a^{I}t^{P}m: a Strategy for Integrating IP with ATM

Guru Parulkar, Douglas C. Schmidt, and Jonathan S. Turner
guru@cs.wustl.edu, schmidt@cs.wustl.edu, and jst@cs.wustl.edu
Department of Computer Science, Washington University
St. Louis, MO 63130, USA
TEL: (314) 935-6160, FAX: (314) 935-7302*

Abstract

This paper describes research on new methods and architectures that enable the synergistic combination of IP and ATM technologies. We have designed a highly scalable gigabit IP router based on an ATM core and a set of tightly coupled general-purpose processors. This a^{I}t^{P}m (pronounced "IP on ATM" or, if you prefer, "ip-attem") architecture provides flexibility in congestion control, routing, resource management, and packet scheduling.

The a^{I}t^{P}m architecture is designed to allow experimentation with, and fine tuning of, the protocols and algorithms that are expected to form the core of the next generation IP in the context of a gigabit environment. The underlying multi-CPU embedded system will ensure that there are enough CPU and memory cycles to perform all IP packet processing at gigabit rates. We believe that the a^{I}t^{P}m architecture will not only lead to a scalable high-performance gigabit IP router technology, but will also demonstrate that IP and ATM technologies can be mutually supportive.

1 Introduction

The Internet protocol suite provides the foundation for the current data communications infrastructure in the United States and much of the rest of the world. The IP protocols have proven to be very flexible and have been deployed widely over the past two decades. As technology makes it possible to communicate at gigabit speeds, it is essential to create scalable, high-performance routers that implement IP protocols. In the past ten years, *Asynchronous Transfer Mode* (ATM) technology has emerged as a key component of next generation networks. ATM offers unprecedented scalability and cost/performance, as well as the ability to reserve network resources for real-time oriented traffic and support for multipoint communication.

Although IP and ATM often have been viewed as competitors, we believe their complementary strengths and limitations form a natural alliance that combines the best aspects of both technologies. For instance, one limitation of ATM networks has been the relatively large gap between the speed of the network data paths and the control operations needed to configure those data paths to meet changing user needs. IP's greatest strength, on the other hand, is its inherent flexibility and its capacity to adapt rapidly to changing conditions. These complementary strengths and limitations make it natural to combine IP with ATM to obtain the best that each has to offer.

This paper describes our research on new methods and architectures for achieving the synergistic combination of IP and ATM technologies. We have designed a highly scalable gigabit IP router based on an ATM core. This a^{I}t^{P}m router integrates the following core architecture components:

- A gigabit ATM switching fabric that is highly scalable in terms of the number of ports and provides optimal hardware support for multicasting [19, 20];
- A multi-CPU embedded system that includes a string of ATM Port Interconnect Controllers (APICs) [4, 5, 6] and allows flexible and high-performance IP packet processing in software.
- A distributed software system capable of forwarding IP packets at gigabit data rates on the ATM substrate and configuring that substrate dynamically to provide efficient handling of IP packet streams.

The paper is organized as follows: Section 2 outlines the hardware and software architecture of the a^{I}t^{P}m router; Section 3 describes how packet processing is carried out; Section 4 describes how various other Internet protocols (such as ICMP, IGMP, and IP version 6) are supported by an a^{I}t^{P}m router; Section 5 compares the a^{I}t^{P}m approach with related work; and Section 6 presents concluding remarks.

2 Architecture of the Gigibit a^{I}t^{P}m Router

2.1 System Overview

An overview of the a^{I}t^{P}m router architecture is shown in Figure 1. Each router is designed using ATM switch and host interface components. These components form a substrate that links a set of IP *Processing Elements* (IPPE). IPPEs handle the IP packet processing and directly control the ATM substrate. The IPPEs are general-purpose processors implementing flexible routing and queuing strategies that are central to high-performance IP networks.

Each router has a number of high-speed ports (1.2 Gb/s) equipped with routing cards that implement the required IP functionality. The main data path of each routing card passes through a sequence of ATM Port Interconnect (APIC) chips. The APIC chip is an extensible, high-performance network interface chip designed to interface directly to the main memory bus of a high-performance computing system. The APIC supports zero-copy semantics, so that no copying is required to deliver data from the network to an application.

*This work was supported in part by Ascom-Timeplex, Bay Networks, BNR, NEC, NTT, Southwestern Bell, and Textronix.

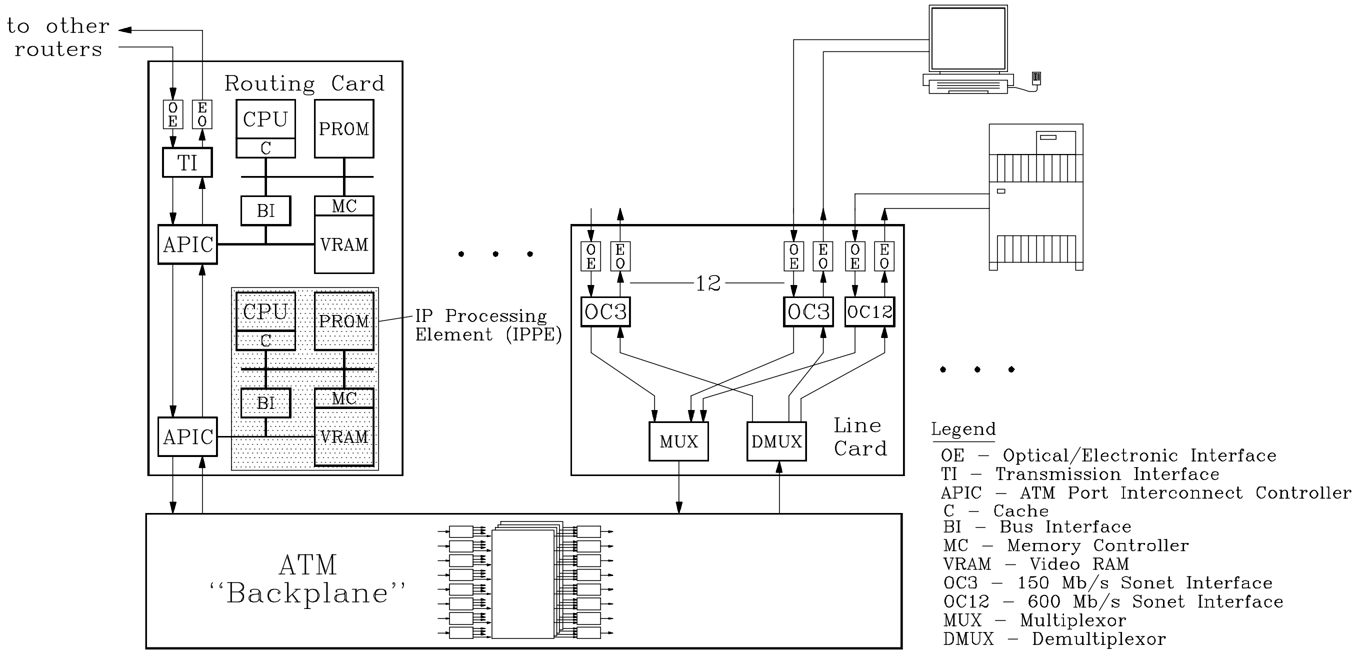

Figure 1: Gigabit a^{I}t^{P}m Router

The gigabit a^{I}t^{P}m router uses a chain of APICs to support a set of embedded microprocessors that perform IP processing.

Each router may also have line cards that support interfaces to workstations, servers and other IP routers (to reach machines on conventional LANs and on the Internet). A line card has 12 interfaces at 150 Mb/s and one at 600 Mb/s. These streams are multiplexed together at the cell level and sent through the ATM backplane. Line cards interfacing directly to shared access LANs (such as Ethernet and FDDI) are also possible.

The basic operation of the router is illustrated in Figure 2. This figure shows an IP datagram being sent from the workstation labeled A across an ATM permanent virtual circuit to the IP processing element labeled B. At B, an IP routing table lookup is done to determine the router's output port where the packet should be sent. The packet is then forwarded to that output port (C). At C, the packet is queued for transmission on the outgoing link and then sent on its way.

The scenario in Figure 2 routes the packet using two passes through the ATM switch. The second pass is required since the line cards have no dedicated IP processing capability. While such functionality could be included in the line cards, the added cost may not be justified by the performance improvement. In particular, the ATM backplane imposes only a small latency for the extra pass ($\approx 10\mu$s).

In some cases, only one pass through the ATM switch is required. For example, one pass is possible when a packet arrives at an input port connected directly to a routing card, and it is destined for an output port connected to another routing card. The second pass may be eliminated in other cases, as well. For example, in the scenario described above, the routing table lookup done at B may determine that the outgoing link connected to B's routing card is the best choice. In this case, the packet is queued for transmission at B, and the extra pass through the switch is skipped.

The a^{I}t^{P}m router architecture permits the IP layer to dynamically configure ATM virtual circuits to optimize the handling of user information flows. To achieve this, we introduce a "cut-through" mechanism to allow more efficient handling of bursts consisting of many packets. For example, suppose an application starts transmitting a file and the resulting stream of packets arrives at the router in Figure 2 on the external link connected to router card B. The IPPE handling the packet stream responds by queueing the packets temporarily while it determines the best outgoing link. It then sends control cells to the embedded ATM switch, instructing it to configure a new virtual circuit on the selected outgoing link. Packets received for subsequent transmission are queued on the newly configured virtual circuit. After the buffer for this flow has been completely flushed, it may instruct the APIC to forward subsequent cells from this flow directly along the virtual circuit, without passing through the IPPE. This approach allows the IP layer to maintain full control over individual packet flows and modify its decisions as necessary, while amortizing the cost of the more complex decision-making processes over many IP packets. The design of the cut-through mechanisms is discussed further in Section 3.

The a^{I}t^{P}m architecture also enables the IP layer to configure multicast virtual circuits by allowing IP multicast to be implemented directly at the hardware level. Consequently, IP multicast applications such as the MBONE can be supported in a highly scalable fashion. This makes it possible for a very large number of multicast applications to operate on the network at the same time.

2.2 Hardware Architecture

An overview of the a^{I}t^{P}m system architecture was described above. This section discusses the various hardware subsystems and components that are used to implement the a^{I}t^{P}m architecture. One central component is the ATM Port Interconnect Chip (APIC). The APIC has been designed as a flexible, extensible, and high-performance ATM host interface chip [4, 5, 6]. It provides direct support for segmentation and reassembly, efficient data transfer across a host processor's memory bus, support for zero-copy to and from an application's address space, and pacing of cell flows for individual virtual circuits. The APIC is designed with two bidirectional ATM interfaces. This design allows multiple APICs to be chained together conveniently, as in the router card application described above.

A block diagram of the APIC is shown in Figure 3. The inputs at the top left are parallel interfaces that transfer ATM cells according

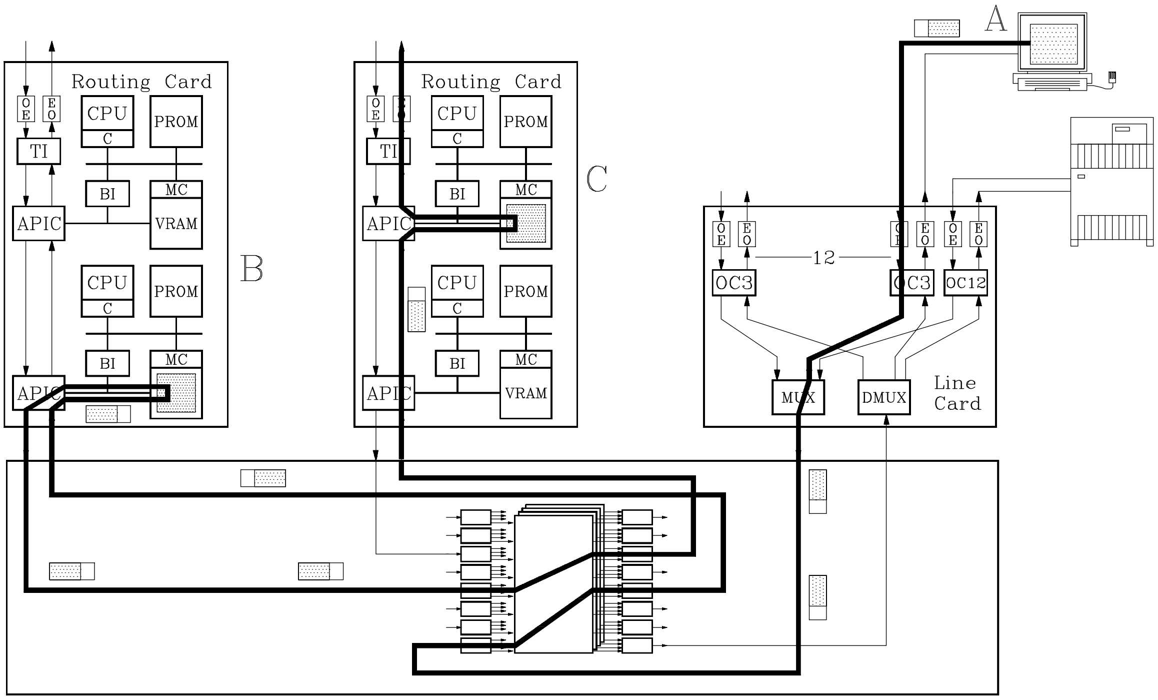

Figure 2: Example Showing Data Flow

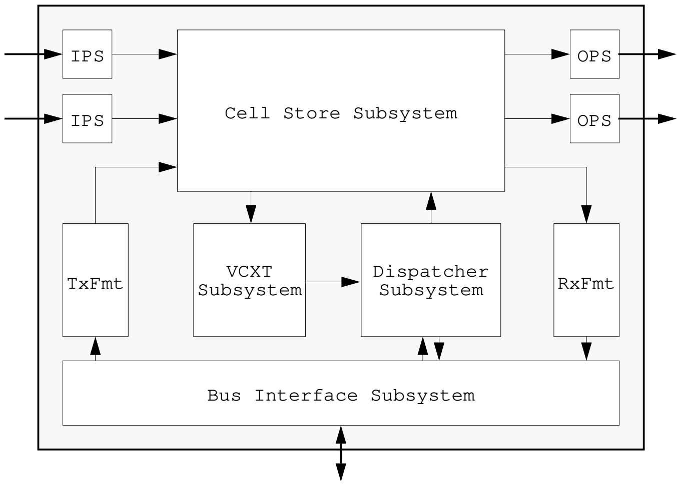

Figure 3: ATM Port Interconnect Chip (APIC)

to the Utopia [2] interface standard. This standard specifies how to connect SONET transmission devices to ATM switches and host interface circuits. The output interfaces at the top right are similar. The bus interface at the bottom of the figure is 64 bits wide. It is designed to be directly compatible with the Sun Mbus specification [18], but may be adapted to other memory buses with auxiliary logic.

Cells arriving from either of the Utopia inputs are placed in the central cell store (its capacity is 256 cells) via the VCXT subsystem. This subsystem performs a table lookup to determine how to process a cell. Cells may be forwarded directly to one of the outgoing Utopia interfaces or they may be directed to the external memory interface (EMI). A single cell may be directed to both. The output framers (OF0 and OF1) and the RX-CRC block schedule the transmission of cells to either of the two Utopia outputs and to the EMI respectively. The pacer is responsible for cell pacing for all active connections to ensure that cells are transmitted at the appropriate rate from the local external memory to the Utopia outputs. For cells directed to the local external memory, the EMI also provides address information and batches multiple cells together to achieve efficient transfer across the bus. The receive CRC block (RX-CRC) computes the AAL5 CRC as cells pass through to the EMI. The transmit CRC block (TX-CRC) computes the outgoing CRC as cells pass to the cell store.

The interfaces between the various inputs and outputs within the central cell store are completely asynchronous. This allows a wide range of link speeds to be accommodated in a straightforward fashion. The APIC permits individual Utopia ports to be configured for either 16 bit or 32 bit operation, with completely independent clocks. This enables the data paths flowing through the APIC chain to and from the core switch to operate at a higher rate than the external links. Therefore, queuing may be managed primarily within the IPPEs. In addition, the VCXT and dispatcher subsystems may cache information relating to specific packet flows. Thus, packets may be forwarded directly along the main data path without processing by the IPPE.

The second key component of the gigabit $a^{I}t^{P}m$ router is the ATM switch at its core, as illustrated in Figure 4. The system comprises a multistage switching network that implements dynamic routing of cells to evenly balance the load from all inputs and outputs over the entire network [19, 20]. The network supports gigabit operation by striping cells across four parallel planes (each cell is divided and transferred in parallel through all four planes simultaneously, minimizing latency). The network also supports an elementary ‘copy-by-two’ operation. Together with a novel cell recycling scheme, this permits an incoming cell to be copied F times with $\log_2 F$ passes through the switch. This architecture is the only nonblocking multicast switch architecture known that

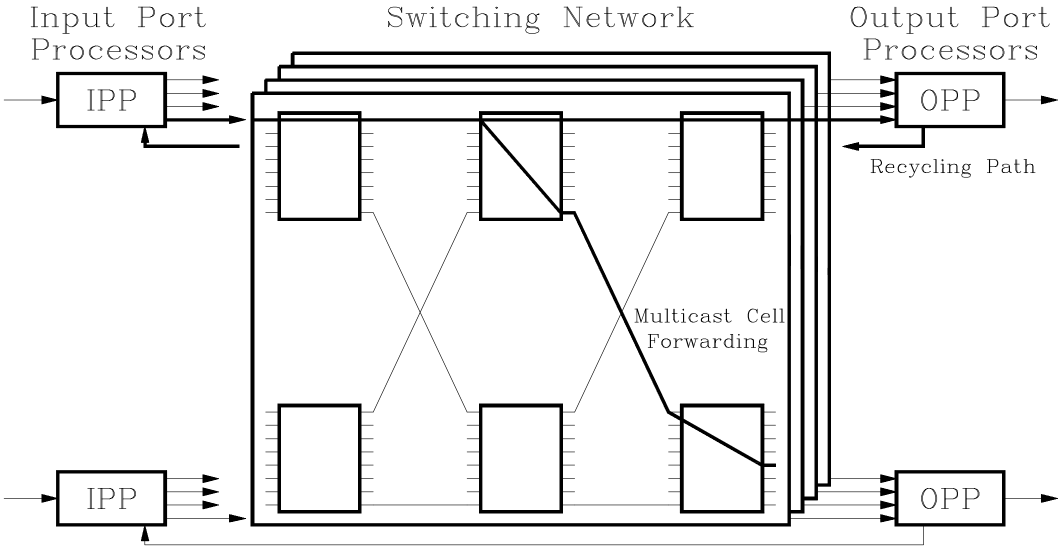

Figure 4: Gigabit ATM Switch Architecture

achieves optimal scaling with respect to interconnection network complexity, routing memory and virtual circuit modification. In large configurations it achieves order-of-magnitude cost improvements over competing multicast switch architectures. See [17] for further details. The architecture is implemented using a set of three custom integrated circuits. One circuit implements the *Switch Elements* making up the switching network. The other two circuits implement the *Port Processors* that interface to the external links and perform cell routing (using virtual circuit identifiers) and cell buffering. The Input and Output Port Processor chips, like the APIC, implement the Utopia interface. This makes it possible to directly connect an APIC to a switch port processor.

The gigabit a^{I}t^{P}m router uses the ATM switch to configure itself by sending control cells over its ATM links. Cells with a special VPI/VCI combination are interpreted as switch control cells for those ports that are enabled to receive them. The contents of their payloads determine the functions to perform, as well as the switch ports where they are to be performed. When a control cell is received from a link, it is forwarded to the port it operates upon. The target port processor then carries out the required operation. Most commonly, the requested operation is to read or write an entry in the port's virtual circuit routing table. In addition, this same mechanism may be used to configure certain hardware options or to access cell counters (these are maintained on both a link and per virtual circuit basis) or other status registers. In typical ATM switch applications there would be a single processor managing the switch resources. However, there is no intrinsic reason why these resources cannot be managed in a distributed fashion by a collection of IPPEs that setup and modify virtual circuits quasi-independently. The IPPEs inform each other of their resource usage and respond cooperatively when conflicts arise. Therefore, it is possible to distribute control in such a way that non-conflict-producing decisions are very fast, allowing virtual circuits to be established in under 100 microseconds. For a ten megabyte information burst (transferred at 1.2 Gb/s per second), the virtual circuit could be established before the first .2% of the burst has been received. Thus, the IPPE would explicitly process only a small part of the entire burst.

2.3 Software Run-time Environment

The common path of IP packet forwarding is relatively simple and may be implemented in less than one hundred RISC instructions [16]. However, operating system (OS) related overheads of packet processing (such as data movement, interrupt processing, and context switching) are significant and may limit the overall packet throughput of a system. Therefore, we have selected a software run-time environment for the a^{I}t^{P}m IPPEs that minimizes OS related overheads. We are using a general-purpose Unix operating system that is tuned for the IPPE environment. Since an IPPE is not a general-purpose workstation, it is possible to "disable" the following unnecessary OS capabilities that represent sources of performance overhead:

- Demand paging is disabled by allowing the kernel code to be locked in the physical memory. This will ensure that there are no page faults during packet processing. In fact, the IP packet processing code may reside in the CPU cache, and thus, save memory and bus accesses during packet processing.
- The interrupt-driven receive interface may be replaced with a polled interface to eliminate any interrupt processing overhead. Note that the APIC also supports a polled interface that allows the APIC to signal packet reception and transmission by modifying status bits in a data structure that is shared between the APIC and kernel memory.

  Also note that the APIC allows zero-copy packet processing so there is no performance cost due to data copying.
- All daemons and other system processes that are not needed on a given IPPE are disabled on it. Since the IPPEs do not have any local disk they boot off of a boot ROM using a remote machine as the boot server.

With these modifications, the Unix environment can be nearly as efficient as a custom embedded software system. One major advantage of the Unix environment is that the software development and runtime environments are essentially the same. This greatly facilitates development, debugging, and testing of various software modules. Also, existing Unix implementations of protocols are easily ported to IPPEs.

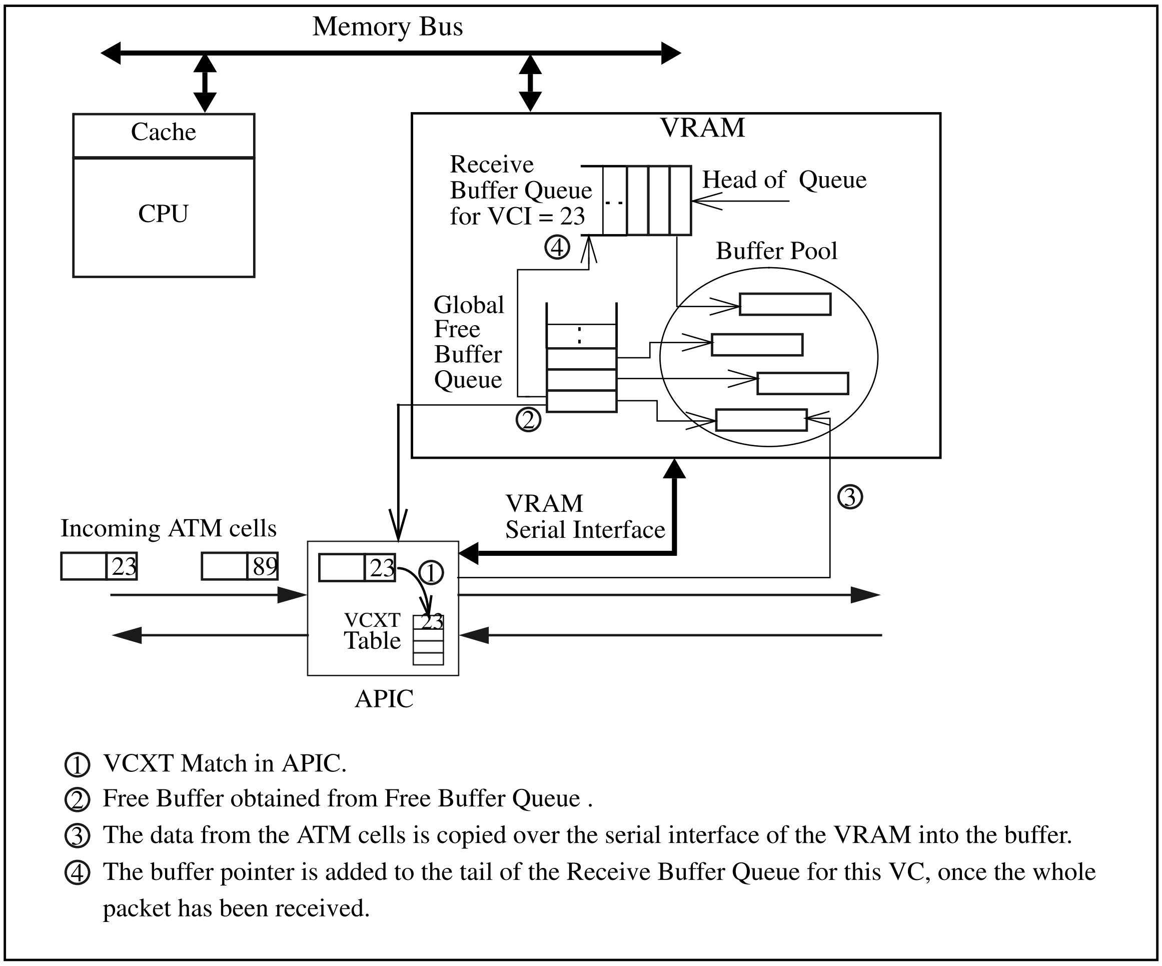

Figure 5: APIC/IPPE Data Transfer

3 Packet Forwarding in an a$^{\mathrm{I}}$t$^{\mathrm{P}}$m Router

Before discussing how packets are forwarded in the a$^{\mathrm{I}}$t$^{\mathrm{P}}$m router, we describe how the APIC moves packets between the ATM data path and the IPPE. IP packets passing along the chain of APICs are carried on several ATM virtual circuits. Different virtual circuits are used to prevent interleaving of cells belonging to separate packet flows and provide traffic isolation. Each APIC is configured with a list of VCs that are the responsibility of its attached IPPE. When a cell is received on one of these VCs, it is forwarded to the attached IPPE. Otherwise, it is passed along the APIC chain.

An APIC maintains a Connection State Block (CSB) for each of its VCs. Among other things, the CSB stores a packet chain pointer that points to a queue of packets received. It also contains a write pointer that points to a buffer indicating where to store the next cell within the IPPE's VRAM. A memory transfer is carried out in batches of cells to maximize VRAM throughput. The APIC also computes the AAL5 CRC as the cells belonging to a packet are being written into the IPPE's memory. If the CRC does not match, the packet is dropped silently. Otherwise, the APIC enqueues the packet in the receive packet queue. The APIC also adds the CSB to a linked list of CSBs (if is not already on the list) to indicates that the VC needs attention from the CPU. Therefore, the APIC does not need to interrupt the CPU for every packet.

The APIC driver executing on the IPPE processor stays in a tight loop checking for additions to the linked list of CSBs containing valid data. As soon as the APIC driver finds a new CSB, it forwards it and the associated packet(s) to the IP packet processing routine.

Packets being forwarded by the IPPE back into the ATM cell stream are handled similarly. Each active outgoing VC has an entry on a linked list of CSBs. The CSB contains a pointer to a linked list of packet descriptors that constitute the transmit packet queue for that VC. Once the IPPE enqueues a packet in this queue, it gives a grant signal to the APIC, causing it to DMA the packet (or packets) from the IPPE's memory, segment it into cells and insert the cells into the ATM cell flow.

For long messages or bursts, we propose to implement IP packet forwarding by using the ATM layer to provide a 'fast-path' that is used for most packets in a burst. The fast path is established by software in the IPPEs at the start of the burst. For short messages (in particular, messages consisting of a single packet), all packet processing is handled by the IPPE software. In this section, we describe the processing of packets for both short messages and longer bursts.

3.1 IP Packet Processing for Short Messages

In Section 3.2, we describe a technique for processing of IP packet flows containing many packets, which allows most of the data to be forwarded directly at the ATM layer without explicit software processing. However, often a router receives short messages that do not benefit from the use of such techniques. In addition, a router may receive packets that for some other reason require explicit software processing at every router.

Each ATM link connecting different routers has one (or possibly more than one) virtual circuit dedicated to carrying packets that require explicit software processing. An APIC at the receiving end of such a link forwards such packets into the memory of its IPPE. In this case, the required IP processing is carried out on the IPPE. This processing includes checking the validity of the packet header, making a routing decision, updating the appropriate header fields (e.g., the time-to-live and header checksum), and enqueueing the packet for forwarding to the proper output port. The bulk of this processing is identical to that performed by conventional IP routers and can be optimized to about one hundred instructions per packet in the common case, as described in [16].

Each input IPPE maintains a dedicated virtual circuit to each of the output IPPEs. When queueing a packet for a particular output port, it selects the virtual circuit corresponding to an IPPE at the proper output port. The output IPPE buffers packets received from different input IPPEs and schedules them for transmission on a single outgoing virtual circuit. The scheduling algorithm performed by the output IPPE is designed to ensure that each packet flow receives the appropriate quality of service.

3.2 IP Packet Processing for Longer Bursts

To allow fast-path processing for longer bursts of IP packets we pre-configure a set of permanent virtual circuits (PVC) joining IPPEs in adjacent routers. That is, we have permanent virtual circuits crossing a single link, joining the IPPEs on the output side of a router to the IPPEs on the input side of adjacent routers. Each PVC may be in one of two states: active or inactive. At the time of initialization or when a PVC is not being used, it is in an inactive state. The APICs at the receiving end have all PVCs in their internal VC tables, and keep track of the state of each PVC.

If a packet is received on an inactive PVC, (meaning that the upstream router has decided to use this PVC), the virtual circuit switches from inactive to active. The APIC sends the packet to the IPPE for processing as shown in Figure 6 for VC = 54. The IPPE does four things:

- The IPPE makes a routing decision to select the output port. This decision may be made based on dynamic information on the status and current loading of the various output ports, as well as static routing information.
- Next, the IPPE exchanges control messages with the IPPE at the selected output port to get an unused PVC to forward packets to the next router. Upon receiving the proper response, it sends a control cell to the ATM switch, configuring it to forward cells received at the input to the proper output, with the proper virtual circuit identifier. Note that this control interaction is purely local to the router and involves no long-latency interactions.
- The input IPPE, in cooperation with its APIC, then forwards packets it has received during the time that has passed back into the main data flow. When all such packets have been forwarded, the APIC begins forwarding cells on that virtual circuit directly, without diverting them through the IPPE. This operation of flushing packets that have accumulated at the input IPPE requires close coordination between the IPPE and the APIC, but poses no fundamental difficulty. (The bandwidth available through the APIC chain is roughly twice as large as the bandwidth of the external link, guaranteeing that the accumulated packets can be flushed rapidly.)

When the burst of packets that established a given connection is completed, the connection may be torn down. This may be done explicitly, through flow maintenance messages (where available), or implicitly. One simple implicit mechanism involves monitoring usage of PVCs on the output side of the router, and allowing the output IPPE to reclaim any PVC that has not been used recently. This would require that the output IPPE inform the input IPPE using the PVC, so the input IPPE can set its incoming PVC to the inactive state (meaning that packets received subsequently will be processed by the IPPE). The output IPPE would also need to send a control cell on the outgoing PVC to force the IPPE at the next router to reset the PVC to the inactive state. This may be accomplished by defining a special resource management cell for this purpose. Reclaiming unused PVCs may be done as a continual background process. Likewise, it may be done on-demand only when an arriving packet requests use of a link in which all PVCs are already in use.

When selecting output ports to receive a given stream of packets, we try to select the port that is best able to accommodate the added traffic. In some cases however, an output link will become overloaded, causing cells to accumulate in the buffer at the APIC on the output side. When this happens, the APIC will start diverting packets to the IPPE. The larger memory capacity at the IPPE allows it to absorb fairly long-lasting overloads. However, it is important that the IPPE schedule the use of the overloaded resource to provide fair treatment of all the competing traffic streams. Since the overload could also lead to congestion in the ATM core, it should send control cells to the IPPE's that are sending it packets, causing them to start buffering packets on the input side and forwarding them on at a reduced rate. The input IPPE may also reconfigure packet streams away from the congested link if there are other acceptable choices available. This requires some coordination with other IPPEs to prevent control oscillations between different links.

One aspect of cut-through handling of IP packets is that the time-to-live field (hop count in IPv6) is processed only in packets that pass through IPPEs. While this violates a strict interpretation of IP protocol processing, we believe it is not a serious violation. The purpose of the time-to-live field is to detect routing loops. Thus, if we process the first packet of a burst in each router on the path, we can still detect routing loops and flush the entire burst. We simply interpret the time-to-live field of the first packet in the burst as applying to the whole burst. A similar argument justifies selective processing of other fields that would normally be processed at every hop.

3.3 Congestion Avoidance and Control

As indicated in the previous section, we seek to avoid congestion as much as possible by routing arriving bursts of packets to output links that are best able to accommodate them. The loading on the various output links can be obtained by polling hardware cell counters in the output port processors of the ATM core. The counters are read using control cells, which are time-stamped when data is read to allow accurate determination of the load during short time intervals.

If information describing the data rate for an IP packet stream is available (through a reservation protocol like RSVP, for instance), this can be used to optimize the output port selection process. In particular, if there are several good choices available (that is, links that can all accommodate the added traffic), it is best to select the busiest link. This policy minimizes bandwidth fragmentation and improves the performance for later-arriving bursts. In the absence of such information, the best choice is the least busy link.

When the load on an output link exceeds its capacity, packets may accumulate in the outgoing IPPE. Careful scheduling is required to ensure that each packet stream receives acceptable performance. A number of scheduling algorithms have been proposed over the past few years [9, 8, 10, 11, 12, 13, 14, 22]. Two of them have received most attention in the Internet community. One was designed

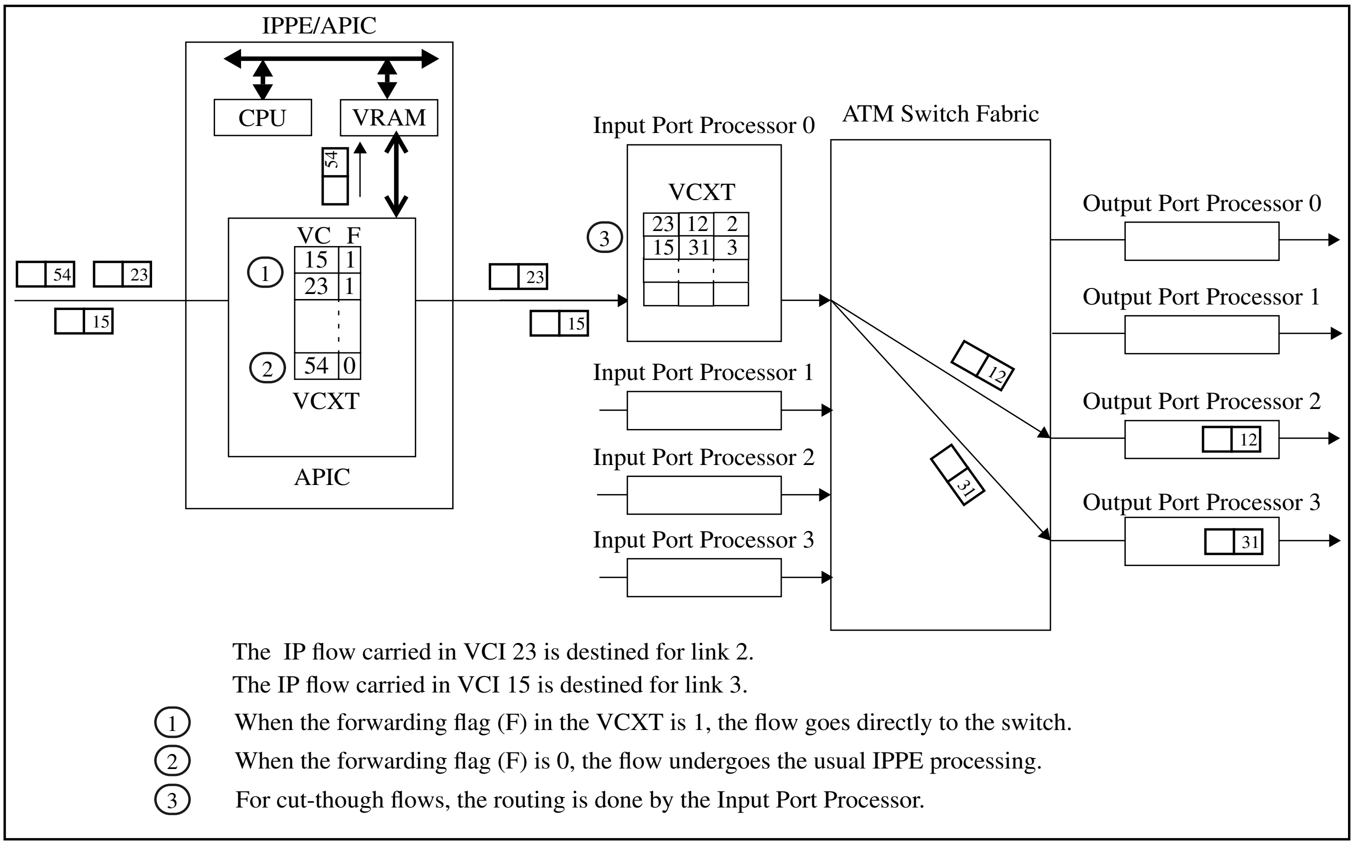

Figure 6: Cut-through Routing

by a collaboration between MIT and Xerox PARC and is generally referred to as the CSZ algorithm [3]; the other was designed at LBL, and is named Class-Based-Queueing (CBQ) [9].

The CSZ algorithm combines three simple building blocks; weighted fair queueing (WFQ), priority queueing, and FIFO. In order to provide guaranteed service to selected traffic streams, WFQ is used at the top level of CSZ scheduling to provide traffic isolation. Priority queueing is used to separate predictive real time services from best-effort traffic, as well as to separate different classes of predictive services. Each predictive class contains multiple real time data flows. Within a class, FIFO queueing is used to take the most advantage of statistical multiplexing. The principle of WFQ is also used to build a *sharing tree*, which is orthogonal to the scheduling architecture, to enforce link-sharing.

In CBQ the basic building blocks are priority queueing, round-robin scheduling, and a novel "borrowing hierarchy" for link-sharing control. CBQ uses priority at the top level of the scheduling control, thus it does not provide guaranteed real time services. Classes within each priority level are served in a round-robin fashion. In addition to this scheduling architecture, there is a separate "borrowing hierarchy" that includes all traffic classes and the allocated bandwidth to each class. Whenever a packet is forwarded, the bandwidth usage of the corresponding class is adjusted. Upon becoming resource overdrawn, a class may either borrow more bandwidth from its parent class (if the parent has any left), or be handled in a predefined manner (such as being suspended for a while, or packets being dropped). This borrowing hierarchy may provide adequate link-sharing control and has an extremely efficient implementation.

The a^{I}t^{P}m router will provide a testbed for evaluating these and other packet scheduling algorithms in a gigabit environment.

3.4 IP Multicast Forwarding

To support IP multicast forwarding, a router must be able to take a single incoming packet and send it out multiple outgoing ports. In conventional, bus-based router architectures, this is an expensive operation. Typically, a CPU must make multiple copies of the packet, which incurs a large number of bus and memory cycles. It also incurs undesirable delay: the last copy of a packet is delayed by the time it takes to make all the other copies. The a^{I}t^{P}m router avoids the cycle-cost and delay of CPU-based packet copying by exploiting the cell-replication capability of the a^{I}t^{P}m ATM backplane to achieve high-performance IP multicast forwarding.

In particular, when the first of a stream of IP multicast packets is received at an IPPE, a multicast route lookup is performed. This yields an ATM VCI that was previously configured using the IGMP and multicast routing protocols and locally bound to the given multicast address. The IPPE then modifies the virtual circuit table in the ATM switch's input port processor to forward cells to the proper multicast VCI. After that, the IPPE hands off to the APIC as before, allowing the remainder of the burst to be processed at the cell level.

3.5 Performance Issues

We estimate that the combination of the APIC and IPPE will achieve sustained packet processing rates of between 100 and 200 thousand packets per second. This is based on the assumption that the IPPE's processor executes instructions at a sustained rate of 40 MIPS while processing packets and that between 200 and 400 instructions are sufficient for each packet. This is fast enough to accommodate a fully loaded 1.2 Gb/s link with average packet sizes of between 750 and 1500 bytes. This implies that traffic loads consisting mostly of small packets can be handled with reasonable efficiency and that IPPEs can accommodate the traffic flows they are required to process during the start of a burst or a congestion build-up.

For long bursts, it's important to understand the control delay incurred at the start of a burst while packets are waiting for the local virtual circuit to be established. We estimate that this control delay will be less than 100 μs per router. Therefore, an end-to-end path through ten routers will involve about 1 ms of control delay. For wide-area network applications, this added delay is not a significant penalty. Notice though, that as a burst progresses through the network, each router adds some additional delay, meaning that the first portion of the burst becomes "compressed" as it moves through the network. Thus, the last router in a ten router path will process about ten times as many packets in software does the first router in the path. This suggests that establishment of the local paths is clearly beneficial only for bursts that use the virtual circuit for a time duration of at least about 10 ms. This need not be continuous. As long as the virtual circuit is used for at least 10 ms before being reallocated to some other traffic stream, we benefit from establishment of the local virtual circuit. This implies that we should hold virtual circuits for as long as possible, once they have been allocated to a particular traffic stream. So long as there are sufficient virtual circuits available, this does not present any difficulty.

4 Other Protocols

To operate correctly, an $\mathrm{a^It^Pm}$ router must implement several protocols besides IP. These include an ATM-PPP protocol, the Internet Control Message Protocol (ICMP), the internet routing protocol(s), telnet, a flow set up protocol such as RSVP, and others. In addition, next-generation routers must support the IP version 6 protocol. The following paragraphs summarize operation of these protocols on $\mathrm{a^It^Pm}$.

ATM PPP Protocol. The preceding description implies the existence of a point-to-point protocol for carrying IP packets between $\mathrm{a^It^Pm}$ routers. This protocol defines the use of AAL5 for carrying IP packets and defines the "reset" control cell used to force a PVC into the inactive state. It also defines the procedures for establishing these PVCs in the first place.

Internet Control Message Protocol. ICMP is used to send error and control messages from a router back to the originator of the packet whose processing led to the ICMP message. In the case of an $\mathrm{a^It^Pm}$ router, the ICMP messages are generated by either an input or output IPPE (the input IPPE is the one that received the original packet and the output IPPE is the one to transmit the packet). If a message is generated at an input IPPE, it may be sent on the same link and does not need to go through the switch. Some ICMP messages may be generated at an output IPPE by packet scheduling and congestion detection algorithms. These messages are sent from the output IPPE to the input IPPE through the switch so that the ICMP message may be returned to the source of the packet.

Unicast and multicast routing protocols. One of the IPPEs, called the "route server IPPE" (RS-IPPE) is responsible for running Internet routing protocol(s) that maintain an up-to-date routing table for the entire router. Routing updates received at all IPPEs are sent to the RS-IPPE. In response to these updates, the RS-IPPE recomputes the routing table and uses a pre-established multicast virtual circuit to broadcast a copy of the table (or only modifications) to other IPPEs within the $\mathrm{a^It^Pm}$. Thus, each IPPE independently makes the routing decision for an incoming IP packet.

Another IPPE, called the "multicast server IPPE" (MS-IPPE) is responsible for maintaining multicast group information within the $\mathrm{a^It^Pm}$. Efficient multicast support is essential for higher-level services such as Multicast Backbone (MBONE), which provides one-to-many and many-to-many network delivery services for applications such as video-conferencing and network audio. Creation of, and modifications to, multicast groups in an $\mathrm{a^It^Pm}$ router happen in response to IGMP messages and multicast routing updates. The MS-IPPE is also responsible for creating and maintaining multicast VCs within the $\mathrm{a^It^Pm}$ for active IP multicast groups. Once these multicast VCs are set up, routing of an input multicast IP packet is done independently by an IPPE without having to go through the MS-IPPE.

Telnet. The Internet model allows each port of the router (and even each IPPE in the $\mathrm{a^It^Pm}$) to have a unique IP address. However, to hide the internal complexity of the router and to conserve IP addresses, we allow only one IP address per router. This means that IPPEs are not addressable individually. Therefore, we plan to have one of the IPPEs per $\mathrm{a^It^Pm}$, called the control IPPE (C-IPPE), run the telnet and other daemons. Thus, telnet'ing to an $\mathrm{a^It^Pm}$ router involves connecting to its C-IPPE. Of course, the C-IPPE has VC connections to all other IPPEs within the $\mathrm{a^It^Pm}$ router. Thus, a remote user with appropriate access rights may access and control any of these IPPEs.

Flow Setup and Reservation Protocols. These functions are handled similarly to the routing protocols. All RSVP messages are sent to a designated IPPE (possibly the same as the RS-IPPE). This IPPE performs the admission control function on flow setup requests, and if a request may be admitted, the IPPE makes an appropriate reservation and informs the concerned IPPEs on the input and output links.

An alternate arrangement could be to process RSVP messages at the appropriate output IPPEs. However, this could lead to synchronization problems, in the case of a multicast flow if some output links may accommodate the flow, and some may not.

IP version 6. IPv6 is designed as the next-generation Internet routing protocol. $\mathrm{a^It^Pm}$ support for IPv6 is straightforward since IPv6 packet formats have been designed to simplify packet processing and help with QoS guarantees, as explained in the following paragraphs:

- The presence of a Flow Label in the IPv6 packet header simplifies per-packet processing. For example, flow labels may be used to do hash table lookup for packet routing. Likewise, they may be mapped directly on to VCs to allow hardware based cut-through routing and provide Quality of Service (QoS) support at the IP level.

- IPv6 eliminates computing the header length and comparing it with a minimum. In IPv4, it is necessary to check each datagram header to see if the header length was set to a value greater than or equal to the minimum.

- IPv6 also eliminates the header checksum. It assumes that each link level protocol (for example AAL5) will provide a hop-to-hop error detection using CRC or something comparable.

- IPv6 eliminates fragmentation at a router. If a router cannot forward an IP datagram because the outgoing interface supports MTU sizes less than the packet size, the router does not fragment the packet. Instead, the router drops the packet and sends an ICMP error message back to the source.

- IPv6 eliminates IP options processing for the common case. If the destination address does not match any local address, then IP option headers do not have to be examined (except for the unusual hop-by-hop options header).

5 Related Work

IP routers have been produced commercially for many years now. The classical router architecture consists of a single general-purpose processor with multiple hardware interfaces to point-to-point links or shared access subnetworks. Over the last decade, commercial router vendors have migrated to architectures in which increasing amounts of processing are placed on the interface cards and a high bandwidth interconnect (usually a high-speed bus or crossbar) provides connectivity among the different interface cards. An notable example of this style of architecture is a recent product by NetStar [15]. The NetStar architecture comprises a central crossbar with serial interfaces operating at about 1 Gb/s per port together with interface cards containing two programmable processors and custom hardware for buffering and selected IP functions.

The a^{I}t^{P}m router differs from this type of architecture in two fundamental ways. First, because it is based on a scalable switch fabric with optimal cost/performance, rather than a crossbar, the a^{I}t^{P}m architecture scales up economically to configurations with thousands of high-speed ports. This allows large networks to be constructed far more economically than is possible by composing many small switches. Networks constructed from large switches require substantially more interface cards, which are a major cost component. The construction of large networks from small switches also leads to better performance, since it minimizes the number of hops required. Most of the commercial architectures have no way to support multicast at the hardware level. Therefore, the software bears the entire load, which significantly restricts the amount of multicast traffic that may be supported.

A second fundamental difference between the a^{I}t^{P}m architecture and conventional routers is its ability to use the ATM core in a dynamic fashion to allow the vast majority of IP packets to be routed directly in hardware, without the requirement for software processing at every hop. While the a^{I}t^{P}m architecture permits all processing to be done in software, the potential for using cut-through packet handling to optimize the normal case (while software processing is triggered for exceptions), raises the possibility of getting substantially higher data throughputs for a given amount of software processing capacity.

Another less basic, but still important, distinction between our approach and conventional routing architectures is that we use general-purpose components in the a^{I}t^{P}m. We expect these components to become commodity ATM parts over the next several years. The APIC is a general-purpose host interface chip, and the extensions required for the a^{I}t^{P}m router require only that it be able to identify packet boundaries, using AAL5. Commercial router vendors are generally moving toward embedding portions of the actual IP processing in custom hardware. We feel this approach will limit their ability to keep pace with future protocol enhancements and technology advances. (A good example of the kind of custom hardware solution we feel is inappropriate is described in [21], which embeds some of the IP protocol processing in custom integrated circuits.)

Our approach also differs significantly from simply implementing an IP overlay network, on top of permanent or semi-permanent virtual circuits provided by an underlying ATM network. The fundamental difference is again that in the a^{I}t^{P}m router, the IP layer can directly establish virtual circuits on the fly for individual data bursts (without end-to-end processing). This enables it to exploit the hardware switching advantages of ATM to dramatically reduce the amount of software processing that is required. Note that this can accomplished without the performance penalty and loss of flexibility associated with end-to-end virtual circuit setup. Moreover, the close physical integration of the IP processing with the ATM switching leads to significant implementation economies. The compatibility of the gigabit switch port processors and the APIC, resulting from their common use of the Utopia interface standard, makes this physical integration particularly beneficial.

A similar gateway architecture, with an ATM fabric at the core for high performance and scalability, was also proposed in [17]. However, [17] had argued for all per-packet processing to be implemented in hardware which is not economical, flexible and necessary, especially if the software per-packet processing can be made efficient enough to support the necessary data rates.

Finally, the a^{I}t^{P}m router differs from commercial efforts in that it provides a flexible testbed for experimentation with the latest versions of the IP protocols at gigabit rates. As such, it is an invaluable source of real experimental results and a tool for protocol researchers that may be tailored to suit evolving needs.

6 Concluding Remarks

This paper describes a strategy for integrating IP with ATM to achieve scalable, gigabit internets. We seek to go well beyond the conventional approach of implementing IP over ATM in a strictly layered fashion. We believe that by allowing the IP processing layer to directly control and manipulate an underlying ATM switch core, IP can directly benefit from the hardware processing efficiencies of ATM switching technology, or looking at it from the other perspective, ATM can enjoy the inherent flexibility and adaptability that are among IP's greatest strengths.

Our work uses ATM technology to build scalable high-performance gigabit IP routers and tightly couples the IP and ATM layers to obtain maximum advantage from each. We plan to use the proposed a^{I}t^{P}m router to implement a variety of IP protocols and control algorithms, including IP version 4, the proposed IP version 6, and various packet scheduling and congestion control algorithms to support both best effort and continuous media traffic. The software implementations will allow us to experiment with and fine tune the protocols and algorithms that form the core of the next generation IP in the context of a gigabit environment. The underlying multi-CPU embedded system will ensure that there are enough CPU and memory cycles to perform all IP packet processing at gigabit rates.

We believe that the a^{I}t^{P}m architecture will not only lead to a scalable high-performance gigabit IP router technology, but will also demonstrate that IP and ATM technologies can be mutually supportive. In addition, the architectural approach developed here, in which powerful general-purpose processors are closely coupled to a high-speed and scalable switching system, offers possibilities that go beyond IP routing. The integration of IP with ATM may be viewed as just a first example of a new form of 'integrated layer processing' that blurs the boundaries between different network layers and between the network and application processing layers. While in the past, the conventional wisdom has been to keep these layers strictly separated, there has been a growing appreciation of the potential advantages that may be obtained by a disciplined and carefully structured blurring of the boundaries in the workstations and servers that use the network. We expect that similar advantages can be obtained by carrying this process into the network core, as well.

Acknowledgments

The authors want to thank Lixia Zhang and Steve Deering of Xerox PARC for technical discussions leading up to this paper, particularly with respect to new Internet packet scheduling, congestion control, and multicast mechanisms. Thanks also to Hari Adiseshu of Washington University for assistance in clarifying certain details of IP version 6 and helping with the figures.

References

[1] Asthana, A., C. Delph, H. Jagadish, and P. Krzyzanowski "Towards a Gigabit IP Router", Journal of High Speed Networks, Vol. 1, No. 4, pp. 281-288, 1992.

[2] ATM Forum, "UTOPIA, An ATM_PHY Interface Specification," Level 1, Version 2.01, March 21, 1994

[3] Clark, D., S. Shenker, and L. Zhang "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", *Proceedings of SIGCOMM'92*, September, 1992.

[4] Dittia, Zubin, Jerome R. Cox, Jr. and Guru Parulkar. "Design of the APIC: A High Performance ATM Host-Network Interface Chip," *Proceedings of IEEE INFOCOM'95.*

[5] Dittia, Zubin, Jerome R. Cox, Jr. and Guru Parulkar. "Catching Up With the Networks: Host I/O at Gigabit Rates," Technical Report WUCS-94-11, Department of Computer Science, Washington University in St. Louis, 1994.

[6] Dittia, Zubin, Jerome R. Cox, Jr. and Guru Parulkar. "Using an ATM Interconnect as a High Performance I/O Backplane," *Proceedings of the Hot Interconnects Symposium*, August 1994.

[7] Dittia, Zubin, Andy Fingerhut and Jonathan Turner. "A Gigabit Local ATM Testbed for Multimedia Applications: System Architecture Document for Gigabit Switching Technology," Applied Research Lab, Working Note 94-11.

[8] Ferrari, D. and D. Verma. *A Scheme for Real-Time Channel Establishment in Wide-Area Networks*, In *IEEE JSAC*, Vol. 8, No. 4, pp 368-379, April 1990.

[9] Floyd, Sally, "Talk given at Maryland High Speed Workshop," March, 1992

[10] Golestani, S. J. *A Stop and Go Queueing Framework for Congestion Management*, In *Proceedings of SIGCOMM '90*, pp 8-18, 1990.

[11] Golestani, S. J. *Duration-Limited Statistical Multiplexing of Delay Sensitive Traffic in Packet Networks*, In *Proceedings of INFOCOM '91*, 1991.

[12] Hyman, J., and A. Lazar. *MARS: The Magnet II Real-Time Scheduling Algorithm*, In *Proceedings of SIGCOMM '91*, pp 285-293, 1991.

[13] Hyman, J., A. Lazar, and G. Pacifici. *Real-Time Scheduling with Quality of Service Constraints*, In *IEEE JSAC*, Vol. 9, No. 9, pp 1052-1063, September 1991.

[14] Kalmanek, C.,, H. Kanakia, and S. Keshav. *Rate Controlled Servers for Very High-Speed Networks*, In *Proceedings of GlobeCom '90*, pp 300.3.1-300.3.9, 1990.

[15] NetStar, Inc. *GigaRouter System Description.* Publication SPD000001, July 1994, Revision 2.

[16] Partridge, C., "Gigabit Networking," Addison Wesley, 1993.

[17] Parulkar, Guru. "The Next Generation of Internetworking," ACM SIGCOMM, *Computer Communication Review*, January 90.

[18] "SPARC MBUS Interface Specification: Revision 1.2", SPARC International, April 1991.

[19] Turner, Jonathan S. "Progress Toward Optimal Nonblocking Multipoint Virtual Circuit Switching Networks," *Proceedings of the Thirty-First Annual Allerton Conference on Communication, Control, and Computing*, September 1993, pp. 760-769.

[20] Turner, Jonathan S. "An Optimal Nonblocking Multicast Virtual Circuit Switch," *Proceedings of Infocom*, June 1994, pp. 298–305.

[21] Tantawy, Ahmed, Odysseas Koufopavlou,Martina Zitterbart and Joseph Abler. "On the Design of a Multigigabit IP Router," Journal of High Speed Networks, vol. 3, no. 3, 1994.

[22] Verma, D., H. Zhang, and D. Ferrari. *Delay Jitter Control for Real-Time Communication in a Packet Switching Network*, In *Proceedings of TriCom '91*, pp 35-43, 1991.