Report Number: WUCS-91-11

1991-10-01

Design of an ATM-FDDI Gateway

Sanjay Kapoor and Gurudatta M. Parulkar

Asynchronous transfer mode (ATM) networks are capable of supporting a wide variety of applications with varying data rates. FDDI networks offer to support similar applications in the LAN environment. Both these networks are characterized by high data rates, low error rates, and are capable of providing performance guarantees. In this paper we present the design of an ATM-FDDI gateway that can provide high performance internetworking between these two important classes of networks. An important part of the gateway design philosophy is to partition the functionality into critical and non-critical paths. The critical path consists of per packet processing and... **Read complete abstract on page 2.**

Design of an ATM-FDDI Gateway

Sanjay Kapoor and Gurudatta M. Parulkar

Complete Abstract:

Asynchronous transfer mode (ATM) networks are capable of supporting a wide variety of applications with varying data rates. FDDI networks offer to support similar applications in the LAN environment. Both these networks are characterized by high data rates, low error rates, and are capable of providing performance guarantees. In this paper we present the design of an ATM-FDDI gateway that can provide high performance internetworking between these two important classes of networks. An important part of the gateway design philosophy is to partition the functionality into critical and non-critical paths. The critical path consists of per packet processing and is implemented in hardware. The non-critical path consists of connection, resource, and route management, and is implemented in software.

Design of an ATM-FDDI Gateway

Sanjay Kapoor and Gurudatta M. Parulkar

WUCS-91-11

October 1991

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

This work is supported in part by the National Science Foundation, and an industrial consortium of Bellcore, BNR, DEC, Italtel SIT, NEC, NTT and SynOptics.

Design of an ATM–FDDI Gateway*

Sanjay Kapoor
kapoor@wuee1.wustl.edu
+1 314 726 4163

Gurudatta M. Parulkar
guru@flora.wustl.edu
+1 314 889 4621

ABSTRACT

Asynchronous transfer mode (ATM) networks are capable of supporting a wide variety of applications with varying data rates. FDDI networks offer to support similar applications in the LAN environment. Both these networks are characterized by high data rates, low error rates, and are capable of providing performance guarantees. In this paper we present the design of an ATM–FDDI gateway that can provide high performance internetworking between these two important classes of networks. An important part of the gateway design philosophy is to partition the functionality into critical and non-critical paths. The critical path consists of per packet processing and is implemented in hardware. The non-critical path consists of connection, resource, and route management, and is implemented in software.

Appeared in the *Proceedings of ACM SIGCOMM'91 Conference on Communications Architectures and Protocols*, September 1991.

*This work was supported in part by the National Science Foundation, and an industrial consortium of Bellcore, BNR, DEC, Italtel SIT, NEC, NTT, and SynOptics.

DESIGN OF AN ATM–FDDI GATEWAY*

Sanjay Kapoor
kapoor@wueel.wustl.edu
+1 314 746 4099

Gurudatta M. Parulkar
guru@flora.wustl.edu
+1 314 889 4621

ABSTRACT

Asynchronous transfer mode (ATM) networks are capable of supporting a wide variety of applications with varying data rates. FDDI networks offer to support similar applications in the LAN environment. Both these networks are characterized by high data rates, low error rates, and are capable of providing performance guarantees. In this paper we present the design of an ATM–FDDI gateway that can provide high performance internetworking between these two important classes of networks. An important part of the gateway design philosophy is to partition the functionality into critical and non-critical paths. The critical path consists of per packet processing and is implemented in hardware. The non-critical path consists of connection, resource, and route management, and is implemented in software.

*This work was supported in part by the National Science Foundation, and an industrial consortium of Bellcore, BNR, DEC, Italtel SIT, NEC, NTT, and SynOptics.

1 INTRODUCTION

The existing communication environment is best characterized as an internet consisting of a number of low speed networks interconnected by gateways. We believe that the future communications environment will continue to be an internet. However, it will include emerging high speed networks, such as ATM and FDDI networks and will need to support a variety of applications, some of these requiring performance guarantees. This places new demands on internet protocols and gateways.

To meet these new demands, we have proposed a novel internet abstraction called the *Very High Speed Internet Abstraction* (VHSI). The VHSI abstraction is aimed at supporting both connection-oriented applications requiring performance guarantees and classical datagram applications. An important component of the VHSI abstraction is the Multipoint Congram-oriented High performance Internet Protocol (MCHIP, *i.e.*, equivalent of DoD IP) that higher level protocols use to communicate across the internet without being concerned with the diversity of underlying networks. Another aspect of the VHSI abstraction is its high speed and high performance gateway architecture. A simple definition of a gateway is a device that interconnects disparate networks and implements the internet functionality. We believe that the next generation internet gateways must be able to provide high throughput and performance guarantees on a per connection basis. An important part of the gateway design philosophy is to partition the functionality into critical and non-critical paths. The critical path consists of per packet processing and is implemented in hardware for speed and performance. The non-critical path consists of connection, resource and route management, which is best implemented in software, because it is complex and because it requires flexibility and fine tuning. Mixing of these paths, as is generally done in present day gateways, is not an efficient approach.

In this paper we present the design of an ATM–FDDI gateway that implements the VHSI functionality and is able to sustain high throughput. The focus of this paper is on the implementation of the critical path of the gateway in hardware. We believe that both ATM and FDDI networks are excellent target networks to explore the VHSI gateway architecture for the following two reasons:

- These networks represent two important component networks of the next generation internet. ATM is gaining tremendous popularity amongst phone companies as a promising switching technology to be used in the future Broadband ISDN. ATM networks are aimed at supporting a wide variety of applications such as digitized voice, full motion video, and interactive imaging for scientific and business applications. FDDI is a promising high speed LAN technology, capable of supporting similar applications as ATM.
- The design of a simple two port ATM–FDDI gateway requires addressing issues typical of high speed gateway architecture because these networks support high data rates with considerable diversity. For example, an ATM network is essentially connection-oriented using small fixed size cells with explicit resource management. A FDDI network on the other hand is a datagram network using relatively large variable size frames with no explicit resource management.

The outline of the paper is as follows: Section 2 describes the VHSI abstraction and presents an overview of the MCHIP protocol. Section 3 summarizes key features of the ATM and FDDI networks. Section 4 describes the architecture of a two port ATM–FDDI gateway. The two important components of the gateway include a segmentation and reassembly (SAR) protocol processor (SPP) and a MCHIP protocol processor (MPP). Sections 5 and 6 explain the designs of the SPP and MPP respectively. Details of these custom chips are not presented for the sake of space limitations. We finally conclude by summarizing the work in progress.

2 VHSI OVERVIEW

In this section we present an overview of the VHSI abstraction, the details of which can be found in [12]. The VHSI abstraction (Figure 1) is similar to the existing internet abstraction in the sense that it also provides a homogeneous networking environment on top of heterogeneous networks. However, it includes a number of significant improvements over the current internet abstraction. The important components of the VHSI abstraction include a multipoint congram-oriented transport facility, resource managers for diverse networks, an internet route server, and interfaces to various network access protocols. The rest of this section gives an overview of each of these components.

2.1 Component Networks

The VHSI can successfully deal with network diversity only if it can impose certain requirements on component networks, which force them to cooperate at the internet level in a number of important ways. Two requirements that the VHSI abstraction imposes on each component network are that it must provide its parametric description to directly connected gateways, and either do internal resource management on a per application conversation basis, or allow its directly connected gateways to do this.

2.2 Internet Routing

The requirements for internet routing in the VHSI include efficient multicast and routing based on resource requirements, access constraints, and policy. Furthermore, there are standard requirements, such as stability, fast convergence, and load balancing, that the routing protocols have to meet. Our work does not include research on internet routing, however, there are a number of promising research efforts in progress which aim at developing routing protocols/models which would include some or all of this functionality.

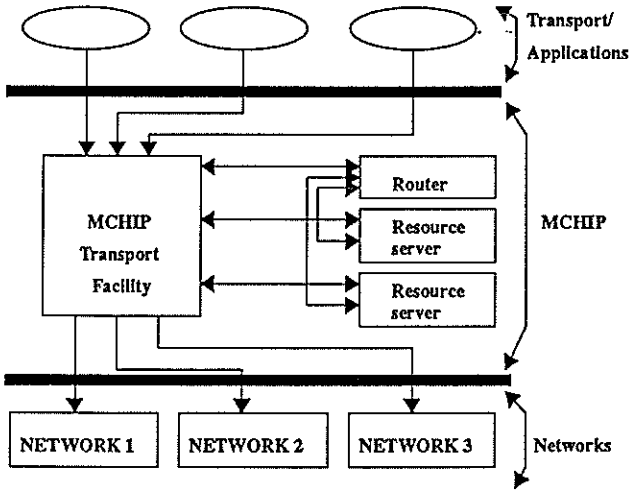

Figure 1: The VHSI Abstraction

2.3 Internet Resource Management

One of the goals of the VHSI abstraction is to be able to make performance guarantees to applications across an internet of diverse networks. To achieve this goal, we argue that it is necessary to monitor and control resource allocation and usage on a per congram/connection basis. However, a number of existing and emerging networks do not have this functionality. Suitable mechanisms can be designed at the internet level to incorporate this functionality, with only minor modifications to the internal operation of the networks. A simple but effective approach is to designate a directly connected gateway to serve as a resource manager of the network, that is the gateway is responsible on behalf of the network for keeping track of resource usage of active congrams, and accepting a new congram only if there are resources to meet the congram's performance needs. We have demonstrated viability of this approach for Ethernet using a simulation study [10].

2.4 Multipoint Congram-Oriented High Performance Internet Protocol (MCHIP)

There is little doubt that the next generation internet protocol must provide high performance with high predictability. One issue that researchers still argue about is that of connection versus connectionless service. We consider an application to be connection-oriented if it requires strict performance guarantees, a large fraction of link bandwidth, and can tolerate set up and termination delays. We consider an application to be datagram-oriented (or connectionless) if it does not require performance guarantees, can use resources on the basis of availability, and requires a relatively smaller fraction of link bandwidth. In other words, a connection-oriented application does not have good multiplexing characteristics, while a datagram application does. We believe that both paradigms are useful for different applications, and we need to effectively support them at the internet level. Therefore, we argue in favor of a service which aims at combining the strengths of both the connection and the datagram.

We introduce an abstraction called a congram, because it incorporates important aspects of connection-oriented and datagram services, and because it avoids any prejudice resulting from using old terms, such as "connection" or "datagram". A congram in our context is a plesio-reliable[1] service with no hop-to-hop flow and error control. A congram only implies a predetermined path for packets with statistically bound resources. Also, appropriate low overhead mechanisms are provided to allow establishment and reconfiguration of the congram path. Note that reconfigurability is important to ensure survivability in the event of network failures. Thus, the important point to note about the plesio-reliable congram abstraction is that the connection aspect of this abstraction provides efficient per packet processing and variable grade service with performance guarantees, and the plesio-reliability aspect provides survivability and flexibility as in a datagram model. In short, this abstraction has the potential to incorporate the benefits of both connectionless and connection-oriented approaches.

MCHIP supports two types of congrams: the user congram (UCon) and the persistent internet congram (PICon)[2]. The UCon corresponds to a soft connection – it requires setup by the user (at some cost), and once the required data transfer is complete, it needs to be terminated. PICons are long lived congrams between MCHIP entities, and their purpose is to allow multiplexing of traffic from a number of users and applications when appropriate, and to carry data for UCons that are being set up or reconfigured. In this respect, PICons are like dynamic leased packet switched internet channels. It is important to note that UCons are created upon request by a user, while PICons are set up by the system.

The management of PICon and UCon are described in detail in References [3, 11], and an MCHIP implementation is in progress. It is important to note that as far as the critical path implementation in a gateway is concerned, the distinction between a UCon and PICon is minimal, and therefore, the rest of this paper uses the congram and connection as a generic term.

3 SUBNETWORKS OVERVIEW

In this section we summarize the important characteristics of the ATM and FDDI networks (Figure 2).

Transmission. Both ATM and FDDI networks use fiber optic as the transmission medium. The proposed data rate on an ATM network varies from 100 to 600Mbps. FDDI supports a data rate of 100Mbps.

Topology. An ATM network topology is best characterized as a mesh, whereas the FDDI network topology is a ring. The maximum number of nodes on the FDDI network is 1000, with the maximum ring distance equal to 200 kilometers. There

[1] *Plesio* comes from Greek, meaning *almost*.

[2] In previous reports, a PICon was referred to as a "*perpetual* internet congram".

	ATM	FDDI
Transmission Medium	Fiber optic	Fiber optic
Data Rates	100 - 600 Mbps	100 Mbps
Network Topology	Mesh	Ring
Resource Allocation	Explicit for each connection	None
Media Access	Connection-oriented	Datagram using timed-token access protocol
Packet Format	Fixed cells; 53 bytes	Variable sized frames upto 4500 bytes

Figure 2: Summary of ATM and FDDI Network Features

is no theoretical limit for the number of nodes on an ATM network. Nodes on an FDDI network can be dual or single attached. In the case of a dual attached node, it has two media access controllers.

Access. An ATM network essentially provides a connection-oriented access to its endpoints. An endpoint uses a signaling protocol to set up and terminate connections. An FDDI network provides datagram access using a timed token protocol. For a station to transmit, it must first get a valid token, and based on the status of MAC timers, it either transmits synchronous or asynchronous frames. Synchronous data transmission is used for time critical applications, whereas asynchronous is used for all other applications. More details on the operation and tradeoffs of the FDDI protocols can be found in [1, 2, 6, 13]. According to Reference [8], all internet traffic on an FDDI network should be sent using asynchronous frames.

Packet Format. An ATM network uses fixed length cells that are 53 octets in length. These cells comprise of a 5-octet header and a 48-octet information field. FDDI frames are variable in size, with the minimum and maximum size equal to 64 and 4500 bytes respectively.

Addressing. An ATM network may or may not support multipoint addressing or communication. FDDI supports three types of addressing: point-to-point, group or multicast, and broadcast addressing. A station can be a member of many groups at a given time, and groups can be dynamically set up.

The Broadcast Packet Network (BPN) is an ATM network being locally designed and prototyped at Washington University [14], and it is the target network for the gateway design. The BPN supports point-to-point and multipoint connections with resource reservations. The BPN has a connection-management (ATM signaling) protocol for connection set up and termination [4, 7].

4 GATEWAY ARCHITECTURE

4.1 Protocol Hierarchy

The gateway essentially implements the ATM and FDDI access protocols and MCHIP as the internet protocol. This subsection explains how these protocols fit together (Figure 3).

The ATM access is divided into three layers: the PHY, segmentation and reassembly (SAR) protocol, and signaling protocol, as shown in Figure 3. The ATM PHY is responsible for synchronizing the incoming ATM cells and providing an error check on the ATM header. The SAR protocol is used to segment higher level protocol

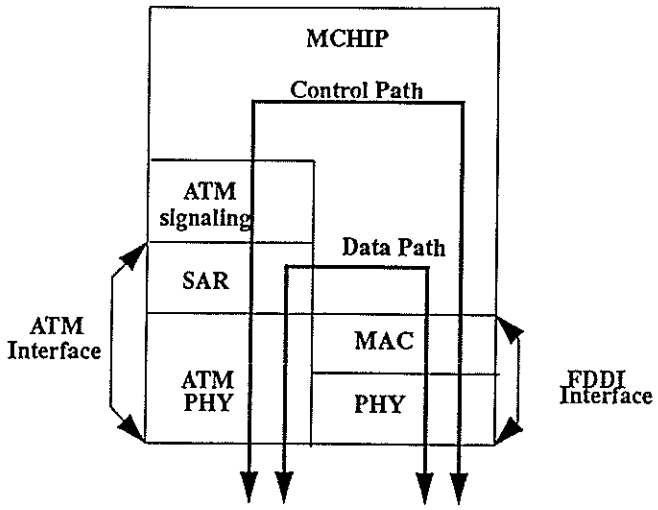

Figure 3: Protocol Structure in a Gateway

data and control frames into ATM cells and reassemble ATM cells into higher level protocol frames [5]. The ATM or BPN signaling protocol is used to do ATM connection management which requires an exchange of a series of messages between the gateway and ATM switch [4, 7].

The FDDI access is divided into two layers: the PHY (Physical Layer), and MAC (Media Access Control) [2]. The FDDI interface is implemented using AMD's SUPERNET[3] chip set [1].

As mentioned before, MCHIP is the internet protocol of the VHSI abstraction, and works in this case on top of the ATM and FDDI access protocols. Communication using MCHIP consists of three phases: congram set up, data transfer, and congram termination.

4.2 Control and Data Paths

For the gateway implementation, it is important to identify the data and control paths. The control path is concerned with the ATM and MCHIP connection (congram) management and the FDDI station management. The control path is the same as the non-critical path and is implemented in software. The data path deals with the packet forwarding on already established connections and is a major part of the critical path. It is implemented in hardware using already available VLSI chips and two new custom chips whose designs are described in subsequent sections.

[3]SUPERNET is a trademark of Advanced Micro Devices Inc.

Incoming ATM cells are reassembled into segments; a segment can be of data or control type. All control segments are forwarded to the appropriate protocol module running on the gateway processor. For data segments, the gateway does an internet channel translation, appends FDDI specific header and forwards the frame to the transmit buffer, which is a part of the FDDI interface. Upon gaining access of a token, the FDDI interface transmits the frame on the ring. The incoming FDDI frames are first stored in the receive buffer. As before, all control frames are sent to the appropriate protocol module running on the gateway processor. For a data frame, a table look up is done to determine the ATM header. The data frame is then fragmented into ATM cells using the SAR protocol, and finally each cell is transmitted on the ATM network with appropriate ATM and SAR headers.

4.3 Hardware Architecture

In this subsection we provide an overview of the hardware architecture of a two port ATM–FDDI gateway with its schematic shown in Figure 4. It consists of the following main components: ATM interface chip (AIC), SAR protocol processor (SPP), MCHIP protocol processor (MPP), node processing element (NPE), reassembly buffer, and SUPERNET (FDDI interface) chip set. We now explain each of these building blocks.

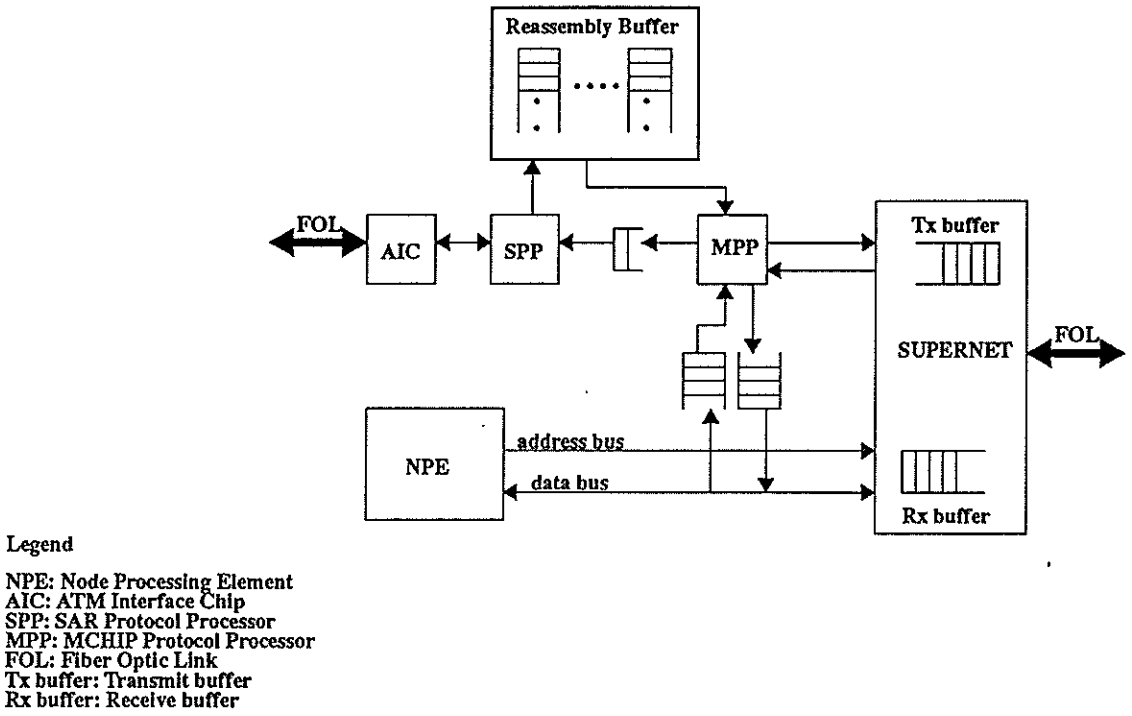

Figure 4: A Two Port ATM–FDDI Gateway

ATM Interface Chip (AIC)

The AIC implements the ATM PHY layer protocol and provides an interface with the ATM network. It synchronizes the incoming ATM cells to the gateway's internal clock (packet cycle). It also performs an error check on the 5-byte ATM header. Any cells with an error in the header are simply discarded. The AIC also generates a CRC for the ATM headers on outbound cells. The AIC is described as the Packet Processor 1 (PP1) in Reference [14].

SAR Protocol Processor (SPP)

The SPP implements the SAR protocol within a single chip with interfaces to the AIC, MPP, and reassembly buffer memory. The SAR protocol is used to segment higher level protocol data and control frames into ATM cells and reassemble ATM cells into higher level protocol frames. Some of SPP's other functionalities include reassembly buffer and timer management. Section 5 explains the working of the SPP in greater detail.

MCHIP Protocol Processor (MPP)

The MPP interfaces with the SPP, NPE, and transmit and receive buffer memories. The SPP forwards the reassembled frames to the MPP which decodes the frame type and routes them appropriately. ATM and MCHIP control frames are routed to the NPE (via FIFO) without any table lookup or header processing. For data frames, the MPP does a table lookup to determine the outgoing destination address on the FDDI network and copies the frame into the transmit buffer memory. For a frame going from FDDI to ATM, the MPP reads the FDDI frame from the receive buffer memory and appends an appropriate ATM header using a table look up. It then forwards this frame to the SPP via the SPP-FIFO. The MPP can also receive control and initialization frames from the NPE. It forwards control and SPP initialization frames to the SPP. Section 6 explains the working of the MPP in greater detail.

SUPERNET

The SUPERNET chip set implements the PHY and MAC layer protocols of the FDDI in VLSI [1]. Station and connection management are not implemented in the SUPERNET chip set, though it provides various registers to keep track of ring statistics, which are used by the NPE to provide this functionality.

After loading frames in the transmit buffer, the MPP sends a transmit request to the NPE, which forwards the request to the SUPERNET. Once the SUPERNET captures a valid token, it transmits the frames on the ring. The SUPERNET stores incoming frames in

the receive buffer memory and informs the NPE. The NPE then reads and processes all the MCHIP control frames and FDDI station management frames. For a data frame, the NPE requests the MPP to read it from the receive buffer memory.

We chose to use a commercially available FDDI chip set because our focus in this effort is on the gateway specific hardware design and not on the FDDI specific development.

Node Processing Element (NPE)

The NPE can be implemented using a standard microprocessor. It will run software implementations of the ATM signaling protocol, the FDDI connection and station management, and the MCHIP congram management. The NPE also performs housekeeping functions for gateway hardware. Some of these functions include processing interrupts, initializing various chips, and configuring the synchronous and asynchronous queues in the transmit and receive buffer memory.

Buffer Memories

There are three distinct buffer memories in the gateway: reassembly buffer memory, transmit buffer memory, and receive buffer memory. The exact size of these buffers will be determined based on results of an ongoing simulation study.

There are also three sets of FIFOs used in the gateway as shown in Figure 4. Two sets of these FIFOs are used between the MPP and NPE to exchange ATM and MCHIP control frames. The third FIFO is used between the MPP and SPP. This FIFO stores either SPP initialization frames or MCHIP frames with suitable ATM headers to be transmitted on the ATM network.

5 SAR Protocol Processor (SPP)

In this section we describe the design of the SPP. The main function of the SPP is to segment MCHIP frames into ATM cells and reassemble ATM cells into MCHIP frames. The fragmentation and reassembly is performed using the SAR protocol as specified in Reference [5].

5.1 Why Fragmentation and Reassembly?

We believe that ATM–FDDI gateways have to perform fragmentation and reassembly because it is extremely

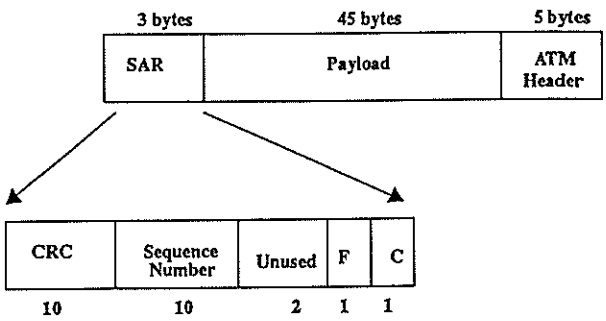

Figure 5: SAR Header

inefficient to carry 53-byte ATM cells on the FDDI network due to the excessive header overhead. It is also important to note that the ATM standard requires (or will require) sequenced cell delivery and very low cell loss rate, thus making the fragmentation and reassembly significantly simpler, which is otherwise considered complex and harmful [9]. We decided to do the fragmentation and reassembly using the SAR protocol as opposed to using MCHIP because the SAR protocol is specifically designed for this job, requires only 3-byte overhead per cell, and can be conveniently implemented in hardware.

5.2 SAR Header

Figure 5 shows the ATM cell format with SAR header. The 3-byte SAR header consists of the following fields: a 10 bit sequence number which specifies the position of the cell in the reassembled frame, a 1-bit frame field that signifies the last cell of the reassembled frame, a 1-bit control field that signifies that the cell is a control cell, and a 10-bit error checksum that covers the entire 48-byte information payload of the ATM cell including the SAR header. If an error is detected, the cell is dropped, and the buffer memory is overwritten. In the SPP design, cells belonging to a particular connection are assumed to arrive in a sequence, which is an assumption consistent with the emerging ATM standard. The SPP detects lost ATM cells using the sequence number and sets an error flag for the corresponding reassembled frame. In the current version of the gateway design, all such frames are discarded. In future, this decision will be left to the MCHIP layer.

5.3 ATM-to-FDDI SPP Pipeline

Figure 6 shows the schematic of the SPP. It essentially consists of two packet processing pipelines, one for each direction: ATM-to-FDDI and FDDI-to-ATM. The ATM-to-FDDI pipeline consists of the Header De-

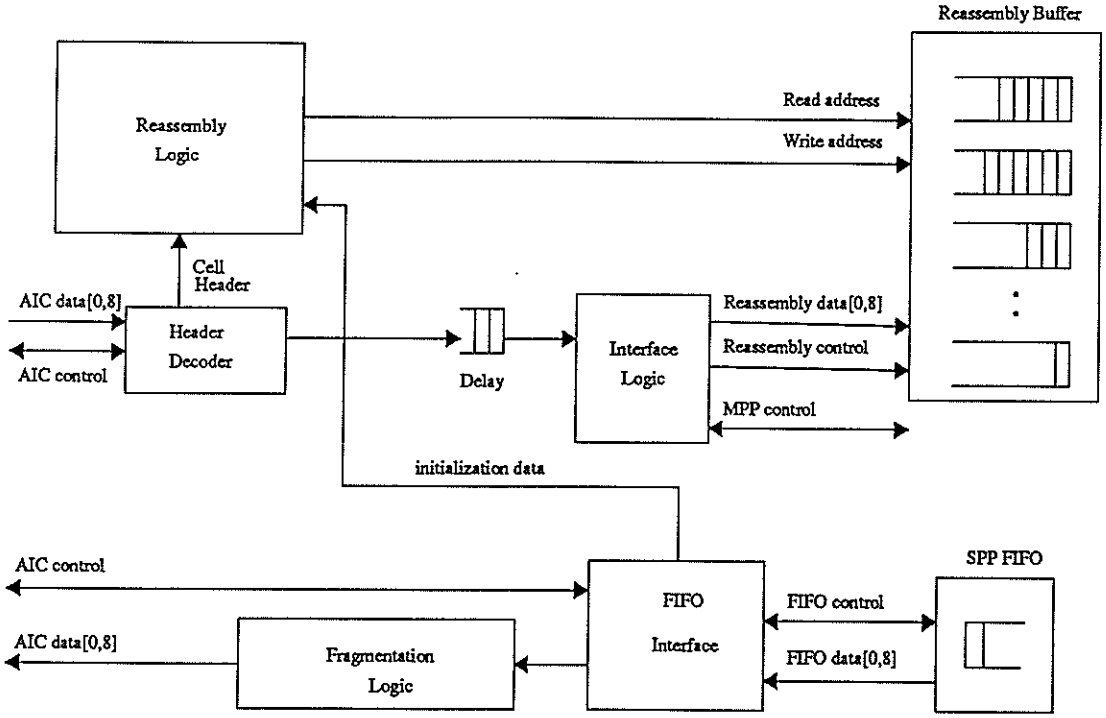

Figure 6: SAR Protocol Processor

coder, Reassembly Logic, CRC Logic, Interface Logic, and Reassembly Buffer. The pipeline operation is explained in this subsection. It is important to note that the SPP allows concurrent reassembly for multiple open connections, and thus maintains state information for partially assembled frames for each connection. In fact, the SPP allows two reassembly buffers per connection, and thus, while a reassembled frame is being queued for transmission over the FDDI network, reassembly of the next frame for the same connection can begin.

The Header Decoder recognizes the ATM and SAR headers in the input cell stream, and forwards them (*i.e.*, VCI, SAR sequence number, and F-bit) to the Reassembly Logic, which keeps the state information and generates write addresses for the reassembly buffer. The Header Decoder also sends the ATM payload to the CRC Logic. The CRC Logic does an error check on the entire 48-byte payload of the ATM cell and forwards the SAR payload. The Interface Logic provides the necessary handshake with the MPP and with the Reassembly Buffer to store the SAR payload.

The Reassembly Logic stores in a lookup table the state information for each open VCI, including the start and end addresses of each reassembly buffer, status (idle or busy) of the reassembly buffer, the write pointer, expected next sequence number, and reassembly timer. The VCI of the input cell is used to do the table lookup and read the state information. The expected next sequence number is compared with the sequence number in the SAR header to confirm the sequenced delivery. Otherwise, an appropriate error flag is set. The write pointer (a part of the state information) is used to generate write addresses to store cells that belong to the same frame in a contiguous memory within the reassembly buffer. Once the incoming ATM cell has been stored, the write pointer is saved for future use. When the end of the frame is detected by decoding the F-bit of the SAR header, the buffer status is updated, the write pointer is reset, and a signal is sent to the MPP to forward the reassembled frame.

The Reassembly Logic also maintains reassembly timers for each active connection. These reassembly timers determine the reassembly timeout value and are initialized by the NPE. If the timer for a particular active connection times out and the last fragment has not arrived, the partially reassembled frame is forwarded to the MPP.

The reassembly buffer memory is configured as a bank of dual-ported memories. The maximum length of the data segment (info field) of an FDDI frame carrying internet traffic should be 4096 bytes [8], which translates to a maximum of 91 ATM cells per reassembly buffer.

5.4 FDDI-to-ATM SPP Pipeline

The FDDI-to-ATM pipeline is relatively simple and consists of the SPP FIFO Interface, Fragmentation Logic, and CRC Generator (Figure 6).

The SPP can receive data, control, or initialization frames from the MPP. The FIFO Interface decodes the type of the received frame. An initialization frame containing reassembly timeout values is sent to the Reassembly Logic. A control or data frame is forwarded to the Fragmentation Logic, where it is fragmented into ATM cells. The *Fragmentation Logic* reads the first five bytes of the frame from the SPP FIFO, containing the ATM header appended by the MPP. The *Fragmentation Logic* makes a copy of this 5-byte header and stamps it on each subsequent 45-byte payload read from the SPP FIFO. It also then adds for each cell the SAR header including the 10-bit sequence number. The partially generated ATM cell is then forwarded to the CRC Generator, which appends the 10-bit SAR CRC and forwards the frame to the AIC for transmission on the ATM network. The FIFO Interface uses the receive frame descriptor to determine the last cell of the frame, and marks the F-bit for this cell. Detailed designs of the SPP blocks have been worked out but not presented here for the sake of brevity and simplicity.

5.5 SPP Delay

The SPP is designed to operate at a clock rate of 25 Mhz, with a 40ns clock cycle. An important performance metric of the SPP is the worst case static delay through SPP pipelines. Our estimates of this delay are as follows:

- **Reassembly processing delay:** It takes 10 clock cycles (400ns) to latch, decode the cell header, and start generating the write addresses. After this initial delay, the 45-byte payload is written into the reassembly buffer in 45 cycles.
- **Fragmentation processing delay:** In the case of fragmentation, the SAR headers including sc crc are appended on the fly as the cell is forwarded to the AIC.

6 MCHIP PROTOCOL PROCESSOR (MPP)

In this section we describe the design of the MCHIP protocol processor (MPP). The main functions of the MPP are to route initialization, control, and data frames appropriately and do MCHIP level packet translation for data frames. Figure 7 shows a schematic of the MPP which essentially consists of two parts: the top half processes frames received from the SPP, and the bottom half processes frames from the NPE and SUPERNET. We describe each part in the following subsections.

6.1 Frames from SPP

The SPP Interface reads a frame from the reassembly buffer of the SPP and determines if it is a control or data frame. A control frame is routed without any header processing to the NPE via the NPE FIFO Interface. The buffer capacity of the NPE FIFO primarily depends on the NPE's processing latency. In the case of a data frame, the MPP does the following header manipulations:

Internet Channel Number Translation. Each MCHIP congram is identified by a 2-octet hop-by-hop identifier called the internet channel number (ICN). At each hop the input ICN is mapped to an output ICN. These ICN translations are stored in a lookup table, called the internet channel translation table for FDDI (ICXT-F), and are a part of the state information for all open congrams.

FDDI Header Generation. In addition to the ICN translation, the MPP needs to also save the FDDI destination address for each open congram. Depending upon the congram, this destination address can be a point-to-point address, a multicast group address, or even a broadcast address. The translation from the input ICN to FDDI destination address for all open congrams is also saved as a part of the ICXT-F.

The MPP also needs to append to the frame the rest of the FDDI standard header. This header is the same for all data frames forwarded by the gateway on the FDDI network for all congrams. It includes the source address, frame control, and LLC specific header. This fixed header is stored in a register of the Header Builder by the NPE at the time of gateway initialization.

The actual header manipulation sequence is described next. When the SPP Interface detects a data frame, it strips the input ICN and uses it as the address into the ICXT-F. The SPP Interface also forwards the rest of the frame to the Header Builder, which builds a valid FDDI header using the output ICN and FDDI destination address received from the ICXT-F and the fixed FDDI standard header stored in its register. It then forwards this frame to the Transmit Buffer Interface. The Transmit Buffer Interface uses DMA to store the frame into the transmit buffer memory of

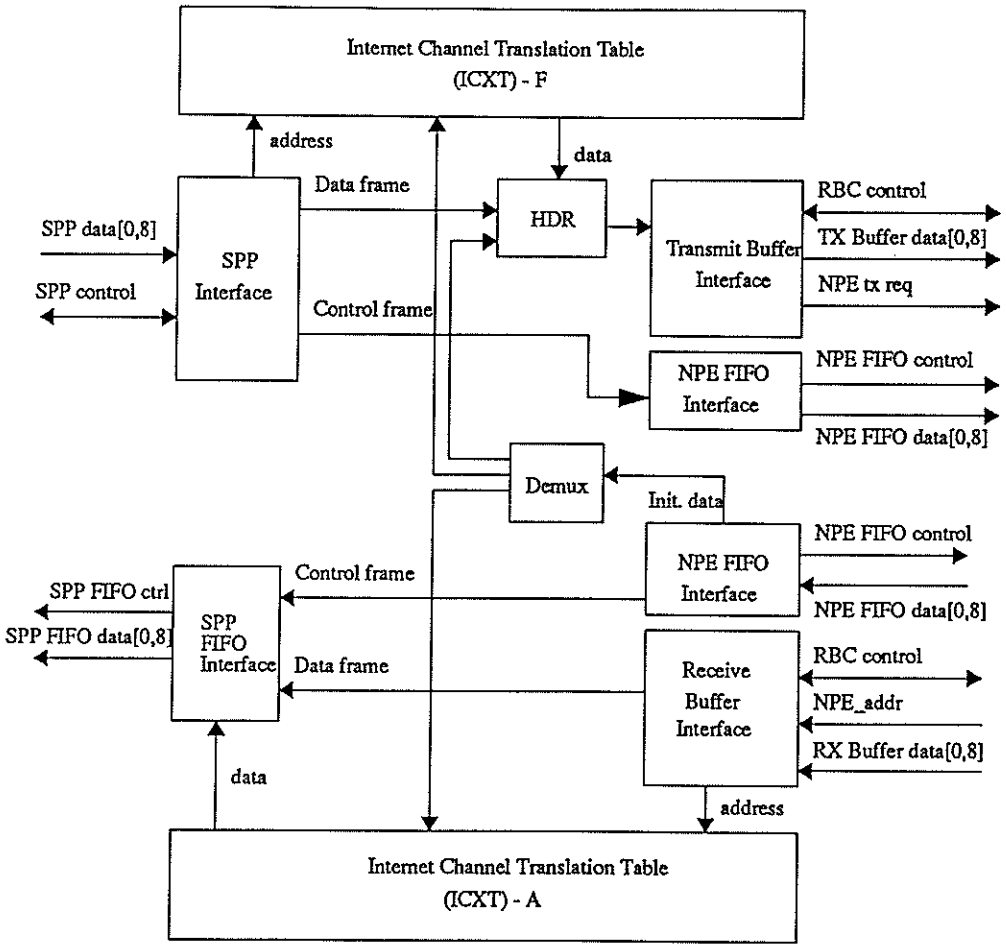

Figure 7: MCHIP Protocol Processor

the SUPERNET. Note that the RAM buffer controller (RBC) of the SUPERNET acts as a DMA controller for this transfer.

The size of the ICXT-F table is $N \times 8$ because the MPP supports a maximum of N open congrams, and for each congram, it stores a 2-byte ICN and a 6-byte FDDI destination address.

6.2 Frames from NPE and FDDI

The MPP can receive frames in the reverse direction from the NPE and SUPERNET. It receives initialization frames and ATM and MCHIP control frames from the NPE via the NPE FIFO Interface. The control and SPP initialization frames are forwarded without any modification to the SPP FIFO. MPP initialization frames are used to update the ICXT-F and ICXT-A.

For MCHIP data frames going from the FDDI to ATM network, the MPP needs to do the ICN translation and generate the ATM header. The MPP uses another lookup table called the internet channel translation table for ATM (ICXT-A) to achieve this.

The Receive Buffer Interface is responsible for reading the frame from the SUPERNET's receive buffer and stripping the FDDI specific header. It then uses the ICN as an address into the ICXT-A and forwards the rest of the MCHIP frame to the SPP FIFO interface. The SPP FIFO interface forwards the frame and the ICN and ATM header received from the ICXT-A to the SPP.

The size of the ICXT-A table is also $N \times 8$ because it stores the state information for N open congrams, and for each congram it stores a 2-byte ICN and a 5-byte ATM header. Details of the MPP have also been worked out, but are not presented here for space limitations and reasons of simplicity.

6.3 MPP Delay

The MPP is designed to operate at a 25 Mhz clock rate, with a 40ns clock cycle. We now provide the worst case static delay estimate for the MPP:

- **ATM to FDDI**: For data frames, there are two components that contribute to the delay within the MPP: delay to decode the type of the frame and make a routing decision; and delay to read the ICXT-F and construct the FDDI header. The worst case delay to decode the frame type and make the routing decision is 2 clock cycles (80ns). The read access time for the ICXT table is approximately 13 clock cycles (520ns). This gives a total processing delay of approximately 600ns for data frames through the MPP. For a control frame, there is no table lookup and the processing delay is approximately 80ns within the MPP.
- **FDDI to ATM**: The delay in this direction is almost the same as above. Data frames take 600ns and control frames 80ns.

7 CONCLUSIONS

In this paper we have presented the design of a two port ATM–FDDI gateway. In doing so, we have also emphasized the design architecture that is required to implement the next generation high speed gateways. The key to the high speed gateway architecture is partitioning of the functionality into critical and non-critical paths. The critical path is implemented in hardware as a packet processing pipeline. For the ATM–FDDI gateway, the two important components of this pipeline are a Segmentation and Reassembly Packet Processor (SPP) and MCHIP Packet Processor (MPP). Designs of the SPP and MPP have been presented in this paper. It is important to note that the latency through the gateway is minimal, and the gateway can process packets at the full FDDI rate. However, the present design does not implement any explicit rate or congestion control.

We are presently developing the simulation model for the gateway to do the functional verification of the design and to quantify its performance with various application traffic patterns. Work is also in progress in scaling the architecture of the gateway to support multiple ports.

References

[1] Advanced Micro Devices, "The SUPERNET Family for FDDI," *1989 Data Book.*

[2] American National Standard, ANSI X3.139-1987, "Fiber Distributed Data Interface (FDDI) - Token Ring Media Access Control (MAC)"

[3] Anderson, J., Parulkar, G.M., Dittia, Z., "Persistent Connections in High Speed Internets," Technical Report WUCS-91-12, Department of Computer Science, Washington University, St. Louis.

[4] Bubenik, R.G., DeHart, J.D., Gaddis, M.E., "Multipoint Connection Management in High Speed Networks," Proceedings of the IEEE INFOCOM'91.

[5] Escobar, Julio, and Craig Partridge, "A Proposed Segmentation and Reassembly (SAR) Protocol for use with Asynchronous Transfer Mode (ATM)," Proceedings of the Second IFIP WG6.1/WG6.4 International Workshop on Protocols for High Speed Networks, Stanford, California, Nov 1990.

[6] Johnson, J.M., "Proof that Timing Requirements of the FDDI Token Ring Protocol are Satisfied," *IEEE Transactions on Communications,* Vol. COM-35, No. 6, June 1987.

[7] Haserodt, Kurt, Turner, J.S., "An Architecture for Connection Management in a Broadcast Packet Network," Technical Report WUCS-87-3, Department of Computer Science, Washington University, St. Louis.

[8] D. Katz, "A Proposed Standard for the Transmission of IP Datagrams over FDDI networks," RFC-1103, June 1989.

[9] Kent, C.A., Mogul, J.C., "Fragmentation Considered Harmful," *Proceedings of the ACM SIGCOMM'87,* August 1987.

[10] Mazraani, T.Y., "High Speed Internet Protocols and Resource Management in the Ethernet," MS Thesis, Department of Electrical Engineering, Washington University, 1990.

[11] Mazraani, T.Y., Parulkar, G.M., "Specification of a Multipoint Congram-oriented High Performance Internet Protocol," Proceedings of the IEEE INFOCOM'90.

[12] Parulkar, G.M., "The Next Generation of Internetworking," ACM SIGCOMM Computer Communication Review, January, 1990. Also, Technical Report WUCS-89-19, Department of Computer Science, Washington University, St. Louis.

[13] Sevcik, K.C., Johnson, M.J., "Cycle Time Properties of the FDDI Token Ring Protocol," *IEEE Transactions on Software Engineering,* Vol. SE-15, No. 3, March 1987.

[14] Turner J.S., "Design of a Broadcast Packet Switching Network," *IEEE Transaction on Communications, Vol.36 No.6,* June 1988.