AAL5 at a Gigabit for a Kilobuck

Greg Watson, David Banks, Costas Calamvokis,
Chris Dalton, Aled Edwards, John Lumley
Networks and Communications Laboratory
HP Laboratories Bristol
HPL-93-37
May, 1993

Gbit/s LAN, B-ISDN,
ATM, AAL5, TCP/IP

We present a novel LAN that has been designed to meet three goals: low cost, standards and protocols, and high performance.

The LAN uses a ring topology to interconnect many workstations to each port on packet switch. The physical layer operates at one Gbit/s. The packet format is that defined by the CITT for B-ISDN, but the length is variable. The network also provides hardware support for ATM Adaptation Layer 5, as this AAL5 will be widely used for data transfer as well as signaling.

We have implemented the ring network and we describe a network interface card which provides hardware support for critical functions such as calculating checksums. This card is used in conjunction with a second card that supports a single-copy implementation of the TCP/IP protocols. The application-to-application throughput has been measured at rates of up to 200Mbit/s between two workstations.

1 Introduction.

Asynchronous Transfer Mode (ATM) has been chosen by the CCITT as the core technology for the future broadband public network (B-ISDN) [1]. ATM has also been embraced by private network vendors as the basis for a future-proof, high-performance local area network that can offer services for emerging applications such as multimedia [2].

The local area network market, though, is very different from that of the public network and is much more sensitive to cost. Novel LAN technologies such as ATM must compete with the well established low-cost technologies such as 10BaseT, where a network connection is available for about $150 including hub costs. Low cost ATM networks which provide from 50 to 150 Mbit/s link rates are under development by several vendors. What is not clear, though, is whether ATM will be cost-effective at the very high rates of 600 to 800 Mbit/s.

One disadvantage of ATM at very high data rates is the use of small fixed-size cells. The segmentation (and reassembly) of variable-size packets into these small cells will require special hardware support in order that the link rates are made available through the end system up to the application. These segmentation and reassembly (SaR) functions are required because current protocols are based on the notion of a variable-size protocol data unit. Rather than modify the current protocols, it has been proposed that an ATM Adaptation Layer (AAL) should be inserted between the ATM cell service and the current protocols in order to maintain a network service that can transfer variable-size packets. Several AALs have been proposed, each offering different services. Of these, AAL5 appears to be the choice of many vendors for an asynchronous data service.

We present a network system, called Jetstream, which was designed to meet three core objectives:

- Low cost
- Standards (AAL5 and TCP/IP)
- High Performance (Gbit/s link rate, high end-to-end throughput)

In order to achieve these objectives we have designed and implemented a complete networking system. The standard TCP/IP protocols were recoded under the assumption that the network interface could support a single-copy protocol stack. A network-independent card was developed to provide the buffer model required for the single-copy TCP/IP. A network-specific card was developed that would attach to the buffer card and provide the Gbit/s link capability as well as special hardware

assistance for AAL5. The entire system has been designed to function as a whole, and has been optimised for a particular group of workstations.

In this paper we focus on the network-specific card, and how it is used to provide AAL5 at Gbit/s rates for a low cost. In section 2 we describe the main sources of cost in an ATM network and we suggest ways to reduce these costs. In section 3 we present the Jetstream network and we discuss several aspects including the access protocol, initialisation procedure and the architecture of the network interface card. In section 4 we briefly describe the other components of the system with emphasis on how architectural support is required to support a single-copy approach to protocol implementation. The measured performance of our system is presented in Section 5.

2 A low cost AAL5 network

While ATM-based networks will, in the future, provide services for voice, video and other multimedia traffic types, it is fairly certain that initial ATM networks will be used as traditional LANs to carry asynchronous data. Consequently, the performance of asynchronous data transfers will be critical to the success of ATM.

ATM networks are based on the idea of cell switching. Every end system is directly connected to a switch via a bidirectional link, and switches may be connected to other switches in arbitrary mesh topologies, as shown in figure 1. Examples of research networks include FALCON [3] and Sunshine [4], although in 1993 several vendors offer commercial ATM LANs at 155Mbit/s rates.

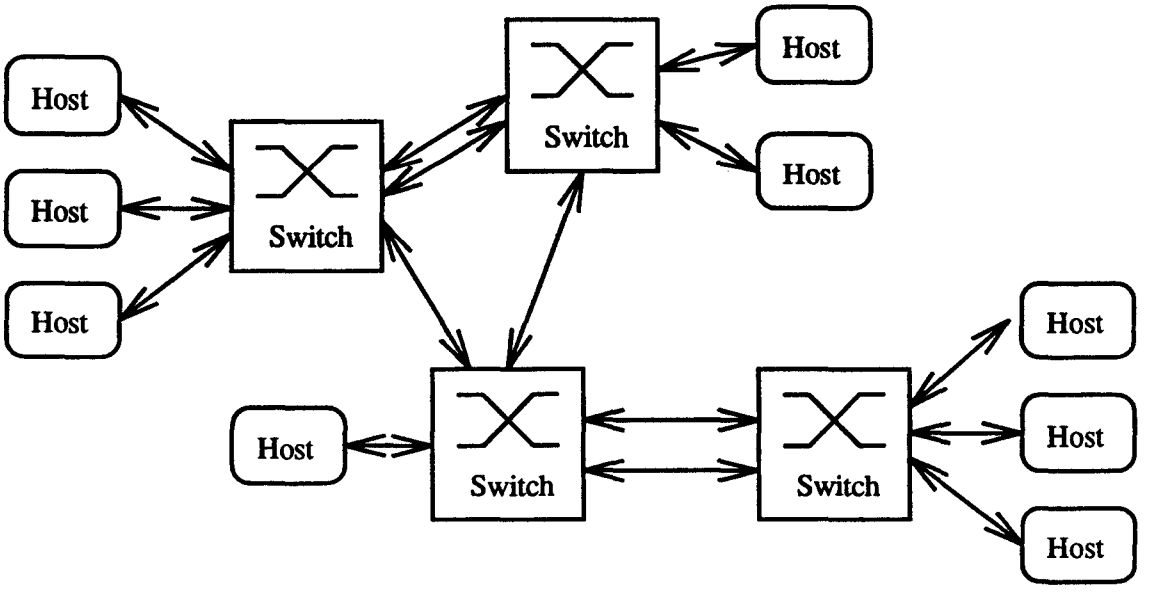

Figure 1: An arbitrary mesh ATM network

The cost of each physical connection in such an architecture is the sum of the various component costs:

- Host interface card
- Host transceiver, cable and switch transceiver
- Port interface card
- A portion of the switch fabric, enclosure, power supplies, etc.

It is impossible to say if the cost of any one component will dominate, but it is clear that significant efforts towards reducing costs must be made if ATM networks are to become as affordable as current technologies. In the remainder of this section we note how some of these costs can be avoided or else shared among many users.

2.1 Host interface card

Very high speed ATM interface cards will need to perform segmentation and reassembly via hardware. With the STS-12c line rate of 622 Mbit/s, a 53-byte cell will arrive approximately every 0.7 microseconds. While this per-cell time may be within the processing capabilities of a workstation CPU, the problem is the interrupt latency within UNIX workstations, which can be of the order of 20 microseconds or more. Consequently, a dedicated processor or some custom hardware is needed to perform the SaR function, and to interrupt the host processor only when a complete AAL5 protocol data unit (PDU) has been assembled in memory [5, 6].

We observe that the real goal of an ATM LAN is to exchange AAL5 PDUs between machines, not ATM cells. Thus we propose that a variable length AAL5 PDU should be the basic unit of transmission. In other words, our network does not use asynchronous transfer mode. The benefit is that no SaR function is required and thus the interface should be cheaper. The price to be paid is potentially greater complexity in the switch, and a potential reduction in the quality of service that the network can provide.

Given that our network transfers AAL5 PDUs up to the maximum size of 64 Kbytes, it becomes feasible to use the host processor to manage the transfer of all PDUs. In particular, applications that transfer large amounts of data will benefit from the provision of large PDUs at the lowest layers.

2.2 Transceiver and cable costs

Much of the cost of any network is that of cables and transceivers. Very high speed networks either require optical fibres and associated laser drivers, or else use

co-axial cable over short distances of 20 to 30 meters. In addition, the encoding, serializing, deserializing and decoding functions must be done by extremely fast circuits which, though available off-the-shelf, still cost hundreds of dollars.

An ATM connection provides a host with dedicated transmit and receive lines to the switch. Consequently each host requires two sets of transceivers, one in the host and one in the switch. A substantial reduction in cost can be achieved by eliminating one of these transceivers.

2.3 Interface card and switch fabric costs

Currently, the cost of the switch represents a substantial part of the cost of each connection. Every host must incur the cost of one interface card in the switch together with at least 1/Nth the cost of the rest of the switch i.e. excluding interface cards. Note that it is at least 1/Nth of the cost since some switch ports may be connected to other switches.

The obvious way to reduce this cost is to allow two or more hosts to share a switch port. This has the inevitable effect that the line rate is now shared among several hosts. In practice this is unlikely to be much of a limitation because, given a 1Gbit/s line rate any host is likely to gain almost instant access to the network. One added complication is that a medium access control protocol is required in order to regulate the use of the shared medium. It is essential that this MAC protocol be simple so that it can be implemented at low cost.

We note that the use of a shared medium is independent of the use of variable-size frames, and there is no reason why conventional 53-byte ATM cells could not be used together with a shared-access network. Many LAN access protocols such as S++ [7], Metaring [8], or CRMA [9] support fixed-size cells, while the various Cambridge ring networks [10] could be claimed to be early examples of such an approach. This would mean that a conventional ATM switch could be used and each port could interconnect several hosts. The only penalty is the expense incurred for hardware to perform SaR functions at the hosts.

We are not the first to propose the use of a switch-based network that transfers variable-size frames. In particular the IBM plaNET/ORBIT project [11] uses a similar approach, where a local distribution LAN is used to connect host computers to the switch. The IBM network supported a variety of addressing modes such as source routing and label switching and was not specifically focussed on a particular standard. Our proposal is based around the provision of an AAL5 service at low cost while delivering very high end-to-end performance between workstations.

In the next section we present a proposal for a network which is designed to provide a high-speed AAL5 service at a very low per-connection cost.

3 The Jetstream network

The Jetstream LAN uses a single unidirectional ring topology to provide low cost access to switch ports, as shown in figure 2. The network is a combination of dedicated and shared links. Each switch port may connect to a ring of up to 15 hosts or to another switch. A switch port has the necessary intelligence to distinguish the various cases. The network specification limits the number of hosts per port to 15 and also limits the total ring length to 1km. These limits are imposed in order to provide certain service guarantees. These limits should not be too onerous because the ring is acting as a local distribution network to interconnect the host computers in a local work group.

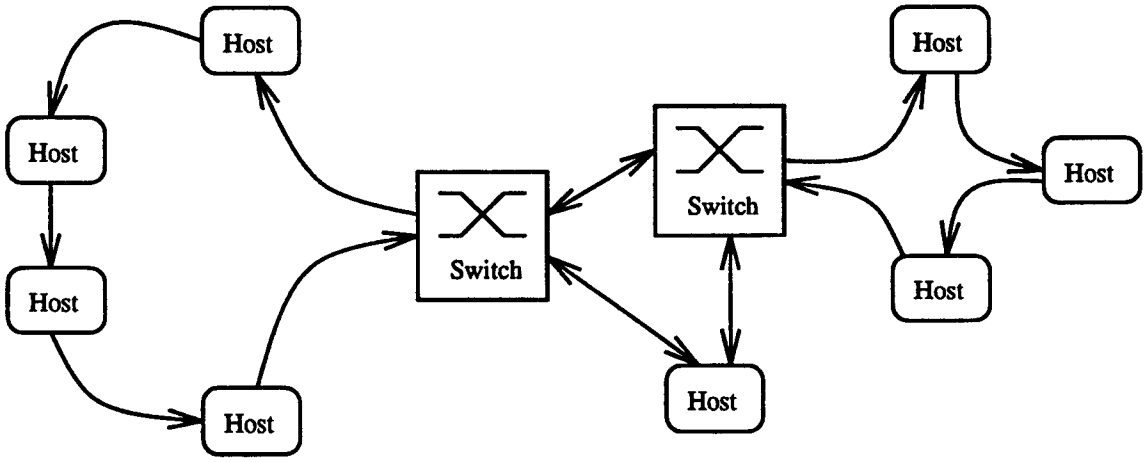

Figure 2: A Jetstream network consists of shared access rings together with individual links for critical connections

The use of a single unidirectional ring is dictated by our goal of a low cost system. The ring provides connectivity at the minimal expense of one receiver and one transmitter per node. The disadvantage of this topology is the lack of redundancy for fault tolerance, and we comment on this issue in section 3.5.

If a switch port is connected to two or more hosts the switch has the responsibility for establishing the integrity of the physical layer, and for starting the MAC protocol. The MAC protocol supports half-duplex transmission, so that only one host may transmit at any instant. If a switch port is connected to only a single device, whether it be a host or another switch, then that link can be used in full duplex mode.

In this paper we present the design and implementation of the ring part of the network, not the switch. A ring LAN can be used on its own, without a switch and this is equivalent to a conventional shared-medium LAN. The choices of variable length packets and a shared medium topology have important ramifications for the

switch architecture, and these are considered in section 3.5. In the remainder of this section the MAC Protocol, PDU format, initialisation, and implementation are examined in some detail.

3.1 The Jetstream MAC protocol

A MAC protocol is required whenever two or more hosts are connected to the same ring. Our requirements for the MAC protocol are twofold:

- it must be amenable to a low-cost implementation,
- it must provide support for guaranteed bandwidth and bounded access delays.

We opted to use the core elements of the FDDI MAC protocol [12], since it is well proven and operates on a ring topology. Much of the FDDI protocol was deemed to be unnecessary, and would only increase the cost of the implementation. Consequently our protocol only uses the token rotation protocol aspects of FDDI.

Two classes of service can be supported: asynchronous and the so-called synchronous service. The asynchronous service provides round-robin access to the ring and although access delays are bounded they may be quite large. The synchronous class of service is supported by the timed token rotation protocol. This ensures that a host that requires a synchronous service will receive some guaranteed bandwidth on every token rotation, and that the token rotation time is strictly bounded.

Strict limits are imposed on the number of hosts and the maximum size of the ring. This enables the access network to provide certain guarantees of service for the synchronous traffic. The timed token rotation protocol is based around the notion of a target token rotation time (TTRT). The TTRT will be the mean token rotation time, as observed by any host, if the load offered to the network is large. The protocol then ensures that the maximum token rotation time will be less than twice the value of TTRT. A corollary of this behaviour is that the token is deemed to be lost if it does not visit every node at least once every 2 x TTRT seconds.

During a token rotation of duration TTRT, the time for useful transmissions will be $T_{\text{transmit}} = \text{TTRT} - T_{\text{prop}}$, where T_{prop} is the propagation delay of the token around the ring, including any latency within each host interface, and including the time required to pass on the token.

This transmission time, T_{transmit}, must be divided between synchronous and asynchronous traffic. Also, once a node has started to send an asynchronous PDU then it may continue to do so, and this must be allowed for as an "overrun" factor. A node must not send a synchronous PDU unless it can complete the transmission

within the time allotted to that node for synchronous traffic. Thus, the transmission time can be expressed as: $T_{\text{transmit}} = T_{\text{asynch}} + T_{\text{synch}} + T_{\text{overrun}}$.

Or, TTRT $= T_{\text{asynch}} + T_{\text{synch}} + T_{\text{overrun}} + T_{\text{prop}}$.

The network specifies a maximum ring length of 1 km. Assuming a one microsecond delay through each of up to 16 interfaces (one switch plus 15 hosts) then the propagation delay T_{prop} will be about 21 microseconds.

The physical layer of our prototype offers a nominal transmission rate of 960Mbit/s. The maximum physical layer frame is slightly larger than 64 Kbytes, which supports a maximum size AAL5 PDU. Thus the overrun time is less than 700 microseconds. The use of a TTRT of one millisecond means that T_{synch} can be as large as 300 microseconds, ignoring T_{prop}. This corresponds to approximately 300 Mbit/s of synchronous bandwidth with a maximum access delay of two milliseconds. We believe this will be sufficient for most 15-host networks for some time. The maximum access delay for asynchronous traffic is approximately (N-1)TTRT, where N is the number of nodes, so this will be about 15 milliseconds.

3.2 Frame format

The Jetstream network has been designed to transfer AAL5 PDUs with a minimum of overhead. The MAC frame format is shown in figure 3, for both an AAL5 frame and a generic frame. The AAL5 frame has three distinct sections: a Jetstream header, a B-ISDN header, and an AAL5 PDU. Note that the network does not specify an AAL5 PDU – any AAL PDU may be used provided that its length is less than or equal to that of a maximum length AAL5 PDU. This is important as there is interest in other AALs which provide service guarantees appropriate for multimedia traffic. However, Jetstream has been designed to optimise support for AAL5 as will be seen later.

The AAL5 PDU section conforms exactly to the format specified by the AAL5 protocol, with control, length and CRC-32 fields in the trailer. The cyclic redundancy check (CRC) provides error detection capabilities over the large PDU. One concern with AAL5 over ATM is whether the CRC will detect a missing cell (though this should be detected by the length indicator) [13]. This cannot occur with Jetstream because PDUs are sent as integral units.

The B-ISDN header conforms exactly to the format specified by the CCITT. In the case of AAL5 the 'user' bit in the B-ISDN payload type field will always be set to indicate the last cell of a PDU, since there will only ever be one 'cell' per PDU.

We have opted to maintain the B-ISDN header for several reasons. The most important is compatibility with B-ISDN ATM networks. B-ISDN ATM networks

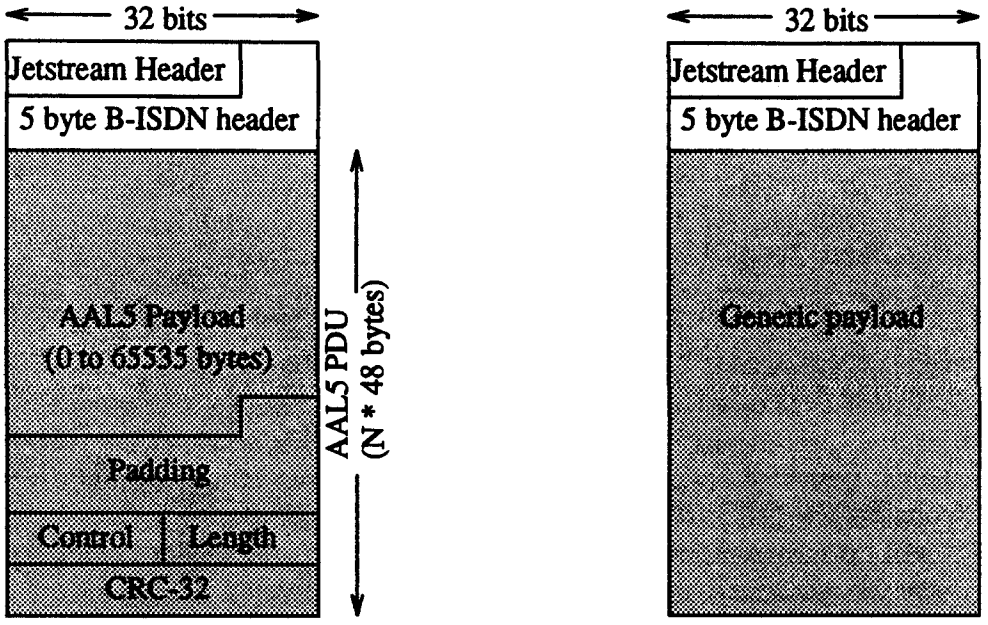

Figure 3: The format of a Jetstream MAC frame

exchange 53 byte cells which appear to a Jetstream network as a particular case of a general Jetstream PDU. Thus, an ATM cell is a valid Jetstream PDU, and so the Jetstream network treats ATM cells in the same way as all other sizes of Jetstream PDUs. This provides a significant benefit in that it is almost trivial to interface Jetstream to an ATM network.

Figure 4 illustrates a Jetstream network connected at one point to an ATM network. The Jetstream/ATM interface unit must convert cells from one network format to the format of the other network. Cells that pass from the ATM network to the Jetstream network require trivial processing – a constant Jetstream header must be prefixed. Cells that pass from the Jetstream network to the ATM network must be segmented into 53-byte ATM cells. This simply involves segmenting the AAL portion of the network into 48-byte cells and prefixing these with a copy of the B-ISDN header that is already present. This operation will be very simple for AAL5 but will be more complex for AAL3/4.

One alternative to using the B-ISDN header would be to use a traditional MAC header such as that used by the IEEE 802.3 or the ANSI FDDI LANs in which each PDU has a 48-bit source and destination address. We believe that the use of B-ISDN virtual channels is superior to this scheme because it provides greater flexibility for the end-systems, and because routing will be simpler.

The Jetstream header format is illustrated in figure 5. The CRC-present bit is set

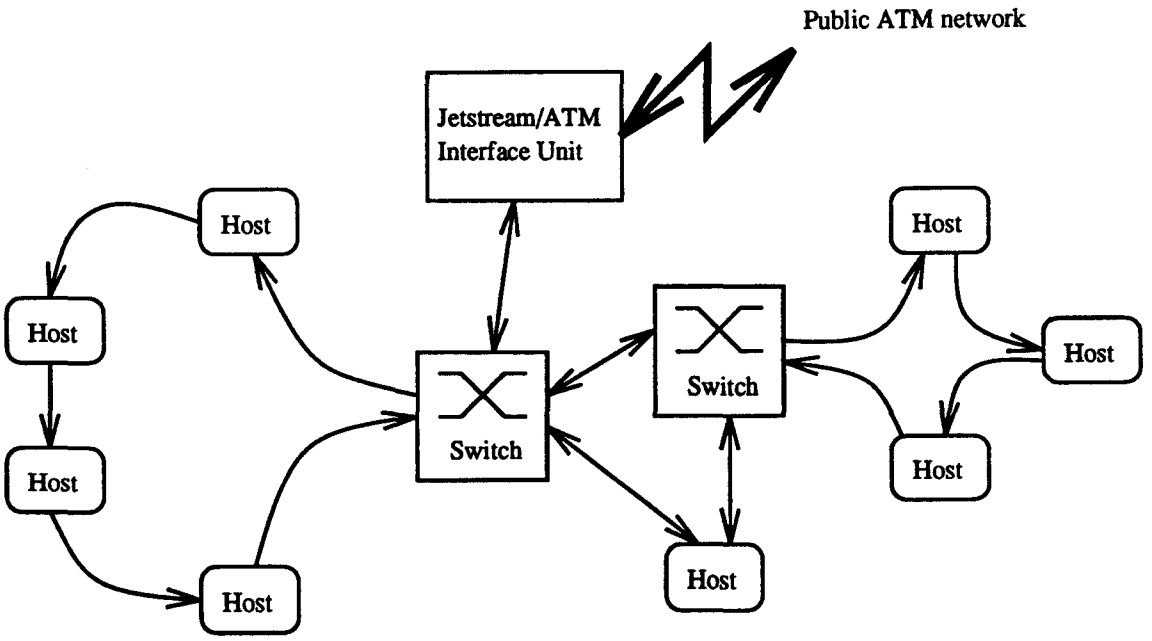

Figure 4: A special interface unit is required to connect Jetstream to the public ATM network

by the host to indicate that an AAL5 CRC-32 should be calculated over the AAL5 PDU and inserted at the end of the PDU. At the destination, an AAL5 CRC-32 is always calculated and appended to the incoming PDU, and is simply ignored if the frame does not contain an AAL5 PDU. The Jetstream header is always three bytes so that the AAL5 PDU is aligned on a 32-bit boundary; this simplifies the task of calculating the 32 bit CRC.

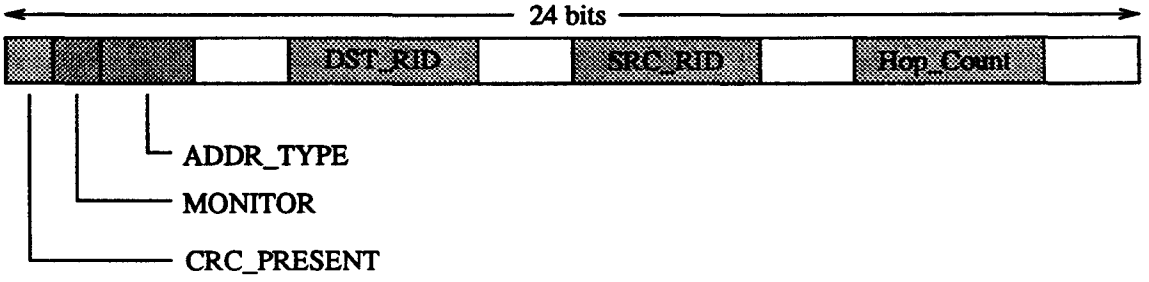

Figure 5: The Jetstream header

The monitor bit is used to detect corrupted frames. Within each Jetstream ring the switch port acts as the unique monitor node. The monitor sets the monitor bit to one in every frame that passes. The monitor then strips any frame that passes with the monitor bit already set to one. This prevents corrupted frames from circulating endlessly. The same technique has been used in other ring networks.

The DST_RID and SRC_RID provide an absolute addressing scheme that is significant only within the local Jetstream ring. Every host is assigned a Ring ID (RID) during the initialisation process described below. The ADDR_TYPE field indicates the address type associated with each frame. Currently the ADDR_TYPE may be set to one of the two values: RID or VC. When ADDR_TYPE is set to RID then the frame contains a source RID and a destination RID, and the node that matches the destination RID will read the frame. The RID addresses are provided so that management operations can proceed when no virtual channels have been established. When ADDR_TYPE is set to VC, a node compares the VPI/VCI field in the B-ISDN header with a local list of active VCs to determine whether to read the frame.

Jetstream uses source stripping to remove frames from the ring. A node strips an incoming frame if it recognises its own RID in the SRC_RID field.

The RID is important for another reason. One fundamental difference between an ATM node and a Jetstream node is that a Jetstream node may see frames which are destined for other nodes. Thus virtual channel identifiers must be unique within the Jetstream ring, and this can be assured by using the RID as four bits of the virtual channel identifier.

3.3 Ring Initialisation

In this section we describe how a Jetstream network is initialised, either at power-up or when a node is added or removed from the network. The initialisation process is complicated by the single unidirectional ring topology – nodes do not have duplex links with their neighbours, and so it is difficult to know when the entire ring is operational.

The ring is built from unidirectional point-to-point links, and the integrity of the entire ring must be established before the MAC protocol is started. This leads to a two-phase initialisation process. In the first phase a master node decides when every link is operational. In the second phase the master node allocates ring IDs and starts the protocol. The master node would be the switch port if a switch is present, otherwise it would be a pre-selected member of the LAN.

The first phase exploits the signaling capabilities of the physical layer. The Jetstream physical layer has been implemented using Hewlett Packard HDMP-1000 Gbit/s transmitter/receivers. These devices can send two special words, FW0 and FW1, which are out-of-band symbols and thus distinguished from ordinary data symbols. At power-up all nodes transmit FW0. Host nodes wait until they receive FW1 before they transmit FW1. When the master node receives FW0 then it sends FW1 to see if all nodes are up. If the master node does not receive FW1 within a certain short period then it assumes that the ring is not up, and starts the process

again with FW0. When the master node receives FW1 it knows that all links are operational. This completes phase one.

In phase two the master node transmits a RING_UP message which contains a Hop_Count value of zero. The Hop_Count field is incremented when the RING_UP message arrives at each node. Every node is initially in promiscuous mode and thus receives the RING_UP message. The Hop_Count field is interpreted as the ring ID for each node so that the node immediately downstream from the master node will acquire a RID of 1, the next node acquires a RID of 2, etc. Thus the ring configuration is achieved with a single message.

When the master node receives its RING_UP message the Hop_Count field indicates the number of hosts that are connected to the ring and the master node then starts the MAC protocol by issuing a token. If the token is ever lost then the master node will re-issue a token. The master node detects the loss of the token within two milliseconds, assuming a TTRT of one millisecond, and issues a new token within the interrupt response time – typically a few tens of microseconds.

3.4 Implementation of the host interface card

While a prototype cannot achieve the very low costs associated with a mass produced product, we have developed a card which suggests that a production version would be extremely cost-competitive with other networks of similar performance.

The prototype is a host interface card that will attach to an HP series 700 workstation. All components are commercially available and all the functions such as the MAC protocol processing and the generation of the AAL5 CRC are implemented in two field programmable gate arrays. There is no processor on the card.

The card provides the following features:

- Implementation of the timed-token rotation protocol,
- Generation of the CRC-32 over the AAL5 PDU,
- Auto-increment of the Hop_Count field to simplify initialisation,
- A 15-bit VC space implemented as a simple RAM look-up,
- Numerous statistics counters,
- An interface to the host processor,
- Support for the TCP checksum for inbound frames.

The prototype does not support the synchronous service because the host interface does not provide a synchronous data service. However, the current implementation will work correctly with future nodes that do exploit the synchronous protocol.

The AAL5 CRC-32 is generated in parallel, 32 bits at a time at 30MHz. The algorithm to perform this is well known [14, 15] but requires the logical equivalent of several hundred 2-input XOR gates. To minimise the cost only one CRC-32 generator is used and this is shared between outbound and inbound data paths. A CRC-32 is generated over the frame, and the final value is always appended to an inbound frame and is optionally appended to an outbound frame. Figure 6 shows the architecture of the interface card.

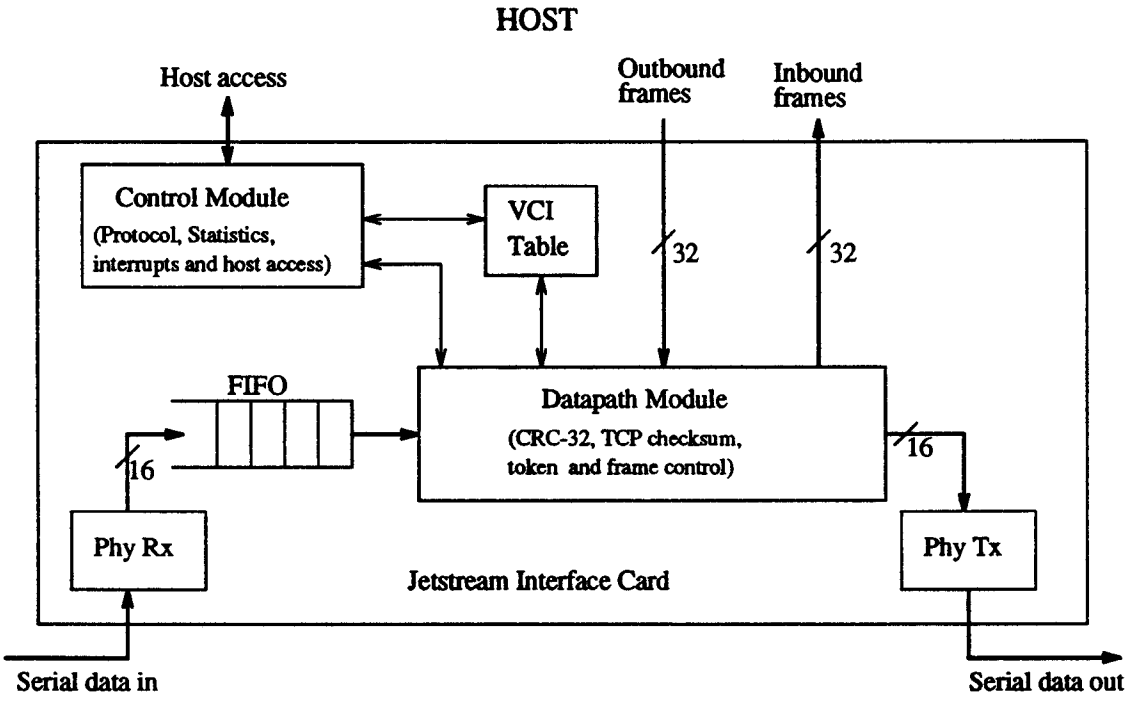

Figure 6: The architecture of the Jetstream card

The FIFO is used as an elastic buffer to compensate for differences in clock rates between adjacent nodes. The node datapath module waits until the FIFO contains four 16-bit words before starting to read; thereafter the node can safely read a word from the FIFO on every clock tick without encountering underrun. Overrun cannot occur because nodes cannot transmit back-to-back packets for indefinite periods of time. Each node introduces a latency of 300 nanoseconds to every frame that passes through from its receiver to its transmitter.

The datapath module is implemented as a programmable gate array and provides all functions that operate on frame data. The datapath module accepts 16-bit words at 60MHz from the ring, and transmits them to the ring at a similar rate.

Internally the module uses 32-bit paths, and the two host data interfaces are also 32 bits wide. The control module is implemented as a second programmable gate array, and provides the protocol functions as well as all statistics and the host interface.

The total device count is about 25 ICs, most of which are simple registers that provide retiming between fast and slow clock domains or conversion between TTL and ECL. The component cost, including board, connectors and all devices, is less than $2,000, assuming the use of a co-axial connection. One ASIC could reduce the total device count to about five ICs, and greatly reduce the total cost.

For software, the Jetstream prototype will initially provide permanent virtual circuits for IP and associated protocols such as TCP and UDP. We plan to develop a simple interface at the AAL5 layer that will support direct access to the card by both user applications and experimental protocols. Signaling protocols and address resolution protocols will be initially proprietary and minimal but we expect to follow ATM Forum recommendations with Q93B when implementations become available.

3.5 The ramifications of a shared medium and variable-size packets

To close this section we consider the effects of the Jetstream approach when compared to a conventional ATM network. Areas of interest include the quality of service (QoS), reliability, and ease-of-implementation.

One reason that ATM has had such impact is that it can offer many qualities of service with a single mechanism - cell switching. The Jetstream network switches variable-size packets, not small fixed-size cells, and so there is a question of whether Jetstream will offer the same QoS as an ATM network.

The variable cell-size has an impact on QoS at the host access point and within the switch. Relevant aspects of QoS include guaranteed bandwidth, bounded access delay, delay jitter, and burst access. Of these, Jetstream can certainly provide bandwidth guarantees equivalent to ATM; the others need to be examined more closely.

Jetstream supports a bounded access delay. The configurations described in section 3.1 indicate a worst-case access for priority traffic of about two milliseconds. While an ATM network could guarantee much smaller access delays, two milliseconds are likely to be sufficient for most applications such as video and voice.

Jetstream cannot provide guarantees of delay jitter other than the guarantees on the access delay (two milliseconds). However, ATM networks may find that the

biggest contributor to jitter is congestion within switches.

One benefit of ATM is that a host can request a certain guaranteed bandwidth, but might be able to exceed that allocation if the network has spare capacity. Jetstream can provide a similar facility: synchronous bandwidth can be allocated to a host, but the host might use some of the asynchronous bandwidth on any given token rotation.

Our conclusion is that although Jetstream cannot provide the same granularity of QoS that an ATM network might offer, it can almost certainly meet the requirements of many emerging applications.

The reliability of the system is also of great interest. A benefit of one host per switch port is that each connection is protected from faults associated with other connections. With Jetstream, a fault in the ring causes the entire ring to be lost. This is a straightforward cost decision. The ring could be replaced by a simple repeating hub that bypasses damaged links. This would be quite low-cost and might even be integrated into the switch.

The other major area of interest is the switch itself. In general, switching variable length packets is no more complex than switching cells of a fixed length, although greater buffering is likely to be required. The main difference is that it is harder to maintain a given QoS.

The problem arises when a high-priority frame arrives at a switch input port and must be transferred to an output port which is already in the process of transferring some large low-priority frame. The delay encountered by the high priority frame will depend on how the switch is constructed. For example, the switch might use a fast bus to transfer one frame at a time. Assuming a line rate of 1 Gbit/s and a bus rate of 8 Gbit/s (for an 8 port switch) then the maximum delay due to a large cell might be about 85 microseconds. However, if the switch performs space switching over eight independent channels, then the delay may be as great as 680 microseconds. Regardless of the implementation, though, we recognise that queueing delays will be longer with large packets and will have greater variance.

An alternative approach is to design the switch with a priority access path, so that a higher priority frame can temporarily suspend the transfer of a lower priority frame. This looks to be a feasible approach; one question is whether the extra expense in the switch might negate the cost savings achieved by the use of the ring.

One effect of using large frames is that the switch will almost certainly need to support cut-through routing, where the switch starts to transfer a frame before it has completely arrived, given that the output port is available. Careful thought needs to be applied to the issue of flow control to ensure that deadlock conditions do not occur.

The whole issue of switch behaviour is very hard to quantify until we can gain a better understanding of the behaviour and requirements of host applications. To gain this knowledge we believe we need to build trial systems and see what is useful and what is not.

4 System performance

A Jetstream interface card is a 'fat pipe' in the sense that it provides raw data rates of nearly 1 Gbit/s. This does not mean that user applications will obtain similar performance because there are many other potential bottlenecks between the network and the applications. We believe that the 'single-copy' approach to protocol implementation is one way to overcome these bottlenecks. In this section we describe how Jetstream is used in conjunction with another card, called Afterburner, which has been designed to enable a single-copy implementation of the network protocols commonly used in UNIX workstations.

4.1 The problem

A few years ago it was the network protocols themselves that were thought to be the bottleneck in network performance. It is now generally accepted that while a poor protocol implementation is part of the problem, other components of the system are the real obstacles. Of these, the main obstacle to high network throughput in current workstations appears to be the data copy rate of the main memory system. The details of this problem have been expounded elsewhere [16, 17], and so we present only a simple overview of the problem.

A common protocol stack that is used within UNIX workstations is shown in figure 7. The socket layer provides a simple interface to the application, and the TCP and IP protocols are used to provide a reliable end-to-end service across the Internet. Figure 7 also shows the data copies that take place in a typical implementation, and it is clear that application data is copied many times.

A proposal, made by Jacobson [18], is to move the data only once, directly from the application buffer to the network interface card. Thus, on transmit the socket layer copies the user data directly to the network card, calculating the TCP checksum as it does so. The various network headers are then added to the data on the card, and the completed packet is transmitted onto the network.

TCP provides a reliable service and so the source must maintain a copy of the user's data until an acknowledgement is received from the destination. This requires a network card that has a reasonably substantial buffer. Jacobson has proposed the WITLESS interface (Workstation Interface That is Low-cost Efficient Scalable and Stupid) [18] that provides such buffering.

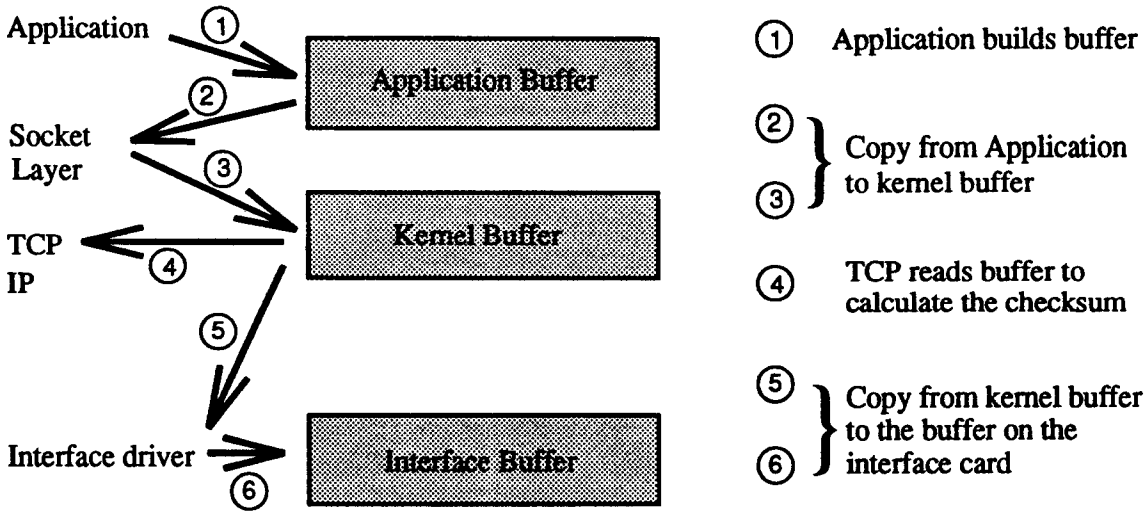

Figure 7: The protocols commonly used in a UNIX workstation. Data are often copied several times before they are transmitted

Jacobson suggests that the processor should move the data since the processor can calculate the checksum 'for free' during the copy. This is particularly true for RISC processors which can execute several instructions for each memory access. One implication of using the processor to move data is that the network buffer must appear to the processor to be normal memory. The performance of the single-copy approach is not subject to the behaviour of the application provided that the network card contains sufficient buffer memory. However, the correct size of this buffer will be determined by the behaviour of the applications, as well as the number of applications that access the network concurrently.

4.2 The Afterburner card

We have constructed a prototype card, called Afterburner, which supports a single-copy protocol stack and can transmit and receive network packets at burst rates of up to 1Gbit/s.

Afterburner was designed for specific workstations - the HP PA-RISC 9000/700 series. The practicalities of workstation engineering meant that Jacobson's ideas for a memory-based network interface had to be modified. Perhaps the biggest difference between the WITLESS model and Afterburner is that Afterburner is attached to an I/O bus, and is not part of the main memory system. The reason for this is that the main memory is designed to maximise cache-to-memory bandwidth whilst minimising the cost. Every aspect of the timing of the main memory RAMs is tightly controlled and there is no scope for providing a dual-ported portion of memory that can support accesses from both the processor and the network.

Consequently, Afterburner occupies a slot in the fast I/O bus within a 700 series workstation. This is not the EISA I/O bus, but rather the internal graphics bus which offers substantially greater throughput. The architecture of the Afterburner card is shown in figure 8.

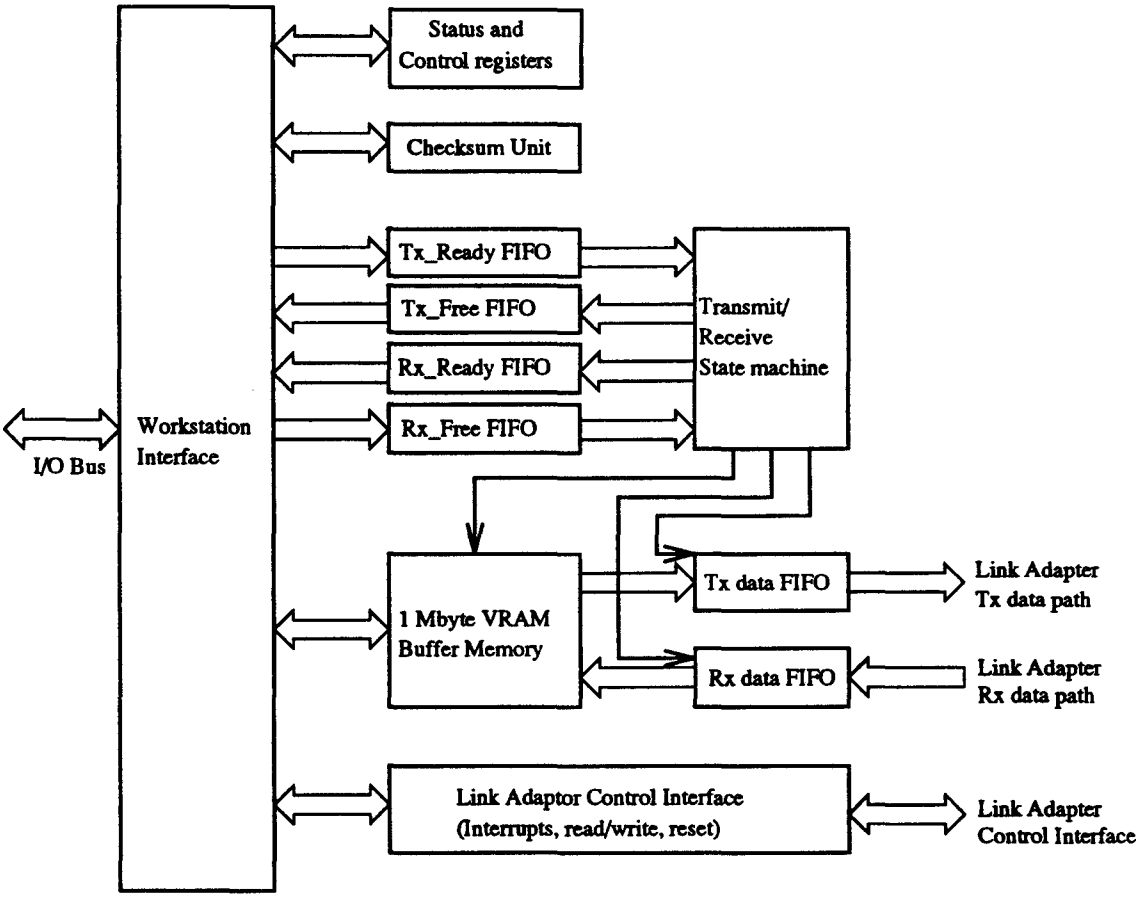

Figure 8: Afterburner Block Diagram

In order to support a single-copy stack, the card contains a one megabyte buffer. Triple-ported VRAMS (video RAMs) were used which provide one standard random access port, and two high speed serial ports. The random access port is used by the CPU while the serial ports provide a fast transmit pipe and a fast receive pipe, each 32 bits wide and capable of bursting at 33MHz. The VRAM is divided into blocks which must be the same size, but this size can be any power of two between 2 Kbytes and 64 Kbytes.

The details of the operation of Afterburner will be provided in another paper, and we provide only a very simple description here. The transmit section has two FIFO queues which contain a list of blocks that are empty (Tx_free) and blocks that are waiting to be transmitted (Tx_ready). The socket layer acquires an empty block

and copies the user data into it. The TCP checksum is calculated as the data is copied. Lower layers then build the packet header in another block. The driver then places the addresses of the full blocks into the Tx_ready FIFO and hardware on the card then manages the transmission of the packet.

The operations occur in reverse for the receive section except that the processor does not move the data itself. Instead, some special block move circuitry is used to move 32 bytes from I/O to main memory. This circuitry is not part of our cards but is included in all series 720 and 730 workstations. The effect of using the block move circuitry is that data ends up in main memory and not in cache.

Afterburner does not contain any network-specific hardware, but has been designed to be used with a variety of networks. Jetstream is just one type of link adapter that can be used with Afterburner; other link adapters under development include ATM and HIPPI.

On the receive side, the TCP checksum is calculated by the Jetstream card while the data is stored in the Afterburner buffer. This is required because TCP is called *before* the data is moved to the application, and thus before the processor can calculate a checksum. The Jetstream card provides the TCP checksum on the receive path only.

5 Performance

In this section we present measurements of the end-to-end performance of Afterburner and our single-copy implementation of TCP-IP. We consider the case of a large one-way data transfer; additional cases, such as request/response, will be covered in another paper.

The tests were performed on HP 9000/730 workstations, each with 32 MBytes of main store and one 420 Mbyte disk. The afterburner cards were connected together via ribbon cables; Jetstream was not used as we are awaiting prototype boards. The operating system was HP-UX 8.07, extended with our single-copy implementation of TCP and with RFC 1323 window scaling [19]. The benchmarks were the only active 'user' processes, but the systems had the usual assortment of daemons in the background, and were attached to the lab Ethernet LAN. The Afterburners were configured for a link rate of 1 Gbit/s and used 16 Kbyte blocks with a maximum TCP segment size of 14 Kbytes.

The measurements were made with a tool called netperf [20] which is used to measure the transfer of data from a generator process to a remote consumer process. The core of the generator program is a tight loop which repeatedly applies the socket `send()` function to the first N bytes of a suitably large buffer. The program does not alter the contents of the buffer in any way, so there are no pauses between

sends. This, together with the absence of other activity on the machine, means that the source for `send()` is always in the data cache.

Similarly, the consumer program repeatedly calls `recv()` to fetch as much data as possible into a suitably large buffer. The program does not inspect the data, so there is again no extra calculation or I/O to consider. With the other circumstances of the tests, such as buffer alignment and the particular amounts of data being moved, this means that the received data do not enter the data cache.

The throughput measurements are made by the generator program. The main loop runs for a specified interval (30 seconds), then the generator shuts down the TCP connection. Timing commences immediately before the first call to `send()`, after the connections are established, and ceases just after the shutdown, ensuring that the consumer has received all the data in the measured interval.

Figure 9 shows the TCP throughput obtained between two 9000/730 workstations (PA-RISC 1.1 @ 66 MHz). We measured the throughput for `send()` sizes of multiples of 256 bytes, using packets carrying 6 Kbytes and 14 Kbytes. A socket buffer of 192 Kbytes was used at both workstations. The test interval for each sample was 30 seconds.

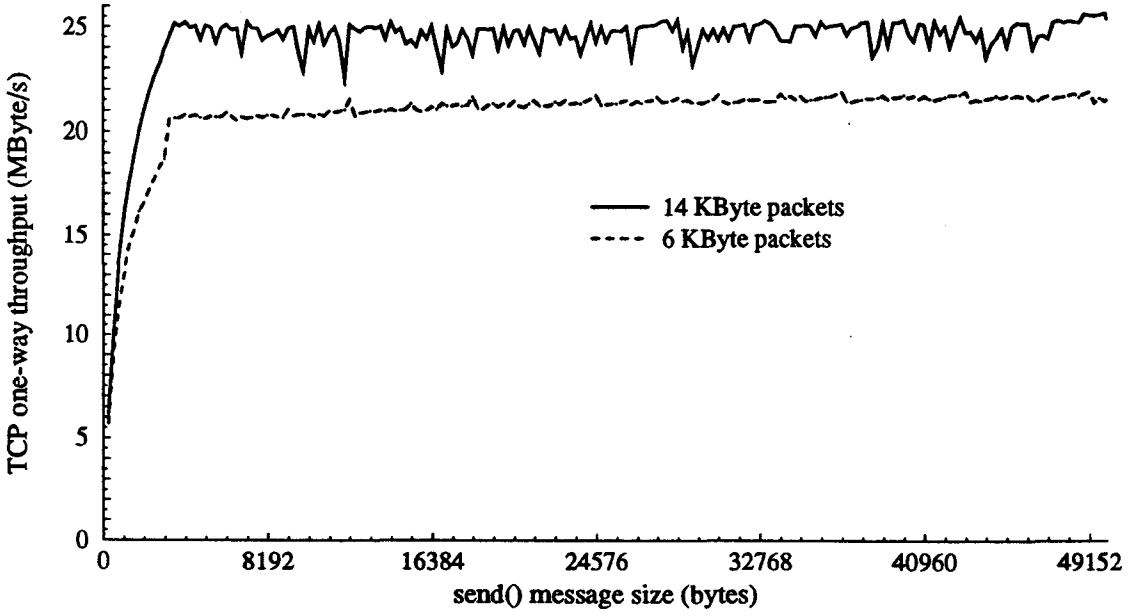

Figure 9: TCP stream throughput

For message sizes of 4 Kbytes or more, the mean measured throughput for the set of samples is 24.7 Mbyte/s (197 Mbit/s). The highest seen in that range was 25.6 Mbyte/s. For comparison, other tests show that a 9000/730 can copy data

from the Afterburner card to main store at a peak rate of 32.3 MByte/s, and it is this that limits performance. The use of 14 Kbyte PDUs shows a 20% increase in throughput over the use of 6 Kbyte PDUs. This improvement occurs because fewer packets are sent and thus there is a reduction in the total per-packet overheads such as protocol processing and interrupt handling.

The figures given above are without the Jetstream link adapter although the use of Jetstream should not change these figures by a significant amount. Increased delay will arise due to two aspects. First, the latency of a packet will increase due to the time required to acquire the token (a few microseconds under light load) plus the latency through the transmit and receive datapaths (less than one microsecond).

The second difference will arise because Jetstream allows only a single host to transmit at any instant, whereas with the back-to-back afterburners both channels can be active concurrently. However, this will only have an effect if nodes really can transmit and receive concurrently. For stream data, only one host sends significant amounts of data — the receiver simply sends acknowledgements. Hence we would expect only a minor reduction in performance when Jetstream is used.

6 Conclusions

We have presented the Jetstream network, a local network that is designed to provide industry standard protocols (AAL5 and TCP/IP) at low cost and with very high performance. Low cost has been achieved by two main changes to the conventional ATM network architecture: the first was to use a shared access distribution network which lowers per-port costs; the second was to use variable-size PDUs as the basic traffic element, thus avoiding the expense of special hardware for segmentation and reassembly.

A prototype network card has been designed for use with HP Series 700 workstations, and provides support for the AAL5 CRC and for the TCP checksum. The Jetstream interface is used in conjunction with the Afterburner card to provide a networking system that can support proven high performance single-copy protocol implementations. Experiments with HP 9000/730 workstations have shown that applications can achieve large transfers at rates up to 25 Mbyte/s.

These performance results suggest that the combination of Jetstream and Afterburner yields a low-cost network that is capable of delivering very high performance for current workstations. We believe this architecture will scale to future workstations to yield throughputs of 1 Gbit/s.

7 References

[1] CCITT, I-Series Recommendation on B-ISDN, Geneva, 1991.

[2] P. Newman, "ATM Technology for Corporate Networks", *IEEE Commun. Mag.*, Vol. 30, No. 4, pp. 90-101, April 1992.

[3] J-Y. Le Boudec, E. Port and H.L. Truong, "Flight of the FALCON", *IEEE Commun. Mag.*, Vol. 31, No.2, pp. 50-56, Feb 1993.

[4] J.N. Giacopelli, J.J. Hickey, W.S. Marcus, W.D. Sincoskie and M. Littlewood, "Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture", *IEEE Jnl. Sel. Areas in Comms*, Vol. 9, No. 8, pp.1289-1298, October 1991.

[5] B.S. Davie, "A Host-Network Interface Architecture for ATM", in Proceedings SIGCOMM 1991, Zurich, Switzerland, pp. 307-315, September, 1991.

[6] C.B.S. Traw and J.M. Smith, "A High-Performance Host Interface for ATM Networks", Proceedings SIGCOMM 1991, Zurich, Switzerland, pp. 317-325, September, 1991.

[7] G. Watson and S. Tohme, "A Performance Analysis of S++: A MAC Protocol for High Speed Networks", in Proceedings of 3rd Int'l IFIP WG6.1/6.4 Workshop on Protocols for High-Speed Networks, pp. 87-102, Stockholm, Sweden, May, 1992.

[8] I. Cidon and Y. Ofek, "Metaring - A Full Duplex Ring with Fairness and Spatial Reuse", in Proceedings IEEE INFOCOM '90, pp.969-981, June 1990.

[9] M.M. Nassehi, "CRMA: An Access Scheme for High-Speed LANs and MANs", Proceedings SUPERCOMM/ICC '90, Atlanta, GA, April 1990.

[10] A. Hopper, S. Temple, and R. Williamson, "Local Area Network Design", Addison-Wesley (pubs.), 1986.

[11] I. Cidon, I. Gopal, P.M. Gopal, J. Janniello and M. Kaplan, "The plaNET/ORBIT High-Speed Network", IBM Research Report 92A005472, August 1992

[12] Fiber Distributed Data Interface (FDDI) - Media Access Control, ISO 9314-2, 1989

[13] Z. Wang and J. Crowcroft, "SEAL Detects Cell Misordering", *IEEE Network Mag.*, VOl. 6, No. 4, pp.8-9, July 1992.

[14] G. Albertengo and R. Sisto, "Parallel CRC Generation", *IEEE Micro*, pp.63-71, October 1990.

[15] T-B. Pei and C. Zukowski, "High-Speed Parallel CRC Circuits in VLSI", *IEEE Transactions on Communications*, Vol. 40, No. 4, pp.653-657, April 1992.

[16] D.D. Clark, Van Jacobson, J. Romkey and H. Salwen, "An Analysis of TCP Processing Overhead", *IEEE Commun. Mag.*, pp. 23-29, June 1989.

[17] D. Banks and M. Prudence, "A High Performance Network Architecture for a PA-RISC Workstation", *IEEE Jnl. Sel. Areas in Comms*, Vol. 11, No. 2, Feb 1993.

[18] Van Jacobson, "Efficient Protocol Implementation", ACM SIGCOMM '90 Tutorial, September 1990.

[19] Jacobson, V.; Braden, R.; Borman, D., RFC 1323, "TCP Extensions for High Performance". 1992 May.

[20] Rick A. Jones, `netperf`: A Network Performance Benchmark, Revision 1.7 Information Networks Division, Hewlett-Packard Co., March 1993. The `netperf` utility can be obtained via anonymous ftp from `ftp.csc.liv.ac.uk` in `/hpux/Networking`