


1992

# Design of a multiprocessor high-bandwidth communication gateway based on a protocol processor pool architecture

Jangkyung Kim  
Iowa State University

Follow this and additional works at: <https://lib.dr.iastate.edu/rtd>

 Part of the [Computer Sciences Commons](#), and the [Electrical and Electronics Commons](#)

## Recommended Citation

Kim, Jangkyung, "Design of a multiprocessor high-bandwidth communication gateway based on a protocol processor pool architecture" (1992). *Retrospective Theses and Dissertations*. 10321.  
<https://lib.dr.iastate.edu/rtd/10321>

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact [digirep@iastate.edu](mailto:digirep@iastate.edu).

## INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

# U·M·I

University Microfilms International  
A Bell & Howell Information Company  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
313/761-4700 800/521-0600



Order Number 9234824

**Design of a multiprocessor high-bandwidth communication gateway based on a protocol processor pool architecture**

Kim, Jangkyung, Ph.D.

Iowa State University, 1992

**U·M·I**

300 N. Zeeb Rd.  
Ann Arbor, MI 48106



**Design of a multiprocessor high-bandwidth communication gateway  
based on a protocol processor pool architecture**

by

Jangkyung Kim

A Dissertation Submitted to the  
Graduate Faculty in Partial Fulfillment of the  
Requirements for the Degree of  
DOCTOR OF PHILOSOPHY

Department: Electrical Engineering and Computer Engineering  
Major: Computer Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Members of the Committee:

Signature was redacted for privacy.

Iowa State University  
Ames, Iowa  
1992

Copyright © Jangkyung Kim, 1992. All rights reserved.

## TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b> . . . . .	x
<b>CHAPTER 1. INTRODUCTION</b> . . . . .	1
Problem Statement and Objectives . . . . .	1
Motivation of the Research . . . . .	2
New high-speed gateway design concept is needed . . . . .	2
The advantages of the transputer approach . . . . .	3
Protocol Processor Pool Architecture . . . . .	4
Goal of the Research . . . . .	5
Design Approach Method . . . . .	6
Organization of the Dissertation . . . . .	9
<b>CHAPTER 2. BACKGROUND OF THE RESEARCH</b> . . . . .	11
Broadband ISDN . . . . .	11
Evolution of the ISDN towards BISDN . . . . .	11
SONET . . . . .	14
FDDI . . . . .	15
FDDI overview . . . . .	15
FDDI standard architecture . . . . .	15
Standard Protocols . . . . .	16

IEEE 802.2 LLC . . . . .	17
IP . . . . .	17
Q.931 . . . . .	18
Parallel Processing Architecture . . . . .	18
von Neumann-based parallel processing architecture . . . . .	19
Bus system . . . . .	20
Current Research on the High-Speed Network Communication Node . . . . .	20
High-speed protocol approach . . . . .	21
Optimized implementations of existing protocol approach . . . . .	24
<b>CHAPTER 3. DESIGN ISSUES OF THE HIPAMG . . . . .</b>	<b>29</b>
Network Protocol Architecture Design . . . . .	29
Protocol architecture of the HIPAMG . . . . .	30
Network protocol architecture . . . . .	31
Address translation . . . . .	31
Call control procedure design . . . . .	31
Analysis of the Communication Protocol Architecture . . . . .	35
Parallelism of the protocols . . . . .	35
Multi-media characteristic . . . . .	37
Multi-media applications . . . . .	37
Connectionless communication (datagram) . . . . .	40
Layered architecture . . . . .	40
Full-duplex communication . . . . .	41
Frame encapsulation . . . . .	41
Protocol Processor Pool Architecture . . . . .	42



<b>CHAPTER 4. HIPAMG HARDWARE ARCHITECTURE . . . . .</b>	<b>45</b>
MC 68332 . . . . .	45
Device overview . . . . .	45
Bus arbitration . . . . .	46
QSM queued serial module . . . . .	46
HIPAMG Hardware Architecture . . . . .	47
Hardware design issues . . . . .	47
Implementation details . . . . .	48
Packet Flow Inside the HIPAMG . . . . .	52
<b>CHAPTER 5. HIPAMG SOFTWARE DESIGN . . . . .</b>	<b>56</b>
Parallelism of the Protocols . . . . .	56
Functions of IP . . . . .	57
Execution time of the IP . . . . .	59
IP Protocol Subdivision . . . . .	59
Finite State Machine of the Subdivided IP Protocol . . . . .	60
Dynamic Path Allocation Algorithm . . . . .	62
Assumptions . . . . .	63
Idea . . . . .	64
Design . . . . .	64
Frame format . . . . .	67
<b>CHAPTER 6. MODELING AND PERFORMANCE SIMULA-</b>	
<b>TION . . . . .</b>	<b>69</b>
Modeling Parameters . . . . .	69
Assumptions of the HIPAMG Modeling . . . . .	70

Workload Model . . . . .	71
Service Time Calculation . . . . .	72
FDDI adapter service time ( $s_1$ ) . . . . .	72
Control frame transfer time ( $d_2$ ) . . . . .	73
FDDI memory manager service time ( $s_3$ ) . . . . .	73
LLC packet shared memory write time ( $s_5$ ) . . . . .	74
LLC manager service time ( $s_7$ ) . . . . .	75
LLC header shared memory read time ( $s_9$ ) . . . . .	75
LLC pool service time ( $s_{10}$ ) . . . . .	76
LLC header shared memory write time ( $s_{11}$ ) . . . . .	76
IP manager service time ( $s_{14}$ ) . . . . .	76
IP header shared memory read time ( $s_{16}$ ) . . . . .	77
IP pool service time ( $s_{17}$ ) . . . . .	77
IP header shared memory write time ( $s_{18}$ ) . . . . .	77
LLC packet shared memory read time ( $s_{21}$ ) . . . . .	77
BISDN adapter service time ( $s_{22}$ ) . . . . .	77
Development of the Simulation Model . . . . .	78
OPNET . . . . .	78
Simulation parameters . . . . .	78
Simulation procedure . . . . .	79
Structure of the HIPAMG network model . . . . .	83
Simulation period . . . . .	83
Simulation Result . . . . .	86
Results with varying traffic . . . . .	87

Results with 100 Mbps traffic . . . . .	89
<b>CHAPTER 7. PERFORMANCE ANALYSIS AND DESIGN OP-</b>	
<b>TIMIZATION . . . . .</b>	<b>91</b>
Analytical Evaluation of the Model . . . . .	91
Jackson's theorem . . . . .	91
Packet transfer delay calculation . . . . .	93
Number of packets in shared memory . . . . .	97
Throughput . . . . .	99
Simulation vs. Analytical Evaluation . . . . .	99
Improvements due to the Dynamic Path Allocation Algorithm . . . . .	100
Packet Transfer Delay . . . . .	100
Number of packets in shared memory . . . . .	101
Design Optimization . . . . .	102
Number of protocol processors in IP pool . . . . .	102
Shared memory size . . . . .	102
Cost of the HIPAMG . . . . .	103
<b>CHAPTER 8. CONCLUSIONS . . . . .</b>	<b>105</b>
Future Work . . . . .	108
<b>BIBLIOGRAPHY . . . . .</b>	<b>110</b>

## LIST OF TABLES

Table 3.1:	The relationships between communication protocol architecture characteristics and hardware architecture . . . . .	36
Table 3.2:	Characteristics of the multi-media traffic streams . . . . .	38
Table 6.1:	Structure of the HIPAMG network simulation model . . . . .	85
Table 6.2:	HIPAMG simulation results . . . . .	87
Table 7.1:	Mean queing time and delay time of the HIPAMG . . . . .	98
Table 7.2:	Simulation and analytical evaluation result . . . . .	100
Table 7.3:	Packet Transfer Delay at 100 Mbps traffic . . . . .	101
Table 7.4:	Number of packets in shared memory at 100 Mbps traffic . .	101
Table 7.5:	Optimal shared memory size at 100 Mbps traffic . . . . .	103

## LIST OF FIGURES

Figure 1.1:	HIPAMG environment . . . . .	7
Figure 2.1:	Reference model for ATM parameters . . . . .	13
Figure 3.1:	Protocol architecture of the HIPAMG . . . . .	30
Figure 3.2:	Network protocol architecture of the HIPAMG control plane	32
Figure 3.3:	Network protocol architecture of the HIPAMG user plane . .	33
Figure 3.4:	The packet call setup procedure . . . . .	34
Figure 3.5:	Three methods implementing multi-media traffic in HIPAMG	39
Figure 3.6:	Multi-media communication connections in HIPAMG . . . .	43
Figure 3.7:	Protocol Processor Pool Architecture . . . . .	44
Figure 4.1:	Block diagram of the HIPAMG hardware architecture . . . .	50
Figure 4.2:	Packet flow inside the HIPAMG . . . . .	53
Figure 5.1:	IP protocol flow diagram . . . . .	58
Figure 5.2:	Subdivided IP protocol flow diagram . . . . .	61
Figure 5.3:	Finite state machine of the IP_PROCESS_1 . . . . .	62
Figure 5.4:	Finite state machine of the IP_PROCESS_2 . . . . .	63
Figure 5.5:	Traffic flow in HIPAMG . . . . .	65
Figure 5.6:	Formal expression of the Dynamic Path Allocation Algorithm	66

Figure 5.7:	Frame formats of the HIPAMG . . . . .	68
Figure 6.1:	hipamg_1 model . . . . .	80
Figure 6.2:	hipamg_3 model . . . . .	82
Figure 6.3:	hipamg_5 model . . . . .	84
Figure 6.4:	Packet Transfer Delay of the hipamg_5 . . . . .	88
Figure 6.5:	Throughput of the hipamg_5 . . . . .	88
Figure 6.6:	hipamg_5 simulation results with 100 Mbps traffic . . . . .	90
Figure 7.1:	Queueing network model of the HIPAMG . . . . .	92
Figure 8.1:	Packet Transfer Delay of the hipamg_6 . . . . .	107
Figure 8.2:	Throughput of the hipamg_6 . . . . .	107

## ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Douglas W. Jacobson, for his insight, guidance, and encouragement for the past five years. He helped me to start this challenging research topic and guided me when I needed it most. His technical excellence and foresight has influenced me to pursue my future goals and ambitions.

I would also like to thank Dr. Johnny S. Wong for being my minor representative of computer science on my committee and for providing suggestions for improvements and insights on learning computer networks.

I would like to thank Dr. James A. Davis for being my committee and for helping me to understand the parallel processing architecture. He has influenced this work in many ways, especially by pointing out the potential problems before I meet them.

Dr. Richard E. Horton served on my committee, taking time to help me with my research. Especially, I would like to thank him for giving me the opportunity to work as a TA; without it I could not have finished this research work.

I would like to thank Dr. Way Kuo for being on my committee and taking time to help me with my research despite his busy schedule.

Dr. Prabhu served on my prelim committee and provided valuable comments in the work. He and Dr. Charles Wright also gave me good guidance and help while I was working as a Computer Science 321 TA.

Thank you all my friends and their families in this student community who made me feel rich by sharing good friendships and made me happy by sharing the good times together.

Our parents deserve more praise than I can give them. They have always been there for me. They supported and encouraged me all the time with their endless love.

Finally, I thank my wife, Youngsun, who filled my soul with hope for the future, filled my heart with love, and provided me with a sweet home where I always be able to take a rest. Thank you my son, Jiwon, for his happy smile and his bright birthday present, eraser, which has almost worn out during my dissertation writing.



## CHAPTER 1. INTRODUCTION

In the recent years, we have seen a large increase in the bandwidth of communication networks. The increase appears to continue with 100 Mbps networks available today and 1 Gbps being available in 2-3 years [1][2]. The high-speed networks currently available include FDDI, Cambridge Backbone Network (CBN), Bellcore METROCORE Network, Local Integrated Optical Network (LION), DQDB, TDM Loop, HYPER Channel-100 Network, and BISDN [3]. In these communication systems, the limited bandwidth on the physical transmission is no longer the performance bottleneck. The most significant limiting performance factor of high-speed networks is now the speed at which a processor can execute a communication protocol inside the network nodes [4]. Among these network nodes, the gateway that connects the various kinds of high-speed network will be the most important and in demand network node system because the proliferation of the high-speed networks is expected in the near future.

### Problem Statement and Objectives

Available implementations of today's gateways are not able to deal with these high data rates. The current communication node or gateway can achieve only 10 Mbit/s or less at the top of the transport layer [5]. To overcome this performance bot-

tleneck inside the gateway, a new design concept of the gateway is needed. To achieve higher throughput at the communication node, a lot of research work is currently being done. The design of high-speed protocols, which support high performance by the use of special protocol mechanisms, e.g., powerful flow control algorithms, is one approach. The other approach is to optimize the implementations of existing protocols to minimize processing time by using special VLSI processors, designing so-called high-speed adapter boards, and using general-purpose processors like the transputer [5]. But these approaches have some disadvantages that need to be solved.

This dissertation describes the Multiprocessor High-bandwidth Communication Gateway based on a Protocol Processor Pool Architecture which can solve the problems of the current approaches described above. The gateway is named High-speed Protocol processor pool Architected Multi-media Gateway (HIPAMG) and designed using the new concept in the design of the gateway architecture called a Protocol Processor Pool Architecture which has a pool of micro-controllers as its processing units. With this architecture, HIPAMG is able to achieve high throughput and flexible to the change of the networks. HIPAMG also uses the standard protocols so existing stations with standard protocols can be used without any change.

## **Motivation of the Research**

### **New high-speed gateway design concept is needed**

The high-speed protocol approach and high-speed implementation approaches described above have some disadvantages. The fact that existing stations with standard protocols can not be used is the one disadvantage of the high-speed protocol approach. Also, the high-speed protocol approaches do not show significantly better

performance than standardized protocols like OSI TP4 or TCP, because the protocols themselves are not the real performance bottleneck [6]. The reason is that the implementation of the state machine building the protocol takes only a small amount of an implementation of the whole protocol layer. Realizing the protocol state machine takes only about 20% to 30% of the entire processing time for a layer [7][8]. Thus, it seems more promising to use better implementation environments that support process scheduling and timer mechanisms in an efficient way than to design new protocols.

A VLSI approach and high-speed adapter board approaches also have disadvantages in that they mostly deal only with the protocol state machine, and thus do not necessarily improve the performance of the protocol layer. Furthermore, these approaches need to design a new special hardware which is expensive and not flexible.

Another important characteristic of the future network is its multi-media capabilities [9]. Therefore, the future gateway should not only be fast enough to work with high-speed networks but also have the architecture that handles the multi-media traffic efficiently. Besides the requirements on the performance and functionality of these gateway, other design issues such as low cost solution must be considered. Therefore, approaches that use super-computers as gateways are not acceptable for general use [10].

### **The advantages of the transputer approach**

Among the optimized implementations of existing protocol approaches, transputer approach is the most flexible approach, that is, this approach does not need to design a special hardware like VLSI processors approach or high-speed adapter board

approach. By replacing the software, it may be used in different communication protocol applications. The main advantages of the transputer approach are its capability of supporting multiple protocols simultaneously and its ability to maintain the scope for change and evolution of protocol architectures. Another big advantage of the transputer approach is the low implementation cost. But the transputer approaches developed so far are not intended to work with the high-speed network and not fast enough to accommodate the speed of the high-speed networks like FDDI and BISDN [4][11].

### **Protocol Processor Pool Architecture**

The gateway proposed in this dissertation is the HIPAMG which has a pool of micro-controllers as its processing units. The idea of the HIPAMG is originated from the transputer approach described above. The HIPAMG has Protocol Processor Pool Architecture which can handle the high-speed traffic much more efficiently than the transputer approach. The basic idea of this proposal is from the fact that the communication protocols have parallelism in a number of places. For example, between protocol layers, within individual protocol layers, and finally within the entire communication architecture. If we divide the gateway hardware architecture into many pieces and relate them properly with the components of the parallelism of the communication protocols, we can build a gateway architecture that takes advantage of the parallelism of the communication protocols. Therefore, one of the important design issues of the HIPAMG is to analyze the parallelism of the communication protocols and determine the characteristics of the communication protocols that can be implemented in hardware.

In this design, one communication layer will be processed by a pool of protocol processors and each packet will be handled by the independent protocol processors in order to increase the speed of the gateway. The different media may be processed by different protocols in the multi-media environment.

The most important idea in this proposal is the Protocol Processor Pool Architecture which significantly reduces the protocol processing time by allocating the processing jobs to many protocol processors. The dynamic allocation algorithm is used to allocate the protocol processing jobs to the protocol processors properly. Many other techniques such as the shared memory, priority-based transfer, packet pointer transfer (instead of a packet itself) and protocol subdivision will be used to achieve the design goals of HIPAMG which are the high performance, efficient multi-media handling ability, low cost, and the flexibility.

### **Goal of the Research**

The goal of this research is to design a gateway which satisfies the requirements of the high-speed communication gateway and has the following characteristics.

- Be fast enough not to be a bottleneck of the high-speed network.
- Handle the multi-media traffic effectively.
- Be cost-effective.
- Have the capability of supporting multiple protocols simultaneously.
- Be flexible to the change, so that it can maintain the scope for change and evolution of protocol architecture.

## Design Approach Method

HIPAMG is a general high-bandwidth gateway with a Protocol Processor Pool Architecture which can be used on any kind of networks. But in this research, we need to choose one specific network environment to design and simulate a detailed HIPAMG architecture. In order to design a HIPAMG, the following design approach method was used.

- **Choosing the Network Environment**

In order to design a detailed HIPAMG architecture, the network environment of the HIPAMG is needed to be selected. The BISDN and FDDI networks were chosen. Therefore, HIPAMG connects FDDI and BISDN in this research.

- **Designing a Network Protocol Architecture**

The network protocol architecture of the HIPAMG and its environment should be designed first before the details of the gateway are designed. The environment in which the HIPAMG works is shown in Figure 1.1. HIPAMG functions as a  $NT2+TA$  between the FDDI and  $T_B$  interface of the BISDN.

To connect the FDDI station and BISDN station, the network configuration and protocols for connection oriented packet switching mode were chosen. The protocol architecture of the gateway and the network architecture of the system will be discussed in detail in Chapter 3.

- **Analyzing the Communication Protocol Architecture**

The good parallel protocol implementation of the gateway can be achieved when the underlying multiprocessor architecture and the way a communication protocol architecture is specified are properly matched. To achieve this,

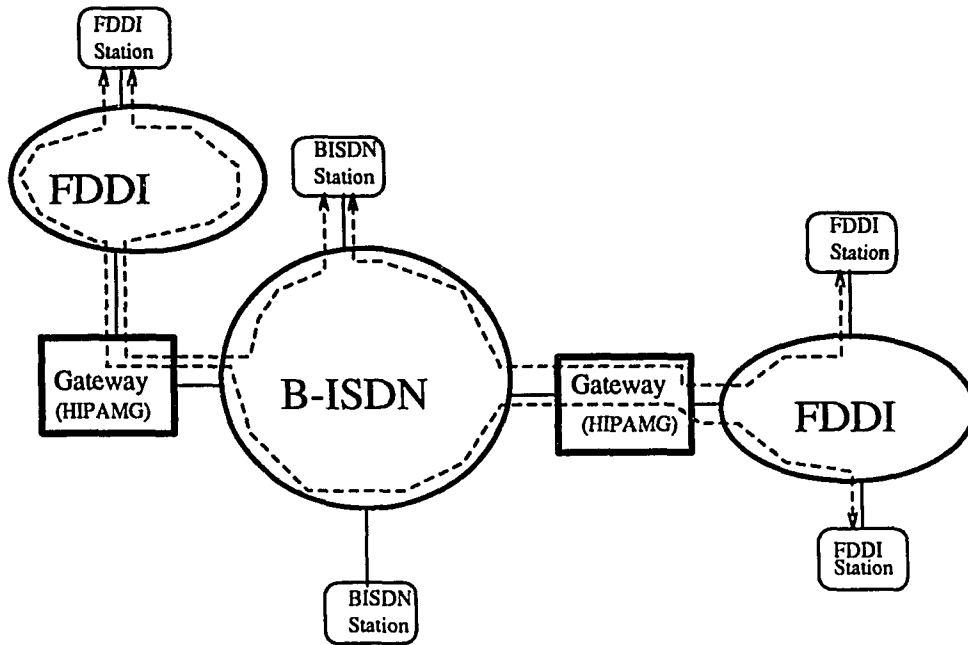


Figure 1.1: HIPAMG environment

the communication protocol architecture of the HIPAMG environment is analyzed and the relationship between the communication protocol architecture and underlying HIPAMG hardware architecture are studied.

- **Finding the Parallelism within the Protocols**

The standard protocols used in this gateway includes IP, LLC, FDDI MAC (X3T9.5), FDDI Physical layer (X3T9.5), Q.931, LAPD, ATM Adaptation Layer (AAL), ATM layer, and PMD layer. These protocols have many parallel factors within the individual protocols. In the Protocol Processor Pool Architecture, each communication layer is executed on the different protocol processor pool. Even within the same layer, many processors are dedicated to each function of the protocols. Therefore, one of the important procedure in

designing this gateway is to find out the parallel factors inside the protocols. In this research, the IP layer is studied and subdivided to make the project manageable.

- **Hardware Design Using Protocol Processor Pool Architecture**

The hardware design is based on a horizontal and vertical subdivision of the communication systems. The HIPAMG architecture consists of several building blocks and supports shared memory concepts. Motorola MC68332 [12] was selected as the processors of this gateway because of its low price, easy accessibility, design flexibility, and separate serial communication link ability. The detailed hardware design of the gateway is performed with the results of the parallelism found from the protocols. New design concept of the gateway architecture in the high-speed multi-media environment so-called Protocol Processor Pool Architecture is proposed to be implemented in the HIPAMG. The main issues in this design procedure are the best hardware granularity, shared memory system implementation, communication method between processors, and priority scheme for different media.

- **Software Design**

Most of the software design is based on the standard protocols. But some modifications to the standard protocols are needed. The main issues in this design procedure are the parallelism of the protocols, IP protocol subdivision, FSM of the subdivided protocol, the Dynamic Path Allocation Algorithm, and the frame format.



- **OPNET Simulation and Design Optimization**

After completing the gateway design, the performance of the gateway was simulated using the OPNET graphic simulator which is the contemporary tightly-coupled CAE system developed by MIL 3, Inc [13]. By using the OPNET we can simulate the gateway system down to the process level, which means the simulation model is very close to the real hardware architecture, that is, minimal abstraction is used inside the simulation model. After the simulation, the analytical evaluation of the HIPAMG model is performed and the results of the simulation and the analytical evaluation are compared. Lastly, the design parameters like the shared memory size, and the number of protocol processors in IP pool are tuned to optimize the gateway design.

### **Organization of the Dissertation**

In Chapter 2, the related backgrounds of this research are discussed. BISDN and FDDI are introduced briefly, then all the standard protocols used in this gateway are discussed. Some topics of the parallel processing architecture is discussed. Lastly, the current research on the high-speed network communication node are explained.

Chapter 3 contains the design issues of the HIPAMG. First, the network architecture design of the HIPAMG is discussed. Second, the communication protocol architecture of the HIPAMG is analyzed to determine the relationships between the communication architecture and HIPAMG hardware architecture. Chapters 4 and 5 describe the HIPAMG hardware and software design issues respectively.

In Chapter 6, modeling issues of the HIPAMG are discussed. Then the performance simulation result of the HIPAMG using OPNET is discussed.

Chapter 7 discusses the analytical evaluation of the HIPAMG model and the performance analysis of the HIPAMG. After that, the design optimization issues are discussed.

Lastly, Chapter 8 concludes this dissertation by discussing the contribution of this research and future work.

## **CHAPTER 2. BACKGROUND OF THE RESEARCH**

In this chapter, the background which is needed to design the HIPAMG is described. Broadband ISDN, FDDI, and some other standard protocols used in HIPAMG are discussed. Then, some topics of the parallel processing architecture are introduced. Lastly, current research on the high-speed network communication node is described.

### **Broadband ISDN**

#### **Evolution of the ISDN towards BISDN**

In 1984, the Plenary Assembly of the CCITT adopted the I series recommendations dealing with ISDN matters. CCITT stated that “an ISDN is a network....that provides end-to-end digital connectivity to support a wide range of services, including voice and non-voice services, to which users have access by a limited set of standard multi-purpose user-network interfaces [14]”. Such an ISDN standard interface was defined (and called basic access), comprising two 64 Kbps B channels and a 16 Kbps signaling D channel. Another type of interface, the primary rate access, with a gross bit rate of about 1.5 Mbps or 2 Mbps, respectively, offers the flexibility to allocate high-speed H channels or mixtures of B and H channels.

The need for services employing bit rates greater than 2 Mbps was clearly seen

when the I series recommendations were written. Therefore, with the ink hardly dry on the first definitive set of ISDN standards, much of the planning and design effort is now directed toward a network concept that will be far more revolutionary than ISDN itself. This new concept has been referred to as Broadband ISDN (BISDN) [15].

While dedicated networks require several distinct costly subscriber access lines, the BISDN access can be based on a single optical fiber for each customer. To meet the requirement for high-resolution video, an upper channel rate of approximately 150 Mbps will be needed. To simultaneously support one or more interactive services and distributed services, a total subscriber line rate of about 600 Mbps is needed. In terms of today's installed telephone plant, this is a very large data rate to sustain. The only appropriate technology for widespread support of such data rates is optical fiber and CCITT defines Asynchronous Transfer Mode (ATM) as the target switching technique toward the BISDN in Recommendation I.121.

**Worldwide unique NNI** The discussion in BBTG began by the definition of the User-Network Interface (UNI) structure and H2 and H4 channel rates. A number of proposals were made on H2 (30-45 Mbps) and H4 (132-138 Mbps) rates, but the agreement could not be reached. For the definition of the basic structure of UNI, two interface bit rates were selected: 155.520 Mbps and 622.20 Mbps in harmonization with Network-Network Interface (NNI) bit rate.

Standardization of the UNI structure at T and S reference points started in the current 1989-1992 study period. As illustrated in Figure 2.1 [16], NT2 for BISDN (B-NT2) may be a distributed type like LAN or may be a centralized type like PABX,

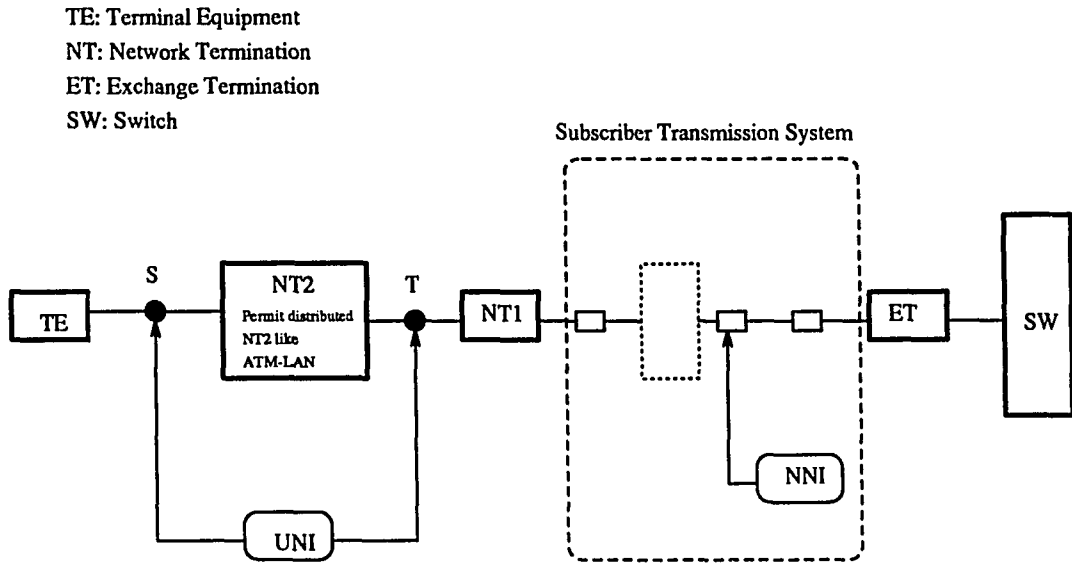


Figure 2.1: Reference model for ATM parameters

and the physical layer of the subscriber network may be based on SDH transmission systems or may be based on existing transmission systems.

**ATM network architecture** ATM is a specific packet oriented transfer mode using asynchronous time division multiplexing technique. The multiplexed information flow is organized in fixed size blocks called cells. Fixed size was selected over variable size because, based on the state of the existing experimental fast packet switching technology, it is believed that fixed size cell can be switched more efficiently. ATM networks provide huge bandwidth with low error rates using optical fiber. In such a high-speed network environment, processing time becomes bottleneck. The conventional OSI 7 layer protocol architecture may be too heavy, i.e., it involves too much processing, and thus a new protocol architecture is needed.

Layered architecture for ATM networks have been studied and some agreement

has been made in the CCITT Study Group XVIII meeting held in June 1989. The protocol hierarchy of the BISDN ATM protocol model consists of the physical-medium-independent (PMD) layer, the ATM layer, the adaption layer (AAL), and the higher service layer. Note that functional layering in the BISDN protocol model does not follow the OSI model.

As mentioned previously CCITT will standardize two physical interfaces to BISDN, one based on SONET and the other based on a variation of ATD. This layer is responsible for the proper bit transmission and performs functions which are necessary to insert/extract the cell flow into/out-of a transmission frame. This layer is also responsible for electro-optical conversion since in BISDN, the physical medium is optical fiber.

## SONET

Synchronous Optical Network (SONET) is the name of newly adopted standard, originally proposed by Bellcore for a family of interfaces for use in Operating Telephone Company (OTC) optical networks. SONET defines standard optical signals, a synchronous frame structure for the multiplexed digital traffic, and operations procedures.

The basic building block and first level of the SONET signal hierarchy is called the Synchronous Transport Signal-level 1 (STS-1). The STS-1 has a bit rate of 51.84 Mbps and is assumed to be synchronous with an appropriate network synchronization source [17]. No physical interface for the STS-1 signal has been defined as yet; the Optical Carrier-level 1 (OC-1) is obtained from the STS-1 after scrambling (to avoid long strings of ones and zeros and allow clock recovery at receivers) and electrical-

to-optical conversion. In BISDN transmission, ATM will be carried within SONET. BISDN proposals using SONET usually use the STS-3c frame. SONET overhead is not embedded within the cell structure, and the SONET payload carries ATM cells multiplexed using ATM techniques.

## **FDDI**

### **FDDI overview**

The accredited standard committee ASC X3T9.5 has produced a standard referred to as Fiber Distributed Data Interface (FDDI) in 1984 [18]. FDDI is a standard for a high-speed ring LAN. Like the IEEE 802.5 standard, FDDI employs the token ring algorithms. There are, however, several differences that are intended to allow FDDI to take advantages of the high speed of its optical fiber ring and maximize efficiency. Transmitting over fiber optic media, FDDI provides a throughput of 100 Mbps and allows a large distance between the two nodes; up to 12 km in multimode fibers and 20 km in single mode fiber. FDDI allows backbone applications where computers and workstations connected to medium speed LANs can exchange data with other LANs or LAN segments in an FDDI extended network configuration. FDDI allows also back-end and front-end applications.

### **FDDI standard architecture**

The FDDI standard encompasses both the MAC layer and the physical layer. This standard assumes the use of the IEEE802.2 standard, LLC [19]. The standard is in four parts:

- Medium Access Control
- Physical Protocol
- Physical Medium Dependent
- Layer Management

The MAC service specification defines in functional terms the service provided by FDDI to LLC or any other higher-level user. The physical protocol (PHY) is the medium-independent portion of the physical layer. This includes a specification of the service interface with MAC. The PHY protocol specifies the NRZI-4B/5B code encoding of digital data for transmission. The physical medium dependent (PMD) sublayer of the physical layer defines and characterizes the fiber optic drivers and receivers, and other medium-dependent characteristics of the attachment of stations to the ring and of the cabling and connections of the ring. Layer management (LMT) provides the control necessary at the station level to manage the processes underway in the various FDDI layers such that a station may work cooperatively on a ring. LMT is part of a broader concept, referred to as station management (SMT), which defines all the management issues of the FDDI including LLC layer and above the LLC layer.

### **Standard Protocols**

HIPAMG includes many standard communication protocols like FDDI Physical layer, FDDI MAC, IEEE 802.2 LLC, IP, Q.931, LAPD, ATM, AAL, and PMD layer. Among these, FDDI Physical layer, FDDI MAC, LAPD, ATM, AAL, and PMD



layer will be assumed to be implemented inside the Adapter Board. Therefore, IEEE 802.2 LLC, IP, and Q.931 are going to be implemented by MC68332 and supporting software. In this section, these protocols which will be implemented by protocol processor pool are described briefly.

### **IEEE 802.2 LLC**

IEEE 802.2 Logical Link Control (LLC) is the highest layer of the local network communications architecture. It is used above all of the Medium Access Control (MAC) standards specified by IEEE 802 and by FDDI. The primary purpose of this layer is to provide a means of exchanging data between LLC users across MAC-controlled link. The LLC standard provides three forms of service to LLC users:

- Unacknowledge connectionless service
- Connection mode service
- Acknowledged connectionless service

All these services are defined in terms of the primitives and the parameters that are exchanged between the LLC entity providing the LLC service and the LLC users that are identified by LLC Service Access Points (SAPs).

### **IP**

The DOD IP, MIL-STD-1777, was developed as part of the DARPA Internet Project. IP provides a connectionless, or datagram, service to IP users (e.g., ISO TP) in stations attached to networks of the internet. Two primitives are defined at the user-IP interface. The IP user requests transmission of a unit of a data with

N-UNITDATA.request. N-UNITDATA.indication is used by IP to notify a user of the arrival of a data unit.

The function of IP includes address translations, routing, datagram lifetime control, fragmentation and reassembly, error control, and flow control. DOD has defined an Internet Control Message Protocol (ICMP), which is a required companion to IP. Basically, ICMP provides feedback about problems in the communication environment.

### **Q.931**

Q.931 (I.451) is a standard for common channel signaling developed by CCITT. The primary application of this standard is for the ISDN. The channel that is reserved for the transmission of control information is referred to as the D channel. Q.931 is the control signaling protocol that is used on the D channel. In OSI terms, Q.931 is a layer 3, or network layer, protocol. It specifies procedures for establishing connections on the B channels that share the same interface to ISDN as the D channel. It also provides user-to-user control signaling over the D channel. Q.931 relies on a link layer protocol to transmit messages over the D channel. Each Q.931 message is encapsulated in a link layer frame. The link protocol is LAPD (I.441) which is very similar to HDLC.

## **Parallel Processing Architecture**

The concept of parallel processing is found in the literature at least as far back as the 1920s [20]. After that, there has been a continuing research effort to understand parallel computation. Such effort has intensified dramatically in the last few

years, with hundreds of projects around the world involving scores of different parallel architectures for all kinds of applications.

There are three basic approaches to parallel computation: von Neumann-based, dataflow, and reduction approaches. A further approach is a hybrid of data flow and reduction. HIPAMG architecture is based on the von Neumann-based approach and this approach is described in this section.

### **von Neumann-based parallel processing architecture**

The von Neumann approach to parallel processing consists of interconnecting two or more von Neumann-type uniprocessors in a variety of configurations. These von Neumann-based parallel processing systems are classified according to how they process the program instruction and data streams: Single Instruction Single Data (SISD), Multiple Instruction Single Data (MISD), Single Instruction Multiple Data (SIMD), and Multiple Instruction Multiple Data (MIMD) systems.

A very important part of the architecture of a parallel processing system is its interconnection network [21][22]. The communications subsystem linking in general processors, memory modules, and I/O controllers in a parallel processing system is one of its most important architectural areas. There are two basic architectural alternatives for the communication system: bus structure and network structure. A shared bus provides the simplest communications subsystem with adequate performance if each processor has its own cache memory and if the number of processors is no more than about 32 with present bus and memory technologies.

For large numbers of processors, the bus bottleneck is eliminated by using a communications network instead of the bus to provide the desired connectivity and

performance. The cross-bar network and the interconnection network are the two approaches of the interconnection network. In the HIPAMG design, the bus structure is used to interconnect the shared memory and many process modules.

### **Bus system**

In a bus-based system, data transfer operations are controlled by the bus interfaces of the sender and receiver. The sender must determine the availability of the bus and then interrogate the destination to establish its readiness to receive the transfer before initiating it. The receiver recognizes its address and responds to the requests of the sender. Due to conditions for the bus, a mechanism must be provided for conflict resolution. The technique used to resolve the bus conflict is bus arbitration. The details of the bus arbitration used in HIPAMG design are described in Chapter 4.

### **Current Research on the High-Speed Network Communication Node**

With the advent of high-speed networking technologies such as fiber optics, a traditional bottleneck in communication, the limited bandwidth of the physical transmission media, has disappeared. Now the processing of communication protocols inside the network nodes is the most significant limiting performance factor of high-speed networks. The network technology offers 100 Mbps or more; however, at the top of the transport layer, currently only 10 Mbps or less will be achievable [5]. A performance loss in the same order of magnitude will hold for the application layer, where a throughput of less than 1 Mbps seems to be realistic. To overcome this performance bottleneck inside the communication software, research work is currently

done on the communication node or high-speed gateway systems in order to improve the protocol processing speed inside the high-speed gateway systems.

Currently, there exist mainly two approaches to achieve high-performance communication node that will be suitable for the future high-speed communication systems. They are the high-speed protocol approach and optimized implementations of existing protocols approach.

### **High-speed protocol approach**

High-speed protocols are new design of communication protocols which support high performance by the use of special protocol mechanisms such as powerful flow control algorithms. Six types of high-speed protocols are described in this subsection. They are Versatile Message Transaction Protocol (VMTP), Xpress Transfer Protocol (XTP), Delta-t, Horizontally Oriented Protocol Structure (HOPS), Very High Speed Internet (VHSI), and Network Block Transfer Protocol (NETBLT).

**VMTP** VMTP was developed within a project at the Stanford University in order to improve the deficiencies in current transport protocols: performance, naming, and functionality. VMTP provides transport communication between network-visible entities via message transactions. It is mainly tuned for traffic patterns as they occur in RPC-based communication environments. Special features of VMTP are the support of multicast communication and the location-independent addressing scheme, which supports process migration. Timer -based connection management as well as rate-based flow control and selective acknowledgement are supported [23].

**XTP** XTP is a recently developed protocol for next generation high-speed networks by Silicon Graphics Inc. XTP was developed with the special goal of implementing it in VLSI. It especially supports real-time datagrams and multicasting. The core of XTP is a light-weight protocol based on the header address and sequence number plus bit flags in the trailer. XTP comprises the functionality of OSI layers 3 and 4. It provides a flexible addressing scheme, which supports the use of different address formats. The long term goal for XTP is to achieve a compact VLSI representation of the complete design.

**Delta-t** Delta-t is a transport protocol designed to support both request-response and stream style of communication in high performance networks and distributed systems [8]. The development of Delta-t is quite old; it started in the late 70's. At the Lawrence Livermore National Laboratory (LLNL), an integrated network and distributed operating system architecture called Livermore Integrated Network Computing System (LINC'S) has been developed [24]. LINC'S was designed to integrate a wide range of heterogeneous micro to super computer systems. Delta-t's design goal was to allow complete requests, replies, or large data buffers to be sent with exactly two packets in the usual case, one packet for the data and one for an ACK. No other packets are required for connection opening or closing.

The functionality of Delta-t is split into a connectionless network level protocol and a transport level protocol. The former handles those services provided in datagram routing nodes (including routing software in the ends) and the latter those services needed just in the ends. Delta-t can be implemented on other connectionless network protocols such as DARPA's IP or ISO's CLNP.

**HOPS** HOPS was designed at the AT&T Bell Laboratories to improve the performance of the current high level protocols [25]. HOPS is an alternative approach to the existing architected models. The main idea behind HOPS is the division of the protocol into functions instead of layers. The functions, in general, are mutually independent in the sense that the execution of one function can be performed without knowing the results of the execution of another. Thus, intercommunication between the functions is substantially reduced. Because of the independence between the functions, they can be executed in parallel, thus reducing the latency of the protocol and improving throughput. The architecture is based on three layers: Network Access Control (NAC), Communication Interface (CI), and Application. CI of HOPS implements in hardware the services defined by layers 4 to 6. HOPS can be implemented as a collection of custom-designed hardware and general-purpose processors.

**VHSI** VHSI has been proposed at the Computer and Communications Research Center of the Washington University [26]. In the ARPA Internet and ISO models, the internet level is responsible for providing a homogeneous networking abstraction on top of diverse networks [27]. The existing internet abstraction is based on best effort datagram delivery which is becoming increasingly outdated for a number of reasons: it can not work well with connection-oriented high speed networks; it does not do any explicit resource management, and thus can not provide variable grade service with guarantees to different applications; and its gateway architecture are not designed to work at very high speeds. VHSI abstraction has been proposed in order to meet these challenges [28]. An important component of the VHSI abstraction is a novel Multipoint Congram-Oriented High Performance Internet Protocol (MCHIP).

Features of this protocol include support for multipoint communication, the Congram as the service primitive which incorporates strengths of both connection and datagram approaches, ability to provide a variable grade of service with performance guarantees, and suitability for high speed implementation.

**NETBLT** NETBLT was developed for high-throughput bulk data transfer at MIT. The connections in NETBLT are unidirectional. An interesting point of this protocol is the separation of data and control flow, which allows an efficient independent implementation of both. Flow control is based on a combination of a window algorithm and rate control. In contrast to that, standard protocols generally use a window-based flow control. Furthermore, NETBLT supports a selective acknowledgement strategy.

### **Optimized implementations of existing protocol approach**

Even though new high-speed protocols have been invented as described in previous subsection, some recent work shows that clever tuning and implementation of existing protocol architectures can also deliver high throughput and quick response time. These implementations has three different approaches: using special VLSI processors, designing so-called high-speed adapter boards, and using general-purpose processors like the transputer. The problem of these three approaches is that they mostly deal only with the existing protocol state machine, and thus do not necessarily improve the performance of the protocol layer. Furthermore, there is still the problem of flexibility of VLSI implementations, including the support for multiple connections on a single chip.



**VLSI processors approach** The goal of using special VLSI chips is to increase the performance and to find an automatic way of translating protocol specification into an implementation.

The first example of the VLSI processors approach is Protocol Silicon compiler (PSi) developed by IBM Watson Research Center at Yorktown Height [29]. The PSi transforms formal protocol specifications into efficient VLSI implementations. PSi consists of software tools that transform high-level specifications of processing elements into efficient and correct VLSI layouts. The two goals of PSi research are to accomplish:

- very high-speed protocol implementations for arbitrary protocols.
- simplified protocol implementation process leading to reduced development time and increased reliability and uniformity of the resulting implementations.

Another VLSI processors approach is Protocol Engine (PE) developed by Silicon Graphics Inc. in connection with the XTP project [7]. XTP was designed with a PE in mind. PE is a hardware architecture for implementing network protocols and system interfaces using VLSI techniques.

There are some other VLSI processors approaches; Modular Communication Machine (MCM) was developed by IBM Watson Research Center at Yorktown [30] and AT&T Bell Laboratory developed the method to translate the protocol specifications into VLSI.

**High-speed adapter board approach** High performance computer communication between multiprocessor nodes requires significant improvements over conventional host-to-network adapters. Current host-to-network adapter interfaces impose

excessive processing, system bus and interrupt overhead on a multiprocessor host. Current network adapters are either limited in function, wasting key host resources such as the system bus and the processors, or else intelligent but too slow. Although processor and memory cycle times keep improving, with communication networks moving to gigabit range, we expect the processing to persist as a bottleneck unless significant improvements in network adapter board and transport protocol designs are achieved.

The Network Adapter Board (NAB) [31] for the VMP multiprocessor [32] was designed to solve above problems at Stanford University. The adapter host interface of the NAB is designed for minimal latency, minimal interrupt processing overhead and minimal data transfer on the system bus. The prototype NAB has been designed using Motorola's MC68020 as the on-board processor, running at 16Mhz clock rate; it uses about 200 hundred standard MSI and LSI components.

Another high-speed adapter board approach is Petrinet-Controller developed by Aachen University [33].

**Transputer approach** The last implementation approach is the general purpose processor approach using INMOS transputer. A transputer is a micro computer with its own local memory and with links for connecting one transputer to another transputer [34][35]. A concurrent system can be constructed from a collection of transputers which operate concurrently and communicate through serial communication links. Both of the transputer approach and the high-speed adapter boards approach are supported by the well suited implementation system adding low overhead to an implementation. The main design issues for the transputer approaches

are using parallelism and realizing a general purpose solution.

The university of Erlangen and IBM R schlikon mainly focus on the implementation of the OSI LLC protocol [4][36]. Hence, they do not support parallelism inside the protocol state machine; but both have built a global memory for transputers, which seems to be necessary for high-performance protocol implementations. The aim of the IBM R schlikon transputer approach is to achieve high performance even when running traditional protocols such as the OSI transport protocol or TCP/IP over FDDI rings (100Mbps) or experimental precursors of BISDN (140 Mbps H4 Channel), e.g. the BERKOM project [37]. Instead of choosing a highly specialized solution, such as the VMP NAB designed specially for VMTP, or the chip set designed for XTP, they decided to develop a general purpose architecture that allows software implementation of different communication protocols. They first exploited the inherent parallelism in communication architectures. Then based on the parallelism in their architectures, a number of protocols were implemented as prototypes on a transputer-based multiprocessor system. The comparatively slow serial transputer links, however, proved to be unsuitable to carry protocol data between the processors of such a multiprocessor system. A shared memory architecture was therefore developed which allows protocol data to be copied directly from the high-speed network interface into a frame memory. This frame memory is shared between the network interfaces, the protocol processors and the host system. Copying of protocol data can therefore be avoided altogether. The result of this implementation shows that it is possible to have pure software implementations of protocols, that can exploit the full bandwidth of the emerging 100 Mbps networks.

In contrast to the previous approach, the approach at the University of Karlsruhe

supports parallelism based on the level of protocol functions, including a global memory concept [11][38]. The design is based on a horizontal and vertical subdivision of communication systems. Transputer networks form the basis of this prototype implementations. A parallel C [39] was used as programming language on the transputers. The parallel architecture consists of several building blocks (e.g. pipeline, array of processors) and supports multiple memory concepts (local and global memory).

### **CHAPTER 3. DESIGN ISSUES OF THE HIPAMG**

In this chapter, some design issues of the HIPAMG are discussed. First, the network protocol architecture design is discussed. Then communication protocol architecture of HIPAMG is analyzed to find out the relationships between the communication protocol architecture and the HIPAMG hardware architecture. Lastly, some issues about these relationships are discussed.

#### **Network Protocol Architecture Design**

The network protocol architecture of the HIPAMG and its environment was decided first before the details of the HIPAMG is designed. The network environment of this project was decided to achieve the communication between the FDDI station and the BISDN station using packet switched call control on the SONET as shown in Figure 1.1. In this network, the BISDN is used as a transparent network between the HIPAMG and the BISDN station. In this design, the basic functions of the HIPAMG are to decapsulate the IP packet from the FDDI MAC packet, and to wrap an IP packet with BISDN protocol header to form an BISDN packet and to deliver the resulting packet to the destination station. At the destination station, the original IP packet will be recovered by simply dropping the BISDN header. With this concept in mind, the protocol architecture of the HIPAMG and its environment network have

IP		
	CONTROL	USER
*	Q.931	*
LLC	LAPD	
FDDI MAC	AAL	AAL
FDDI PHYS	ATM	ATM
	PMD	PMD

Figure 3.1: Protocol architecture of the HIPAMG

been designed.

### Protocol architecture of the HIPAMG

The protocol architecture of the HIPAMG is shown in Figure 3.1. As shown in this figure, the communication protocols of the HIPAMG include FDDI Physical layer, FDDI MAC, IEEE 802.2 LLC, IP, Q.931, LAPD, AAL, ATM, and PMD layer. This figure also shows that the BISDN side of the protocol architecture consists of two parts: control plane and user plane. The control plane is used when it creates and clears the call connections. The user plane is used when it transfers data.

## Network protocol architecture

The network protocol architectures of this project are shown in Figure 3.2 and Figure 3.3. As shown in Figure 3.2, the control plane of the network architecture is used to connect or clear the call connections.

The function of the HIPAMG is to connect the FDDI station to Local Exchange of the BISDN. FDDI station and HIPAMG have IP, LLC, FDDI MAC, and FDDI physical layer in common. The HIPAMG and Local Exchange are connected through the NT1 using  $T_B$  interface. The common protocols of the HIPAMG and Local Exchange on the control plane are Q.931, LAPD, AAL, ATM, and PMD.

Figure 3.3 shows that the user plane doesn't have the link-by-link level protocols like LAPB. This is because it was proved that edge-to-edge scheme performs better than the link-by-link scheme in ATM networks, where the effects of propagation delay and processing time are significant [40].

## Address translation

When an IP datagram arrives at the HIPAMG, the HIPAMG analyzes the IP header to determine whether this datagram contains control information intended for the gateway, or data intended for the destination station. In the latter case, the HIPAMG carries out address translation by performing a table lookup.

## Call control procedure design

The destination BISDN station's BISDN number obtained via address translation is provided to the control plane's Q.931 *call control procedures for packet switched calls* to establish and terminate ATM packet switched virtual connection between the

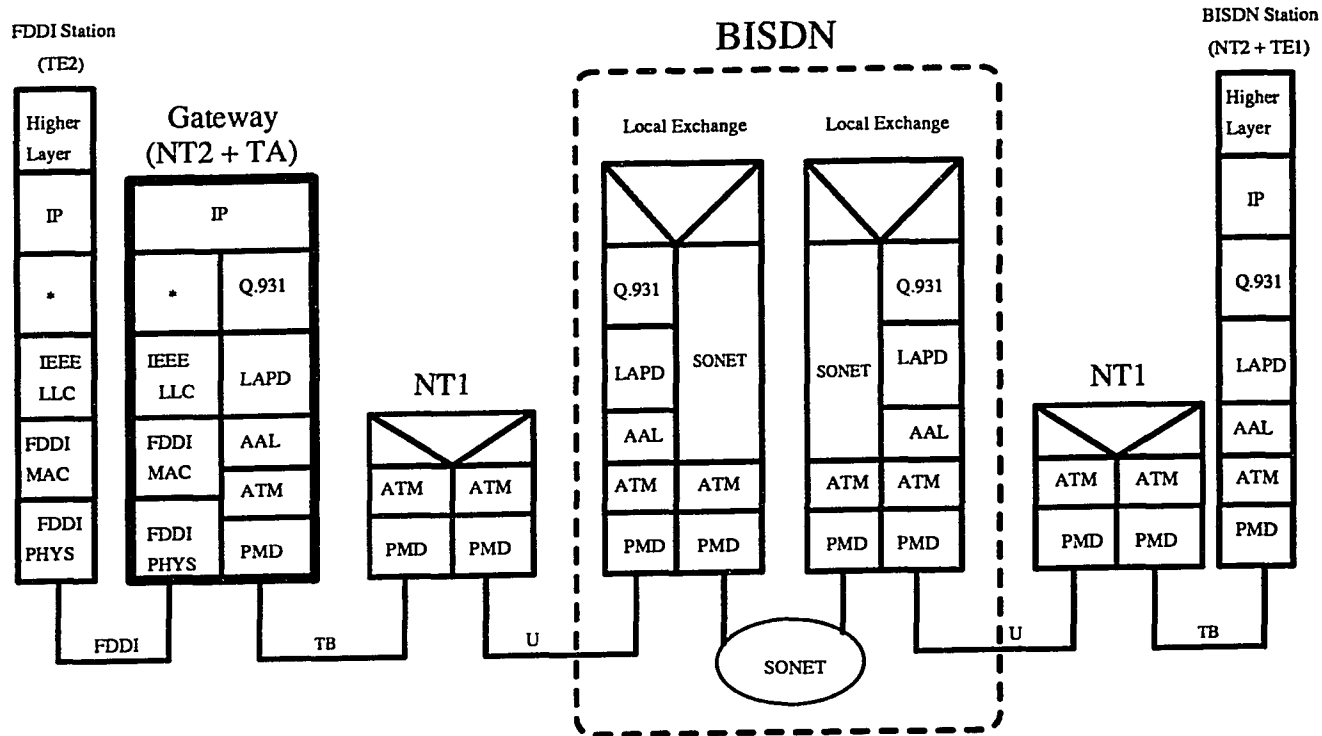


Figure 3.2: Network protocol architecture of the HIPAMG control plane



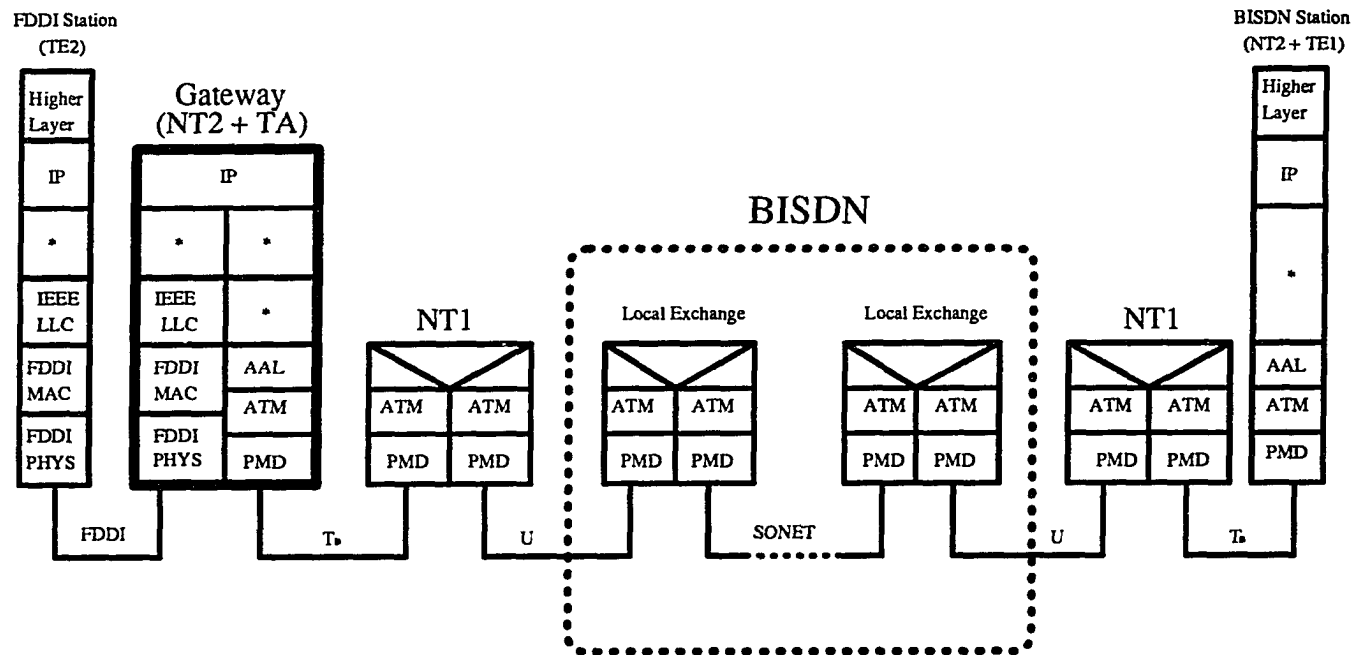


Figure 3.3: Network protocol architecture of the HIPAMG user plane

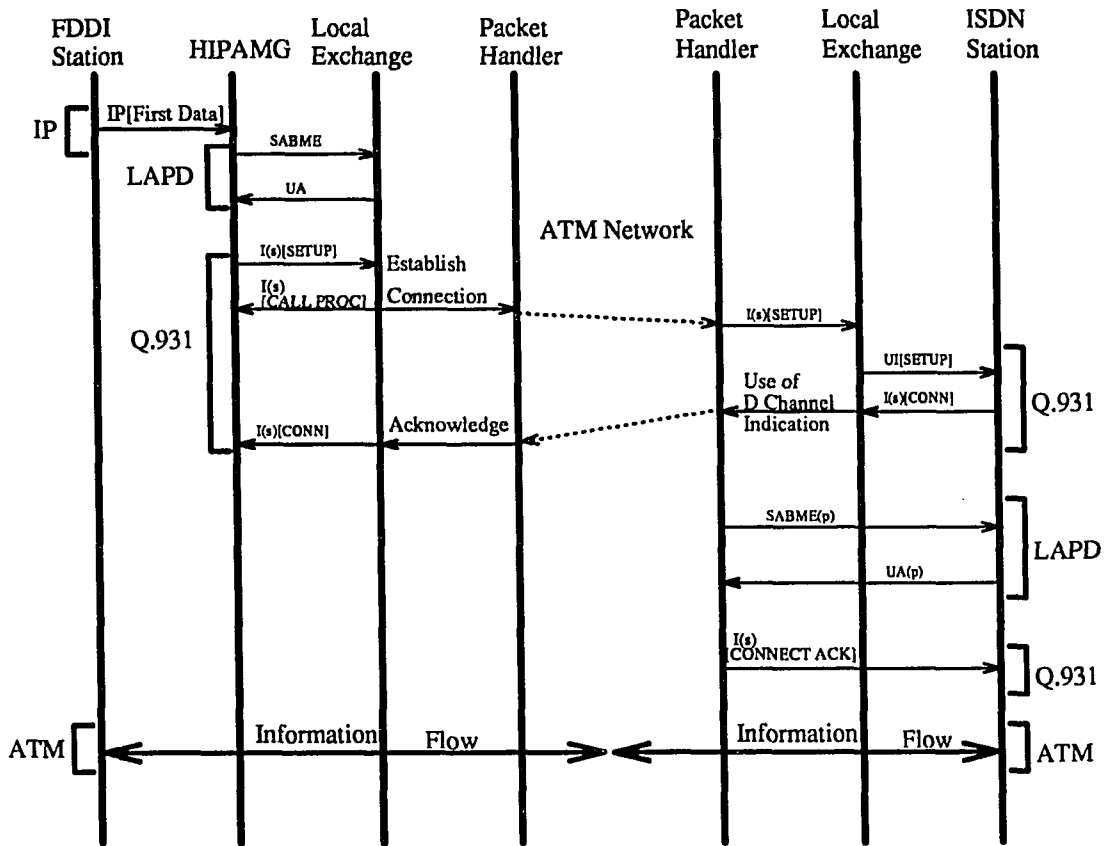


Figure 3.4: The packet call setup procedure

HIPAMG and the destination BISDN station [41]. The packet call setup procedure are shown in Figure 3.4.

As mentioned above, before data can be sent between interconnected stations the ATM packet switched virtual connection must be setup. However, since IP is a connectionless protocol, it gives no indication when to open and close a connection. The approach to solve this problem is to let the HIPAMG monitor the incoming IP datagrams, and the first datagram destined for a remote station triggers the circuit

to initiation of a virtual connection setup to the destination station. Once the virtual connection is established, it is used by the gateway to deliver subsequent datagrams to the remote station. However, since IP provides no disconnect information, the gateway must make a decision when to terminate the virtual connection based on its perception that traffic on the virtual connection has ceased. The simple connection management algorithm for this application was developed in [42]. Therefore, to implement this algorithm to the HIPAMG design, the modification of the standard IP protocol is needed.

### **Analysis of the Communication Protocol Architecture**

The best parallel protocol implementation of the parallel architected gateway can be achieved when the underlying multiprocessor architecture and the way a communication protocol architecture is specified are properly matched. To achieve this, the communication protocol architecture of the HIPAMG is analyzed and the relationship between the communication protocol architecture and underlying HIPAMG hardware architecture are studied. The characteristics of the communication protocol architecture that can be related to the hardware architecture of the HIPAMG are tabulated in Table 3.1 and explained in detail in the rest of this chapter.

#### **Parallelism of the protocols**

Communication protocol architectures are typically structured and described in the form of hierarchical protocol layers as exemplified by the OSI reference model. Such architectures exhibit parallelism in a number of places: between protocol layers, within individual protocol layers, and finally within the entire communication archi-

Table 3.1: The relationships between communication protocol architecture characteristics and hardware architecture

Communication architecture	Hardware architecture
Parallelism of the protocols	Multiple protocol processors
Multi-media characteristic	Multiple communication paths
Connectionless communication	Processor pool architecture
Layered architecture	Pipeline architecture
Full-duplex communication	Separate Rx and Tx hardware
Frame encapsulation	Shared memory and packet pointer transfer

ture. Therefore, to design a parallel architected gateway, we should find out the parallelisms inside the protocols, divide the protocol functions into many pieces and assign each function to a hardware component of the parallel architected gateway.

**Protocols of the HIPAMG** HIPAMG includes many protocols like IP, LLC, FDDI MAC, FDDI Physical, Q.931, LAPD, AAL, ATM, and PMD. Among these protocols, I assume the FDDI MAC, FDDI Physical, LAPD, AAL, ATM, and PMD are implemented in the adapter board. Therefore, IP, LLC, and Q.931 should be implemented using the protocol processors. In this section, I will show how the protocols are divided into many pieces and implemented by giving the examples of IP protocol.

**Parallelisms in IP** As described in Chapter 2, IP has many functions like addressing, routing, datagram lifetime, fragmentation and reassembly, error control, and flow control. In addition to them, IP layer needs some more functions like header format analysis, memory management, receiving and forwarding the data pointer to the lower layer, communication with other IP and checksum calculation when it is implemented as a hardware. Those functions were analyzed and divided into groups

to let them be executed in parallel. The detailed description of this procedure is explained in Chapter 5.

### **Multi-media characteristic**

Another important characteristic of the HIPAMG is its ability to handle the multi-media traffic effectively. Therefore, some topics of the multi-media are described in this section.

### **Multi-media applications**

The term multi-media, as used in this dissertation, refers to the computer network traffic with widely varying data traffic characteristics. For instance, voice and video data have to be transferred in real time, but may tolerate a relatively high error rate; the transfer of numerical data must be error free, but in general not in real time. Some characteristics of these traffics are tabulated in Table 3.2 [43].

The currently evolving networks with data rates in the 100 Mbps range are considered to be an enabling technology. They are expected to foster the development of a whole new class of computer based network applications. The most important difference when comparing these new applications with existing ones is the integrated use of different information media, such as voice, video, still image, and data files. For this reason these applications are generally referred to as multi-media applications. CCITT has categorized evolving multi-media applications into the following service classes [44].

- Conversational services
  
- Messaging services

Table 3.2: Characteristics of the multi-media traffic streams

QOS	Maximum delay (s)	Average Throughput (Mbps)	Acceptable bit error rate	Acceptable packet error rate
voice	0.25	0.064	$10^{-1}$	$10^{-1}$
video (TV)	0.25	100	$10^{-2}$	$10^{-3}$
compressed video	0.25	2 - 10	$10^{-6}$	$10^{-9}$
data (file transfer)	1	2 - 100	0	0
realtime data	0.001 - 1	10	0	0
image	1	2 - 10	$10^{-4}$	$10^{-9}$

- Retrieval services
- Distribution services
- Collection services

**Multi-media characteristics of HIPAMG** As described before, today's standard communication protocols are not well suited to the multi-media environment and many new applications are appearing on the horizon, which shift the emphasis from *mono-media-communication* like telephone to *integrated multi-media-communications*. Therefore, future communication systems should be designed to handle these multi-media requirements effectively in order to get appropriate services and also to allow the communication system to satisfy these different requirements [45].

HIPAMG is supposed to work in the multi-media environment, and so HIPAMG should be able to handle the multi-media effectively. The basic idea used in designing the multi-media characteristics of the HIPAMG is to convey the different media traffic

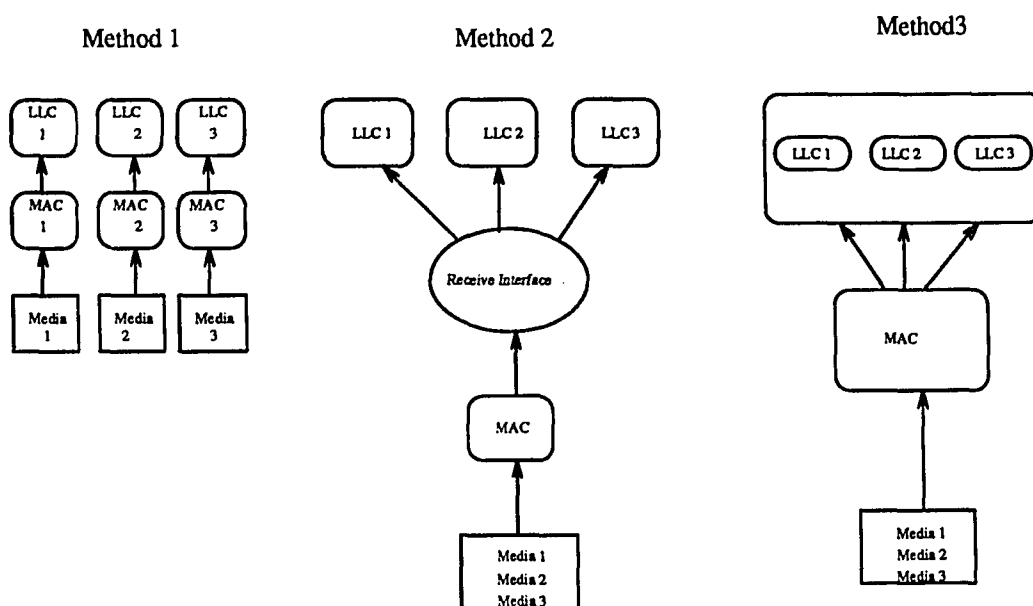


Figure 3.5: Three methods implementing multi-media traffic in HIPAMG

using the different communication paths which is best suited to the different traffic.

The following three methods have been considered:

- Method 1: Using the different MAC address for different media
- Method 2: Using the different LLC SAP for different media
- Method 3: Using the different priority for different media

Each method is depicted in Figure 3.5.

It was decided to use method 2 which provides different LLC SAP for different media. As shown in method 2 of Figure 3.5, when the packets are arrived from the MAC layer, memory manager decides which communication paths should be used

depending on the type of the media the packets are carrying and direct the arriving packets to the proper shared memory module and LLC pool. This idea is implemented in the HIPAMG design with the Protocol Processor Pool Architecture which will be described in detail in the next chapter.

### **Connectionless communication (datagram)**

The communication architecture used in HIPAMG is IP datagram approach in which each packet is treated independently. It is possible that the packet will be delivered to the destination in a different sequence from the one in which they were sent. The destination should have the ability to reorder them using higher layer. This also means that the HIPAMG doesn't need to worry about the sequence of the packet and this make it possible for HIPAMG to handle the packet independently by the separate protocol processor. Therefore, to improve the processing speed, the processor pool architecture which is a collection of processors is used in HIPAMG.

### **Layered architecture**

Communication architecture of the HIPAMG is a layered architecture as shown in Figure 3.1. This vertical subdivision of the communication architecture results in a pipeline structure of communication systems, where each pipeline stage implements a communication layer or sublayer. Such a layer-pipeline performs overlapped computation to exploit temporal parallelism. A single packet moves sequentially through the different pipeline stages, but different packets can be processed in parallel in different pipeline stages. This layered architecture of the communication is utilized in the HIPAMG by implementing the pipeline architecture into it.



### **Full-duplex communication**

One of the characteristics of the HIPAMG communication protocol architecture is its full-duplex communication. This means the send and receive parts of a layer can together be realized as separate layer machine. To increase the performance of a layer, two separate layer machines of the same layer can be processed in parallel and this is implemented in HIPAMG by using a separate receive and transmit hardware to a separate layer machine.

### **Frame encapsulation**

In the layered architecture of the communication protocols, packets are encapsulated when they go down to the lower layer and decapsulated when they go up to the higher layer. Therefore, each layer needs to handle only the header part of its packet. With the help of this architecture, HIPAMG can utilize the shared memory architecture and packet pointer transfer technique.

Another big issue to design a HIPAMG is shared memory architecture. In conventional communication gateway, the packets go through all the protocol layers, that means every packet should be copied between the protocol layers and this consumes a lot of time. To reduce the packet copying time, new technique currently used is to put the packets in the shared memory and only the pointers to the packet is transferred between the protocol layers. It was decided to use this technique in the HIPAMG to improve the throughput and minimize the delay by allowing minimal data movement between the processors. In most shared memory systems, shared memory itself becomes the bottleneck of the system. To avoid this problem, multiple bus architecture was used.

### Protocol Processor Pool Architecture

To satisfy all the design issues of the high-speed multi-media communication architecture discussed so far, we propose a new design concept of the gateway architecture, which is a Protocol Processor Pool Architecture. With this architecture, each communication layer is processed by protocol processor pools which have many protocol processors that execute the different protocol functions independently. In the multi-media environment, every different media will be transferred through separate communication paths from layer 2 and above. And every different communication paths will be handled separately by different protocol processor pools as shown in Figure 3.6.

Here, protocol processor pool is a collection of processors which have their own local memory and execute the protocols independently. To increase the speed of the total protocol processing, we need to increase the number of protocol processors in the pools. Then many packets can be handled in parallel. But if we assign one communication connection to only one protocol processor statically, some protocol processor may be idle even if some of the other protocol processor are too busy to keep up with the speed of the traffic. Therefore, the basic idea of Protocol Processor Pool Architecture is to distribute the packet processing evenly to the all protocol processors dynamically using the Dynamic Path Allocation Algorithm and the Protocol Processor Pool Architecture. The detailed work on this architecture and Dynamic Path Allocation Algorithm will be discussed in the following chapters. Figure 3.7 shows the block diagram of the Protocol Processor Pool Architecture.

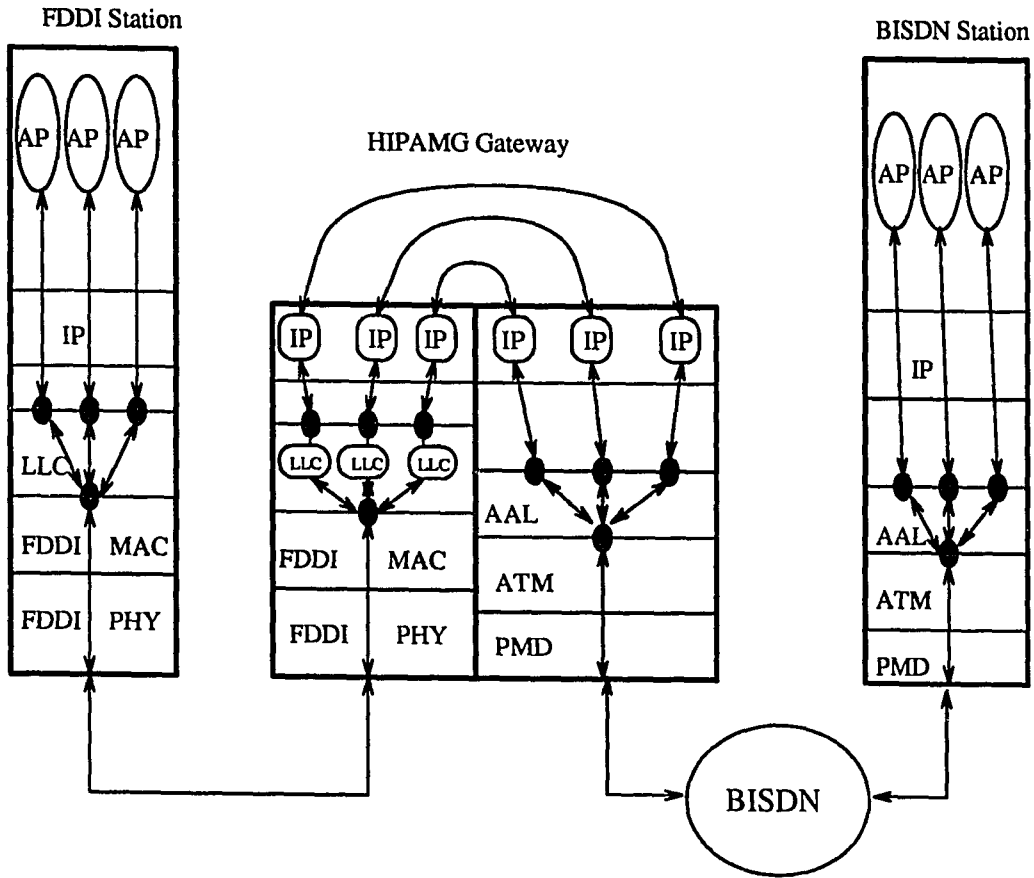


Figure 3.6: Multi-media communication connections in HIPAMG

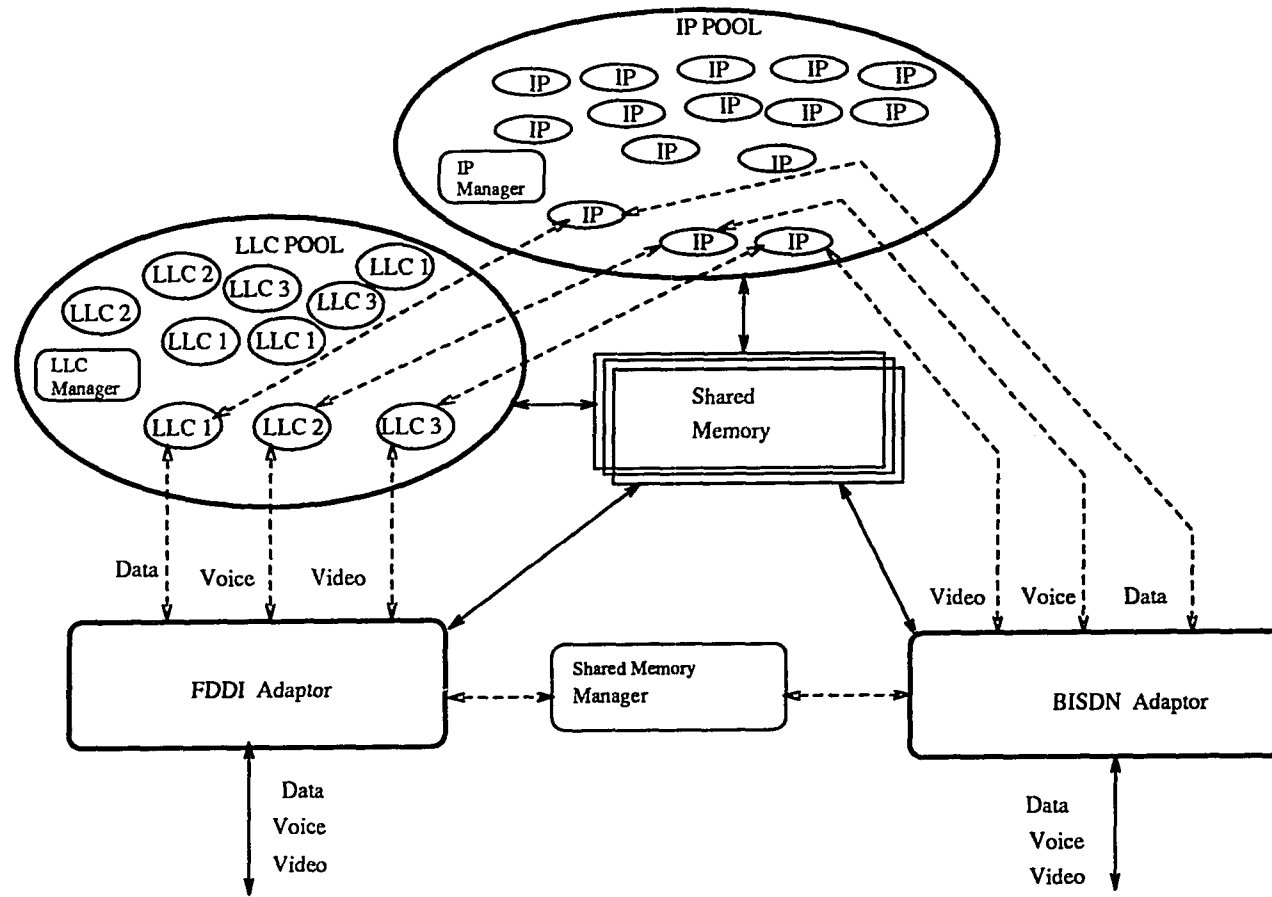


Figure 3.7: Protocol Processor Pool Architecture

## CHAPTER 4. HIPAMG HARDWARE ARCHITECTURE

In this chapter, the detailed hardware design of the HIPAMG is discussed. The design of the multiprocessor gateway architecture so called Protocol Processor Pool Architecture that satisfies all the design issues described in Chapter 3 is proposed and explained in this chapter. Motorola MC68332 micro-controller was selected as the processor of this gateway because of its low price, easy accessibility, design flexibility, and separate serial communication link ability.

### MC 68332

#### Device overview

The MC 68332 is a 32-bit integrated micro-controller, combining high-performance data manipulation capabilities with powerful peripheral subsystems [46]. The MC68332 is the first member of the M68300 family of modular embedded controllers featuring fully static, high-speed complementary metal-oxide semiconductor (CMOS) technology. The MC 68332 contains intelligent peripheral modules such as the time processor unit (TPU), which provides 16 microcoded channels for performing time-related activities. High speed serial communications are provided by the queued serial module (QSM) with synchronous and asynchronous protocols available. Two kilobytes of fully static standby RAM allow fast two-cycle access for system and data stacks

and variable storage with provision for battery backup. Twelve chip selects enhance system integration for fast external memory or peripheral access.

### **Bus arbitration**

The MC 68332 has the ability to access shared memory by using the bus arbitration. Bus arbitration is the protocol by which an external device becomes bus master. The sequence of the protocol is:

1. An external device asserts the bus request signal.
2. The Microcontroller Unit (MCU) asserts the bus grant signal to indicate that the bus is available.
3. The external device asserts the bus grant acknowledge signal to indicate that it has assumed bus mastership.

In the HIPAMG design, protocol processors get the packet from the shared memory by using bus arbitration technique.

### **QSM queued serial module**

The queued serial module (QSM) provides the MCU with two serial communication interfaces divided into two submodules: the queued serial peripheral interface (QSPI) and the serial communication interface (SCI). The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced by the addition of a queue for receive and transmit data. The SCI is a full-duplex universal asynchronous receiver transmitter (UART) serial interface. In the HIPAMG design, the protocol processors exchange control information through QSPI.

The QSPI submodule communicates with external peripherals and other MCUs via a synchronous serial bus. A programmable queue allows the QSPI to perform up to 16 serial transfers without CPU intervention. And four peripheral chip select pins allow the QSPI to access up to 16 independent peripherals by decoding the four peripheral chip select signals. The QSPI internally generates the baud rate for SCK, the frequency of which is programmable by the user. When a 16.78 MHz system clock is used, the actual SCK frequency is from 33 KHz to 4.19 MHz. In HIPAMG design, 4.19 MHz is used.

## **HIPAMG Hardware Architecture**

### **Hardware design issues**

Some hardware design issues that should be considered are

- Granularity of the protocol functions in a protocol processor pool
- Shared memory implementation
- Communication method between processors
- Priority scheme for different media

**Granularity of the protocol functions in a protocol processor pool** In the HIPAMG design, IP protocol was studied and subdivided. The granularity of the IP protocol was decided to be a function level of the IP. Two protocol processors are used to process one IP packet at the same time.

**Shared memory implementation** Shared memory with bus arbitration were implemented in this HIPAMG design. The simulation result showed that three shared memory modules are needed in both FDDI part and BISDN part to prevent the shared memory from becoming the bottleneck of the system at 100 Mbps traffic. To access these multiple shared memory modules, multiple bus system is used. Three memory modules are intended to store three different multi-media packets; voice packet, compressed video packet, and data packet.

**Communication method between processors** Message passing technique is used to exchange the informations between the processors in contrast with the shared memory technique which is used to share the packet between the processors. The actual communication path is QPSI of the MC'68332 and communication speed is 4.19 MHz.

**Priority scheme for different media** Three different kinds of media are assumed to exist in the HIPAMG network environment. They are compressed video, voice, and data whose characteristics are tabulated in Table 3.2. The priority of the compressed video packet is the highest and the priority of the data packet is the lowest. The implementation of this priority scheme are realized by using the Dynamic Path Allocation Algorithm which transfers the higher priority packet faster than the lower priority packet.

### **Implementation details**

The hardware design issues described above were considered and implemented into the HIPAMG. The hardware architecture of the HIPAMG is based on the Pro-



protocol Processor Pool Architecture. Each protocol processor pool will include many protocol processors which execute the communication protocol independently. By using the shared memory, copying the data units between the protocol layers, as is done in many protocol implementations, will be avoided to improve the throughput and minimize the delay. The pointers to the data units stored in the shared memory are to be transferred between the protocol layers through the serial communication link. Each protocol processor pool has its own local memory which is used to execute the protocol. In this design, FDDI MAC, FDDI Physical layer, LAPD, BISDN AAL, ATM, and PMD layer are assumed to be implemented in high speed adapter board. The simplified block diagram of the hardware architecture is shown in Figure 4.1.

The functions of each blocks of the HIPAMG are explained below:

**FDDI adapter** FDDI adapter is an interface board to the FDDI. It executes FDDI MAC and FDDI physical layer protocols.

**BISDN adapter** BISDN adapter is an interface board to the BISDN  $T_B$  interface. It executes LAPD, BISDN AAL, ATM and PMD layer protocols.

**Shared memory manager** This block manages the shared memory by keeping the information about the shared memory. It decides which block of the shared memory is available when asked by the FDDI adapter or BISDN adapter.

**LLC pool** LLC pool includes LLC manager and many LLC protocol processors. LLC manager assigns the job to the LLC protocol processors and communicates with FDDI adapter board and IP manager in IP pool. LLC protocol processors exe-

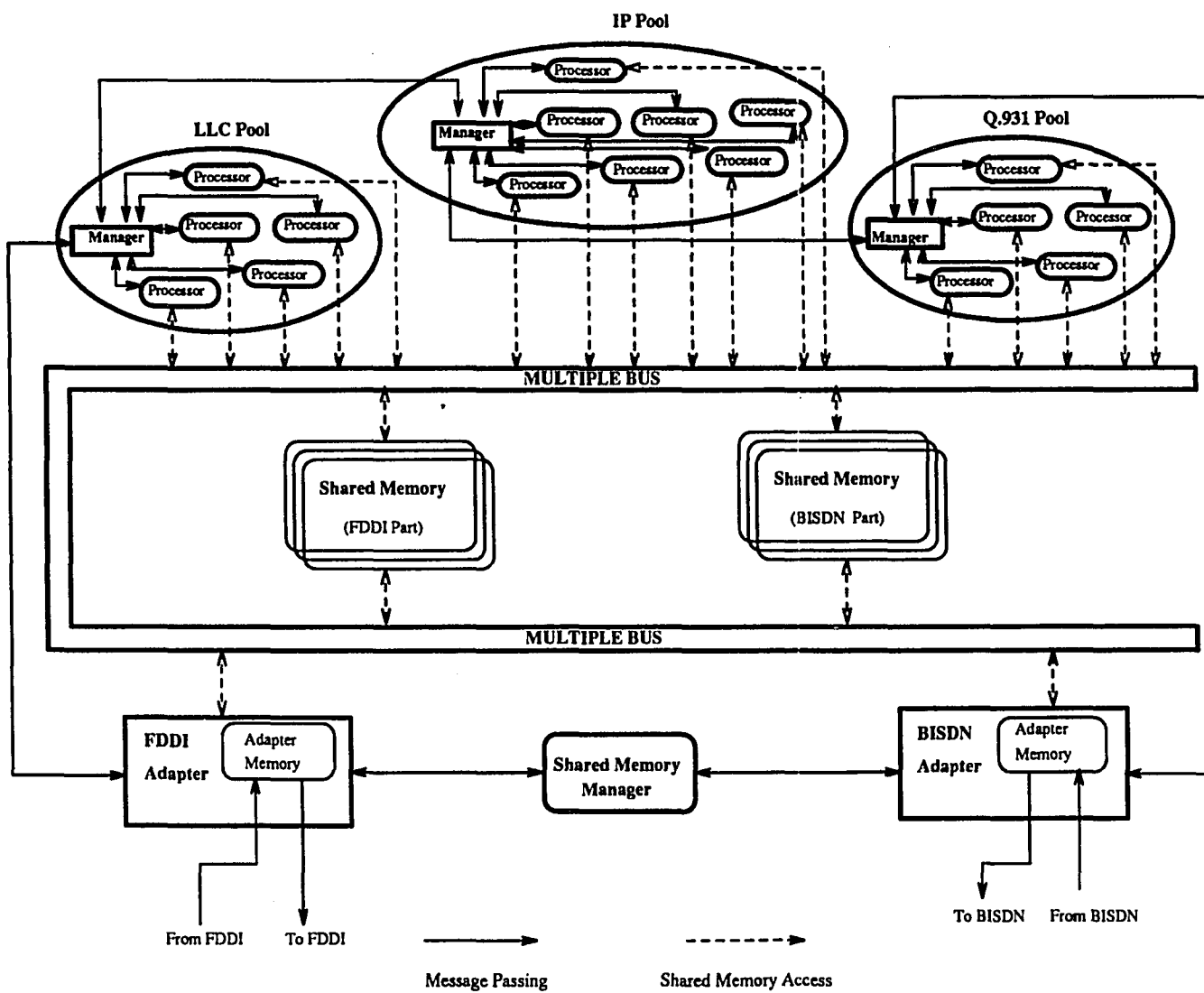


Figure 4.1: Block diagram of the HIPAMG hardware architecture

cuter LLC protocol.

**IP pool** IP pool includes IP manager and many IP protocol processors. IP manager assigns the job to the IP protocol processors and communicates with IP manager, Q.931 manager or BISDN adapter. IP protocol processors execute IP protocol.

**Q.931 pool** Q.931 pool includes Q.931 manager and many Q.931 protocol processors. Q.931 manager assigns the job to the Q.931 protocol processors and communicates with IP manager or BISDN adapter. Q.931 protocol processors execute Q.931 protocol. This pool works only for the control packets on the control plane.

**Shared memory (FDDI part)** This shared memory keeps the LLC packets until they are sent to the BISDN adapter. It provides the LLC header to the LLC protocol processors and IP header to the IP protocol processors. Shared memory is accessed by BISDN adapter, LLC protocol processors, IP protocol processors, Q.931 protocol processors and BISDN adapter through the bus using bus arbitration.

**Shared memory (BISDN part)** This shared memory keeps the Q.931 packets and IP packets depending on the control phase or data transfer phase until they are sent to the FDDI adapter. It provides the Q.931 header to the Q.931 protocol processors. This shared memory also are accessed by the all protocol processors and adapters.

### Packet Flow Inside the HIPAMG

After the packets arrive at one of the adapters, they go through many stages of the HIPAMG and leave through the other adapter. In this section, I will describe how the packets are processed inside the HIPAMG. Figure 4.2 shows the detailed path of the packet flow inside the HIPAMG when the data packets travel from the FDDI to BISDN.

The packet flow procedure inside the HIPAMG are to be explained step by step using the sequence numbers of the Figure 4.2.

1. The receiver of the FDDI adapter receives FDDI MAC packet from the FDDI network and executes FDDI MAC function.
2. After executing the MAC function, FDDI adapter sends control packet to the shared memory manager to interrogate the available location of the shared memory.
3. Shared memory manager decides the available shared memory location where the packets should be saved until it leaves the gateway. Dynamic Path Allocation Algorithm is used to decide the proper shared memory module.
4. Shared memory manager sends this shared memory location information to the FDDI adapter so that the FDDI adapter uses this information to locate the LLC packet at the proper location of the shared memory.
5. FDDI adapter writes decapsulated LLC packet to the shared memory using the bus arbitration.

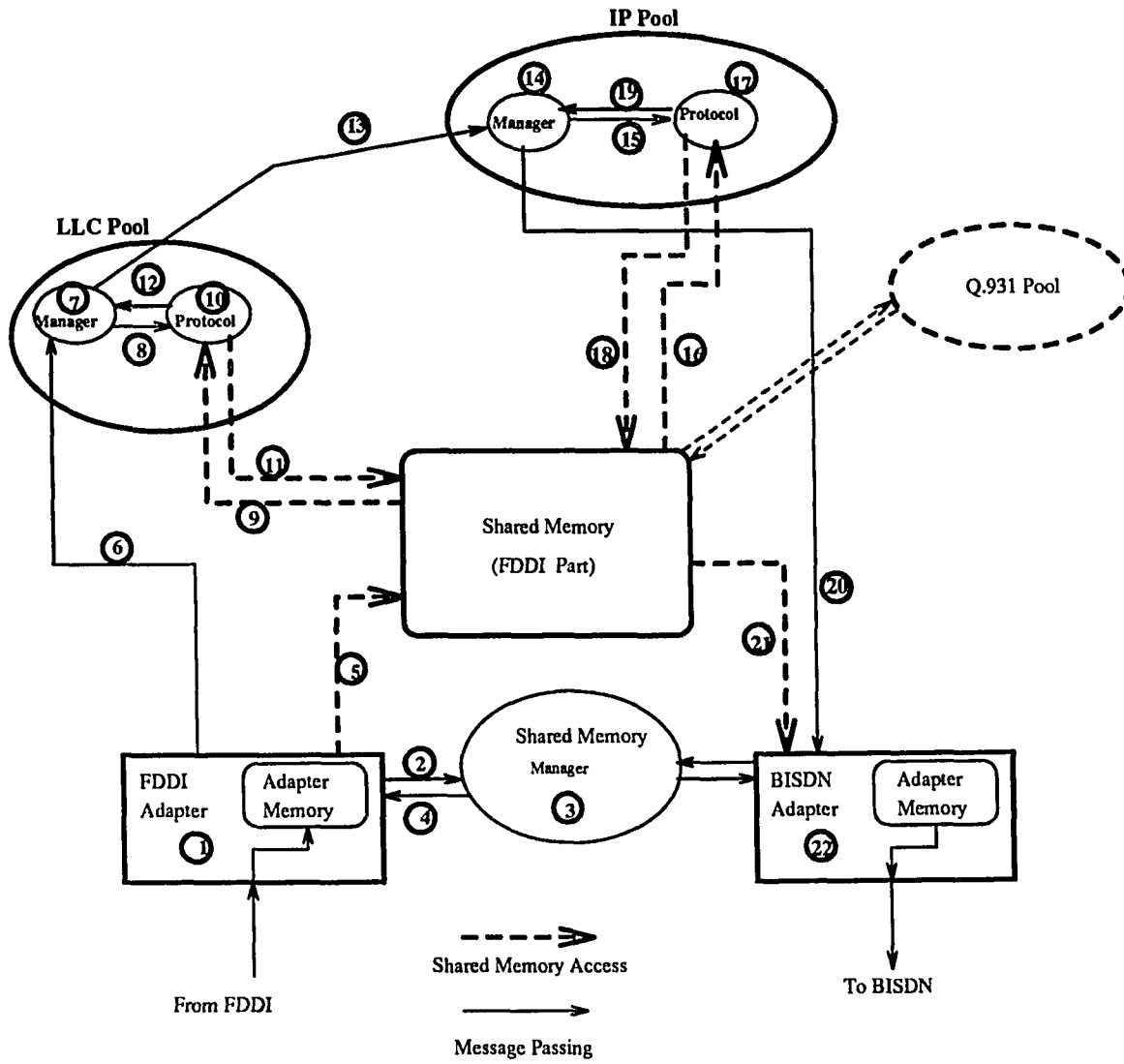


Figure 4.2: Packet flow inside the HIPAMG

6. FDDI adapter sends control packet to the LLC manager to inform the location of the LLC packet inside the shared memory. This control packet is sent through the serial communication link.
7. LLC manager decides which LLC protocol processor should be assigned for this packet.
8. LLC manager sends information about the packet (location at the shared memory) to the selected LLC protocol processor.
9. LLC protocol processor reads header part of the LLC packet from the shared memory.
10. LLC protocol processor executes the LLC protocol and changes the header part of the LLC packet.
11. LLC protocol processor writes the modified LLC packet back to the shared memory.
12. LLC protocol processor sends the control packet to the LLC manager to report the completion of the LLC protocol processing.
13. LLC manager sends the control packet to the IP manager to inform the completion of the LLC protocol processing and the location of the packet inside the shared memory.
14. IP manager decides which IP protocol processor should be assigned for this packet. In this design, round robin technique is used to allocate the packet to the IP protocol processor.

15. IP manager sends information about the packet (location at the shared memory) to the selected IP protocol processor.
16. IP protocol processor reads header part of the IP packet from the shared memory.
17. IP protocol processor executes the IP protocol and changes the header part of the IP packet.
18. IP protocol processor writes the modified IP packet back to the shared memory.
19. IP protocol processor sends the control packet to the BISDN adapter to inform the completion of the IP protocol processing and the location of the packet inside the shared memory.
20. BISDN adapter reads the LLC packet from the shared memory and inform the shared memory manager that the location inside the shared memory which was occupied by the LLC packet is now available. Shared memory manager now updates the shared memory information.
21. BISDN adapter executes the BISDN AAL, ATM, PMD protocol processing and constructs the STS-3c packet. Finally it sends this packet to the BISDN  $T_B$  interface out of the HIPAMG.

The above procedures explain the steps the packet should go through inside the HIPAMG. The packets which travel from the BISDN to the FDDI basically go through the same procedure as above in the opposite direction. These packets are saved in BISDN part of the shared memory and handled independently from the packets which travel from FDDI to BISDN.

## CHAPTER 5. HIPAMG SOFTWARE DESIGN

All of the communication protocols used in HIPAMG are standard protocols. Therefore, no new protocol design is needed in the HIPAMG design. However, some modifications to the standard protocols are needed. Also we need to design the separate finite state machine for each pieces of the protocol functions which are divided according to the parallelism. In this research, IP layer is studied and divided.

As a result, some software design issues that are considered include:

- Parallelism of the protocol
- IP Protocol subdivision
- Finite state machine of the subdivided IP protocol
- Dynamic Path Allocation Algorithm
- Frame format

### Parallelism of the Protocols

The communication protocols have parallelism in a number of places: between protocol layers, within individual protocols, and within the entire communication



architecture. Among them, parallelism within individual protocols were studied during the HIPAMG software design by analyzing the parallelism within the IP layer protocol and implementing IP protocol to the protocol processors.

### **Functions of IP**

The functions of IP are:

- Check sum calculation
- Routing
- PDU life time control
- Fragmentation

When IP layer receives the IP packet, it should process above functions on the IP packets. In addition to the above IP functions, IP layer should execute some more jobs like

- IP header access from the shared memory
- Header format analysis
- Frame composition
- IP header write to the shared memory
- Internet Control Message Protocol (ICMP)

If all the functions are executed by one processor serially, IP protocol should be executed like the flow diagram in Figure5.1.

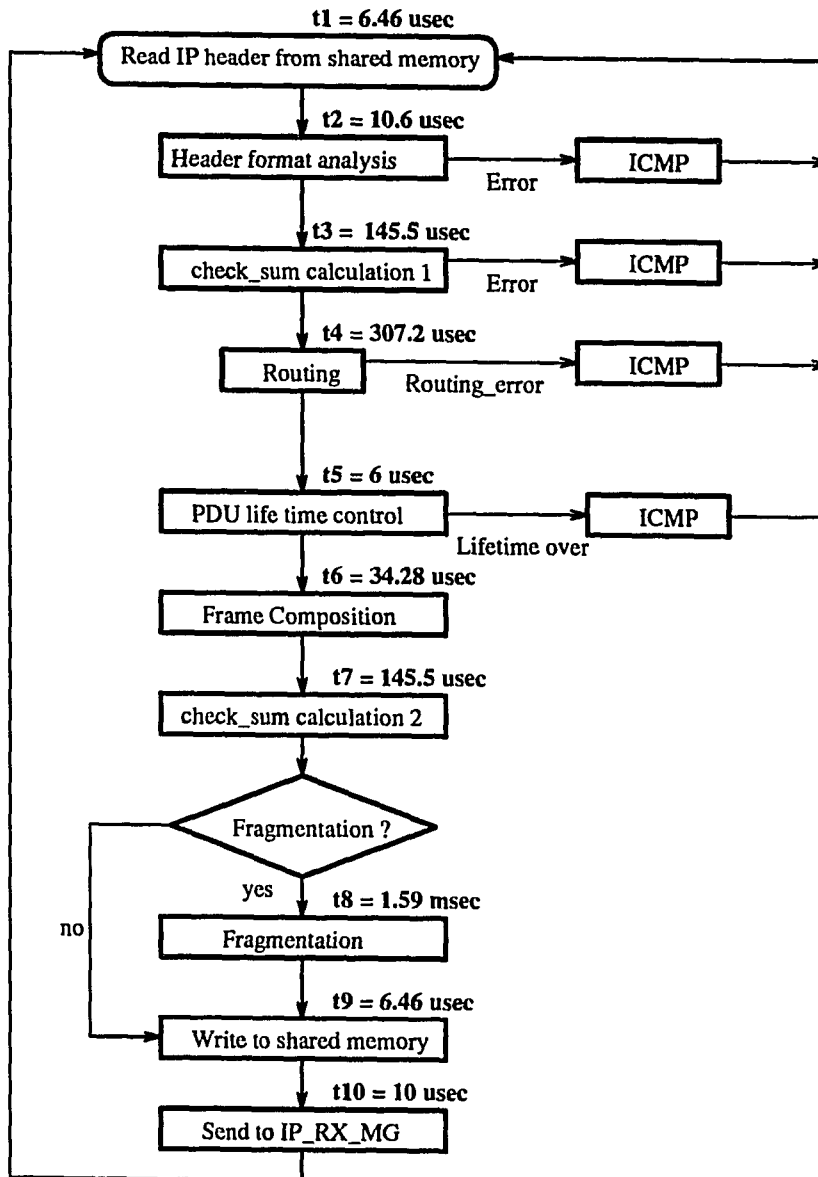


Figure 5.1: IP protocol flow diagram

### Execution time of the IP

The execution time of each IP function was measured using the M68332EVS evaluation system. The IP protocol was implemented in C language on the Apollo workstation with UNIX operating system and downloaded to the M68332EVS board and executed. The execution time of each function is measured using HP 1650B logic analyzer. The IP protocol implemented here is from the Xinu operating system which is described in [47]. The execution times of the IP functions are shown in Figure 5.1. The total execution time of IP with the assumption of no fragmentation and no waiting time is:

$$T_{IP1} = t_1 + t_2 + t_3 + t_4 + t_5 + t_6 + t_7 + t_9 + t_{10} \quad (5.1)$$

$$= 6.46 + 10.6 + 145.5 + 307.2 + 6 + 34.28 + 145.5 + 6.46 + 10 \quad (5.2)$$

$$= 672(\text{usec}) \quad (5.3)$$

when MC68332 runs on 16.78 MHz clock.

When protocol functions of the IP layer are executed in parallel, the speed of resulting pipeline will only be as fast as the slowest functions. Hence, in order to maximize the performance of an implementation, careful analysis of the protocol is required.

### IP Protocol Subdivision

In order to improve the execution time of the IP, IP protocol should be executed in parallel. Therefore, careful subdivision of the IP protocol is needed. Among the functions of the IP, routing takes longest time to execute which is 307.2 usec. Also fragmentation, frame\_composition, check\_sum\_calculation\_2 and write\_to\_shared\_memory

should be executed after the routing. This shows that execution time of the routing plus fragmentation is longer than the sum of the execution time of all other functions of the IP. Therefore, the minimum execution time of the IP should be the sum of the routing and fragmentation execution time. The best subdivision I can get is shown in Figure 5.2. As shown in Figure 5.2, IP protocol is divided into two parts which are IP\_PROCESS\_1 and IP\_PROCESS\_2. The two processes of the IP should be synchronized by exchanging the control frames and those signals carried by control frames are shown in Figure 5.2. The length of control frame is assumed to be 8 bits. Therefore, the control packet send delay time,  $d_1$ , is calculated by dividing the size of the control packet (8 bits) with serial link speed (4.19M).

With this subdivision, the total execution time of the IP is:

$$T_{IP2} = t_1 + t_4 + t_6 + t_7 + t_9 + t_{10} + d_1 + d_2 + d_3 \quad (5.4)$$

$$= 6.46 + 307.2 + 34.28 + 145.5 + 6.46 + 10 + 1.9 + 1.9 + 1.9 \quad (5.5)$$

$$= 515.6(\text{usec}) \quad (5.6)$$

This execution time is the best execution time of the IP, when we assume the minimum subdivision level is function, that is, the function is not to be subdivided further. As a result, the IP protocol subdivision improves the execution time of the IP by 30.33%.

### Finite State Machine of the Subdivided IP Protocol

After subdividing the IP process into IP\_PROCESS\_1 and IP\_PROCESS\_2 processes, the Finite State Machines (FSMs) of these two processes are drawn like in Figure 5.3 and 5.4 [48]. These two FSMs are used to implement the protocol in

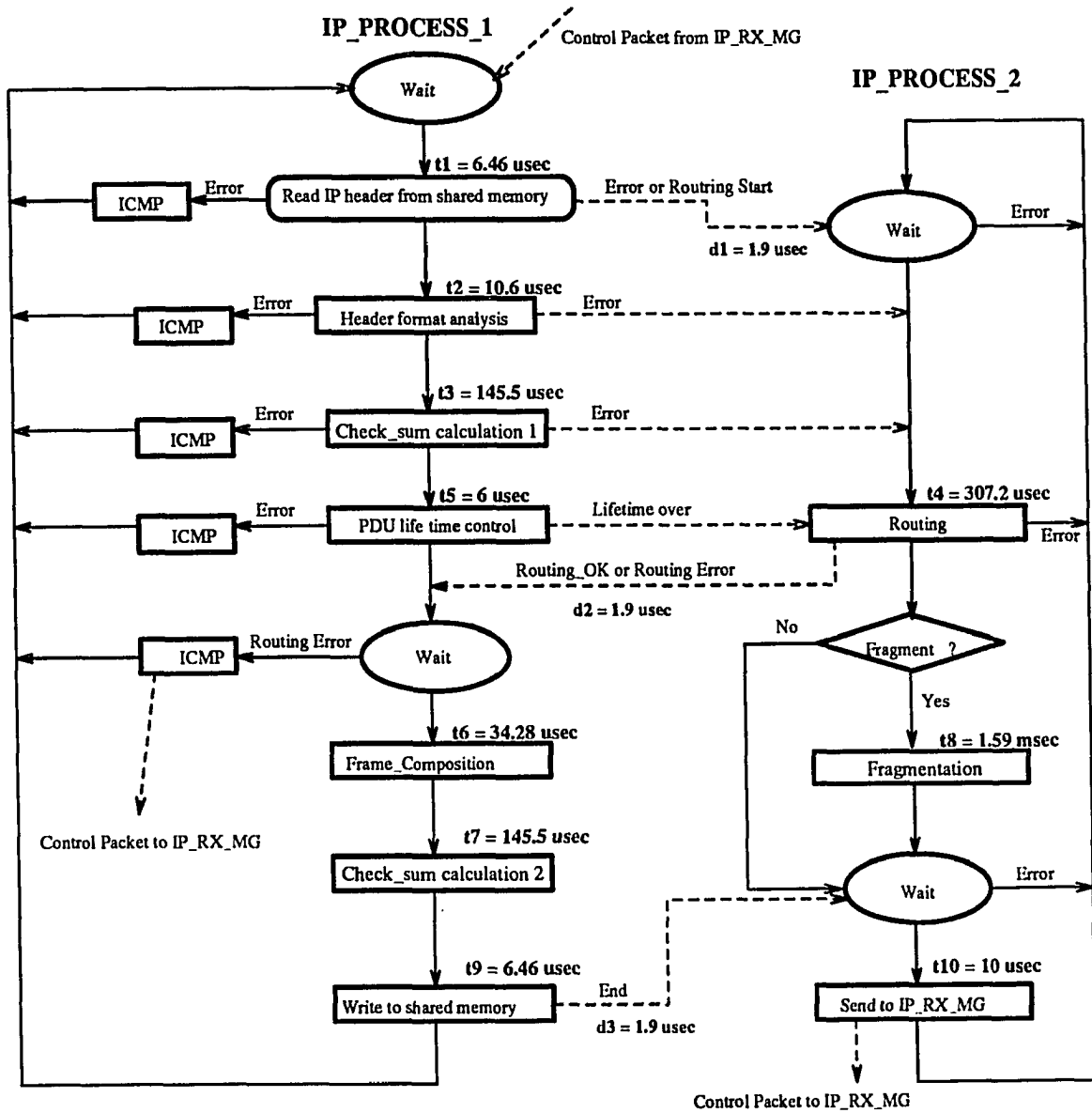


Figure 5.2: Subdivided IP protocol flow diagram

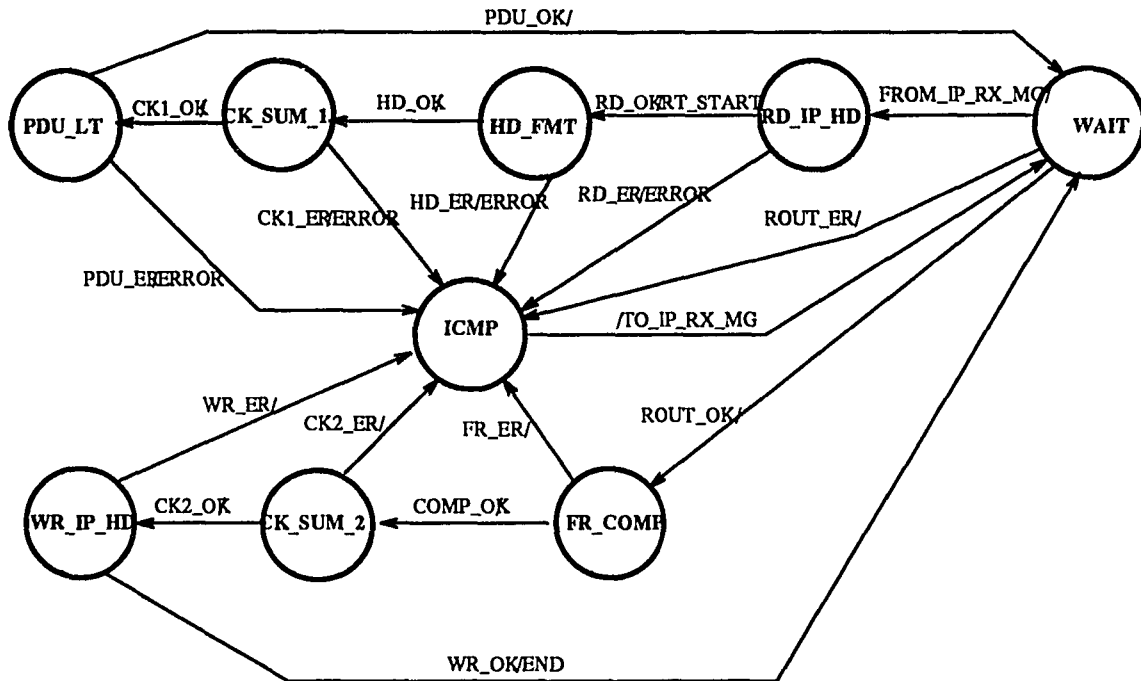


Figure 5.3: Finite state machine of the IP\_PROCESS\_1

software program and are implemented at the simulation stage.

### Dynamic Path Allocation Algorithm

As shown in Figure 3.6, HIPAMG allocates different communication paths for the different media traffic in multi-media environment. When the FDDI adapter or BISO adapter receive the packets from the network, they need to decide which communication path should be used first, then send the packets to the selected path. The proper path decision algorithm should be used to get the best performance and

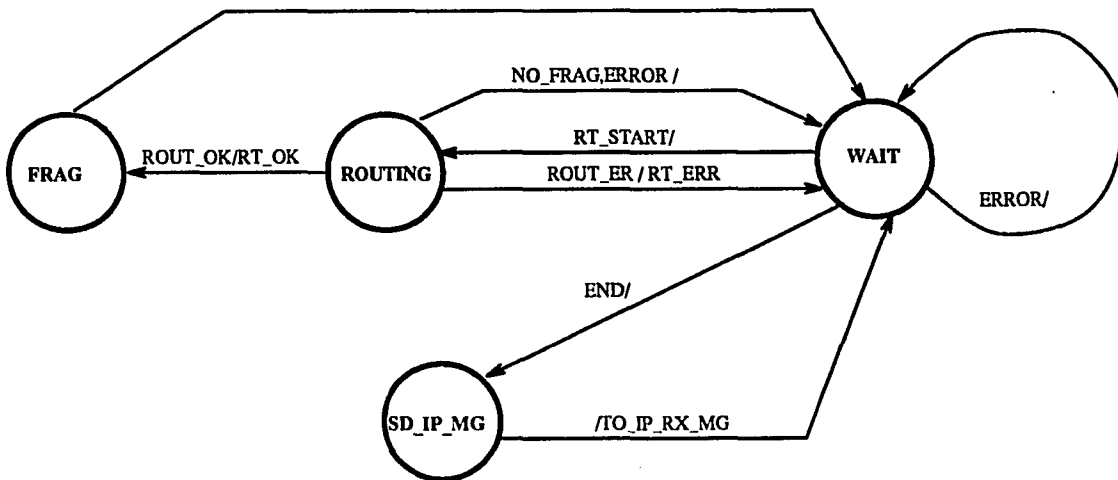


Figure 5.4: Finite state machine of the IP\_PROCESS\_2

minimum Packet Transfer Delay through the gateway.

### Assumptions

To design an allocation algorithm, we need to assume the characteristics of the traffics on this research. The assumptions are:

- The traffics are multi-media traffic of voice, compressed video, and data.
- The characteristics of the traffics are same as the characteristics shown in Table 3.2.
- The average inter arrival times of packets of different media are same and have the exponential distribution.

- The average sizes of packets of different media are same and have the exponential distribution.

## **Idea**

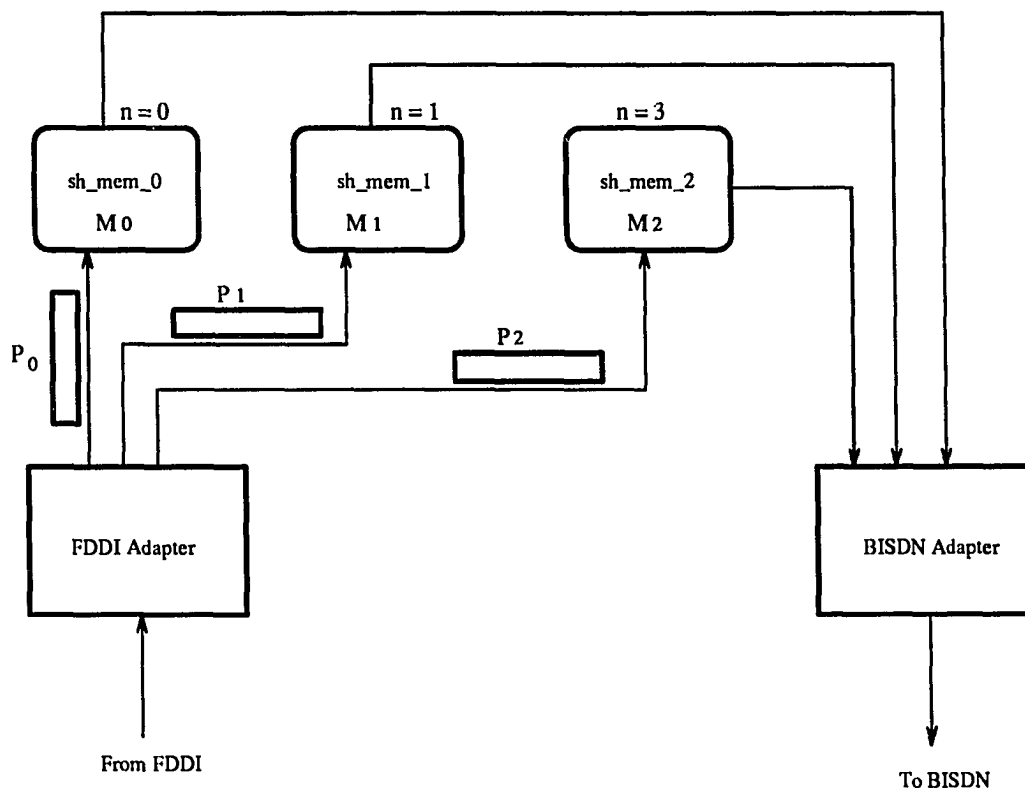
The basic scheme of the traffic flow is shown in Figure 5.5. The shared memory manager has the responsibility to decide the proper path when asked by the adapters. Therefore, the allocation algorithm is executed on the shared memory manager.

The idea of this scheme is to allocate one dedicated path to one specific traffic in normal condition, so that HIPAMG has three communication paths for the three different traffics. For example, in normal operating condition, the voice path can only be used by a voice traffic. But for some reason, if the traffic of one media rises, we need to allocate another path to this traffic even though that path is not originally intended for this traffic. Therefore, we need some rule for this and Dynamic Path Allocation Algorithm handles this situation.

## **Design**

First, we need to assign the priorities to each different traffic. From the characteristics given in Table 3.2, the highest priority (0) is given to the compressed video and the lowest priority (2) is given to data according to the maximum delay and acceptable bit error rate. The voice is given the priority 1. The best allocation can be achieved when the traffic is monitored and the packet is assigned to the proper path. Any best static allocation algorithm without monitoring the traffic can not beat the simple dynamic allocation algorithm which monitors the traffic and decides the paths according to this. The logic of this algorithm is:





$M_n$  Shared memory module  $n$   
 $N_n$  Number of queued packets in  $M_n$   
 $P_n$  Packet whose path is  $n$   
 $MAX_n$  The maximum number of packets allowed in  $M_n$   
 $n$  Path number

Figure 5.5: Traffic flow in HIPAMG

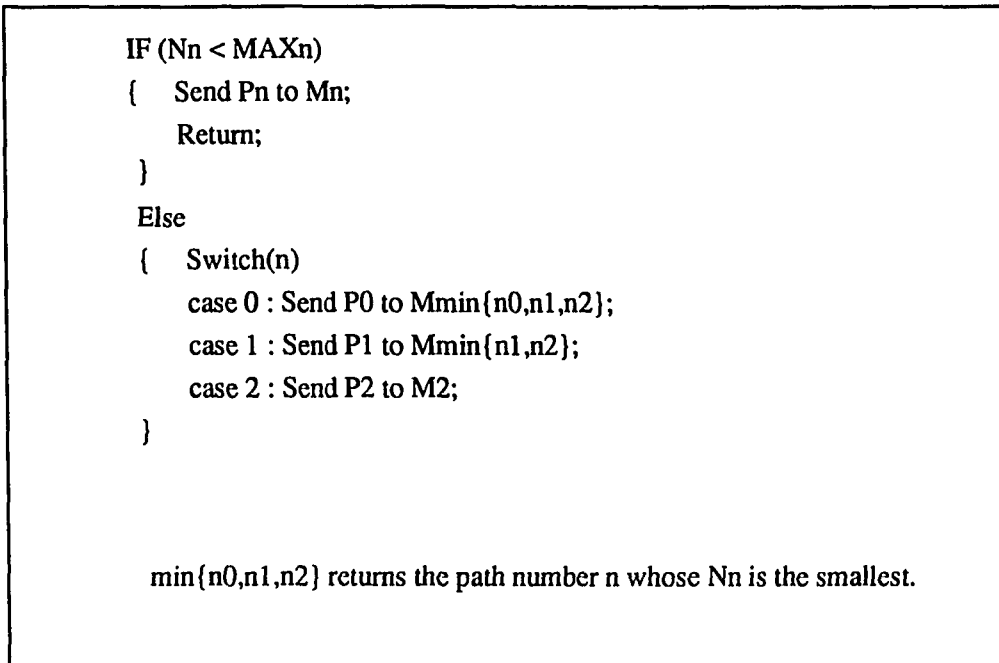


Figure 5.6: Formal expression of the Dynamic Path Allocation Algorithm

1. Send the packet  $P_n$  to its own shared memory module  $M_n$  when the number of packets  $N_n$  in  $M_n$  is less than the maximum number of packets  $MAX_n$  allowed in  $M_n$ .
2. If  $N_n$  in shared memory module  $M_n$  is greater than  $MAX_n$ , then send it to the memory module which has the smallest number of packet in it among the memory modules  $M_n$  to  $M_2$ .

The formal expression of the Dynamic Path Allocation Algorithm is shown in Figure 5.6. The value of  $MAX_0$  is decided to be 9 which is the average number of packets inside the shared memory module 0. The value of  $MAX_1$  is decided to be 12 following the result of the simulation which shows the best compromise on the number of packets in the shared memory module 1 and 2 .

### **Frame format**

All the frame formats used in HIPAMG are the standard frame formats defined in the protocol standards. The frame format used are IP packet, LLC packet, FDDI MAC packet, BISDN ATM packet, and STS-3c packet. On the other hand, control informations are transferred between the protocol processors using the control packets which were designed during this research. All the frame formats are depicted in Figure 5.7.

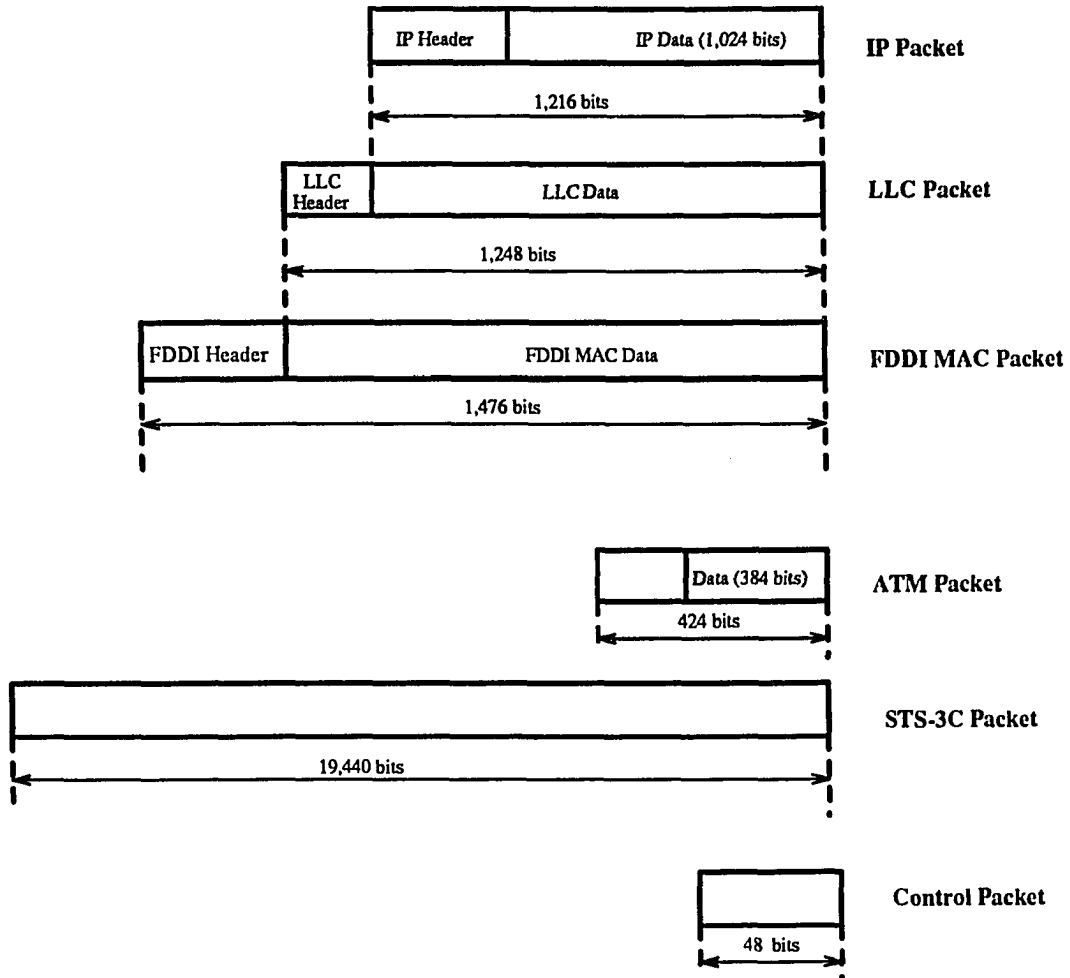


Figure 5.7: Frame formats of the HIPAMG

## CHAPTER 6. MODELING AND PERFORMANCE SIMULATION

After completing the HIPAMG design, the performance of the gateway is evaluated using the analytical evaluation and simulation to verify and validate the results of each one. The simulation was performed using the OPNET graphic simulator which is the contemporary CAE system developed by MIL3., Inc. [13] and has the power to simulate the performance of the HIPAMG system down to the process level. The goal of this performance simulation is to perform the measurements of the Throughput and the Packet Transfer Delay of the HIPAMG system. The simulation result is discussed in this chapter and the analytical evaluation is discussed in the next chapter.

### Modeling Parameters

For the performance simulation of the HIPAMG, the development of a simulation model is needed. The performance of a HIPAMG will be modeled with the following parameters.

- Bandwidth of the channel
- Communication buffering capacity
- Buffer management policy

- Arrival rate of packets
- Packet size
- The processing speed of each processor
- The communication speed between the processors
- Priority handling for different media
- Shared memory usage

Some assumptions and abstractions are needed to build the model. The performance evaluation in this research is concentrated on data transfer phase because the high-speed performance concerns arise in data transfer phase.

### **Assumptions of the HIPAMG Modeling**

In order to get the best performance simulation result, we need to build the best model of a HIPAMG. Even though the best model seems to be the model which is exactly same as the real HIPAMG, it is impossible and not necessary to build the model which is exactly same as the real system. To build the concise and good model, we need some abstractions and assumptions. The following assumptions are made to build the HIPAMG model.

- HIPAMG works in data transfer phase
- No error occurs inside the HIPAMG communication
- FDDI adapter, BISO adapter, memory managers, LLC managers, IP managers, and LLC pool are fast enough not to be a bottleneck of the system. The

utilization of these stage is assumed to be 0.4. (The calculation of the service rate of each stage is discussed in the next section.) Practical considerations usually limit the input rate for a single server to 70 to 90 % of the theoretical maximum [49].

### Workload Model

The workload to the HIPAMG model is the packet traffic. In this section, the parameters of the traffic models are discussed. There are two kinds of traffic in this model. One traffic is the stream of the packets flow from the FDDI network to the BISDN network. The other traffic is the stream of the packets flow from the BISDN network to the FDDI network. The maximum speed of the FDDI network is 100 MBPS [19]. It is assumed that the traffic from the FDDI and the BISDN are about same. Also we decided to use the worst case workload, that means the average traffic of this network is same as the maximum traffic of the network.

The maximum bandwidth of the BISDN network is 600 Mbps. However, the HIPAMG model developed here connects one FDDI network to one BISDN network. Therefore, the average traffic of the BISDN network is assumed to be 100 Mbps which is the maximum bandwidth of the FDDI network.

The followings are the characteristics of the workload model of HIPAMG network.

- The average data rate of the traffic arriving from the FDDI is 50 Mbps and the interarrival time has an exponential distribution.
- The average data rate of the traffic arriving from the BISDN is 50 Mbps and

has an exponential distribution.

- Average packet size of the FDDI MAC packet is 1476 bits and the interarrival time has an exponential distribution.
- Packet of the BISDN network is STS-3c packet which has the constant size of 19,440 bits.

### Service Time Calculation

As shown in Figure 4.2, packets travel through the HIPAMG and they wait until they are serviced at each stage. Each stage has input buffer which queues the input packets until they are serviced. The service times of each stages of the HIPAMG are calculated in this section. The sequence numbers of the Figure 4.2 are used as the subscripts of each parameters which are service time ( $s$ ), packet arrival rate ( $\lambda$ ), control packet delay time ( $d$ ), and utilization ( $\rho$ ).

#### FDDI adapter service time ( $s_1$ )

The average data rate of the arrival traffic from the FDDI is 50 Mbps. Therefore, the average packet arrival rate from the FDDI is;

$$\lambda_{11} = \frac{50M}{1476} \quad (6.1)$$

$$\approx 33,875(\text{packets}/\text{sec}) \quad (6.2)$$

FDDI adapter also should handle the packets from the BISDN. The average packet arrival rate from the BISDN,  $\lambda_{12}$ , is also 33,875 (packets/sec). The packet service time of the FDDI adapter is decided to make utilization  $\rho = 0.4$  at the average input



traffic.

$$s_1 = \frac{\rho_1}{\lambda_{11} + \lambda_{12}} \quad (6.3)$$

$$= \frac{0.4}{33,875 + 33,875} \quad (6.4)$$

$$\approx 5.904(\mu sec) \quad (6.5)$$

### Control frame transfer time ( $d_2$ )

When we designed the frame format, the length of the control packet is decided to be 48 bits. Also, the maximum speed of the serial link of the MC68332 is 4.19 MHz as discussed in Chapter 4.

Control packet delay time ( $d_2$ ) = Size of the control packet / serial link speed

Therefore,

$$d_2 = \frac{48}{4,190,000} \quad (6.6)$$

$$\approx 11.456(\mu sec) \quad (6.7)$$

All the control packet delay times  $d_2, d_4, d_6, d_8, d_{12}, d_{13}, d_{15}, d_{19}$ , and  $d_{20}$  are same and  $11.456 \mu sec$ .

### FDDI memory manager service time ( $s_3$ )

To make the utilization  $\rho_3 = 0.4$ ,

$$s_3 = \frac{0.4}{\lambda_3} \quad (6.8)$$

$$= \frac{0.4}{33,875} \quad (6.9)$$

$$\approx 11.808(\mu sec) \quad (6.10)$$

### LLC packet shared memory write time ( $s_5$ )

The service time at this stage,  $s_5$ , is the time taken to write the LLC packets to the shared memory. Here are some architecture parameters we implemented in HIPAMG.

- Shared memory access from FDDI adapter is DMA data transfer.
- Memory access is done with the data size of 16 bits.
- While waiting the bus grant signal, the waiting packets are queued in the output buffer of the protocol processors.
- Assume the shared memory access time is 60 nsec.

Average LLC packet size is 1248 bits. Therefore, we need  $1248/16$  which is 78 memory accesses to write the LLC packet to the shared memory. Average memory access time for the LLC packet is the product of one memory access time and Number of memory access. Therefore,

$$s_5' = 60 \times 10^{-9} \times 78 \quad (6.11)$$

$$\approx 4.68(\mu sec) \quad (6.12)$$

We need 300 nsec for the bus arbitration. Therefore, the total memory access time,  $s_5$ , is  $4.68 + 0.3 = 4.98$  ( $\mu sec$ ).

**LLC manager service time ( $s_7$ )**

To make the utilization  $\rho_3 = 0.4$ ,

$$s_7 = \frac{\rho_7}{\lambda_7} \quad (6.13)$$

$$= \frac{0.4}{33,875} \quad (6.14)$$

$$\approx 11.808(\mu sec) \quad (6.15)$$

If we use MC68332 as the LLC manager, the average number of clock per instruction is 5 and clock rate is 16.78 MHz. Therefore, the average number of instructions it can execute during 11.808  $\mu$  sec is

$$num = \frac{11.808(\mu sec) \times 16.78M}{5} \quad (6.16)$$

$$\approx 40(instructions) \quad (6.17)$$

From this result, we can conclude the service time of the LLC manager is reasonable value.

**LLC header shared memory read time ( $s_9$ )**

The service time at this stage,  $s_9$ , is the time needed to read the LLC header from the shared memory. The size of LLC header is 32 bits. Therefore, MOVE.L (A1),(A0) instruction can be used to read the LLC header from the shared memory to local memory of the LLC pool. I measured the execution time of this instruction using the HP 1650B logic analyzer and got 160 nsec execution time. The time needed for the bus arbitration is 300 nsec. Therefore, the total LLC header read time,  $s_9$ , is  $160 + 300 = 460$  (nsec).

**LLC pool service time ( $s_{10}$ )**

In this HIPAMG design, the traffic flows are handled by 3 independent LLC pools and three independent LLC pools process compressed video, voice, and data traffic respectively. The processing speed of each LLC pool is different because the different media traffics are assumed to be processed using different class of protocols which are best suited to the different traffic. To get the LLC pool service time for the data traffic, the utilization  $\rho_{10}$  is assumed to be 0.4. The average packet arrival rate to llc pool,  $\lambda_{10}$ , is 11,292 (packets/sec). Therefore, LLC pool service time for the data is;

$$s_{10} = \frac{\rho_{10}}{\lambda_{10}} \quad (6.18)$$

$$= \frac{0.4}{11,292} \quad (6.19)$$

$$\approx 35.423(\mu sec) \quad (6.20)$$

The LLC pool service time for the compressed video and voice are assumed to be 17.712  $\mu sec$  and 11.808  $\mu sec$  respectively which are half and 1/3 of the LLC pool service time for the data traffic.

**LLC header shared memory write time ( $s_{11}$ )**

The service time at this stage,  $s_{11}$ , is the time to write the LLC header to the shared memory and this time is same as  $s_9$  which is 460 nsec.

**IP manager service time ( $s_{14}$ )**

IP manager service time,  $s_{14}$ , is decided using the same calculation as LLC pool service time which is 35.423  $\mu sec$ .

**IP header shared memory read time ( $s_{16}$ )**

The service time at this stage,  $s_{16}$ , is the time needed to read the IP header from the shared memory. The size of IP header is 192 bits. I suppose to use the following instructions for the memory read.

```

LOOP:  MOVE.L  (A1)+,(A0)+
        DBF    D0,LOOP

```

where, this loop should be repeated  $192/32 = 6$  times. I measured the execution time of this loop using HP 1650B logic analyzer and got  $6.16 \mu\text{sec}$  execution time. Therefore, the total IP header read time,  $s_{16}$ , is  $6.16 + 0.3 = 6.46 \mu\text{sec}$ .

**IP pool service time ( $s_{17}$ )**

The service time of this stage,  $s_{17}$ , is the IP protocol execution time on the IP protocol processor. As discussed in Chapter 5, the execution time of the subdivided IP protocol is  $515.6 \mu\text{sec}$ .

**IP header shared memory write time ( $s_{18}$ )**

The service time at this stage,  $s_{18}$ , is the same as  $s_{16}$  which is  $6.46 \mu\text{sec}$ .

**LLC packet shared memory read time ( $s_{21}$ )**

The service time at this stage,  $s_{21}$ , is the same as  $s_5$  which is  $4.98 \mu\text{sec}$ .

**BISDN adapter service time ( $s_{22}$ )**

The service time at this stage,  $s_{22}$ , is same as  $s_1$  which is  $5.904 \mu\text{sec}$ .

## **Development of the Simulation Model**

The simulation model of the HIPAMG is developed with the parameters and assumptions described at the beginning of this chapter. OPNET graphic simulator is used as the simulation tool of the HIPAMG.

### **OPNET**

OPNET is the contemporary CAE system developed by MIL3, Inc. The OPNET system is a set of tools which can be divided into three functional areas: Specification, Simulation, and Analysis. The specification area consists of the five graphical editors with which users specify their designs; these are the Network Editor, Node Editor, Process Editor, Parameter Editor, and Probe Editor. The simulation area consists of the Simulation Tool and Simulation Kernel. The analysis area consists of the Analysis Tool, which processes and graphically presents simulation results, and the Filter Editor, which is used to construct specialized result-processing filters. These three areas are supported graphically by an encompassing window management system called the Tool Environment [50].

### **Simulation parameters**

The performance of the HIPAMG is evaluated by finding how quickly the packets can be processed in the HIPAMG (Packet Transfer Delay) and how many packets can be processed by the HIPAMG (Throughput). The number of packets in shared memory is another important parameter which is used to optimize the size of the shared memory.

Packet Transfer Delay is defined as the sum of mean queueing times of all the

stages and the control packet send delay times of all the stages. In this simulation, Packet Transfer Delay is decided by measuring the times one packet spends in the HIPAMG.

Throughput is defined as the number of STS-3c packets which leave the HIPAMG per second. Throughput is calculated by dividing the total number of STS-3c packets which leave the HIPAMG by the simulation period.

Number of packets in shared memory is calculated by averaging the number of packets in three shared memory modules.

### **Simulation procedure**

The simulation model of the HIPAMG is constructed by combining the process models to the node model and then the node models to the network model using OP-NET. Simulation was started with the simplest HIPAMG model and the simulation model was improved by adding more modules to the HIPAMG model. The simplest HIPAMG node model is `hipamg_1` and the complete final model is `hipamg_6`. The simulation was performed by increasing the total traffic rate from 3.33 Mbps to the final value 100 Mbps. The simulation procedure of the HIPAMG is discussed next.

**hipamg\_1 model** Simulation started with the simple HIPAMG model shown in Figure 6.1. This module has one shared memory module and one protocol processor in IP pool. All the stages are modeled following the assumptions described at the beginning of this chapter. The simulation result of this model shows that IP pool becomes bottleneck when the total traffic from the FDDI and BISDN reaches at 5 Mbps.

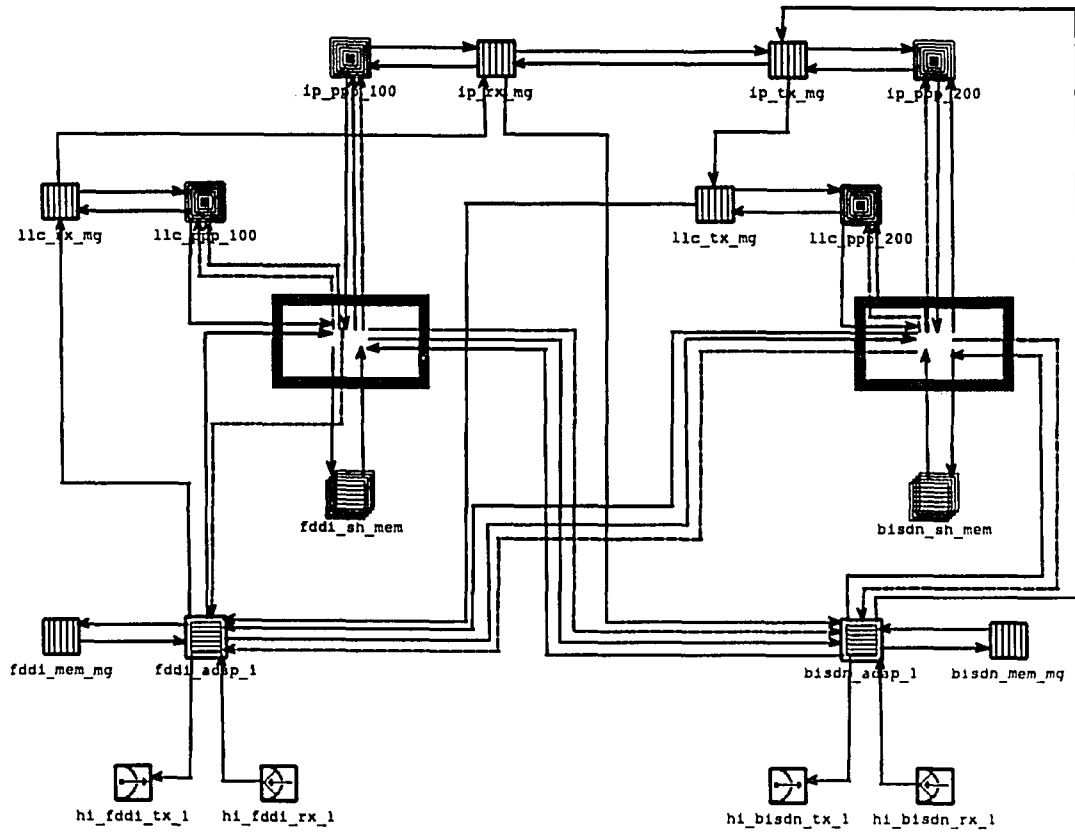


Figure 6.1: hipang\_1 model



**hipamg\_2 model** To increase the speed of the IP protocol processing, protocol processor pair is implemented in this model. Each processor of the IP protocol processor pair processes the IP\_PROCESS.1 and IP\_PROCESS.2 FSM respectively which are shown in Figure 5.1 and 5.2. This model shows a little improvement and IP pool begins to be a bottleneck when the total traffic from the FDDI and BISDN becomes 6.6 Mbps.

**hipamg\_3 model** As shown in Figure 6.2, the number of IP protocol processors of hipamg\_3 model was increased to 6. As expected, the speed of the IP pool is increased linearly as the number of IP protocol processor is increased. The IP pool begins to be a bottleneck when the total traffic from both networks becomes 20 Mbps.

**hipamg\_4 model** The number of IP protocol processors are increased to 18 in this simulation model. But in this model, another bottleneck appears when the total traffic reaches at 50 Mbps. This bottleneck is caused because of the shared memory access. New design decision should be made because by only increasing the number of IP protocol processors, we can not improve the performance of the HIPAMG. The new design decision was made to use multiple memory modules. Three memory modules with three bus systems at each side of the HIPAMG are implemented to satisfy the multi-media characteristics of the HIPAMG as described in Chapter 3.

**hipamg\_5 model** This model is the HIPAMG model which satisfies all the requirements of the HIPAMG. The number of IP protocol processors in IP pool is increased until the packets in shared memory stops stacking up with the maximum

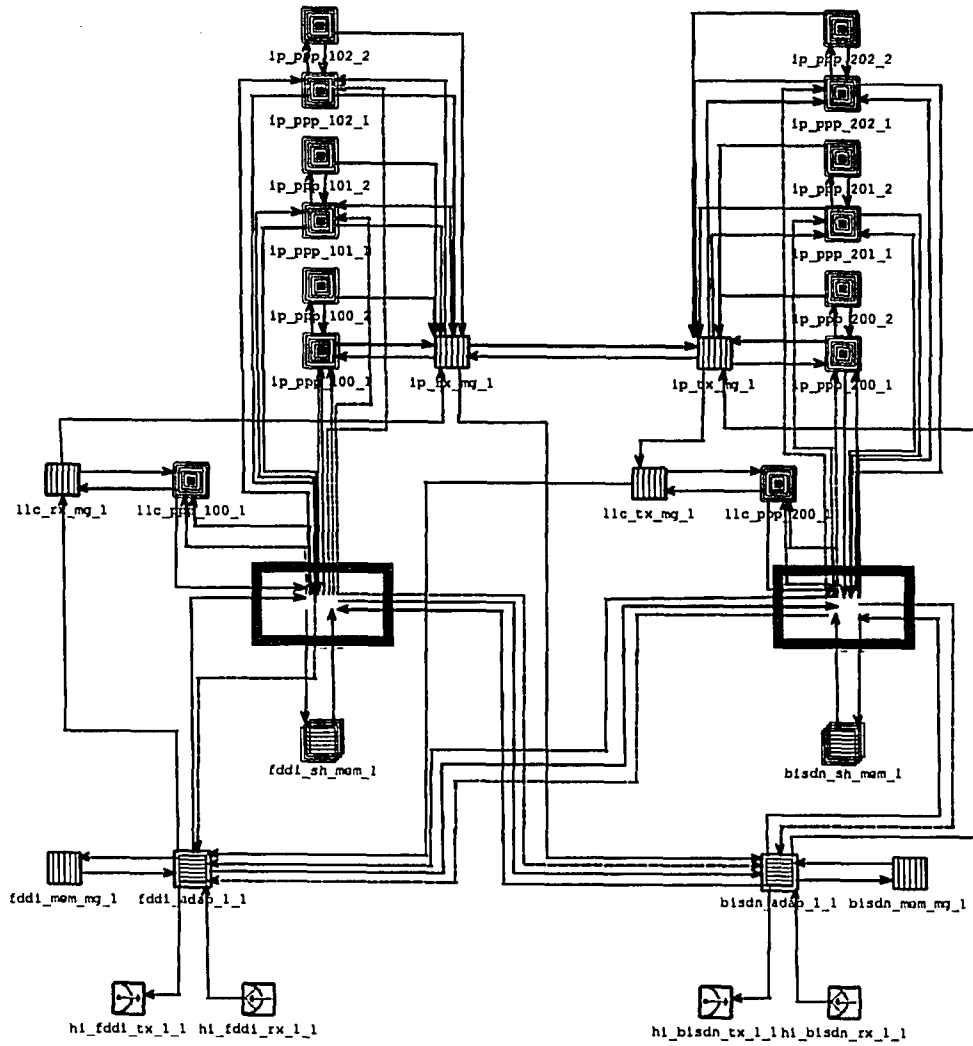


Figure 6.2: hipang-3 model

workload of 100 Mbps. The number of IP protocol processors in IP pool was decided to be 42 as shown in Figure 6.3.

**hipamg\_6 model** hipamg\_6 is the complete final model of the HIPAMG. This model is same as hipamg\_5 model except that the Dynamic Path Allocation Algorithm is implemented in this model. This model shows much better performance than the hipamg\_5 model. This performance improvement is discussed in Chapter 7.

### Structure of the HIPAMG network model

All kinds of simulation models which are used to build the hipamg\_6 model which is the final HIPAMG network model are tabulated in Table 6.1.

### Simulation period

The proper simulation period should be used to get the reasonable simulation result. The transient state of the simulation should not be included in the final computations of the simulation. In this simulation, truncation method [51] was used to remove the transient state from the simulation result analysis if there exists transient state. This method is based on the assumption that the variability during the steady state is less than that during the transient state, which is generally true. Given a sample of  $n$  observations  $\{x_1, x_2, x_3, \dots, x_n\}$ , the truncation method consists of ignoring the first  $l$  observations and then calculating the minimum and maximum of the remaining observations. This step is repeated for  $l = 1, 2, \dots, n - 1$  until the  $(l + 1)$ th observation is neither the minimum nor maximum of the remaining observations. The value of  $l$  at this point gives the length of the transient state. Based on

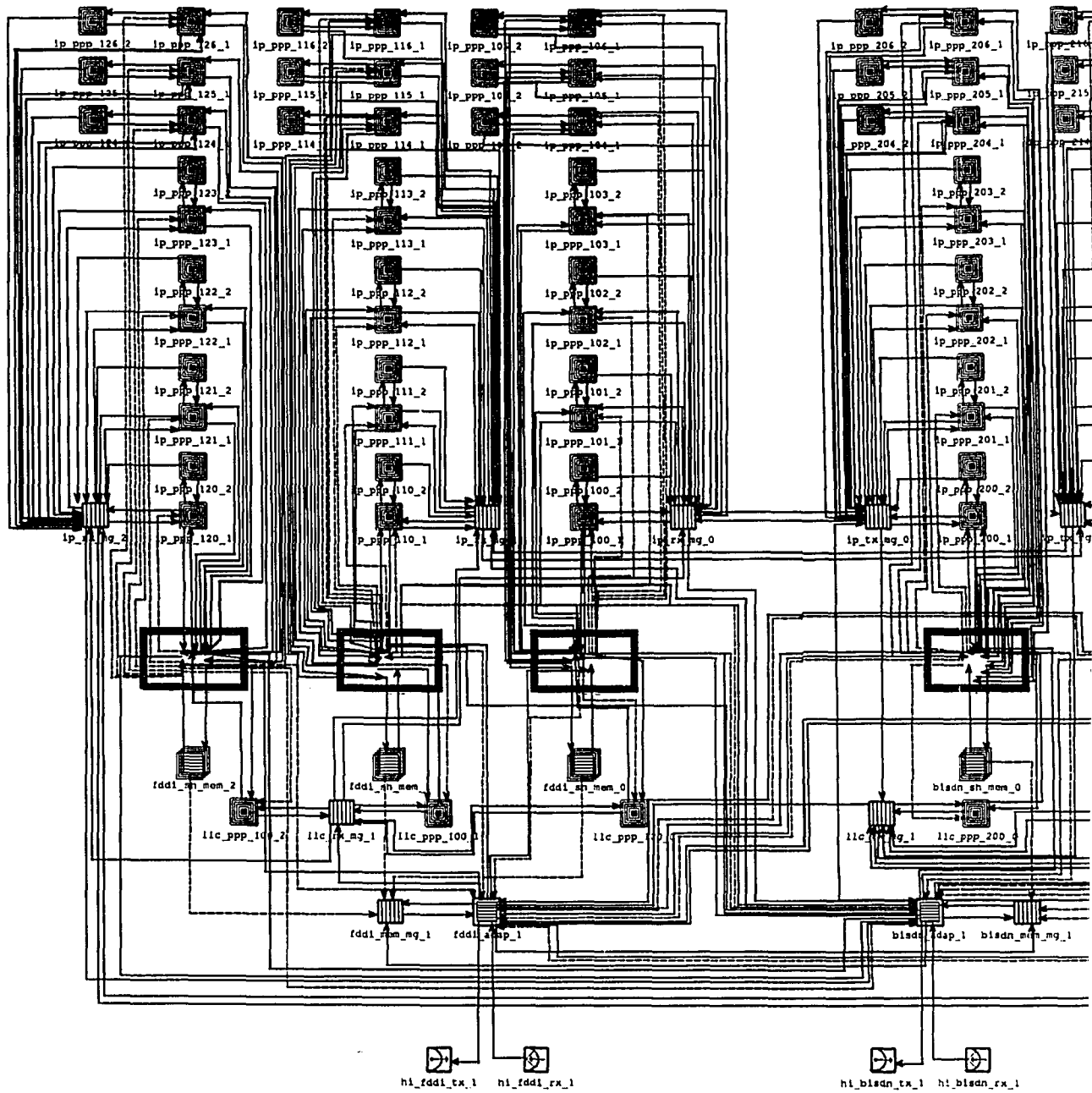
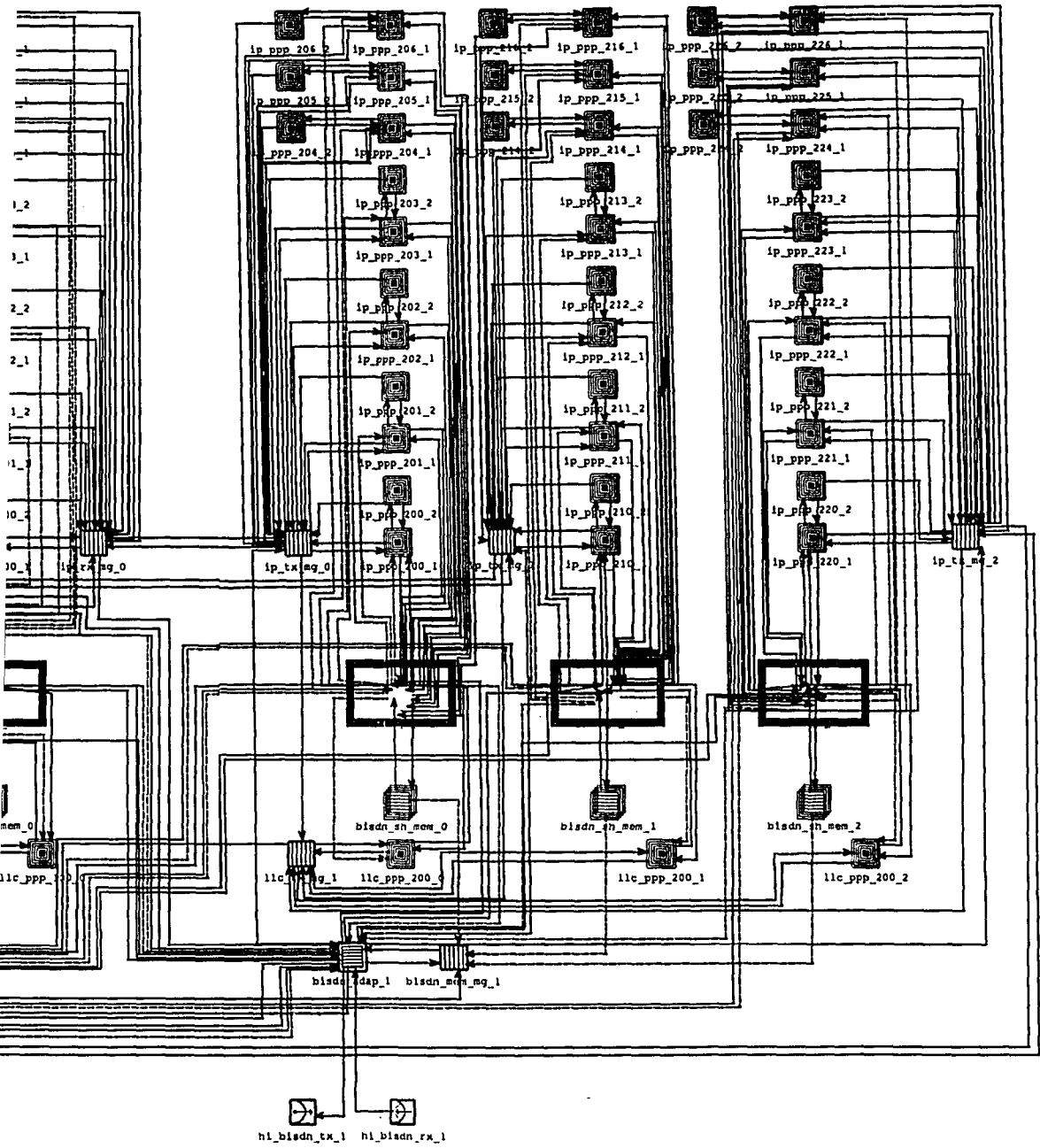


Figure 6.3: hipamg\_5 model





del



Table 6.1: Structure of the HIPAMG network simulation model

Network Model	Node Model	Process Model
net_6	fddi_node_3	fddi_proc_1 ideal generator ptp_receiver ptp_transmitter sink tx_fifo
	hipamg_6	bisdn_adap_5 bus_3 fddi_adap_5 ip_pool_21 ip_pool_22 ip_rx_mang_5 ip_tx_mang_5 llc_pool_1 llc_rx_mang_5 llc_tx_mang_5 mem_mang_5 ptp_receiver ptp_transmitter sh_mem_5
	bisdn_node_3	bisdn_proc_1 ideal generator ptp_receiver ptp_transmitter sink tx_fifo



this method, the simulation period of this simulation starts from 0 because negligible transient state was found in this simulation result.

To choose the length of the simulation is also important. If the simulation is too short, the results may be highly variable. On the other hand, if the simulation is too long, computing resources and manpower may be unnecessarily wasted. Simulation should be run until the confidence interval for the mean value narrows to a desired width. In OPNET simulation, the OPNET gives the confidence intervals of every data automatically using analysis tool facility. Therefore, I chose 0.06 seconds randomly and tried with this simulation period at which all the simulation result data were seen as stabilized. The simulation results shows that 0.06 second simulation period gives narrow enough confidence interval. In this period, 2,032.5 ( $33,875 \times 0.06$ ) FDDI packets are processed when the traffic is 50 Mbps. Therefore, the simulation period was decided to be 0.06 seconds.

### **Simulation Result**

In this section, the simulation result of the hipamg\_5 model is discussed. This result will be compared with the analytical evaluation result in the next chapter. The reason why hipamg\_5 model was chosen here is that the analytical evaluation was done on this model. Even though hipamg\_6 model is the final model, it can't be analytically evaluated because the Dynamic Path Allocation Algorithm is used in hipamg\_6 model and it is impossible to perform the analytical evaluation on this algorithm. Comparing the results of the simulation and the analytical evaluation of hipamg\_5 model is good enough to verify the simulation result of the HIPAMG. The simulation was performed with different seed values which were selected randomly

Table 6.2: HIPAMG simulation results

Traffic (bps)	Average Number of Packets	Packet Transfer Delay (usec)	Throughput (PPS)
20M	2.0067	704.858	597.407
60M	5.1042	750.954	1794.96
80M	6.9653	834.407	2367.4
100M	9.2227	905.512	2983.53
140M	80.7175	5210.43	3595.42

[51]. The average number of packets in the shared memories, the Packet Transfer Delay, and the Throughput were measured for the traffics that flow from the FDDI network to BISDN network. The packet flow in the opposite direction was not analyzed in this dissertation because the result is very similar to the traffic that flow from FDDI to BISDN.

### Results with varying traffic

The simulation parameters were measured by changing the input traffic which is applied to the HIPAMG. Traffic rates of 20 Mbps, 60 Mbps, ,80 Mbps, 100 Mbps, and 140 Mbps are used for this simulation. The simulation results are tabulated in Table 6.2. The plots of the Packet Transfer Delay and Throughput with the varying traffic are shown in Figure 6.4 and 6.5. Figure 6.4 shows that the Packet Transfer Delay is increased very slowly until the traffic reaches 100 Mbps and then is increased suddenly. Figure 6.5 also shows that the Throughput increases linearly up to 100 Mbps traffic then the increasing rate of the Throughput is decreased. These two results show that HIPAMG is fast enough to handle 100 Mbps traffic and becomes the bottleneck when the traffic is more than 100 Mbps.

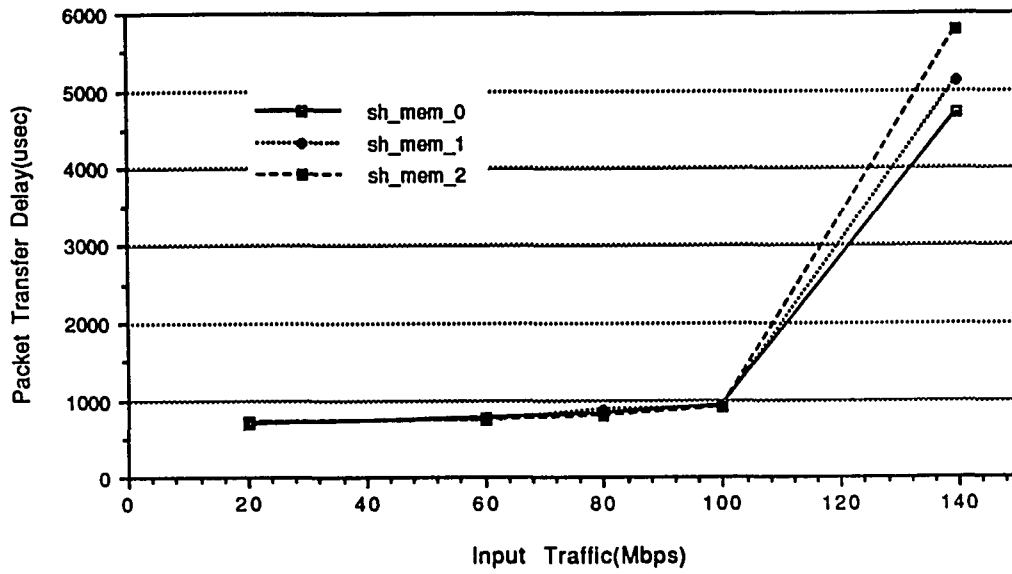


Figure 6.4: Packet Transfer Delay of the hipamg\_5

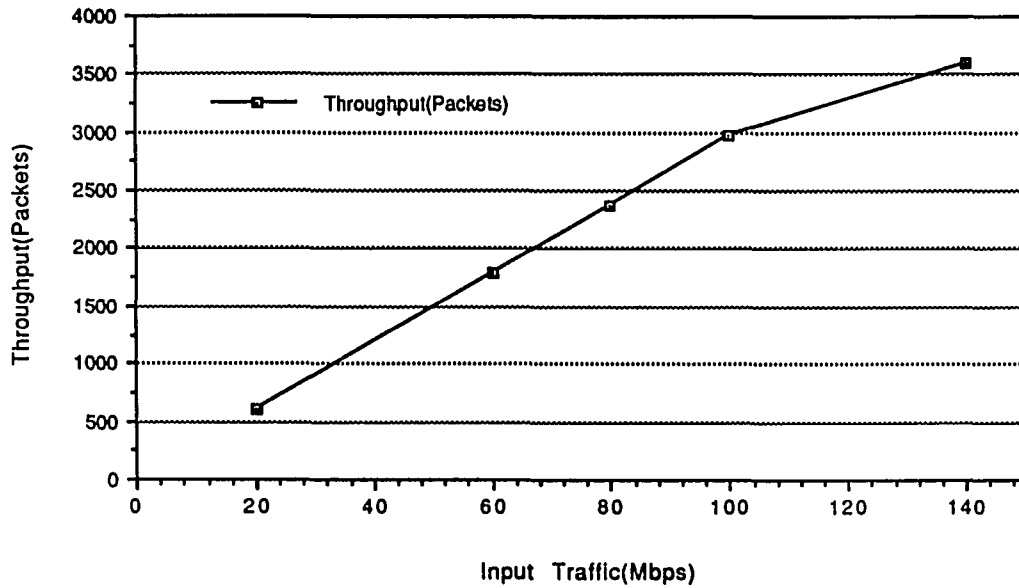


Figure 6.5: Throughput of the hipamg-5

### **Results with 100 Mbps traffic**

To show the change of the simulation parameters with respect to the simulation time at the maximum workload, one of the simulation results with seed value of 4,327 and 100 Mbps traffic is graphed in Figure 6.6. Three graphs on the first row of this figure show the average number of packets in the shared memory module 0,1, and 2 from the left to right. Three graphs on the second row show the Packet Transfer Delay of the compressed video packets, voice packets, and data packets respectively from the left to right. The bottom graph shows the Throughput of the HIPAMG. As shown in this figure, the number of the packets in shared memory and the Packet Transfer Delay are in stable state as simulation time increases. This means the HIPAMG is not a bottleneck with this workload. The detailed simulation result analysis will be discussed in the next chapter.

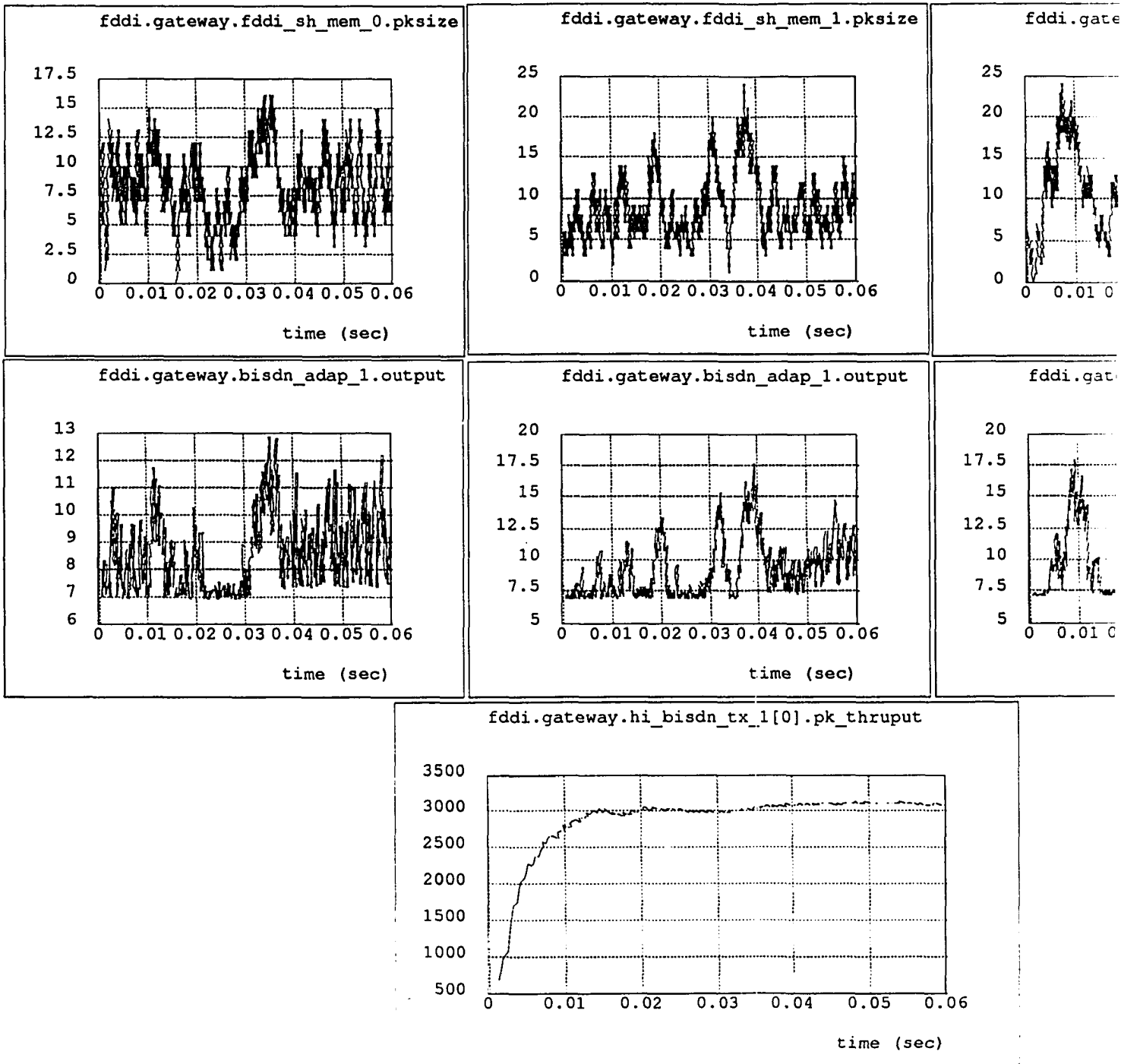
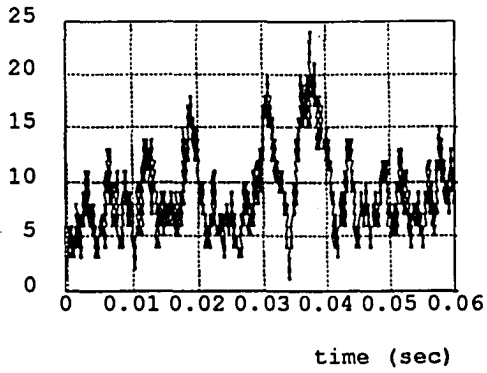


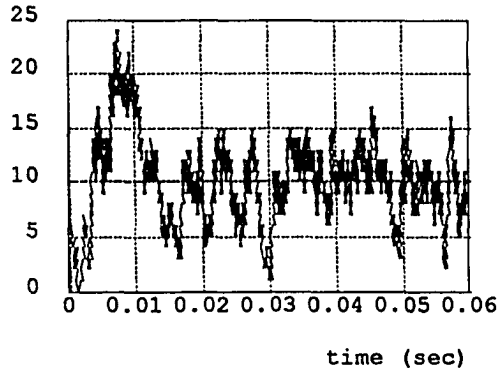
Figure 6.6: hipamg\_5 simulation results with 100 Mbps traffic



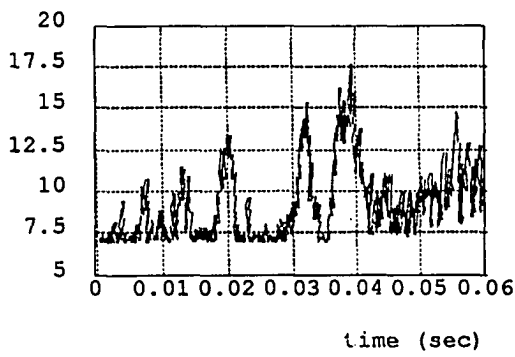
fddi.gateway.fddi\_sh\_mem\_1.pksize



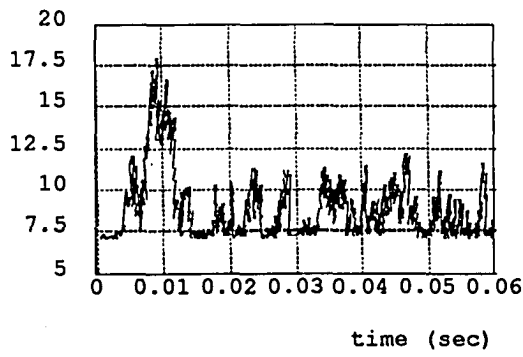
fddi.gateway.fddi\_sh\_mem\_2.pksize



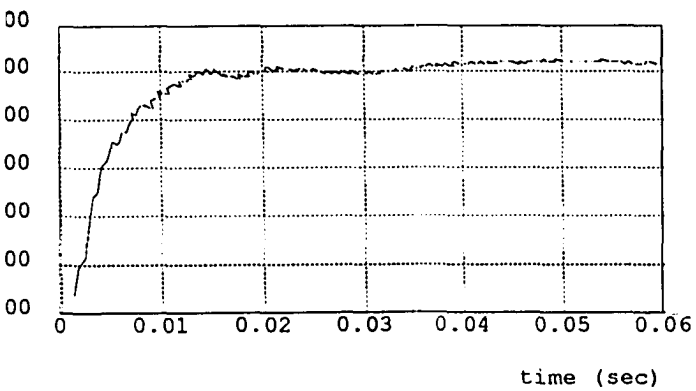
fddi.gateway.bisdn\_adap\_1.output



fddi.gateway.bisdn\_adap\_1.output



fddi.gateway.hi\_bisdn\_tx\_1[0].pk\_thruput



with 100 Mbps traffic





## CHAPTER 7. PERFORMANCE ANALYSIS AND DESIGN OPTIMIZATION

### Analytical Evaluation of the Model

The analytical performance evaluation of the HIPAMG is performed to verify and validate the results of the performance simulation result. The evaluation parameters are same as the simulation parameters which are the Packet Transfer Delay, the number of packets in the shared memory, and the Throughput. As shown in Figure 4.2, packet travels through the HIPAMG and at each stage it waits until it is serviced. Each stage has input buffer which queues the input packets until they are serviced. Therefore, the HIPAMG can be modeled as a queueing network and evaluated using queueing theory [52][53]. The queueing network model of the HIPAMG is shown in Figure 7.1 and the service time of each stage is shown on each stage. The sequence numbers shown in Figure 7.1 are the same numbers appeared in Figure 4.2. Analytical evaluation was performed for the maximum workload of 100 Mbps. The data traffic was chosen among three multi-media traffics for the analytical evaluation.

#### Jackson's theorem

For the analytical evaluation of the HIPAMG, Jackson's theorem was used. To apply this theorem, the following assumptions should be made to the HIPAMG.

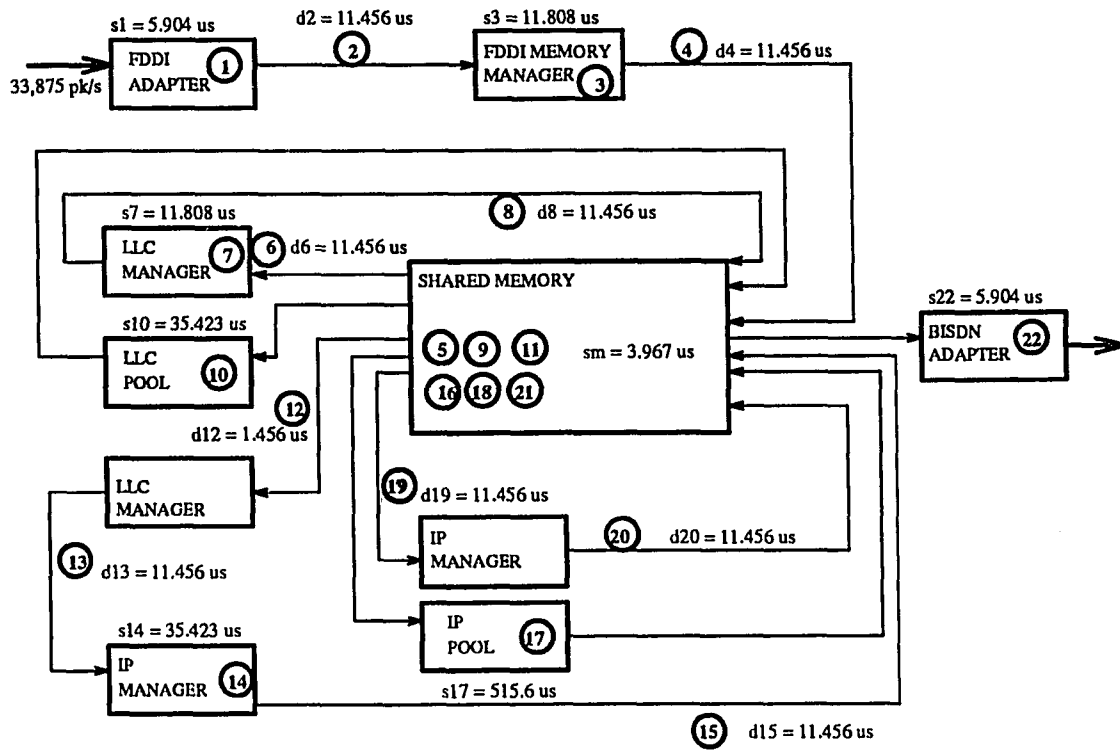


Figure 7.1: Queuing network model of the HIPAMG

- HIPAMG is a queueing network which consists of many nodes, each of which provides an independent exponential service.
- Packets arriving from outside the HIPAMG to any one of the nodes arrive with a Poisson rate.
- Once served at a node, an item goes (immediately) to one of the other nodes with a fixed probability, or out of the HIPAMG.

Jackson's theorem states that in the above network of queues, each node is an independent queueing system, with a Poisson input determined by the principles of partitioning, merging, and tandem queueing. Thus each node may be analyzed separately from the others using the M/M/1 or M/M/N model, and the results may be combined by ordinary statistical methods. Mean delays at each node may be added to derive system delays [54].

### Packet transfer delay calculation

By using the service time of every stages in HIPAMG which were calculated in Chapter 6, we can calculate the Packet Transfer Delay of the HIPAMG. In queueing theory, mean time an item spends in system which is called the mean queueing time,  $t_q$ , is the sum of the mean time an item spends waiting for service and mean service time. In M/M/1 queue,

$$t_q = \frac{s}{1 - \rho} \quad (7.1)$$

**FDDI adapter mean queueing time ( $t_{q1}$ )** The mean queueing time of the FDDI adapter is,

$$t_{q1} = \frac{s_1}{1 - \rho_1} \quad (7.2)$$

$$= \frac{5.904}{1 - 0.4} \quad (7.3)$$

$$\approx 9.84 \mu sec \quad (7.4)$$

Using the same calculation, we can get  $t_{q22}$ .

**FDDI memory manager mean queueing time ( $t_{q3}$ )** Mean queueing time in FDDI memory manager is

$$t_{q3} = \frac{s_3}{1 - \rho_3} \quad (7.5)$$

$$= \frac{11.808}{1 - 0.4} \quad (7.6)$$

$$\approx 19.68 \mu sec \quad (7.7)$$

The mean queueing time in LLC manager,  $t_{q7}$ , is same as  $t_{q3}$ .

**LLC pool mean queueing time ( $t_{q10}$ )** Mean queueing time in LLC pool,  $t_{q10}$ , is

$$t_{q10} = \frac{s_{10}}{1 - \rho_{10}} \quad (7.8)$$

$$= \frac{35.423}{1 - 0.4} \quad (7.9)$$

$$\approx 59.038 \mu sec \quad (7.10)$$

The mean queueing time in IP manager,  $t_{q14}$ , is also 59.038  $\mu sec$ .

**Shared memory mean queueing time ( $t_{qm}$ )** In this HIPAMG system, three shared memory modules are implemented to handle the multi-media environment. Each shared memory module works independently and stores compressed video, voice, and data packets respectively as described in Chapter 3. Therefore, the shared memory module can be modeled as three M/M/1 queue. The mean queueing time of the single memory module,  $t_{qm}$ , is;

$$t_{qm} = \frac{1}{\frac{1}{s_m} - \lambda_m} \quad (7.11)$$

where,  $\lambda_m$  is the total packet arrival rate to one shared memory module and  $s_m$  is the average service time of the shared memory module. As shown in Figure 7.1, the total packet arrival rate to the one shared memory module,  $\lambda_m$ , is the sum of all the packet arrival rate to one shared memory module. Therefore,

$$\lambda_m = \frac{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6}{3} \quad (7.12)$$

$$= 33,875 \times 2 \quad (7.13)$$

$$= 67,750(\text{packets}/\text{sec}) \quad (7.14)$$

The average service time of the shared memory,  $s_m$ , is the average of all the  $s$ 's.

$$s_m = \frac{s_5 + s_9 + s_{11} + s_{16} + s_{18} + s_{21}}{6} \quad (7.15)$$

$$= \frac{4.98 + 0.46 + 0.46 + 6.46 + 6.46 + 4.98}{6} \quad (7.16)$$

$$= 3.967(\mu\text{sec}) \quad (7.17)$$

Therefore, the mean queueing time of the shared memory module,  $t_{qm}$ , is;

$$t_{qm} = \frac{1}{\frac{1}{3.967 \times 10^{-6}} - 67,750} \quad (7.18)$$

$$= 5.42(\mu\text{sec}) \quad (7.19)$$

**IP pool mean queueing time ( $t_{q17}$ )** The mean queueing time of the IP pool when we assume there is only one IP protocol processor pair can be calculated using

$$t_{q17} = \frac{s_{17}}{1 - \rho_{17}} \quad (7.20)$$

$$(7.21)$$

But

$$\rho_{17} = \lambda_{17} \times s_{17} \quad (7.22)$$

$$= 33,875 \times 515.6 \times 10^{-6} \quad (7.23)$$

$$= 17.466 \quad (7.24)$$

This shows that with only one IP protocol processor pair, IP pool can not process the packets fast enough.

Therefore, the simulation was used to decide the proper number of IP protocol processor pairs. As described in previous chapter, IP pool include 42 protocol processors in it. Because IP pool consists of three parts, each parts handles one third of the total traffic. Each parts of the IP pool has 7 IP protocol processor pairs and each part of the IP pool can be modeled as M/M/7 queue. Therefore, the mean time a packet spends in the IP pool is;

$$t_{q17} = s_{17} + \left[ \frac{\frac{(r_{17})^8}{7}}{\lambda_{17} \times 7! \times \left(1 - \frac{r_{17}}{7}\right)^2} \right] \times P_0 \quad (7.25)$$

$$= 515.6 \times 10^{-6} + \left[ \frac{\frac{(5.82)^8}{7}}{11,292 \times 5040 \times \left(1 - \frac{5.82}{7}\right)^2} \right] \times P_0 \quad (7.26)$$

where,

$$\frac{1}{P_0} = \sum_{i=0}^6 \frac{r^i}{i!} + \frac{r^7}{7!} \times \frac{1}{1 - \frac{r}{7}} \quad (7.27)$$

$$= 1 + \frac{r}{1} + \frac{r^2}{2} + \frac{r^3}{6} + \frac{r^4}{24} + \frac{r^5}{120} + \frac{r^6}{720} + \frac{r^7}{5040} \times \frac{1}{1 - \frac{r}{7}} \quad (7.28)$$

$$= \frac{1}{2.082 \times 10^{-3}} \quad (7.29)$$

Therefore,

$$t_{q17} = 515.6 \times 10^{-6} + 0.1163 \times 2.082 \times 10^{-3} \quad (7.30)$$

$$= 757.71 \times 10^{-6} \quad (7.31)$$

**Mean packet transfer delay** The delay time between the stages are calculated in equation (6.6). The results of the above calculations are shown in Table 7.1. From this table, the mean Packet Transfer Delay is the sum of all the  $t_{qn}$  and  $d_n$ , which is 1070.8 ( $\mu sec$ ) as shown in Table 7.1.

### Number of packets in shared memory

The packets are stored in the shared memory at step 5 of the Figure 4.2 and are removed from the shared memory at step 21. We can model this situation as M/M/7 queue whose average packet arrival rate,  $\lambda$ , is 11,292 (packets / sec) and the packet service time,  $s$ , is;

$$s = \sum_{i=6}^{20} d_i + \sum_{i=7}^{18} t_{qi} \quad (7.32)$$

$$= 80.192 + 917.496 \quad (7.33)$$

$$= 997.688(\mu sec) \quad (7.34)$$

Table 7.1: Mean queuing time and delay time of the HIPAMG

	Stage	Mean Queuing Time $t_{qn}(\mu sec)$	Delay Time $d_n(\mu sec)$
1	FDDI.ADAPTER	9.84	-
2	1 to 2	-	11.456
3	FDDI.MEM.MG	19.68	-
4	3 to 5	-	11.456
5	SH.MEM.WRITE	5.42	-
6	5 to 7	-	11.456
7	LLC.MANAGER	19.68	-
8	7 to 9	-	11.456
9	SH.MEM.READ	5.42	-
10	LLC.POOL	59.038	-
11	SH.MEM.WRITE	5.42	-
12	11 to LLC.MANAGER	-	11.456
13	LLC.MANAGER to 14	-	11.456
14	IP.MANAGER	59.038	-
15	14 to 16	-	11.456
16	SH.MEM.READ	5.42	-
17	IP.POOL	758.06	-
18	SH.MEM.WRITE	5.42	-
19	18 to IP.MANAGER	-	11.456
20	IP.MANAGER to 21	-	11.456
21	SH.MEM.READ	5.42	-
22	BISDN.ADAPTER	9.84	-
Sub Total		967.696	103.104
Total			1070.8



The packet service time is calculated from the values in Table 7.1. Therefore, the number of packets inside the shared memory which is the number of packets in service is;

$$r = \lambda \times s \quad (7.35)$$

$$= 11,292 \times 997.688 \times 10^{-6} \quad (7.36)$$

$$= 11.266(\text{packets}) \quad (7.37)$$

### Throughput

Throughput is defined as the rate (packets per second) at which the packets can be serviced by the system. In this HIPAMG system, the Throughput is the number of packets which leave the HIPAMG system to the BISDN network per second. The BISDN adapter makes one STS-3c packet after it received 11 LLC packets. Therefore, the throughput is  $33,875/11 \approx 3,080$  (packets/sec).

### Simulation vs. Analytical Evaluation

The simulation results and analytical evaluation of the HIPAMG are compared and discussed in this section. These two performance evaluations are performed using the data traffic with the traffic rate of 100 Mbps. As shown in Table 7.2, the results of the simulation and the analytical evaluation are close. The differences of the two results are due to the differences of the simulation model and the analytical evaluation model. For example, most of the stages of the analytical evaluation models are assumed to be M/M/1 queues, but service times of some stages of HIPAMG simulation model are implemented as having constant distribution (not exponential

Table 7.2: Simulation and analytical evaluation result

	Packet Delay time( $\mu sec$ )	Number of Packets in shared memory	Throughput (packets/sec)
Simulation	905.512	9.2227	2,983.53
Analytical	1070.8	11.266	3,080
% Difference	18.25	22.156	3.23

distribution) in order to make the simulation model same as the actual HIPAMG. As a result, these two results are good enough to verify and validate the results of each result.

### Improvements due to the Dynamic Path Allocation Algorithm

The Dynamic Path Allocation Algorithm is implemented in hipamg\_6 model to improve the performance of the hipamg\_5 model. In this section, the performance improvement of the HIPAMG due to the implementation of the Dynamic Path Allocation Algorithm is described.

#### Packet Transfer Delay

The simulation result of the hipamg\_6 model shows some improvements in Packet Transfer Delay when compared to the result of the hipamg\_5 model. Table 7.3 shows the Packet Transfer Delay in hipamg\_5 and hipamg\_6 model at the 100 Mbps traffic. The Packet Transfer Delay of the compressed video and voice are improved by 9.917% and 3.419% respectively. Here the % improvement is calculated by using the following equation.

$$\frac{PTD_{hipamg_5} - PTD_{hipamg_6}}{PTD_{hipamg_6}} \times 100 \quad (7.38)$$

Table 7.3: Packet Transfer Delay at 100 Mbps traffic

	Packet Transfer Delay ( $\mu sec$ )		
	compressed video	voice	data
hipamg_5	930.669	892.529	893.339
hipamg_6	846.701	863.021	959.957
% improvement	9.917 %	3.419 %	-6.94 %

Table 7.4: Number of packets in shared memory at 100 Mbps traffic

	Number of packets in shared memory		
	sh_mem_0	sh_mem_1	sh_mem_2
hipamg_5	9.65342	9.00969	9.0049
hipamg_6	7.33985	8.17516	10.3452
% improvement	31.52%	10.21%	-12.95%

These two traffic need to be delivered as fast as possible as shown in Table 3.2. But the data traffic can endure some delay inside the HIPAMG. As a result, the Dynamic Path Allocation Algorithm makes good improvement in Packet Transfer Delay of the HIPAMG.

#### Number of packets in shared memory

The shared memory stores the packets while they are processed inside the HIPAMG. Therefore, by analyzing the number of packets in shared memory, we can find out the minimum size of the shared memory because the size of the shared memory should be large enough to keep all the packets being processed. As shown in Table 7.4, by using the Dynamic Path Allocation Algorithm, the number of packets in shared memory module 0 and shared memory module 1 are decreased by 31.52% and 10.21% respectively. The number of packets in shared memory module 2 is increased by 12.95%.

## Design Optimization

Two of the HIPAMG design parameters, the number of protocol processors in IP pool and the size of the shared memory, are tuned to optimize the gateway design.

### Number of protocol processors in IP pool

To increase the processing speed of the IP pool, the number of protocol processors should be increased. But the price and the complexity of the HIPAMG will go up with more protocol processors. Therefore, the optimal number of protocol processors should be decided to build the best HIPAMG. The optimal number of IP protocol processors is the minimum number of IP protocol processors with which the HIPAMG is able to handle the maximum traffic (100 Mbps) of the HIPAMG. This number is decided following the result of the simulation. As described in Chapter 6, 42 IP protocol processors are used in HIPAMG. As shown in Figures 6.4 and 6.5, the HIPAMG with 42 IP protocol processors is not a bottleneck of the network system up to 100 Mbps traffic.

### Shared memory size

To optimize the size of the shared memory, the minimum size of the shared memory which is not overflowed should be decided. From the simulation result of the hipamg\_6 model with 100 Mbps traffic, the 99 % confidence interval of the number of packets in shared memory and the maximum number of packets in shared memory are tabulated in Table 7.5

In this design, we propose 1.5 times of the maximum number of packets as the actual size of the shared memory to give enough margin to the size of the shared

Table 7.5: Optimal shared memory size at 100 Mbps traffic

	99% confidence interval (packets)	maximum number of packets (packets)	Proposed memory size (bytes)
sh_mem_0	6.8571 - 7.8226	15	3,510
sh_mem_1	8.1667 - 9.2636	18	4,212
sh_mem_2	9.4326 - 11.2577	18	4,212

memory. Therefore, the size of the shared memory in bytes is  $(1.5) \times (1248/8) \times$  (size of the shared memory in packets). The result is shown in Table 7.5.

### Cost of the HIPAMG

One of the goals of this research is to design a gateway which is cost effective. With respect to the number of protocol processors on a HIPAMG implementation, the cost of a system grows almost linearly. The performance of this system grows again almost linearly with respect to the number of protocol processors. The HIPAMG designed in this research can connect one FDDI network to one BISDN network and the simulation result shows that 42 protocol processors are needed in FDDI side of the IP protocol processor pool of the HIPAMG. BISDN side of the IP protocol processor pool also needs 42 protocol processors. Therefore, the total number of protocol processors needed in IP layer pool of HIPAMG is 84.

In the real HIPAMG hardware implementation, MC68332BCC [46] can work as a protocol processor. The cost of the IP layer pool of HIPAMG is then,

$$84 \times \$100^1 = \$8,400 \quad (7.39)$$

In order to estimate the price of the HIPAMG system, we assume the price of the

<sup>1</sup>The retail unit price of the MC68332BCC in July 1992.

LLC layer pool and Q.931 layer pool are same as the price of the IP layer pool. We also assume the prices of FDDI adapter and BISDN adapter are same as the price of the IP layer pool and the prices of the shared memory, bus system, and other miscellaneous parts are negligible. Therefore, the estimated price of the HIPAMG is

$$\$8,400 \times 5 = \$42,000 \quad (7.40)$$

By comparing this price (\$ 42,000) to the prices of the general purpose computers which have been tried to be used as a high-speed communication gateway; \$1.8 million (*Butterfly<sup>TM</sup>* parallel computer with 128 processors) [55] and \$17 million (Cray-2 super computer) [56], we can get the idea how the HIPAMG can be the cost effective high-bandwidth communication gateway system.

## CHAPTER 8. CONCLUSIONS

A multiprocessor high-bandwidth communication gateway based on a Protocol Processor Pool Architecture has been designed and its performance has been simulated and analyzed. HIPAMG was designed based on a new design concept of the high-speed communication gateway so-called Protocol Processor Pool Architecture which has a pool of micro-controllers as its processing unit. The design goals of the HIPAMG are the high-performance, efficient multi-media handling ability, low cost, and the flexibility.

The best parallel protocol implementation of the parallel architected gateway can be achieved when the underlying multiprocessor architecture and the way a communication protocol architecture is specified are properly matched. To achieve this, the communication protocol architecture of the HIPAMG is analyzed and the following hardware architectures are implemented in HIPAMG.

- Multiple processors for one protocol process (processor pair)
- Many independent communication connections paths
- Processor pool architecture
- Pipe-line architecture
- Separate  $R_x$  and  $T_x$  hardware

- Shared memory and data pointer transfer

To improve the processing speed of the communication protocol, parallelism inside the protocols should be studied and implemented using the many processor protocol processors. In this research, IP protocol was studied and the processing speed of the IP is improved by 30.33% by using the IP protocol processor pair. After finishing the simulation and analytical evaluation, the performance evaluation of both are verified and validated by comparing two results.

Lastly, the HIPAMG design parameters are tuned to optimize the design. The number of protocol processors in IP pool needed to support the maximum traffic (100 Mbps) is 42. The shared memory size was decided after implementing the Dynamic Path Allocation Algorithm. The Dynamic Path Allocation Algorithm improves the Packet Transfer Delay of compressed video and voice by 9.917% and 3.419% respectively. It also decreased the number of packets in shared memory module 0 and shared memory module 1 by 31.52% and 10.21% respectively. But the number of packets in shared memory module 2 is increased by 12.95%. Therefore, this makes it possible to reduce the size of the shared memory module 0 and shared memory module 1. The performance results of the optimized HIPAMG (hipamg\_6) are shown in Figure 8.1 and 8.2 by graphing the Packet Transfer Delay and Throughput with the varying input traffic.

The primary contributions of this research can be described as:

- to design a multiprocessor high-bandwidth communication gateway which has the characteristics of high performance, efficient multi-media handling ability, low cost, and the flexibility.



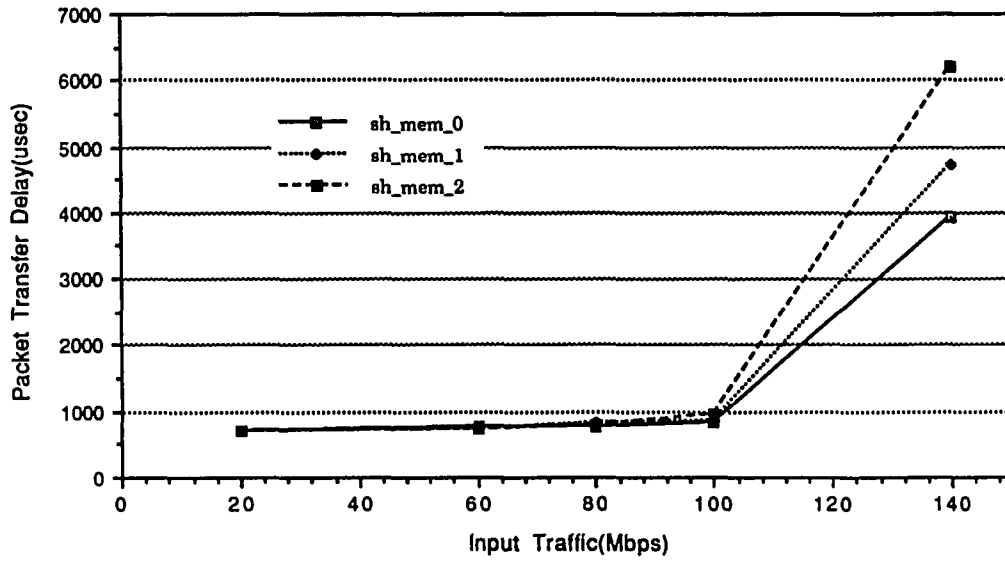


Figure 8.1: Packet Transfer Delay of the hipamg\_6

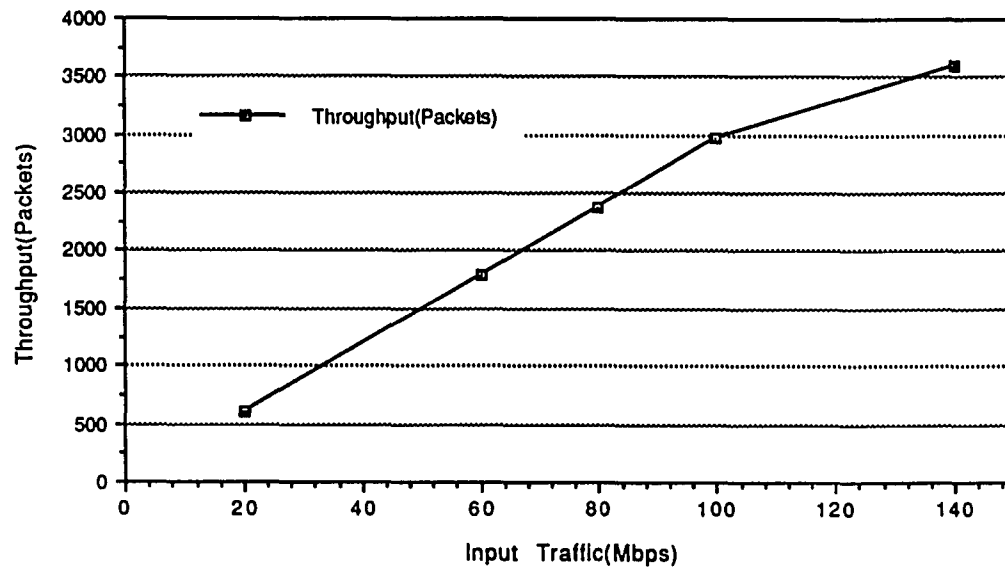


Figure 8.2: Throughput of the hipamg\_6

- to propose a new concept of Protocol Processor Pool Architecture in the high-speed parallel architected gateway design.
- to develop the Dynamic Path Allocation Algorithm.
- to perform the process level simulation using OPNET graphic simulation package.

As a whole, the conclusion of this research is: The multiprocessor high-bandwidth communication gateway based on a Protocol Processor Pool Architecture was designed and the simulation result showed that HIPAMG satisfies all the design goal of this research that is to build the high-bandwidth communication gateway which has the characteristics of high performance, efficient multi-media handling ability, low cost, and the flexibility.

### **Future Work**

This research is the first step to realize the HIPAMG in the real world. In this research, the basic idea was proposed and the functionality and performance are proved.

More work that should be done in the future includes software design for the control phase of the communication and the subdivision and implementation of the LLC, Q.931 protocols. As the Dynamic Path Allocation Algorithm gives good improvement to the HIPAMG performance, the improved algorithms for path allocation and protocol processor allocation inside the protocol processor pool may be developed. Then detailed hardware circuit design should be finished to build the real HIPAMG hardware. The BISDN adapter and FDDI adapter should be available.

HIPAMG designed in this research can handle the traffic up to 100 Mbps. However, HIPAMG design is flexible enough to improve its performance by increasing the number of processors in protocol processor pool using the processors currently available. The very high-bandwidth communication gateway for the 1 Gbps network is recommended to be studied with the concept of the HIPAMG.

## BIBLIOGRAPHY

- [1] Partridge, Craig. "How Slow Is One Gigabit Per Second?." In *Proceedings of the SIGCOM '89 Symposium - Computer Communications Review* Vol. 20, No. 1 (1990): 44-53.
- [2] Jain, Niraj, Mischa Schwartz, and Theodore Bashkow. "Transport Protocol Processing at GBPS Rates." In *Proceedings of the SIGCOM '90 Symposium - Computer Communications Review* Vol. 20, No. 4 (1990): 188-199.
- [3] Skov, Morten. "Implementation of physical and Media Access Protocols for High-Speed Networks." *IEEE Communications Magazine*, June 1989, 45-53.
- [4] Giarrizzo, Dario. "High-speed Parallel Protocol Implementation." In *Proceedings of the IFIP WG 6.1/WG 6.4 International Workshop on Protocols for High-speed Networks, Zurich, Switzerland, May 1989*, edited by H. Rudin and R. Williamson, 165-179. New York: North-Holland Publishing Company, 1989.
- [5] Zitterbart, Martina. "High-speed Transport Components." *IEEE Network Magazine*, January 1991, 54-63.
- [6] Svobodova, L. "Measured Performance of Transport Services in LANs." *IBM Research Report RZ 1799(#64663)*, March 1989.
- [7] Chesson, G. "XTP/PE Design Considerations." In *Proceedings of the IFIP WG 6.1/WG 6.4 International Workshop on Protocols for High-speed Networks, Zurich, Switzerland, May 1989*, edited by H. Rudin and R. Williamson, 27-33. New York: North-Holland Publishing Company, 1989.
- [8] Watson, Richard W. "The Delta-t Transport Protocol: Features and Experience." In *Proceedings of the IFIP WG 6.1/WG 6.4 International Workshop on Protocols for High-speed Networks, Zurich, Switzerland, May 1989*, edited by H.

- Rudin and R. Williamson, 3-17. New York: North-Holland Publishing Company, 1989.
- [9] Malek, Manu. "Integrated Voice and Data Communications Overview." *IEEE Communications Magazine*, June 1988, 5-15.
- [10] Borman, D.A. "Implementing TCP/IP on a Cray Computer." In *Proceedings of the SIGCOM '88 Symposium - Computer Communication Review*, Vol. 19, No. 2 (1989): 11-15.
- [11] Zitterbart, Martina. "High-speed Protocol Implementations Based on a Multi-processor Architecture." In *Proceedings of the IFIP WG 6.1/WG 6.4 International Workshop on Protocols for High-speed Networks, Zurich, Switzerland, May 1989*, edited by H. Rudin and R. Williamson, 151-163. New York: North-Holland Publishing Company, 1989.
- [12] Motorola Inc. *CPU 32 Reference Manual*. Phoenix, Arizona: Motorola Literature Distribution, 1989.
- [13] Cohen, Alain J. *Modeling Manual - OPNET Optimized Network Engineering Tools*. Washington, D.C.: MIL 3, Inc., 1990
- [14] CCITT COM XVIII-228-E, Mar. 1984.
- [15] Händel, Rainer. "Evolution of ISDN Towards Broadband ISDN." *Current advances in LANs, MANs, and ISDN*, edited by B.G.Kim, 302-308. Norwood, MA: Artech house Inc., 1989.
- [16] Kano, Sadahiko, Kitami Ken'ichi, and Kawarasaki Masatoshi. "ISDN Standardization." In *Proceedings of IEEE*, Vol. 79, no. 2, (1991): 118-123.
- [17] Spragins, John D. *Telecommunications: Protocols and Design*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1991.
- [18] Yuklea, Harry. "An FDDI Overview:Some Questions Answered." *Computer Networks and ISDN Systems*, Vol. 19, (1990): 228-232.
- [19] Stallings, William. *Handbook of Computer Communications Standards, Local Network Standards*. New York: Macmillan Publishing Company, 1987.
- [20] DeCegama, L. Angel. *The Technology of Parallel Processing: Volume I, Parallel Processing Architectures and VLSI Hardware*. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

- [21] Reed, Daniel A., Richard M. Fujimoto. *Multicomputer Networks: Message-Based Parallel Processing*. Cambridge, Massachusetts: MIT Press, 1987.
- [22] Siegel, Jay Howard. *Interconnection Networks for Large-Scale Parallel Processing*. Lexington, Massachusetts: Lexington Books, 1985.
- [23] Cheriton, David R. "VMTP as the Transport Layer for High-Performance Distributed Systems." *IEEE Communications Magazine*, June 1989, 37-44.
- [24] Fletcher, J.G. *Introduction to LINCS*. Available through the Lawrence Livermore National Laboratory Computer Center as Chapters 1-12, Lawrence Livermore National Laboratory, Tentacle April 1982 to March 1983.
- [25] Haas, Zygmunt. "A Protocol Structure for High-Speed Communication over Broadband ISDN." *IEEE Network Magazine*, January 1991, 64-70.
- [26] Mazraani, Y. Tony. "Specification of a Multipoint CONGRAM-Oriented High Performance Internet Protocol." *Technical Report WUCS-89-13*, St. Louis: Department of Computer Science, Washington University, 1989.
- [27] Clark, D. "The Design Philosophy of the DARPA Internet Protocols." In *Proceedings of the ACM SIGCOMM'88 Symposium - Computer Communication Review*, Vol. 20, No. 1 (1990): 106-114.
- [28] Parulkar, G.M. "The Next Generation of Internetworking." In *Proceedings of the ACM SIGCOMM'88 Symposium - Computer Communication Review*, Vol. 20, No. 1 (1990): 18-43.
- [29] Abu-Amara, H., T. Barzilai, and Y. Yemini. "PSi: A Silicon Compiler for Very Fast Protocol Processing." In *Proceedings of the IFIP WG 6.1/WG 6.4 International Workshop on Protocols for High-speed Networks, Zurich, Switzerland, May 1989*, edited by H. Rudin and R. Williamson, 181-195. New York: North-Holland Publishing Company, 1989.
- [30] Meleis, H.E., A.H. Tantawy. "The Modular Communication Machine(MCM): An Architecture for High Performance Network." *Research Report RC 14541 (#65062)*, Yorktown Heights, NY: IBM Research Division, T. J. Watson Research Center, 1989.
- [31] Kanakia, Hemant and David Cheriton. "The VMP Network Adapter Board(NAB): High Performance Network Communication for Multiprocessors." In *Proc. SIGMETRICS '88*, 175-187. New York: Association for Computing Machinery, 1988.

- [32] Cheriton, D.R., Gert Slavenberg, and Patrick Boyle. "Software-Controlled Caches in the VMP Multiprocessor." *Technical Report STAN-CS-86-1105*, Computer Science Dept., Stanford University, 1986.
- [33] Rupperecht, M., M. Dresen, and F. Fehlau. "A High-Throughput LAN-Controller for High Bit Rate Systems." In *Proc. EFOC/LAN '88, Amsterdam, Holland, July 14-16, 1988*.
- [34] INMOS Limited. *Transputer Reference Manual*. Hertfordshire, Great Britain: Prentice Hall International(UK) Ltd., 1988.
- [35] Harp, Gordon. *Transputer Applications*. London, Greate Britain: Pitman Publishing, 1989.
- [36] Ulrich, R., R. Hinze, and H. Dietsch. "Erfahrungen bei der Durchsatzoptimierung eines Transputer-Netzwerkes für ISO-OSI Architekturen am Beispiel der LLC-Teilschicht." *GI/ITG Fachtagung Messung, Modellierung und Bewertung von Rechensystemen*, Braunschweig, Sept. 1989.
- [37] Butscher, B., P. Egloff, R. Popescu-Zeletin. "BERKOM - A Broadband ISDN Project." In *Proceedings of the 1988 International Zurich Seminar on Digital Communications, March 1988*, 77-86. 1988.
- [38] Zitterbart, Martina. "A Multiprocessor Architecture for High-Speed Network Interconnections." In *Proc. INFOCOM'89, Ottawa, Canada, Apr. 1989*, 212-218. Washington D.C.: IEEE Computer Society Press, 1989.
- [39] Definicon Systems Inc., *The Parallel Transputer System, Operation and Installation Manual*. Release 2.0, 1988.
- [40] Bhargava, A., J. F. Kurose, D. Towsley, and G. Vanleemput. "Performance comparison of error control schemes in high-speed computer communication networks." *IEEE J. Select. Areas Commun.*, vol. 6, (1988): 1565-1575.
- [41] Verma, Pramode K. *ISDN Systems: Architecture, Technology, and Applications*. Englewood Cliffs, New Jersey: Prentice Hall, 1990.
- [42] Zhang, Xian-Yu, Robert H. Deng. "Gateway Design for LAN Interconnection via ISDN." *Computer Networks and ISDN Systems*, Vol. 19, (1990): 43-51.
- [43] Hehmann, B. Dietmar. "High-speed Transport Systems for Multi-media Application." In *Proceedings of the IFIP WG 6.1/WG 6.4 International Workshop on Protocols for High-speed Networks, Zurich, Switzerland, May 1989*, edited

- by H. Rudin and R. Williamson, 303-321, New York: North-Holland Publishing Company, 1989.
- [44] CCITT Study Group XVIII (Broadband Task Group), *Part C of the Report of the Seoul Meeting, Jan. 25 to Feb. 5, 1988*. CCITT COM XVIII-R55(C)-E, 1988.
  - [45] Burren, J.W. *The Interaction of Computing and Communications*. edited by R., Exeter, Devonshire, England: Scantlebury, Wheaton & Company Limited, 1984.
  - [46] Motorola Inc., *MC 68332 System Integration Module User's Manual*. Phoenix, Arizona: Motorola Literature Distribution, 1989.
  - [47] Comer, Douglas E., David L. Stevens. *Internetworking with TCP/IP: Volume II, Design, Implementation, and Internals*. Englewood Cliffs, New Jersey: Prentice Hall, 1991.
  - [48] Tarnay, K. *Protocol Specification and Testing*. New York: Plenum Press, 1991.
  - [49] Stallings, William. *Data and Computer Communications*. New York: Macmillan Publishing Company, 1991.
  - [50] Cohen, Alain J. *Tool Operations Manual - OPNET Optimized Network Engineering Tools*. Washington, D.C.: MIL 3, Inc., 1990
  - [51] Jain, Raj. *The Art of Computer Systems Performance Analysis*. 454-455. New York: John Wiley & Sons, Inc., 1991.
  - [52] Nussbaumer, Henri. *Computer Communication Systems*. Vol. 2. West Sussex, England: John Wiley & Sons Ltd., 1990.
  - [53] Garzia, Ricardo F. and Mario R. Garzia. *Network Modeling, Simulation, and Analysis*. New York: Marcel Dekker, Inc., 1990.
  - [54] Kleinrock, L. *Queueing Systems, Vol. II: Computer Applications*. New York: Wiley, 1976.
  - [55] Juedes, David, Jangkyung Kim, and Susan Mayer. "The *Butterfly*<sup>TM</sup> Parallel Processor." Presented at the Computer Science Department, Iowa State University, Ames, IA, November, 1988.
  - [56] Oehler, Julie, Tajinder Pal Singh, and Xuehuang Qiu. "Supercomputers." Presented at the Computer Science Department, Iowa State University, Ames, IA, November, 1988.