Interface

It is not clear that any further worthwhile features can be added to an interface unless it implements the sorting function. A sorting interface may either accumulate cells in its own local memory, or may use direct insertion into the main host memory. An advantage of the former approach is that a CRC may be calculated without interruption when data is copied once from the controller to host memory. Advantages of the latter approach are intrinsically shorter bus holding times and 'cut-through' where data is ready in host memory as soon as possible.

Sorting interfaces may only be able to support a subset of the available VCI space since it is desirable to use directly-mapped memory to look-up VCIs. In all of our current MSNA data-link services (our ATM switches and rings), this problem is ameliorated since receivers are able to allocate their own incoming VCIs. If this is not possible when directly connected to B-ISDN, then it should at least be possible when using a local ATM LAN. The ATM LAN may be connected to B-ISDN at one or more points. The entity at the B-ISDN to ATM LAN boundary can then use associative techniques if necessary[11]. An interface which can support 64000 active VCIs is deemed sufficient by [3], and we would agree.

Custom and semi-custom VLSI are being used to implement an ATM LAN controller. These devices connect to OEM host bus controllers, such as those available for EISA, Turbochannel, VME etc. and to inexpensive fibre-optic physical layer devices, such as the AMD FOXI parts, or other standard line interface units (e.g. for T3, E3 or SDH). A full ATM interface ASIC performs sorting, adaptation, rate control and probably buffer management of local memory and scatter-DMA in host memory. Example specifications are available as data sheets from the manufacturers PMC Sierra Inc of Burnaby BC and Base 2 Systems of Boulder CO and from Cambridge UK in [5] and [7].

7 Future directions.

The increasing stability of the user plane within the CCITT recommendations for B-ISDN and the wide applicability of the B-ISDN 'ATM-layer' specification to all types of ATM system enables the definition and implementation of ATM interface ASICs. The ever falling cost of VLSI implies that a variety of interface controllers for ATM end systems will become available.

Users of private ATM LANs can employ an approach, such as MSNA, where the simple semantics of an end-to-end virtual circuit enable users to insulate themselves from variations in B-ISDN specifications. Heterogeneous styles of ATM network, including B-ISDN, proprietary ATM LANs and mobile optical and radio ATM systems, may inter-operate provided each is able to offer the basic ATM layer service of shipping 48 byte cell payloads (plus the bit in the header) without miss-ordering, along a virtual circuit. For signalling, one requires entities which perform signalling adaptation, from one signalling domain or protocol to another. These entities have no in-band presence and so do not introduce a restriction for data throughput. Local islands of heterogeneous ATM technology will achieve wide-area interconnection using the empty adaptation layer offered by B-ISDN [8].

[11] The B-ISDN switch manufacturers are faced with the same VCI/VPI look-up problem, so are probably able easily to control VCI space. The question is whether they will make VCI non-sparseness guarantees to the customer.

References

[1] R Breault and D E McDysan. *ATM User-Network Interface Specification*. From the ATM forum.

[2] CCITT. *I.363: B-ISDN AAL Layer Specification*, 1993. Draft.

[3] Bruce S Davie. A host-network interface architecture for atm. In *Proceedings of SIGCOMM 91 Zurich*, September 1991.

[4] MJ Dixon. System support for multiservice traffic. Technical Report 245, Cambridge University Computer Laboratory, 1991. Order by email to "tech-reports@cl.cam.ac.uk".

[5] David Greaves. *Specification of a Workstation ATM interface ASIC*. OSI95/ORL/TN/B3/7/P from Esprit project OSI 95.

[6] David J. Greaves, Andy Hopper[D, and Dimitris Lioupis. The cambridge backbone ring. In *Proceedings of IEEE Infocom*, 1990. San Francisco.

[7] DJ Greaves. *Specification of a high performance ATM workstation interface*. Available for ftp as 'quince.cam-orl.co.uk:/pub/yesv3.ps'.

[8] DJ Greaves and D McAuley. Private atm networks. In *IFIP 3rd International Workshop on Protocols for high speed networks*, May 1992. Stockholm.

[9] DJ Greaves and ID Wilson. Cambridge hslan protocol review. In H Rudin and R Williamson, editors, *IFIP WG6 International Workshop on Protocols for high speed networks*. Elsevier, 1989. IBM Ruschlikon.

[10] MD Hayter and DR McAuley. The desk area network. *Operating Systems Review*, October 1991.

[11] A Hopper and RM Needham. The cambridge fast ring networking system. *IEEE Transactions on Computers*, 37(10), October 1988.

[12] Protocol Engines Inc. *XTP protocol definition revision 3.6*, January 1992.

[13] *Proposal for Asynchronous Transfer Mode Based Medium Access Method and Physical Layer Specifications*, July 1992. ISO/IEC JTC1/SC6 N7662.

[14] Ian Leslie and Derek McAuley. Fairisle: An atm network for the local area. In *Proceedings of SIGCOMM*, September 1991. Zurich.

[15] Ian M Leslie, Derek R McAuley, and David L Tennenhouse. Atm everywhere. *IEEE Network*, March 1993.

[16] DR McAuley. Protocol design for high speed networks. Technical Report 186, Cambridge University Computer Laboratory, 1989. Order by email to "tech-reports@cl.cam.ac.uk".

[17] Jeff Mogul. Network locality at the scale of processes. In *Proceedings of SIGCOMM*, September 1991. Zurich.

[18] Brendan Traw and Jon Smith. A high performance host interface for atm networks. In *Proceedings of SIGCOMM*, September 1991. Zurich.

This work was partly funded by Esprit project OSI 95: High performance OSI protocols with multimedia support on HSLANS and B-ISDN.