TECHNICAL RESEARCH REPORT

OPNET Simulation Model of the ALAX

by I.Y. Chong, A. Makowski, P. Narayan, J. Kim,
T. Christofili, G. Charleston, S. Rao, R. Gupta

CSHCN T.R. 95-16
(ISR T.R. 95-113)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

OPNET Simulation Model of the ALAX

Jangkyung Kim ; *jkkim@pec.etri.re.kr*
Protocol Engineering Center
Electronics & Telecommunications Research Institute
Taejon, 305-600, South Korea

Thomas Christofili ; *christof@src.umd.edu*
Institute for Systems Research
University of Maryland, College Park MD. 20742

Release 1.0
December 1995

Laboratory
for
Advanced Switching Technologies

ISR, University of Maryland
College Park, MD 20742

Project team members:

Dr. Il-Young Chong	ETRI, Korea
Dr. Armand Makowski	ISR, UMCP
Dr. Prakash Narayan	ISR, UMCP
Dr. Jangkyung Kim	ETRI, Korea
Mr. Thomas Christofili	ISR, UMCP
Mr. Giles Charleston	ISR, UMCP
Mr. Sandeep Rao	ISR, UMCP
Mr. Rajarsha Gupta	ISR, UMCP

1. Introduction

This document describes the OPNET simulation model of the ATM LAN Access Switch (ALAX) based on IEEE 1355. ALAX was designed at the Laboratory for Advanced Switching Technologies (LAST) by the collaborative research project team which consists of Protocol Engineering Center (PEC) of Electronics and Telecommunications Research Institute (ETRI), Institute for System Research (ISR) of the University of Maryland at College Park (UMCP) and Modacom Co., Ltd. during the first year of the research project called "A Standardization and Research Project on An ATM/B-ISDN Switching Fabric System". The OPNET simulation modeling job of the ALAX was performed during the first period of the second project year by the project team of the PEC and ISR.

In this project, ALAX system was designed but is not built yet and is too complicated for us to predict the detailed behavior of the system easily before we build it. Therefore, we decided to use the OPNET simulation in order to evaluate the ALAX system before we actually build the system. The performance of the ALAX can be predicted by analyzing the results of the OPNET simulation. With this analyzing results of the OPNET simulation, we are able to get the optimal architecture of the ALAX by devising the proper number of transputers and the proper packet buffer size of the ALAX system.

This report is organized as follows: In the next Chapter, the overview of the ALAX are described. We then follow up with a description of the OPNET simulator, OPNET Simulation modeling procedure, detailed description of the OPNET model program and conclusion.

2. Overview of the ATM LAN Access Switch (ALAX)

The main function of the ALAX is to provide the interface between legacy LAN and ATM world. The most important design characteristics of the ALAX is the adoption of the IEEE P1355 Standard for Heterogeneous Inter Connect(HIC)[1] as the data communication paths within the ALAX switching system. This makes it possible for this project team to design the PC-based parallel architectured high performance switching system in conformance with emerging open technologies and related standards. The system hardware architecture of the ALAX system is described in the documentation titled "ATM LAN Access Switch: System Architecture."[2] The protocol architecture of the ALAX is described in the another documentation titled "ATM LAN Access Switch: Protocol Architecture (LAN Emulation Version)."[3]

The ALAX system will provide easy expandability by adopting the modularity design concept in it. Many applications can be added to the system by inserting the interface modules into the slots of the ALAX. Therefore, the modular design is a very important feature of this system. The new interfaces can be inserted into this system without changing of entire architecture with the help of scalable, modular and parallel structure of IEEE P1355 serial bus. The future interfaces which will be implemented in this system include the interfaces to the MPEG2 over ATM, NISDN, Multiprotocol over ATM, ATM terminal interface, Satellite communication, PCS and Circuit Emulation etc..

2.1 Hardware Architecture of the ALAX

ALAX system will be built on the PCI Bus[4][5][6][7] based IBM PC with the IEEE P1355 serial bus data communication capability in it. The IEEE P1355 is a new serial bus standard that provides the data

interconnect between communication modules with scalability, flexibility and simplicity. By adopting the P1355 serial bus into this system, the bottleneck problem within the high-speed communication system can be solved. The first phase design target of the ALAX system is to build the ATM hub system that has the capability of LAN Emulation Over ATM for the Ethernet LAN. The detailed block diagram of the first version of the ALAX is shown in Figure 1. The system modules which are needed to build the first LAN Emulation version of the ALAX are as follows;

- ATM Interface Module
- Switch Matrix Module
- Ethernet Interface Module

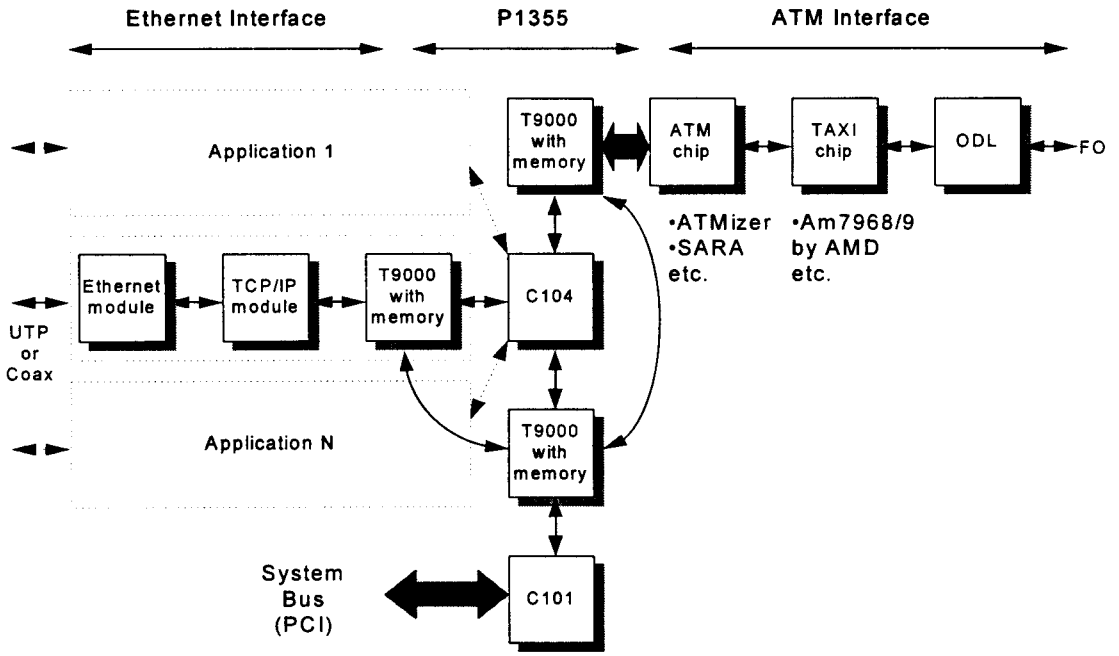

Figure 1. Block diagram of the first version of the ALAX

2.2 Network Architecture of the ALAX

The format of the packets in the LAN environment and ATM environment are different. Therefore, the main function of the ALAX is to convert the input packet to the proper output packet format. All the packets which are transferred through the ALAX should have the P1355 packet format. The packet from the LAN side has the IEEE 802.3 Ethernet MAC format. Therefore, this packet should be converted to the P1355 packet first then it should be converted to the ATM cell format before it is sent out to the ATM network. On the other hand, the ATM cells arrived at the ALAX should also be converted to the P1355 packets then to LAN packets. There are many things that should be considered during this procedure. They are addressing, routing, packet format etc.

The function of the ALAX will be the ATM hub which conveys the LAN packets to the ATM world as well as the ATM cells to the legacy LAN environment. ALAX will also connect two separate legacy LANs by conveying the LAN packets between those legacy LANs. The network architecture of the LAN emulation environment is shown in Figure 2. In this Figure, the packet conversion protocol to P1355 is omitted because P1355 has the meaning only with the internal packet transmission within the ALAX.

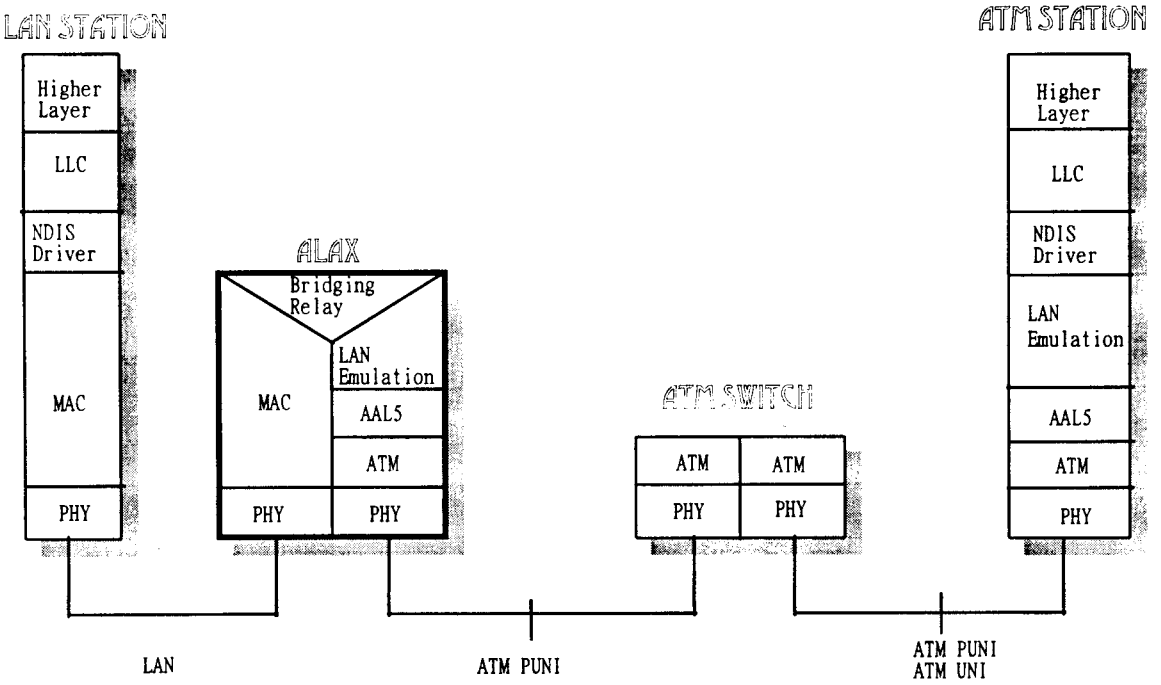

Figure 2. Network Architecture of the ALAX in LAN Emulation Environment

2.3 Protocols of the ALAX

The protocols needed within the LAN Emulation version of the ALAX include LAN Emulation, Bridging Relay function, P1355, MML, MAC, LAN Physical, AAL5, ATM and ATM Physical layer[8][9][10]. The conversion protocol between P1355 and MAC which is named MAC Mapping Layer (MML) protocol are also needed. Other protocols needed in the ALAX include the ATM signaling which is defined in ATM Forum UNI 3.0, Network Management(SNMP or CMIP) and some control functions of the ALAX. All the protocols described above should be divided and run among many adapter boards. Allocating the protocols into the proper adapter boards is one of the most important ALAX system's design issues. The main control functions and network management functions will be performed at the main CPU board of the IBM PC. The other protocols will be executed within each adapter board. Some protocols of the ALAX will be described in this section.

2.3.1 LAN Emulation

Some significant progress for the inter networking problem between legacy LAN and ATM was achieved in the ATM Forum. They are Multi Protocol Over ATM(MPOA) and LAN Emulation(LANE). The first phase of the ALAX design is focused on the LAN Emulation environment. The LAN to ATM interface protocol of the first phase ALAX will be the LAN Emulation which is now documented as LAN Emulation Over ATM Specification - Version 0 by the LAN Emulation SWG Drafting Group at the ATM Forum[11]. The legacy LAN which we are considering in the first phase is the IEEE 802.3 Ethernet LAN[12][13].

2.3.2. IEEE P1355 Protocol

IEEE P1355 is the standard for Heterogeneous InterConnect (HIC) which is the low cost, low latency scalable serial interconnect for parallel system construction. This protocol was also submitted to the ISO/IEC JTC1 standard committee and accepted as the standard. IEEE P1355 specifies the physical connectors and cables to be used, the electrical properties of the interconnect and a cleanly-separated set of logical protocols to perform the interconnection in the simplest possible way. The purpose of this standard is to enable high-performance, scalable, modular, parallel systems to be constructed with low cost, where 'cost' must include not only the price of components, but also the engineering effort required to use them successfully. In ALAX, the P1355 is used to interconnect many modules within the system.

2.3.3. MAC Mapping Layer (MML) Protocol

MML protocol is the interface protocol between LAN and P1355 which is being developed within this project team. The main functions of the MML are like followings;

- Local Addressing
- Conversion to the P1355 packet format

Local addressing function is used to get the local address from the MAC address. This local address is one to one matched to the port number of the C104 Asynchronous Packet Switch fabric chipset. The conversion to or from the P1355 packet format is performed at the DS Link module of the T9000 Transputer[14][15][16][17][18].

2.3.4 Bridging Relay Function

The bridging relay function inside the ALAX is very simple. This protocol just relay the MAC packet from the LAN side to the ATM side and vice versa. Actually the complete bridging function within the ALAX is performed by the combination of the LAN Emulation protocol and the Bridging Relay function.

3. OPNET Simulation

In this Chapter, we described the simulation technique in general. Then the purpose of the OPNET simulation is described and the OPNET graphic simulator is described in detail.

3.1 Introduction to Simulation

The three techniques for performance evaluation are analytical modeling, simulation and measurement. If something similar to the proposed system does not exists as the ALAX system of this project, analytical modeling and simulation are the only techniques from which to choose. Simulation is a useful technique for computer systems performance analysis. If the system to be characterized is not available, as is often the case during the design or procurement stage, a simulation model provides an easy way to predict the performance or compare several alternatives. However, simulation models often fail; that is, they produce no useful results or misleading results. This is either because the model developers are proficient in statistical background or because they are proficient in statistical techniques but are not aware of good software development techniques.

In this project, ALAX system was designed but is not built yet and is too complicated for us to predict the detailed behavior of the system easily before we build it. Therefore, we decided to use the OPNET simulation in order to evaluate the ALAX system.

3.2 Choosing the OPNET Simulation Package

Selecting a proper language is probably the most important step in the process of developing a simulation model. An incorrect decision during this step may lead to long development times, incomplete studies and failures. There are four choices: the first choice is a simulation language like GPSS, SIMAN, SIMSCRIPT, SIMULA etc. The second choice is a general purpose languages like FORTRAN, C, PASCAL etc. The third choice is the extension of a general-purpose languageGASP (for FORTRAN). The last choice is the simulation packages like QNET4, RESQ, OPNET etc. Each choice has its own advantages and disadvantages[19]. However, in this project, the simulation package, especially OPNET was chosen to simulate the performance of the ALAX. Simulation packages like OPNET allow the user to define a model using a dialogue. The packages have a library of data structures, routines and algorithms. Their biggest advantage is the time savings that they provide. The graphic simulation package OPNET has many other advantages as described in the next section.

3.3 OPNET Simulator

The OPNET simulator can be used to evaluate the performance of a network design. The design based decisions must be predicated on some performance criteria that can be gotten through the simulator. The number of interrupts is commensurate with the degree of investigation for this discrete event simulator. Therefore, decisions should be made to investigate the modules and parameters that most greatly affect these design based decisions.

The OPNET simulator operates at three hierarchical levels to describe andcontrol the network to be analyzed. These are the network level, the node level, and the process level. The network level consists of network nodes that can be connected by means of unidirectional or bidirectional communications links and

bus links. The bus links can provide service to an unlimited number of users. The distance between nodes can be specified, and this has ramifications for link delay at the network level.

The node level is the level where the simulation designer should begin. Here, the packets can be generated, terminated, stored in queues, and routed. Specifically, the queue and processor modules can be altered by means of a finite state machine at the process level. The finite state machine can be created in the processor modules to read the packet off the stream, read information from the packet, process the packet, and send the packet on an outgoing stream. The queues differ from this in so far as they are able to store and service the incoming packets based on various input-output schemes. There exists sources at the node level to inject new packets into the network. These sources consist of ideal and clocked generators. The packet definitions for these sources must be defined in the Parameters section. The ideal generator specifies typical interarrival types. Non-standard traffic models can be constructed by means of a processor module via self interrupts. These self interrupts and packet lengths are specified in the finite state machine to concur with the desired traffic model. In addition, the processor modules can act as sinks to destroy the packet as it leaves the network. This node level has various types of transmitters and receivers in order to interact with the higher network level.

The process level operates on the packets as they go through the processor and queue modules of the higher node level. The control of the finite state machine exists in one of the states(or circles) until an interrupt arrives. Upon receival of an interrupt, conditions on the links between states determine which state the control will revert to next. In each state, there exists pseudo-"C" (Proto-C) code to perform some of the operations stated earlier. For example, a typical scenario would be where one state could initialize parameters, one could read the packet off of the stream and store it, and another might send the packet onto an output stream after a delayed interrupt. Furthermore, the parameters and constants are initialized in areas called state variables, temporary variables, and the header block. Finally, subroutines, accessible by the finite state machine, can be written in the function block.

The analysis section of the simulator allows the programmer to view multiple traces that are distinguishable by color. Typical performance measures of interest in the node level queue could be queue occupancy, delay, throughput, and packet loss. These are specified in the Probe section of the simulator, and the results for these are written in vector files when the executable of the simulation is run. In addition, results can be written to scalar files. A typical application of this would be the need to find the total delay across the network. This, of course, cannot be done by "probing" one queue or processor. In actuality, the delay is written at the finite state machine level when the packet arrives at the external part of the network. Detailed trace information of each element written to the output files can be obtained in this analysis section. Also, there is a filter module that can perform functions on the output such as averaging and adding of traces.

4. OPNET Model of ALAX

In this Chapter, the motivation of the OPNET simulation is described first. Then OPNET simulation modeling procedure of the ALAX is described.

4.1 Motivation of the OPNET Simulation

The motivation for use of the OPNET simulation software is based on the need to evaluate the performance of the design of the suggested system. Design based decisions must be predicated on some

performance criteria that can be gotten via the software simulator. The decisions concerning speed and memory size of the T9000 will have ramifications to the consumer. It should be noted that in most simulations there is a diminishing returns on the increasing level of investigation. In fact, the number of interrupts is commensurate with the degree of investigation for this discrete event simulator. Therefore, decisions have been made to investigate those modules and parameters that have most significance for these design based decisions. Certain parameters in the T9000 module are of considerable importance for this. Specifically, buffer size and speed of the transputer will be of considerable importance given that this design will be implemented with commercially available hardware. It should be noted that the designs of the C104 at the network center and the bursty traffic sources at the network periphery will also have considerable importance in getting valid results.

4.2 Modeling of the ALAX

Figure 3 shows the whole block diagram of the first version of the ALAX system. As shown in this Figure, we first simulate the 4 by 4 C104 switch. Therefore, the first OPNET simulation version has one ATM interface module and three Ethernet interface modules. The ATM interface module is simplified to have three submodules which are TAXI, ATMizer and T9000 Transputer. Some other modules like ODL module is omitted because of its small delay compared to the other modules. The Ethernet interface module is simplified to have two submodules like T9000 Transputer and MAC. Among these, TAXI, ATMizer and MAC modules are simulated as the one queue model. The T9000 and C104 modules are divided further to get the correct simulation results because these two modules has the most complicated features and affects significantly the behavior of the whole ALAX system. The detailed models of these T9000 and C104 modules are described in detail in Chapter 5.

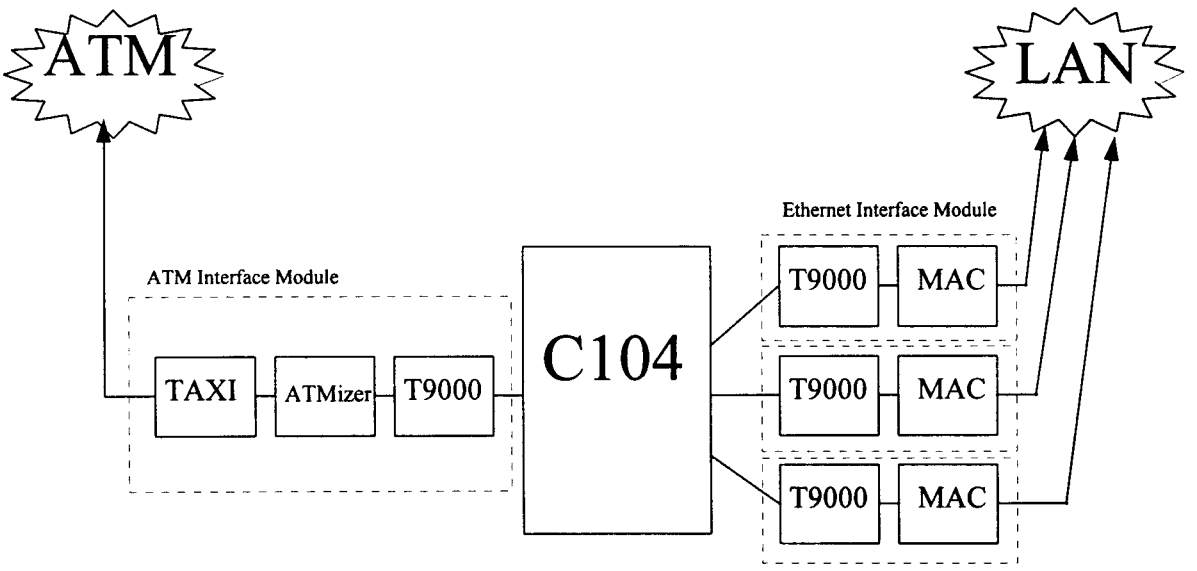

Figure 3: Block Diagram of the First Version of the Whole OPNET Simulation Model

4.3 ATM Interface Module Model

In this section the ATM interface module modeling procedure is described. Figure 4 shows the block diagram of the actual hardware of the ATM interface module. When we build the OPNET simulation model, we need to keep it simple to get the compact simulation model. In order to achieve this, we omit the unnecessary parts from the ATM interface hardware module when we build the OPNET simulation model. The most important and significant parts of the ATM interface module is the T9000 Transputer because it spends most of the packet processing time within the ATM module. Shared memory is also important module because we need to get the proper packet buffer size to get the optimal architecture of the ALAX system. The shared memory can be included within the T9000 module. We decide to include the ATMizer and the TAXI module by making them a simple queue model. ODL module is omitted because of its small delay time. The private memory and PCI chipset are also omitted because these chipsets don't have direct relationship with the packet transmission which is the most important factor we want to analyze from OPNET simulation. Therefore, we decided to include only three modules inside the ATM interface module. The result of this simplification procedure for the ATM interface module is shown in Figure 5.

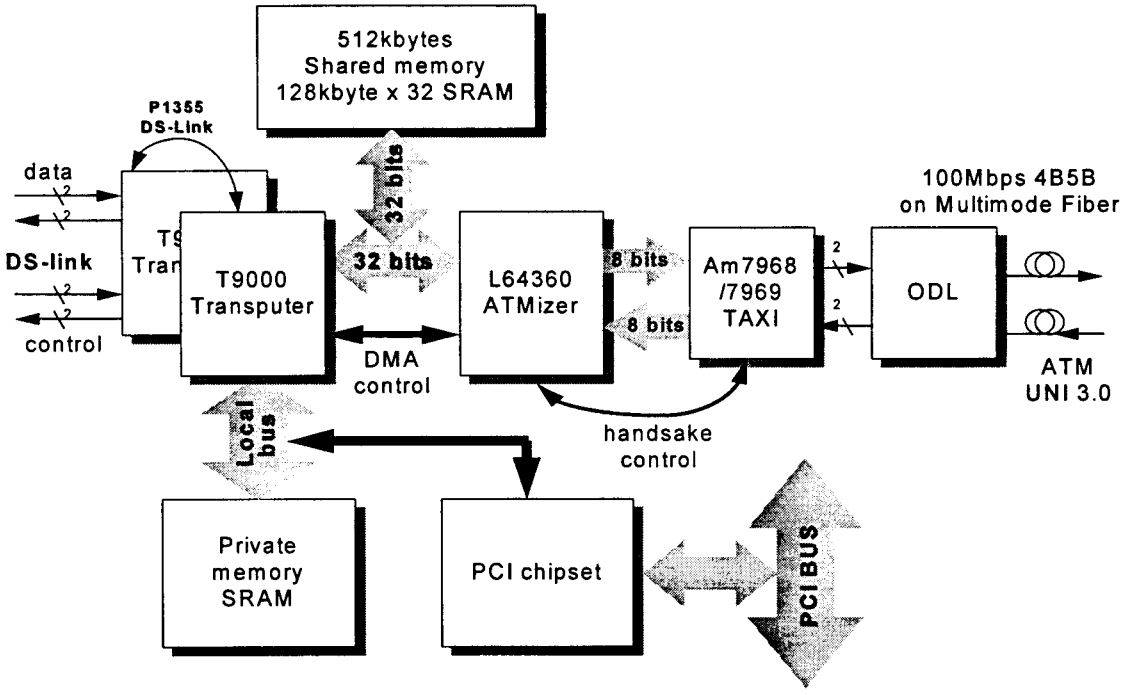

Figure 4: Block Diagram of the ATM Interface Module

4.4 Ethernet Interface Module Model

In this section the Ethernet Interface module modeling procedure is described. We also omit the unnecessary parts from the Ethernet interface hardware module when we build the OPNET simulation model as the same way we did for the ATM interface module. The most important and significant parts of the

Ethernet interface module is again the T9000 Transputer because it spends most of the packet processing time within the Ethernet module. We decide to include the MAC module by making them a simple queue model. The other modules are omitted because of their small delay time. Again the private memory and PCI chipset are also omitted from the OPNET model of the Ethernet module. Therefore, we decided to include only two modules inside the Ethernet interface module. The result of this simplification procedure for the Ethernet interface module is shown in Figure 5.

4.5 T9000 Model

As we described above, T9000 is the most important limiting factor in the behavior of the ALAX system. Therefore, we decided to model the T9000 in more detailed manner. As shown in Figure 5, the T9000 of the ATM interface module and Ethernet interface module are more divided into many subqueues. We assumed the shared memory is included within the T9000 module. The shared memory of the ATM interface module is divided into many subqueues which are ATM input buffer, P1355 output buffer, ATM output buffer and many P1355 input buffers as shown in Figure 5. The shared memory of the Ethernet interface module is similarily divided into many subqueues like P1355 input buffer, many Ethernet output buffers, Ethernet input buffer and P1355 output buffer. In this Figure, all the processing performed by the T9000 and all the packet formats of the ALAX system are also shown.

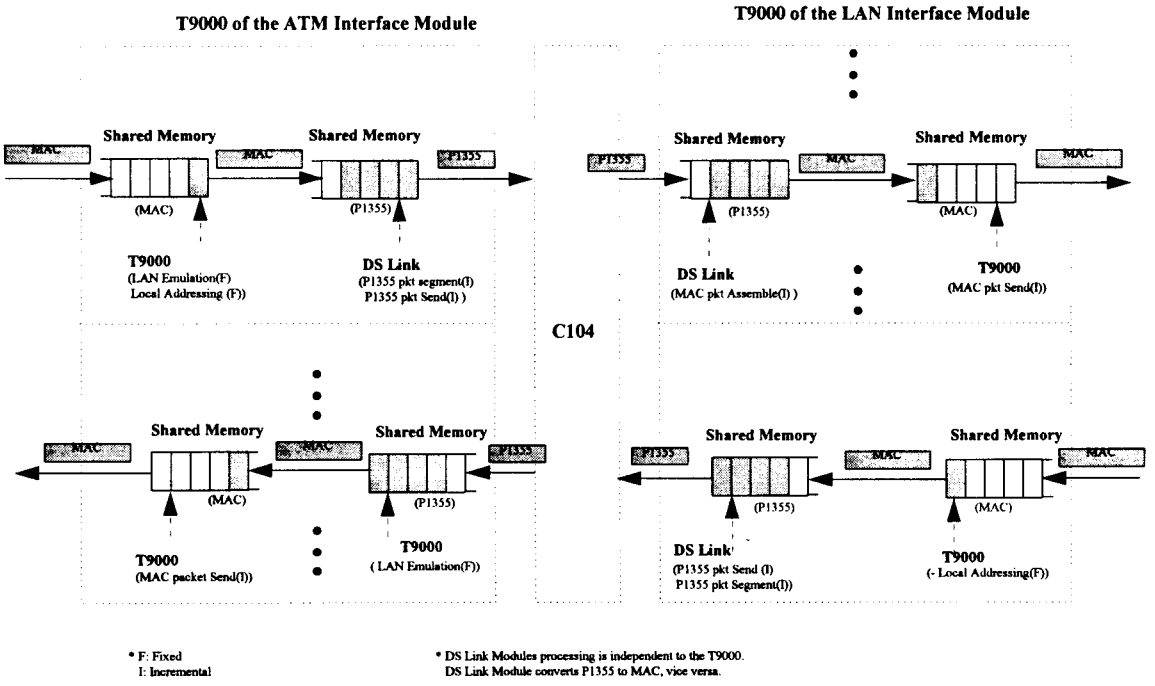

Figure 5: OPNET Model of the T9000 and C014 of the ALAX system

4.6 Final ALAX Model

The final OPNET model of the T9000 and C104 parts of the first version of the ALAX is shown in Figure 6. As you can see here, the final T9000 and C104 OPNET model consists of three Ethernet interface modules, one ATM interface module and one C104 module.

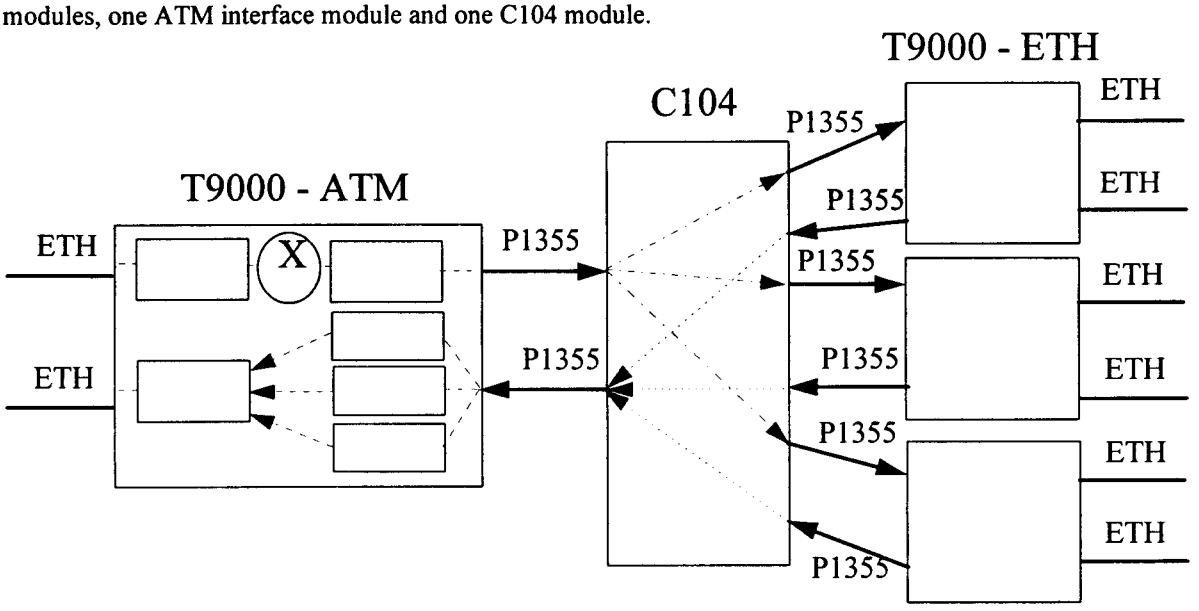

Figure 6: Final OPNET Simulation Model of the T9000 and C104 parts of the ALAX system.

5. OPNET Model Program Description

5.1 Modeling of the T9000

5.1.1 T9000 Hardware Design Considerations

The T9000 could act as a limiting factor on network throughput and packet loss. It is necessary to consider two things to prevent this. Namely, T9000 speed to perform such tasks as LAN emulation must not limit the input rate to the C104, and the T9000 memory size must be large enough to prevent packet loss. There will be a tradeoff between these two factors. For example, less memory will be required as the processor speed of the T9000 increases. Also, there could be diminishing returns from over-emphasizing only one of these requirements.

One technique has been suggested to alleviate the speed-memory requirements. Specifically, it was suggested that the bottom half of the T9000, the receiving end for 1355 packets from the C104, queue 1355 packets corresponding to their origination. Ordinarily, with multiple originations going to one destination, there would inevitably be undue packet delay in the situation where the head of queue(HOQ) P1355's Ethernet packet has not completely arrived and a non-HOQ 1355's Ethernet packet has arrived and received LAN emulation. In essence, splitting this one incoming queue into multiple queues that correspond to each origin is equivalent to using preemptive queueing versus non-preemptive queueing; whereby, the group of 1355s that have completed an Ethernet packet and been serviced are given highest priority. Ultimately, this technique will alleviate the processor speed requirements for the T9000. It is noted that the packet connection between the T9000 and the C104 is mediated by backpressure. This will prevent packet loss in the circumstances when the C104 is operating at capacity.

Various types of traffic will be sent through this proposed network. The ATM and Ethernet traffic types are accomodated by sending them through the C104 switch as 1355 packets. There is no theoretical limit to this 1355 packet length. However, the existing hardware limits the length to approximately 32 bytes. The present design produces the 1355 packets upstream at the T9000 transputer. As stated, the function of the T9000 will be to perform LAN emulation on the various incoming traffic types and provide a buffer to detain the 1355 when there is backpressure from the C104. Secondly, the T9000 will be expected to receive 1355s from the C104 and create the Ethernet packets that are to be sent to the different network types at the network periphery. These details will be described later.

5.1.2 Large Level Operation of T9000

The T9000 consists of a top and bottom half. The top half services the traffic toward the C104(upstream), and the bottom half services the traffic away from the C104(downstream). There is a central roving server that performs the service to be given out between the two parts. In addition, the incoming traffic on the top half consists of Ethernet size and the outgoing traffic consists of 1355 packet size (32 bytes); whereas, the incoming traffic on the bottom half consists of 1355 packet size and the outgoing traffic of the Ethernet size.

There are multiple layers of operation for the roving server. Namely, the roving server first checks either the top or bottom halves for service. For example, the roving server checks "subqueue 0" in Figure 7. If an Ethernet packet exists here without service, then it gets service. If this is not the case, then the roving server checks the bottom half for service. If service is not required on the bottom half then it checks the top half after a delayed self interrupt. However, to check if the bottom half requires service, all of the subqueues - subqueues 2, 3, 4, 5 in the diagram - have to be checked. This is performed in a sequential manner. If one subqueue is encountered whose first queued 1355 packet does not have service, then it is given service and the roving server reverts to the top half of the T9000. Otherwise, the roving server must check all of the bottom half subqueues for service. If it arrives at "subqueue 5", then the initial "subqueue 2" will be checked next.

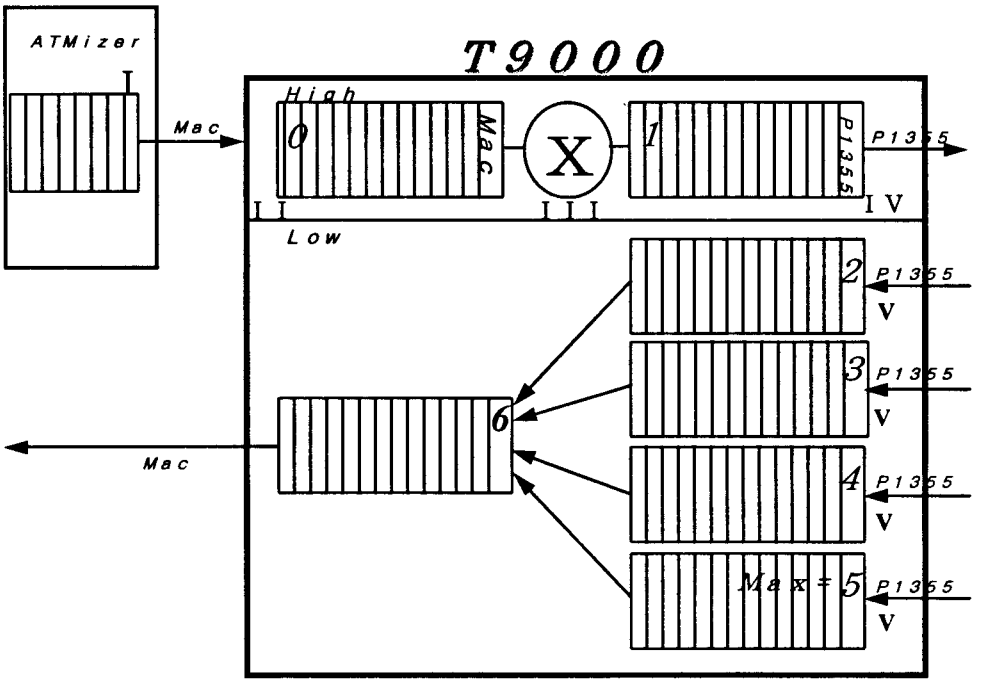

Figure 7: T9000 Model

Next, after "subqueue 0" is served, the corresponding number of 1355s contained in the Ethernet packet are sent downstream to "subqueue 1". These packets wait here if there is backpressure from the C104. Otherwise, they will be sent directly to the C104. Lastly, on the bottom half, once one subqueue has service and the correct number of 1355 packets to constitute the Ethernet packet are in the subqueue, then an Ethernet size packet is sent to "subqueue 6". Here, a delay can occur if the T9000 receives backpressure.

5.1.3 Process Model Description of T9000

The finite state machine for the T9000 is displayed in Figure 8. The sections of the finite state machine are described in Figure 8. The code for each of the finite state machine sections is included at the end of the report.

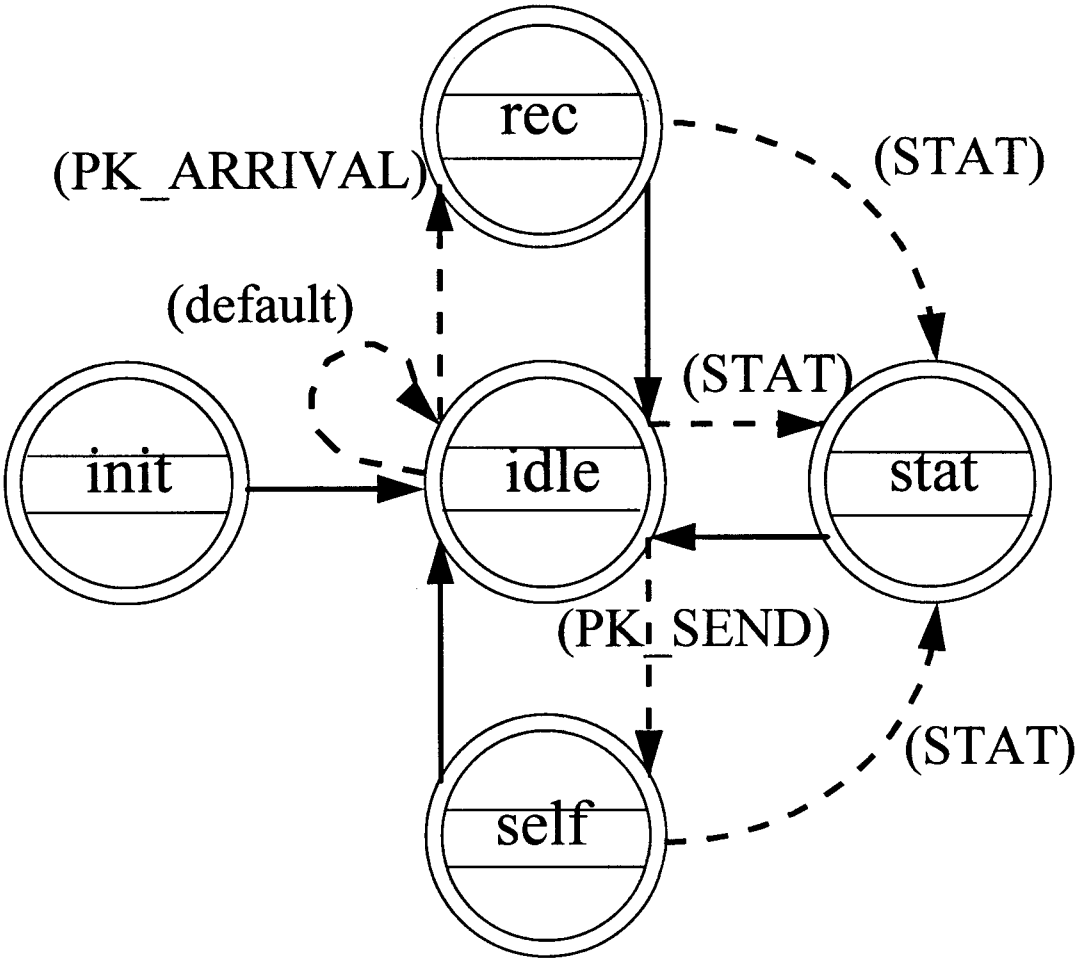

Figure 8: Process Model of the T9000 Model

INIT (initialize): This finite state machine module initializes parameters, reads promoted values, and sets constants based on these retrieved promoted values.

REC (receive): This finite state machine module takes the packet off of the stream and places it in a corresponding subqueue. On the bottom part of the T9000, if a P1355 arrival is sufficient to form an Ethernet packet and service has been given, then an Ethernet can be sent and the P1355s destroyed. On the top part of the T9000, a flag is set to indicate if there are packets in the "subqueue 0". This will assist in determining the exact length of the service delay. For example, if the packet has been in the queue longer than the time that it takes for the first 16 bytes to arrive (the bytes necessary to start LAN emulation), then this delay does not have to be incorporated in the service delay of the packet.

IDLE: This module checks the conditions to send a P1355 from the T9000 to the C104. These conditions are that P1355_SEND_TIME (described in the next section) has transpired, the sbackpressure from the C104 is low, the queue occupancy is non-zero, and the tail of the Ethernet packet that corresponds to the P1355 has arrived. Also, this modules checks for non-zero queue-occupancy in "subqueue 6" to send Ethernet packets

on the upstream away from the C104 from the low part of the T9000.

STAT (stat wire): This module changes the status of the stat_flag in correspondence with the backpressure from the C104. If backpressure is given by the C104, then this parameter is set high (stat_flag = 1). Otherwise, it is set low (stat_flag = 0).

SELF (self interrupt or SEND): This module accepts self interrupts and mediates roving server control which is initiated by a self interrupt at simulation commencement. Specifically, the constants CODE0, CODE1, CODE2, and CODE3 have been defined to demarcate the various possible interrupts. The interrupt that corresponds to CODE0 indicates that service attempted on the top half or the bottom half of the T9000. If, indeed, one half requires service, then an interrupt that corresponds to CODE1 or CODE2 is generated for a delayed time into the future. When a CODE1 type interrupt is received, the Ethernet packet is removed from "subqueue 0" and a corresponding number of P1355 packets is placed in "subqueue 1". At the end of the segment for CODE1, the conch (described in the next section) is switched to check the low part of the T9000 on the next service attempt. When a CODE2 type interrupt is received, the subqueues on the low part of the T9000 are checked for possible required service as described in the above section on the T9000. If, indeed, one of the subqueues is given service and it has all of the P1355 packets from the Ethernet packet that generated them, then the P1355 packets are taken out of this subqueue and sent to "subqueue 6" as an Ethernet packet. Finally, before the CODE2 section is left, the conch constant is switched to check the high part of the T9000 on the next service attempt. The specific action of the roving server is described in the previous section that describes the T9000. Lastly, the CODE3 type interrupt signals that a P1355 packet to be sent to the C104 had not waited long enough in "subqueue 1" and a self interrupt had been initiated for this purpose. Of greater importance, the finite state machine control has to leave and return to the IDLE state in order to recheck the condition to send a P1355 packet. The backpressure is checked in CODE3, and the packet is sent to the C104 when control returns to the IDLE state under the condition that backpressure was low when it was checked at the CODE3. Although, IDLE will be entered quite often, it is more precise to send a specific interrupt to recheck the conditions after an exact desired delay.

5.1.4 Parameter Level Description of the T9000

The code for the finite state machine for the T9000 is contained at the end of the paper. Here, significant parameters will be described with respect to their functions in the finite state machine.

flag_HOQ_down, where HOQ means "head of queue", is an array that indicates whether service has been given to the ith subqueue on the low part of the T9000. The values it takes lay in {0,1}.

conch is a parameter that indicates that the roving server is stationed at the high part or low part of the T9000. The values it takes lay in {0,1}.

queries is a parameter that keeps track of the number of times the roving server has checked the top part and low part of the T9000 unsuccessfully. If (queries=1) and the roving server is again unsuccessful at subsequent half of the T9000, then a self interrupt is initiated for the roving server to check the next half of the T9000 after a delay. The motivation for a large delay would be to minimize interrupts. However, too large a delay might force a packet to wait unduly. The compromise is a heuristic choice. Definitely, it won't be zero. Otherwise, there will be self interrupts ad infinitum. Under the conditions of traffic arrival close to traffic service, it would be rare that the top part and bottom part both do not require service. The values it takes lay in {0,1}

last_arrival_time is a parameter that reads the "tail arrival time" from the packet after it is served in the head

of the queue of "subqueue 0" before it is sent to "subqueue 1". Then, it is used when the next Ethernet packet is to be served. One of the conditions that is checked before the Ethernet packet is served and sent to the "subqueue 1" is that the tail of the last packet has arrived. This would not be required except for the suggestion that we chart the tail arrival.

stat_flag registers backpressure from the C104. The values it takes lay in {0,1}.

self_flag prevents generation of multiple, unnecessary interrupts in the IDLE state. If the the IDLE state is entered, a P1355 is in the "subqueue 1", and the tail of the corresponding Ethernet packet has not arrived, then a delayed interrupt is generated to check the availability of sending it in the future. Of course, it is undesirable to generate multiple similar interrupts between this time and the time that the P1355 should be checked again. The values it takes lay in {0,1}. This would not be required except for the suggestion that we chart the tail arrival.

MAX_EXT_MODULES represents the maximum external modules. For example, if there were one ATM modules to the C104 and three Ethernet modules to the C104, then this would take the value four.

T_DEL represents the delay that evolves because of the need to wait for the first 16 bytes of the Ethernet packet before LAN emulation commences.

mult is currently a test factor included by the group member who has created the long range dependent traffic source.

bit_interarrival_atm is the inverse of the maximum ATM speed.

T_LANE is the time duration to give service to the high part of the T9000

SERV_TIME_DOWN is the time duration to give service to the low part of the T9000

UPDATE is the delay for the self interrupt under the conditions described in detail under self_flag. Namely, if the tail of the Ethernet hasn't arrived, then don't send the P1355 packet and set a self interrupt for UPDATE seconds into the future. Of course, this would not be necessary except for the suggestion that the tail of the Ethernet packet be charted.

SERV_DIFF represents the delay for the self interrupt under the condition that neither the top part or bottom part of the T9000 require service. This condition is described in more detail in the previous section of the paper.

P1355_SEND_TIME represents the required delay interval in between P1355 packets that are sent to the C104

t_1 records the simulation time of the last P1355 sent to the C104; whence, another P1355 is not sent until (op_sim_time() - t_1) > P1355_SEND_TIME is true.

ete_gsh* are basic global statistic handles that enable results to be written to the output scalar file

5.2 Modeling of the C104

5.2.1 OPNET Model Development

The simple model of the C104 was developed from the actual C104 hardware schematic diagram. The model of the C104 is shown in Figure 9. As shown in this Figure, we first developed 4 by 4 switch model. This consists of 4 input queues and 4 output queues.

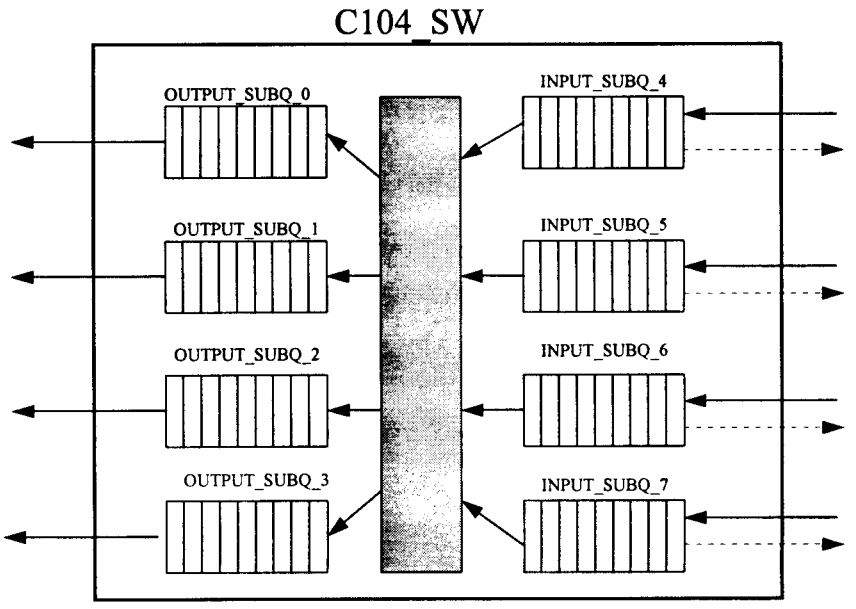

Figure 9: OPNET Model of the C104

5.2.2 Model Coding

The C104 model described above was coded using C programming language inside the OPNET program. The process model of the C104 is shown in Figure 10. The receiving part of the C104 was coded following the flow chart shown in Figure 11. When the packet arrives through the input stream, that is when the input stream interrupt occurs, the receiving procedure starts. Then we first read the header part of the input packet to decide the destination output subqueue. Then we check if the corresponding destination output subqueue is empty. If the destination output subqueue is empty, we put the incoming packet into the subqueue and set the timer to simulate the transfer time of the packet through the C104 switch and the packet service time at the output subqueue. After the timer expires, that is the transfer time elapses, the timer interrupts and invokes the transmit procedure. If the destination output subqueue is not empty, we put the packet into the input subqueue to wait the output subqueue is available.

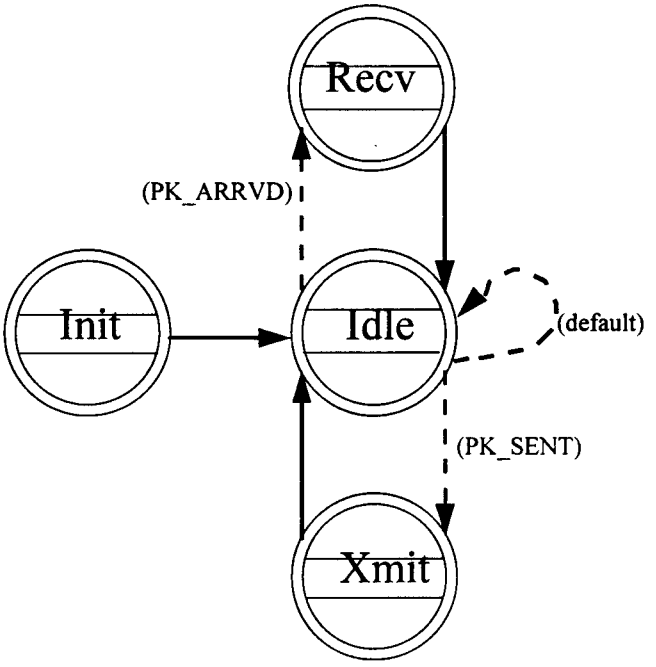

Figure 10: Process Model of the C104 Module

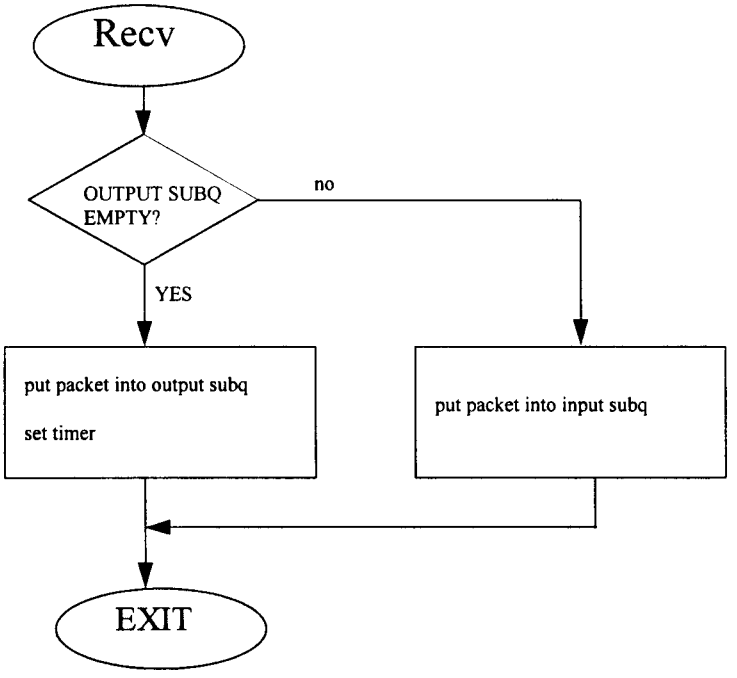

Figure 11: Flow Chart of the Receiving Part of the C104

The transmitting part of the C104 was coded following the flow chart shown in Figure 12. This procedure is invoked when the timer expires. We first remove the packet from the corresponding output subqueue and send the packet to the next stage. Then we check if there is any input subqueue which has the packet inside it. If there is no packet inside any of the input subqueue, we exit from this transmitting procedure. The algorithm we use to check the input subqueue is round robin manner. If any one of the input subqueue has the packet waiting, we check if the corresponding output subqueue is empty. If the output subqueue is not empty, we just exit. If it is empty, we move the packet to the corresponding output subqueue, set the timer and exit. The reason we set the timer here is to simulate the transfer time of the packet through the C104 switch and the packet service time at the output subqueue.

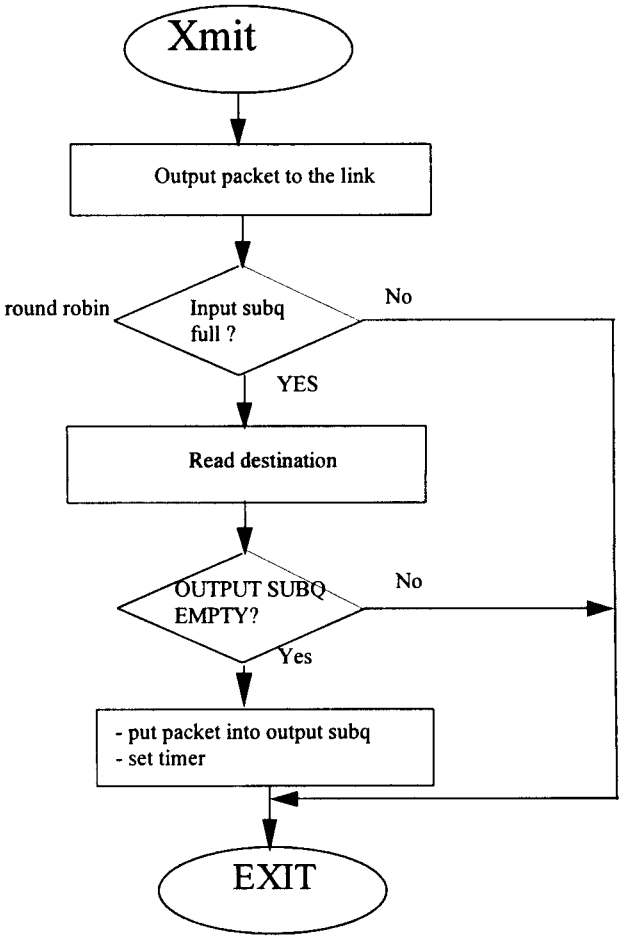

Figure 12: Flow Chart of the Transmitting Part of the C104

5.2.3 Description of the **define**

In this section, all the definitions used in the C104 OPNET model are described.

MAX_NO_LINK defines the number of the DS-Links of the C104 fabric. When we want to increase the number of the DS-Links of the C104 OPNET model later, we can get the new C104 model of any number of DS-Links by just changing this definition to the proper number.

DELAY defines the time for the P1355 packets to wait at the output subqueue before they leave the C104 fabric to the next stage. In this simulation model, we define this number as 1 usec.

PK_ARRVD tells the interrupt type just happened is OPC_INTRPT_STRM type. This interrupt type is true when the packet arrives at the C104 fabric.

PK_SENT tells the interrupt type just happened is OPC_INTRPT_SELF type. This interrupt type is true when the interrupt which was invoked by the **op_intrpt_schedule_self()** occurs at the C104 fabric. This notify the P1355 packet is ready to move from the input subqueue to the output subqueue.

5.2.4 Description of the **State Variable**

In this section, all the State Variables used in the C104 OPNET model are described. State Variable is the Static Global Variable which can be used by any program modules. State Variable does not change its value when the program execution exits the current module.

curr_q[MAX_NO_LINK]: This integer array keeps the input subqueue index which is going to be examined next time to see if there is any packet within the input subqueue. This index is needed to scan the input subqueue in the round robin manner. Each element of the array keeps the input subqueue index of the corresponding output subqueue.

rcvd_pkts keeps the number of packets which are received at all of the input subqueue of the C104. This is used to analyze the packet behavior inside the system.

xmitd_pkts keeps the number of packets which leave from all of the output subqueue of the C104.

5.2.5 Description of the **Temporary Variable**

In this section, all the Temporary Variables used in the C104 OPNET model are described. Temporary Variable is the Dynamic Variable which can only be used by current program module and does not leep its value when the program execution exits the current module.

tmp_pkptr is the temporary buffer which is used to keep the packet pointer temporarily.

pkptr is the buffer which keeps the packet pointer of the received packet at the C104.

out_pkptr is the buffer which keeps the packet pointer of the packet which is to be transmitted from the C104.

count is used to limit the number of scanning repetition when we scan the input subqueue to check if there is

any packet which waits to be serviced. This is increased from 0 to MAX_NO_LINK if there is no packet waiting to be serviced.

out_ds_link keeps the output DS-Link index which will be used this time. It is loaded with the interrupt code number at the beginning of the **Xmit** module.

out_subq keeps the output subqueue index which will be emptied this time. It is loaded by the interrupt code number at the beginning of the **Xmit** module.

dest_address is filled with the destination address of the packet which will be moved from the input subqueue to the output subqueue.

curr_input_lnk keeps the current input stream index where the packet is arriving from. It is loaded with the stream interrupt number at the beginning of the **Recv** module.

pk_transfer_time is loaded with the time which is calculated by dividing the packet length with the transfer speed (100 Mbps). This number indicates the delay which the packet needs in order to go through the C104. Therefore, the **Xmit** routine will be invoked at **pk_transfer_time** after **Recv** routine or **Xmit** routine finish.

pk_len is loaded with the packet size which is got through **op_pk_total_size_get**.

6. Conclusion.

The issues related with the OPNET simulation modeling of the ATM LAN Access Switch (ALAX) based on IEEE 1355 was described. The OPNET simulation modeling job of the ALAX was performed during the second project year of the research project called "A Standardization and Research Project on An ATM/B-ISDN Switching Fabric System" by the project team of the PEC and ISR.

We performed the OPNET simulation to predict the behavior and the performance of the ALAX system before we build the actual system. With the results of the OPNET simulation, we are able to get the optimal architecture of the ALAX by devising the proper number of transputers and the proper packet buffer size of the ALAX system. This results will be compared with the analytical performance evaluation results of the ALAX system which is concurrently performed within this project to prove the validity of the simulation results.

REFERENCES

[1] IEEE, "IEEE Draft Std P1355, Standard for Heterogeneous InterConnect (HIC) (Low Cost Low Latency Scalable Serial Interconnect for Parallel System Construction)", January 1995.

[2] Man Geun Ryu, Jangkyung Kim, "ATM LAN Access Switch (ALAX): System Architecture", Laboratory for Advanced Switching Technologies, College Park, MD., May 1995.

[3] Jangkyung Kim, "ATM LAN Access Switch (ALAX): Protocol Architecture (LAN Emulation Version)", Laboratory for Advanced Switching Technologies, College Park, MD., June 1995.

[4] PCI Special Interest Group, "PCI Local Bus Specification, Revision 2.0", April, 1993.

[5] PCI Special Interest Group, "PCI Multimedia Design Guide, Revision 1.0", March, 1994.

[6] PCI Special Interest Group, "PCI BIOS Specification, Revision 2.1", August, 1994.

[7] PCI Special Interest Group, "PCI to PCI Bridge Architecture Specification, Revision 1.0", April, 1994.

[8] de Prycker, "Asynchronous Transfer Mode, Second Edition", Ellis Horwood, 1993.

[9] Byeong Gi Lee, Minho Kang & Jonghee Lee, "Broadband Telecommunications Technology", Artech House, 1993.

[10] David E. McDysan, Darren L. Spohn, "ATM theory and application", McGraw-Hill, Inc., 1995.

[11] LAN Emulation SWG Drafting Group, "ATM Forum/94-0035R7, LAN Emulation Over ATM Specification - Version 0", ATM Forum, December 1994.

[12] William Stallings, "Handbook of Computer Communications Standards - Local Network Standards", Macmillan Publishing Company, 1987.

[13] IEEE, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", American National Standard ANSI/IEEE Std. 802.3, 1985.

[14] Ian Grabam & Tim King, "The Transputer Handbook", Prentice Hall, 1990.

[15] Ronald S. Cok, "Parallel Programs for The Transputer", Prentice Hall, 1991.

[16] Jeremy Hinton & Alan Pinder, "Transputer Hardware and System Design", Prentice Hall, 1993.

[17] SGS-Thomson, "The T9000 Transputer Hardware Reference Manual", 1993.

[18] SGS-Thomson,"T9000 Development Tools, Preliminary Data sheets", 1993.

[19] Raj Jain, "The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling", John Wiley & Sons, Inc., 1991.