Simulation of an Optical Network System for a Space Based High Performance Computer System

By:

Eric J. Mitchell

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science at the Massachusetts Institute of Technology

May 22, 2002

Author_____
Department of Electrical Engineering and Computer Science
May 22, 2002

Certified By_____
David C. Ngo
BAE SYSTEMS Thesis Supervisor

Certified By_____
Vincent Chan
M.I.T. Thesis Supervisor

Accepted By_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Simulation of an Optical Network System for a Space Based High Performance Computer System

By:
Eric J. Mitchell

Submitted to the
Department of Electrical Engineering and Computer Science

April 14, 2002

In Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Electrical Engineering and Computer Science.

ABSTRACT

An important future addition for a Space Based High Performance Computer System is a high-speed optical network for faster data transmission. The purpose of this project is to research and simulate next generation computing applications on high-speed optical networks. The research has been performed in the System Application context involving embedded high-performance computing applications and optical networking technology to guide future research and development of advanced optical devices. The research addresses advanced processing system issues in bandwidth, latency, protocol, topology, and fault tolerance in relation to high performance systems. The reference distributed computer, provided by BAE SYSTEMS in Nashua New Hampshire, consists of multiple processing nodes connected by a Myrinet copper network. The advanced embedded computing applications include Space-Based Radar Corner Turn processing, Synthetic Aperture Radar Back End processing, and Random Workload software models. Two Optical networks have been developed as part of this research to replace the reference Myrinet network, a Ring based network and a Star based network. Both networks employ redundancy to provide an alternate direct optical path between each pair of nodes. Of these networks the Ring design failed due to packet collisions and due to the need for a complex networking protocol. The Star Optical network design performed well in comparison to the reference network design. Overall, network latency was reduced and the internode data distribution speed was dramatically increased. Also, the memory usage for each of the three software models was analyzed and each has definite bound that will help future development. Although the results of this research are favorable, the eventual future design and implementation of a Space Based High Performance Computer System would benefit from additional research on a number of topics.

Table of Contents

Chapters:

List of Figures

List of Tables

Acknowledgments

There were many people and companies that helped this research. We would like to thank them for their contributions:

I performed this research at the Canal street facility of BAE SYSTEMS, in Nashua New Hampshire. BAE SYSTEMS supplied the computers, software, office space, and reference designs of the architecture discussed in this project. This research was performed with help from BAE SYSTEMS Space Electronics and Integrated Optical Electronics groups. Most importantly I would like to thank my project supervisor David C. Ngo for his help and patience. I would also like to thank Mike Harris, Mark Law, Todd Birkebak, Milton Young, and Jason Boucher.

We would also like to thank Foresight Systems Inc. and Beat Zenerino for graciously giving me a temporary student license to run their Foresight co-design software on my laptop in Cambridge. This was key in debugging the network models after my time at BAE SYSTEMS.

Last but not least, I would like to thank my thesis advisor Prof. Vincent Chan for putting up with me and pushing me to finish this project.

Chapter 1

1 - Introduction

1.1 - Purpose:

Networks are an important part of everyday life. Every time you make a phone call, send an email, or turn on a light, you are using a network. One of the most common networks used are those computers communicate through. These networks transfer data across the long distances between computers. Most computer networks today are copper wire based systems. However, we have begun to reach the limit of how much data a copper wire can transmit over long distances. Networking companies have addressed this issue with optical technologies. Optical technology use in LAN and WAN applications has demonstrated significant increases in data bandwidth capacity. However, use of optical technology for embedded system network application has yet to mature due to greater challenges in size, weight and power, and the electrical interfacing cost (e.g. latency) associated with bridging between the optical and CMOS domain. The objective of this work is to explore new optical LAN architectures for improved embedded network performance.

1.2 - Background

1.2.1 – Reference Design for Space Applications:

A lagging technological aspect of space systems is the computer systems onboard space satellites. This lag is due to severe requirements beyond those placed on ground based, commercial systems; space-based processors must be radiation tolerant, low power, light weight, and tough enough to handle the physical stresses of launch (with high reliability). Because of such factors space based processors are typically several generations behind equivalent terrestrial systems. Several space system applications are

currently being envisioned which require significant advances in the on-board processing available to satellite designers.

Our research is in the context of a high-performance space computer, developed by BAE SYSTEMS, which serves as a reference design for this thesis[1]. Part of this research includes the modeling of a distributed computer composed of multiple processing and input nodes connected by a Myrinet switch network. Myrinet is a byte-wide path-based interconnect protocol and technology.[2] Each node on the Myrinet network is composed of a Node Controller (NC) and a resource element such as: processing elements, input/output elements, or external hardware control elements. The NC is the heart of a Node, and handles all internal message and data passing as well as any external networking needed by the elements attached to it. Each node is connected to the Myrinet switch network through redundant ports for fault tolerance. The current instantiation of this network is limited by its specification to 160 Mega-bytes per second, or equivalently 1.28 Giga-bits per second (Gbps). (See Figure 1.1) The next generation node in support of elements of planned Space Systems is expected to need at least 5x increased network bandwidth to maintain the internal node transfer bandwidth based on newer PCI buses 64 bits across and running at 66 MHz. This will translate to a maximum data throughput of 4.2Gbps. Since the latest version of Myrinet will be able to handle 2Gbps of bandwidth, there will still be a large difference in ability versus need. One possible solution for the network bandwidth problem is an optical network.

[1] See: A Reliable Infrastructure Based on COTS Technology For Affordable Space Applications
[2] See: Myrinet Network Specification Draft Standard

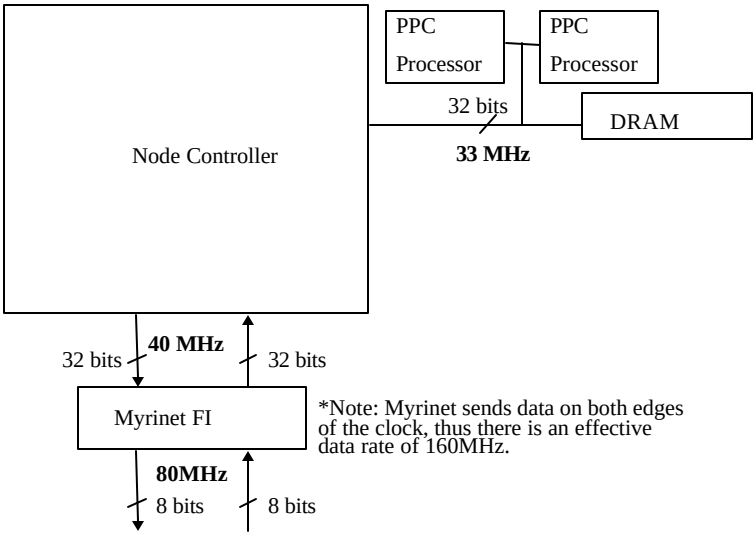


Figure 1-1: *Basic black box outline of a processing node. Notice the Myrinet Connection and current data rates at the bottom.*

1.2.2 – Integrated Optical-Electronics Technology Reference:

The core technology of our optical architecture is based on Vertical Cavity Surface Emitting Laser (VCSEL) transmitters and diode detectors devices. A test chip has been developed containing these devices numbering 1024 VCSEL's and 1024 detectors named Demo-2 [3]. This device contains four test areas with approximately 128 pairs of devices per area. An alternate device to the Demo-2 is the current test model of TerraConnect Inc. This device contains 48 transmitters or 48 detectors per optical transmitter or receiver device. A downside of the current TerraConnect test model is that the transmitters and receivers are not integrated onto a single chip. A benefit of the Demo-2 device is that it has more elements per area and can test different data communication paths. Both the TerraConnect device and Demo-2 are technology demonstration models only and have been implemented in few real world applications. For the purpose of this research, however, we use the specifications as if they were commercial devices.

LAN Optical Networking, especially for embedded system applications remains an area of research. It is not known what sort of requirements need to be integrated into

[3] BAE SYSTEMS Demo -2 White Paper

today's development products to produce a LAN Optical Network. This work develops a system application concept for product research and development on a commercial scale. Results are used to recommend future requirements for devices like the demo-2 device.

Figure 1-2: *The BAE SYSTEMS Integrated Optical Electronics Department Demo 2 transceiver layout (left) and TeraConnect's TC-48 Transmitter/Receiver (right).*

1.2.3 – Foresight Modeling Tool:

The modeling tool used in this project was based on the Foresight v5.1.3 modeling tool from Foresight Systems Inc.[4] This tool allows a user to model both hardware and software design with a single tool. For this reason it was chosen as the tool in which to develop the reference design. Using such a system node, a system developer can then run the software model on the hardware model, simulating the interaction between the software and hardware. The interface for using Foresight is the Software Simulator Testbed developed by BAE SYSTEMS.

1.3 – Project Design

The goal of this project is achieved through the development of new optical LAN models to replace the Myrinet network currently used in the distributed computer described above. To achieve this goal we analyzed the current hardware and software models of the reference system as well as the current optical technology available. From

[4] Release Notes: Foresight v5.1.3

this research we decided that the network needed to be entirely optical due to the latency and size issues resulting from an optical-electrical-optical conversion, and it is hypothesized that either a star or ring network configuration will best suit the needs of the network and the computer. A system model of both a star network and a ring network were developed and then simulated using the same software models tested on the reference model.

1.3.1 – Project Description

A future addition to high performance computing technology for embedded application is a high-speed optical network for faster data transmission. This project first utilized current optical technology capabilities and a space system application as guided by a high performance distributed computer architecture to develop two new high-speed optical network models. Data was generated by testing the new network models on high-performance processing applications. These applications include Synthetic Aperture Radar processing, Radar Corner Turn processing as it is applied in future Space-Based Radar systems, and a randomly generated workload and transmission application. Both the Synthetic Aperture Radar processing and Space-Based Radar Corner Turning are data volume and bandwidth intensive applications used to process radar images from both air and space based radar in real time. Data from these simulations has then been used to address system network issues such as: bandwidth, latency, protocol, topology, and fault tolerance. Ultimately, an Optical System Network Architecture was developed to guide future optical device development. The two BAE SYSTEMS groups involved each have a stake in this research. The Space Electronics group is interested because they see an important future in optical networking for light speed data communication systems as well as fast light command and control systems. The Integrated Optical Electronics group wishes to determine a system application concept for their product research and development on a commercial scale. Thus they wish to find out what requirements devices like the demo 2 device might be required to meet prior to application to a product design.

There are two main domains of concern in this project. The first domain is optical networking technology. Networking technology includes optical parallel fiber

transmitters and receivers, optical fibers, optical switching or filtering devices, and high-speed network protocols. Some of these technologies are already commercially available like the Infiniband and RapidIO protocols while some like the TeraConnect TC-48 optical transmitter/receiver devices and the Agilent Photonic Switch are still in development. As it stands, none of this hardware will be needed to complete this project, their specifications will be used to simulate optical network systems.

The second domain of this project is the system application domain. This domain is made up of the application requirements used by BAE SYSTEMS to simulate the high performance distributed computer architecture. The requirements are that it will use COTS (Commercial Off The Shelf) parts, support multiple industry standard high level protocols, be configurable to meet unique system architecture needs, and enable high bandwidth, flexible topology, and be reliable, fault tolerant, and power efficient. These requirements paired with semi-radiation hardening will allow a distributed computer to run well in space. These requirements are derived from earlier research projects to determine the requirements of a high performance distributed computer architecture for space applications. This research and these simulations are from some of the next generation space processing research programs.

To perform this project we need the new Optical Networks to replace the existing Myrinet network in the hardware models of the simulator tools. A main design goal is to not require the alteration of any of the higher-level software or hardware. The reason for this goal is that the design of the distributed computer system strives to be configurable to meet unique system needs. If a node can be swapped between a copper network and an optical network relatively easily, then this goal has been met. Exchanging the Node Controller should be sufficient to alter the node for this purpose. The Node Controller has to remain capable of communicating with the other hardware elements of a node requiring that it still communicate with the same higher-level protocols. This means replacing the physical and the link layers while leaving the higher communication layers virtually untouched.

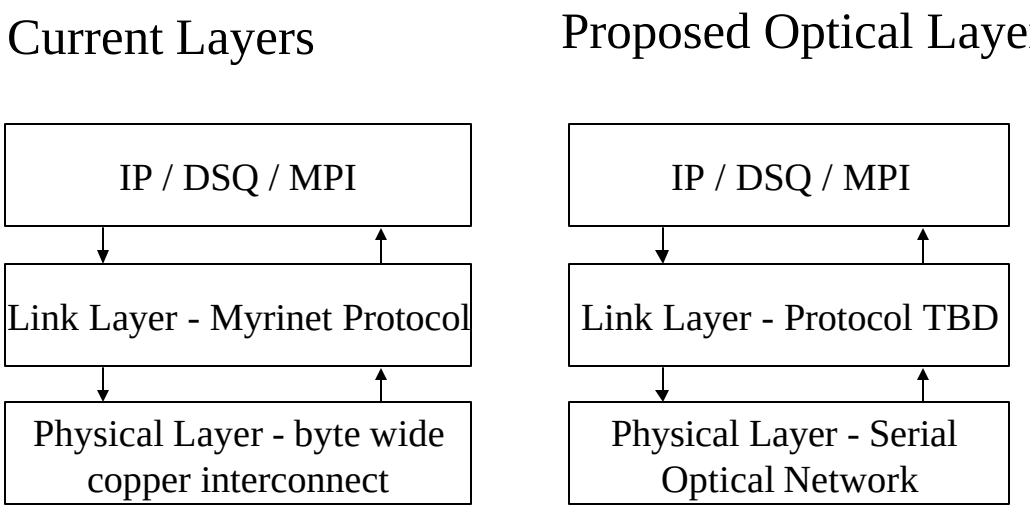


Figure 1-3: *Network layers before and after the new optical network would be added. The Link Layer will most likely utilize the Infiniband Protocol and the physical layer will be either channel or wavelength differentiated.*

By only altering the lowest two network layers we can leave the higher level of the high performance distributed computer architecture alone, meeting the requirement of utilizing the same high-level protocols and hardware. However, we need to determine the characteristics of the Link and Physical layers. The Link Layer will need to contain the protocol to transmit and receive data reliably. The Physical Layer will need to handle the electrical-optical-electrical conversion as well as how to handle the sending and receipt of bits.

1.3.2 – Design Goals

From the description of the project as given above, a list of design goals can be specified. These goals are the core elements to be aware of as the project is being performed. The design goals for this project are:

- **Use of COTS parts:** The project must strive to use available parts to minimize system cost and development time.

- **Support multiple industry standard high-level protocols:** The systems must strive to be transparent to the higher level protocols used on them. In this way any protocol used in the reference system could run on the altered system.
- **Be configurable to meet unique system architecture needs, flexible topology:** The systems must be modular and be able to support many different configurations as needed for different applications of the distributed high-performance computer.
- **Enable high bandwidth and be reliable and fault tolerant:** The optical networks replacing the current copper network should be able to handle well the new requirements of throughput at 580 MBps. The design should be able to recover from faults gracefully.
- **Provide an Optical System Network Architecture:** Will be used to guide future optical device development as well as future development of distributed high-performance computers.

Chapter 2

2 – System Designs

2.1 – Reference (Myrinet Based) Model:

The current research model is a distributed computer composed of multiple processing and input nodes connected by a Myrinet switch network.[5] A diagram of the computer with a Myrinet network can be found in figure 2-1. The basis for this model is the distributed system as described in section 1.2.1, each node is composed of a Node Controller (NC) and either processing elements, input/output elements, or control elements. However the model of this system is much simpler than the design it follows. The model is designed to show and analyze the network aspect of Myrinet and the input/output characteristics of the current design. A node consists of a software module and a node controller module. The software module reads in a software description and then generates events such as Send, Receive, or Process for the node to perform. The node control module is where all network functions are performed in the system.

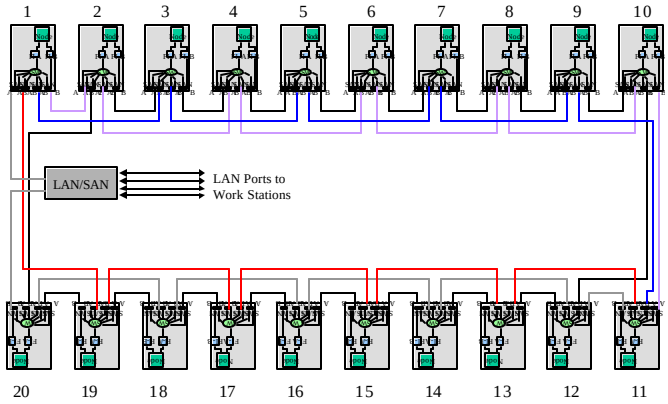


Figure 2-1: *Diagram of the current research model. The network is a Myrinet hypercube and each block is a processing node with multiple Myrinet interfaces.*

[5] See: Myrinet Network Specification Draft Standard

All of the interesting modeling, like: processing, sending and receiving, and the network fabric, are performed in the node control module. I have been purposely vague in the description of this model as it is not my design nor does it fall under my copyright protection. The descriptions above are to help relate the new work to the earlier work they are based on.

2.2 – New Network Models

When developing the specifications of these networks there were many needs to keep in mind. The most important is that we are aiming to use today's or tomorrow's optical technology. There are currently no all-optical packet switches or routers available or hinted at by companies working on optical devices. This means that once a light packet has been sent to the network, there is no current technology to redirect its travel, besides devices that convert light to electricity and work with it in that domain. A transfer of this sort would negate the benefits of optical technology in a local area network; Transmission lag and power consumption would be increased, as would be the weight of the system. The simplest ways to create end-to-end transmissions that are all optical is to use redundant optical hardware to create a direct optical path between source and destination for each message sent.

From the research two novel network models were decided on for the Optical System Network that encompass the idea described above. The two models are a star network and a ring network.

2.2.1 –Star Optical Network Design:

An optical star network for a distributed high-performance computer will rely on one central optical switch while leaving most of the current node architecture untouched. At each node there will be an optical converter device such as the IOE Demo-2. Attached to the optical converter will be a bundle of optical fibers that connect the node to the central switch. This converter will contain the physical layer conventions and directly connect to a link layer interface handling all data communication needs of the higher network levels within the node controller. The interface will transfer data to

individual transmitters in serial at a rate of 4.2 GHz per device. The receiver to which the data is sent depends on the destination node of the data.

Since no intelligent all-optical switches are available that can switch light 'by packet', this star network would need to be connected differently then most star networks. Both the Lucent Wavestar and Agilent Optical switch, among others, have slow optical switching technology that could be used to build the central switch for this system. The slowness of switching could be a problem, however in this novel configuration the switched paths only require an initial setup and relatively infrequent maintenance. Since each optical converter device has many transmitters and receivers it is possible to use an individual pair of these devices as an address for each of the other nodes from the sending node. At initialization of the system the host node will use a low speed copper backbone interface to provide the central optical switch with the correct switch path setup. This setup will provide a direct path from a sending node along a unique optical fiber through the optical switch and then along another unique fiber to the receiving node. Thus, there will be a direct, unique, optical connection between each node, termed Channel Division Addressing. Upon an error of a channel or transmitter or receiver, the system can use the backbone to allocate a new transmitter, receiver, or channel set in the central switch. To recover from an upset, error recovery will take as long as it takes to detect the error plus the relatively long time it takes to ask the central switch to rearrange its internal optical connections.

An alternative in the future will be wavelength division addressing. A different wavelength of light can be the address of another node and the light will be switched per wavelength in a next generation optical switch. The benefit of this alternative is that all the wavelengths of light may be transferred on the same optical fiber and multiplexed onto and off of that fiber at each end node. Additionally, the reduction of the number of optical fibers would decrease weight and simplify wiring of the system. However, the switch will need to be more complex as each wavelength will need to be divided first from the main stream to be directed by the optical switch.

In this network the link protocol used will be Infiniband, however any protocol of equal performance and specifications would suffice because the physical network is

transparent to the protocol used. Figure 2-2 below illustrates the star design described above.

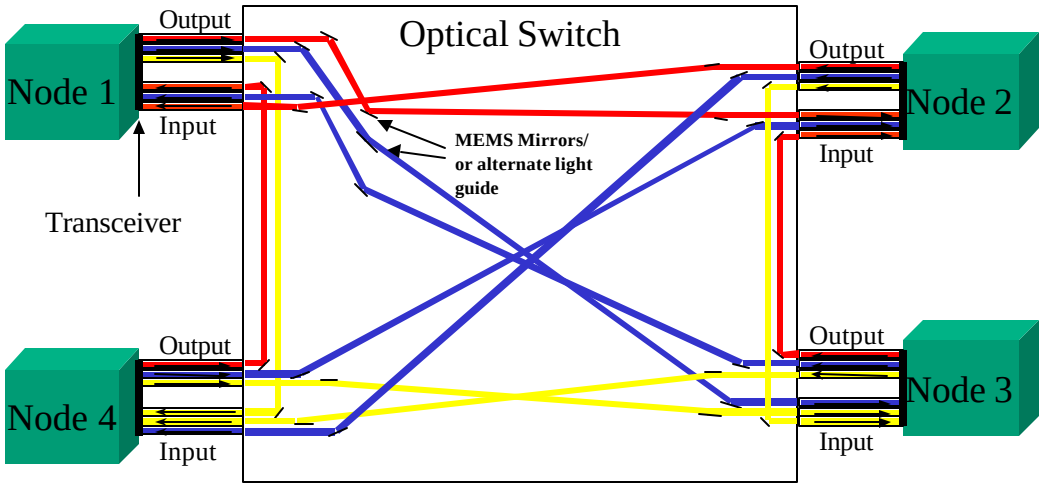


Figure 2-2: *A visualization of the optical switch and Channel Division addressing scheme of the star optical network. (Note: Drawing not to scale.)*

There is a possible problem with this system. There will be a dedicated full speed optical connection between each and every node. Each node will also be able to send data at maximum speed on these connections. A problem would develop when many nodes are all sending data to one node at high bandwidth. The node may not be able to consume the incoming data fast enough and thus drop some of the data. This type of system would best be employed in a command and control system where small amounts of data need to have guaranteed delivery in an extremely short time frame.

The optical switch can be built today with either Lucent's Wavestar Lambda Router or Agilent Technologie's Photonic Switch. (There may be other devices available to perform this, but these two are my focus.) However, even though the technology is here, it would take a good deal of research and development to build a device from either of these products. For the time being we will generalize the Optical Switch as having many channel/wavelengths per port that can be individually routed to a unique output channel/wavelength at any other port. Each port is bi-directional, connected to one node, and contains multiple channels. Light can pass unimpeded through the switch instantly

and configuration of the internal switching is done automatically and before runtime of the simulation.

2.2.2 – Star Optical Network Model:

The modeling of the star network was intended to replace as little of the reference model as possible to try and conform to the goal of only replacing the network aspect of the model. In this way only the fabric and node configuration were originally going to be altered. However, as the reference model was delved into deeply, it became apparent that every module of the reference model relied on other modules and that certain functions needed in this new design were missing or incompletely implemented.

First, the Star Optical Network system model includes 8 work nodes, a host node, and an optical switch as the hub of the star. A diagram of this system can be found in figure 2-3. The nodes are modeled as described below. The optical switch forwards packets to their destination without collisions.

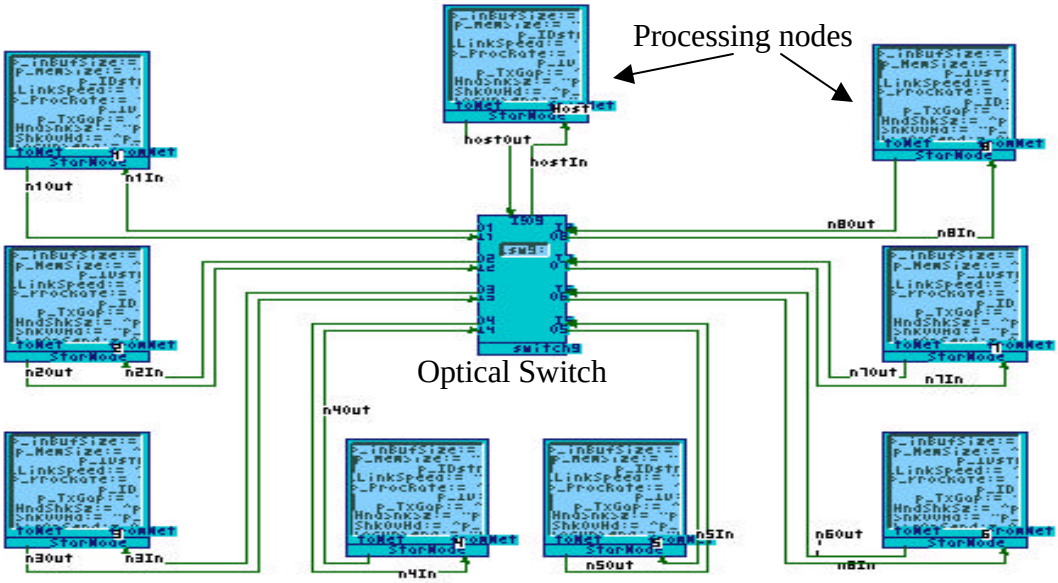


Figure 2-3: *The system setup of the Star Optical Network. The central module is the optical switch, and the outlying modules are the processing nodes*

The highest level of the system is similar to the reference model described above. Figure 2-4 shows the updated model. The AppFromFile module is the software module. The software models that the system run are fed into this module and then parsed into the

specific commands for the node to perform. The node controller module contains the hardware and software modeling of the sending and receiving processes as well as the optical interface and node processor and memory models. The optical network that replaced the Reference Myrinet network can be seen at the right of the node controller module. The multiple channels have been modeled as one connection allowing only one packet of each channel ID on it at a given time.

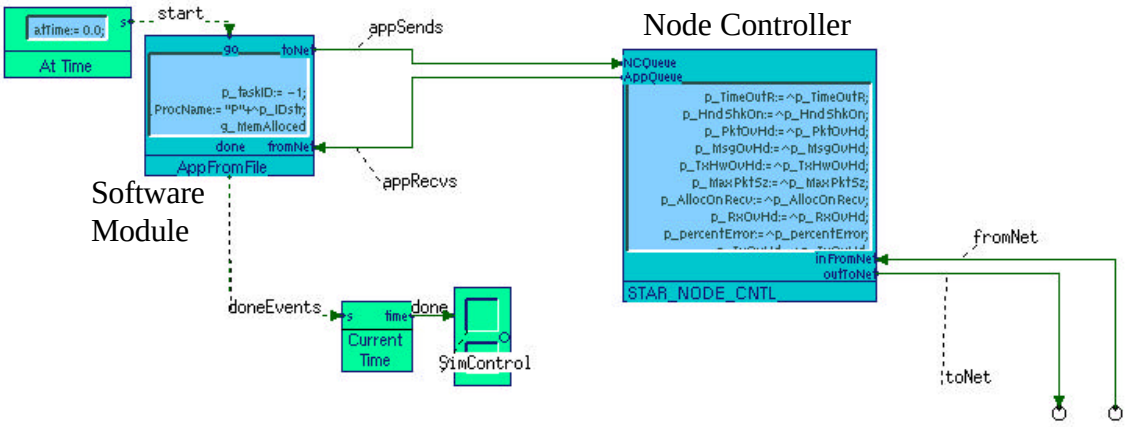


Figure 2-4: *Diagram of a node in the star system.*

The new architecture of the node is better seen inside the node controller module. Figure 2-5 shows the new architecture of the star model's node control module. Initially the only change here was to be replacement of the Reference Myrinet fabric module with a star fabric module, S_fabric. The goal of this replacement was to alter the network makeup without touching the main modeling of the node. Upon further analysis, all of the modules needed updating and upgrading. Most changes were required by the CPU module.

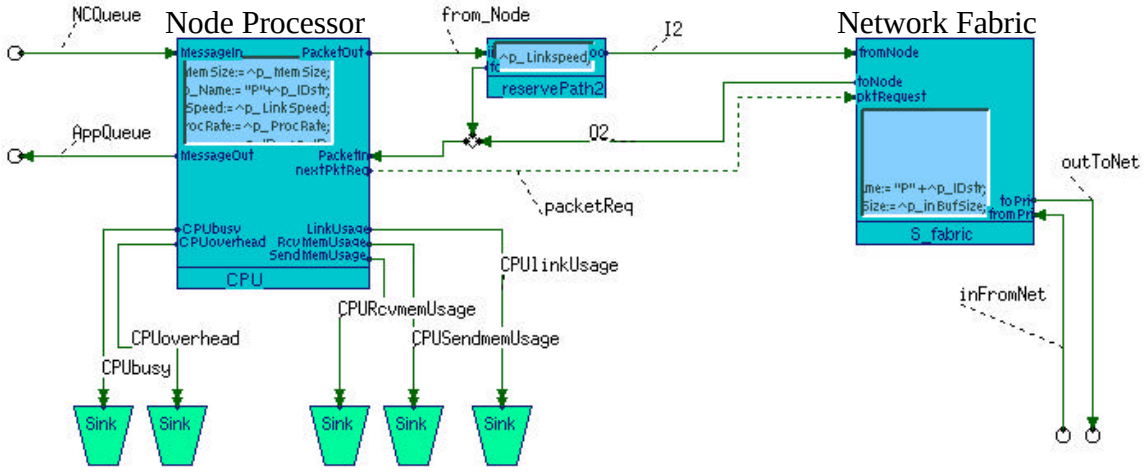


Figure 2-5: *Hardware model of the node control module. This module is similar to that found in the Reference model, however all of the modules herein have much more ability.*

Alterations made to the CPU module include the completion of an acknowledgement protocol, the addition of an automatic retransmission protocol, and modeling of delay and size increases based on error correction coding, serialization, deserialization, and error correction decoding. Furthermore, the addition of send and receive memories was required to allow for the modeling and logging of memory usage during sending and receiving messages. Besides these major alterations, there were many small changes to the CPU module to change the network type from path-based to a packet-based protocol. Some of these alterations were later removed from the CPU module and placed into a new module called reservePath2. This module contains the link activity logging and internal packet passing modeling of the node.

In general the CPU module works by first receiving a message send command. If the send hardware is not currently busy sending another message, the new message is read into the system where it is parsed into packets and sent out onto the network. The protocol for acknowledgements is Stop and Wait. This protocol is used because of the unique architecture of the system. A message must be read out of the DRAM memory and passed along a PCI bus to be sent out of the node. There is no memory available to hold messages at the interface as they wait to be sent. This means that there is only ever one path from DRAM memory available to send a message out of the node. Thus, the system is forced to use a stop and wait protocol and only send one packet at a time.

As a packet enters the CPU it is first parsed for type. If the packet is a Request to Send, Clear to Send, or Acknowledgement packet it is passed to the send hardware module. If the packet is a data packet, it is parsed and stored in partial message memory. When the entire message is received, only then is it passed to the rest of the node, in this model the rest of the node consists of the AppFromFile module.

The S_fabric module is the largest change to the node control module. Figure 2-6 is the Foresight diagram of the new star fabric module. The purpose of the star fabric module is to model the transmission of packet data on the individual channels as guided by the destination address. For incoming packets, there are many channels that could possibly have a packet for the node. To handle this many to one aspect, there is an input FIFO of packets that is modeled by the inputBuffer queue. As packets are fully received on a channel the ABselect module places them in the inputBuffer, and the nextPktDoor module is signaled that packets are waiting to be consumed. If the system is not currently busy processing a packet, the first packet in the queue is read out of the inputBuffer at the new PCI speed of 66Mhz by 64 bits across. If the inputBuffer ever reaches maximum size it will pass packets into the overflow sink, dropping them from the network.

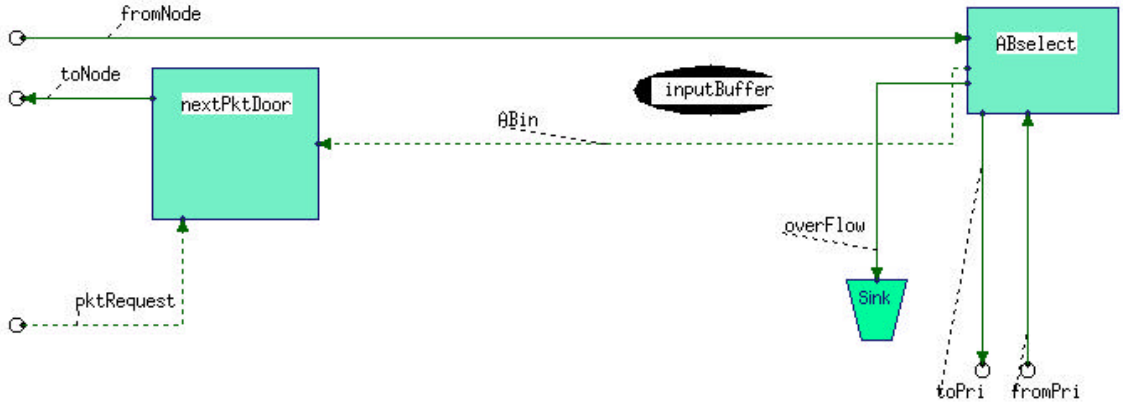


Figure 2-6: *S_fabric module of a star node. This module handles modeling of the input FIFO and output transmission.*

Data is collected from this model using a set of logging functions callable from any module in the model. These functions record the time delay of packets and messages into a log file specified at system initialization. Additionally, there are functions to

record link activity, memory allocation and deallocation, and model debugging. An example of these functions is in Appendix A along with all of the Star Optical Network diagrams and code.

2.2.3 –Ring Optical Network Design:

The ring network is similar in build to the star optical network in that it also utilizes channel division addressing and strives to leave most of the node untouched. However, the difference in this system is that each node has a single incoming channel addressed to it and the specialized optical hardware, instead of an optical switch, is a channel add/drop filter. The add-drop filter contains quick optical switching for each light channel passing through it. In this model, data communication is performed by allocating a single channel to each node. If a different node has a packet to send to the first node it transmits the packet on the channel that is addressed to the receiving node. The receiving node then always guides light on its channel onto a receiver at its optical interface. In this way each node has a full speed optical input, and if need be, could send at full speed to any other node. Broadcast messages will be handled by transmitting on more than one transmitter based on the message header. Figure 2-7 is a graphical representation of an add-drop channel filter for the ring network.

An add-drop filter is not currently available. However, this device could be made easily with Agilent Technologie's Photonic Switch. Additionally, Lucent Technologies WaveStar Lambda Router could also fulfill this job, but the size and price of the unit is prohibitive. Similar to the Star network, the actual product is not needed for this simulation, simply the technical aspects of the devices being used. In the case of the Add/Drop filter, all data transmission through the device is instantaneous and the receipt of data is non-blocking. The transmission of a packet will block data flowing on the channel already, causing a collision. Collision resolution is handled by the link protocol. When packets collide, they will be corrupted and dropped from the network. After a certain retransmission timeout, the data packet will be retransmitted. If the packet was a control packet, that control packet will be attempted again after a longer timeout.

Data collision is the single glaring problem with this setup. Unlike in the star network, a collision can happen during a node's attempt to send a packet. In the ring network each node adds its data for a node to the same channel. Additionally, due to the speed of light and technology limitations it is hard to detect when a packet is passing by. Thus, we have to add data and collide with any packets that are passing as we begin

transmitting. It will be interesting to see what sort of data load degrades the data transmission enough to cripple the network.

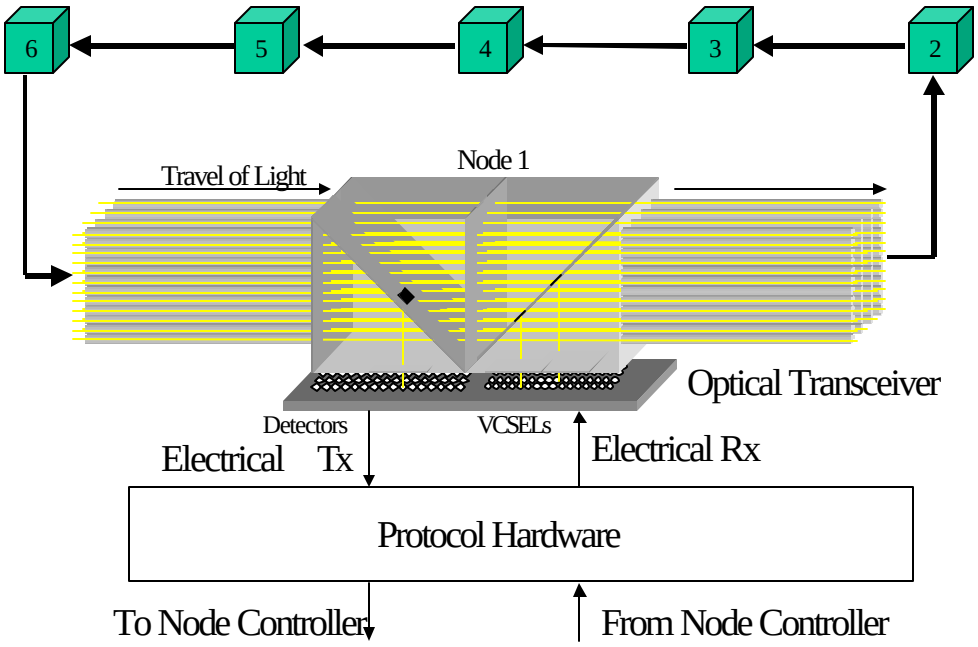


Figure 2-7: *A drawing of the add-drop filter and optical interface to a node in the ring optical network. The receive channel for this node can be seen being switched into the receivers for the node and then being transferred into the node through the Infiniband Protocol hardware. Data being sent out can be seen getting switched onto the channel for the node it is being sent to.*

2.2.4-Ring Optical Network Model.

The modeling of the ring network, as in the star network, was intended to replace as little of the node model as required to replace the system network. However, from the earlier work building the Star Optical Network Model it was found that there were many changes to be made to update the node to work with these models. However, the work done on the Star Optical Network Model gave me a node with all of the functionality required. This functionality included packet acknowledgments, packet retransmission, serialization as well as error correction coding, and message memory allocation. Thus,

from the Star Model, besides a few timing issues, the only modeling needed was the fabric of the star network.

At the system level a ring network is obviously a much different model. However, the ring architecture still contains 8 work nodes and a host controller node. Figure 2-8 is a diagram of the Ring Optical System Model. This network is based on two bi-directional rings.

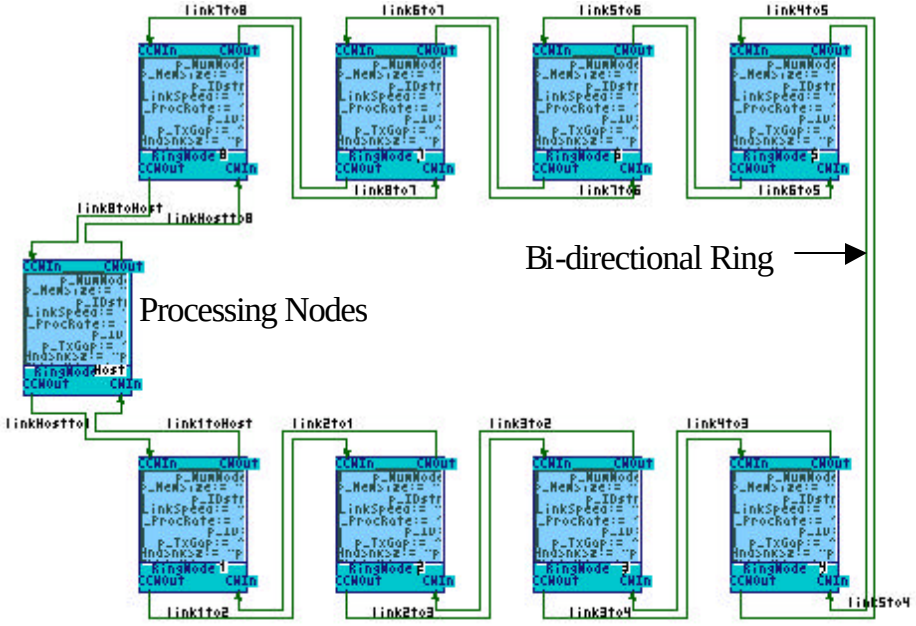


Figure 2-8: *Top level System diagram of the Ring Optical Network Model. Each module is a processing node connected by a bi-directional ring network.*

As most of the node has remained the same from the star model, the high level node model similarly contains the software parsing module and a node controller module. Figure 2-9 shows the Foresight diagram of a ring node. The node controller module is different as it now contains a different fabric as well as the dual ring ports for the bi-direction optical rings.

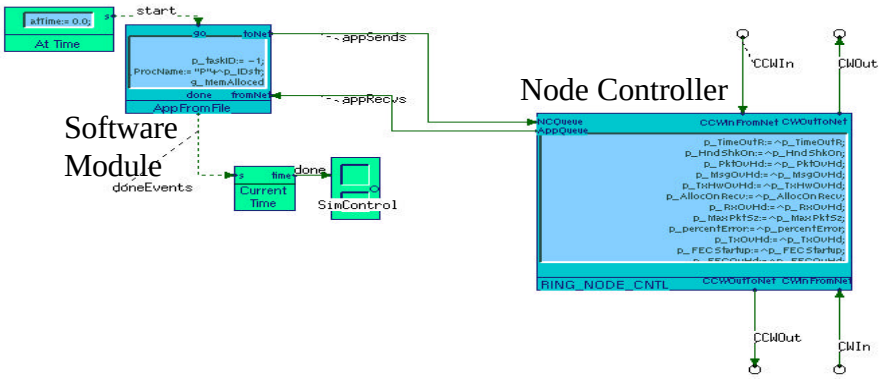


Figure 2-9: *Top level node diagram of a node in the Ring Optical System Network.*

Additionally, the node controller module of the ring model is generally the same as the node controller module of the star model. The difference of course is the replacement of the star fabric with a ring fabric to control the bi-directional rings. The CPU module is the same as that described in section 2.2.2 above. Figure 2-10 is a diagram of the ring node controller module.

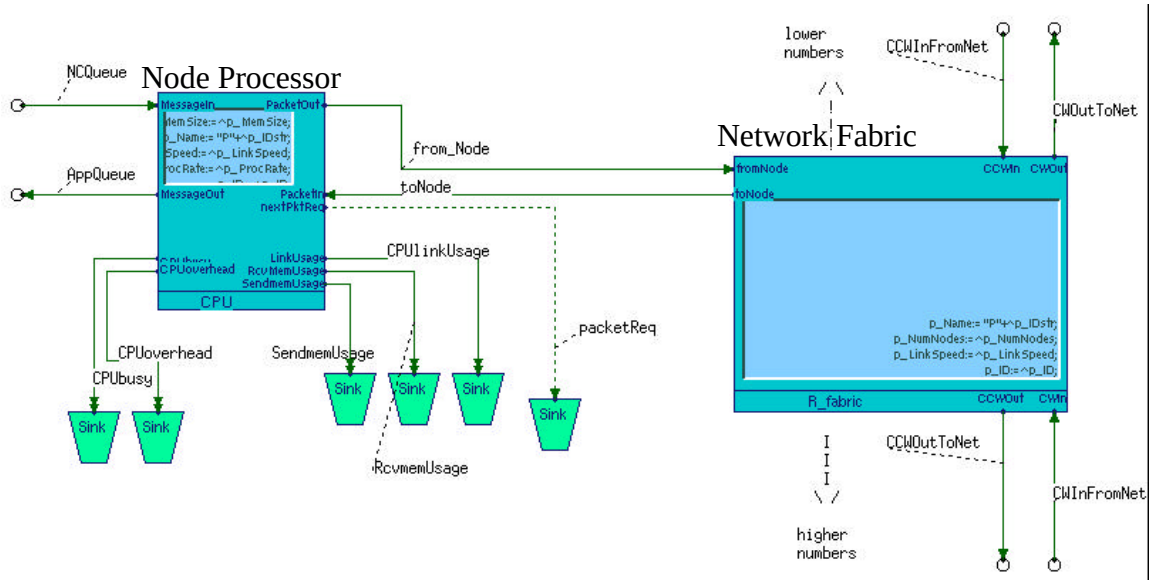


Figure 2-10: *Foresight diagram of the ring architecture node controller module.*

The interesting difference between this second architecture and the star network architecture is the ring fabric node, shown in figure 2-11. The function of this module is to model the different optical channels in each of the two bi-directional rings as well as to control the flow of packets onto and off of these channels at the correct time. This part is

easy to model. However, the colliding of packets and keeping track of how long a packet is on a channel at each node turned out to be much more difficult. Most of the modules shown in figure 2-11 are used to perform these timing and collision events. To describe this model the description of how a packet is transmitted from one node to another node on a channel is used.

The modeling of the passing of a packet is done using two data events and many state variables per node. As the packet is passed to the ring fabric module it is first passed to the gatekeeper module and then passed to either the clockwise (CW) ring or the counterclockwise (CCW) ring Send queue (CWSend or SSWSend). The ring the packet is sent to is decided by the distance to the sending node. This routing was decided on as all channels on both rings should have approximately the same throughput, and the fewest other nodes would be passed thus remove collision possibilities. As the packet is entered into the correct ring's send queue, the same ring's keyMaster module (keyMasterCW or keyMasterCCW) is signaled that an event is waiting to be parsed. At this time the keyMaster module will notice that it is a send that is waiting to be performed (the size of its send queue is greater then 0) and begin sending the packet. Additionally, if there was already a packet passing this node on the same channel it will set both packets to be corrupted and set the channel state to error for logging purposes. To time the end of the packet a timer is started and the packet is copied into the EndSend (CWEndSend or CCWEndSend) queue for that ring. When the timer signals that the packet is finished sending, a "end of packet" event is sent to the receiving node on the correct channel. If along the ring this packet finds its channel in the error state, it will mark itself as corrupted.

As a packet passes through a node on a channel other than that node's receive channel, it is immediately passed on to the next node. If that channel is currently being used to send a packet, it will mark both packets as corrupted and the channel in error. Otherwise, everything is fine and the packet remains uncorrupted. When a node receives a packet on that node's receive channel, it is passed to the gateKeeper. The gateKeeper then waits for the "end of packet" event for that packet. If the "end of packet" event is corrupted, then entire packet is marked corrupted and dropped from the system. If that event is not corrupted, then the packet is passed to the node.

Figure 2-11 is a top level description of the ring fabric. The full code of this model can be found in Appendix B along with all other code for the Ring Optical Network model.

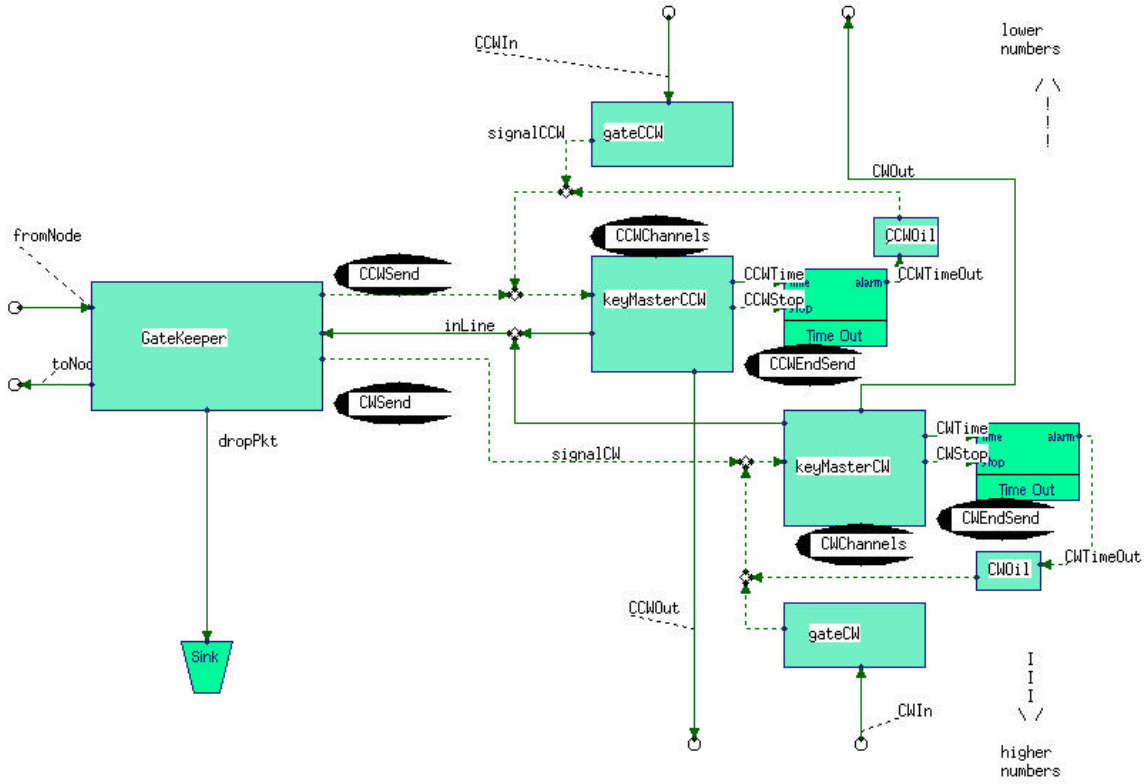


Figure 2-11: *Foresight diagram of the network fabric in a Ring Optical Network node.*

2.3-Software

The software used in the simulations run on the Star Optical Network Model and the Ring Optical Network Model are high performance computing applications. These models are coded in PERL to allow short amo unts of code that build large processing task lists. The PERL codes are evaluated at run time and the task list is passed to Foresight as the software model. These specific applications were chosen because they already had distributed architectures suited to multiprocessing. Furthermore, the radar applications were chosen as the benchmark of these systems because these applications are today performed at ground stations after collecting large amounts of data from space based radar platforms. If these applications could be performed using a high-speed redundant distributed computer onboard the satellite, then the processing of images

would be much faster. Also, the time needed to download all of the data from the satellite for one of these applications is a current bottleneck of a radar system. If instead it was only necessary to download the final image, resources could be preserved. The debugging software models can be found in Appendix C. The Space-Based Radar Corner-Turn, Synthetic Aperture Radar, and Random Workload software models were built by BAE SYSTEMS and are not included in this paper.

2.3.1- Space-Based Radar Corner-Turn

This software model is based on a corner turn processing algorithm for a Space-Based Radar application. A corner-turn is common in most signal processing applications and involves a NxN matrix "frame" of data in row-major order. The rows are then divided equally between the nodes and processed to redistribute the data into column-major order. This benchmark requires a large amount of inter-node traffic as data is passed out, rotated, and then redistributed. Obviously, if a faster network is involved, then the time to redistribute data should be reduced. This software model uses a matrix of 1024x1024 pixels per frame, with eight bytes per pixel, and processing a total of four frames.

2.3.2-Synthetic Aperture Radar – Back End Processing.

This software model is based on the back end processing of synthetic aperture radar. However, this model is optimized for software pipelining and independent Fast Fourier Transform (FFT) processor nodes. The process includes partial pulse compression, polar reformatting, corner-turning, multiple FFTs, and autofocus and multilook averaging. This application is being used for the same reasons as the corner-turn application. This could be run on a hardware system developed from these network simulations. The system will process eight frames containing 16 primary rate interfaces containing 59200 range values per interface. This software model contains no randomness so one run on each model will provide sufficient data.

2.3.3-Random Workload

The random workload software generates N "phases" of computation, where one phase consists of processors doing a random amount of work, then exchanging data in a random pattern. The random work is distributed according to one of several distributions, including: uniform, normal, and exponential. The mean and standard deviations for a given distribution are also specified. The random communication pattern is specified with a single parameter that represents the probability that a given source sends a message to a given destination on any given phase. The size of the message is determined by three parameters that specify the distribution, mean, and standard deviation. The parameters used for this software model are normal processing and communication distributions. The mean and standard deviation of the processing distribution are both 1000 cycles. The mean and standard deviation of the communication pattern are 10000 and 1000 for the probability of sending and number of bytes in the message. This software model was performed four times with 16 phases per model. The results were than averaged for each model.

2.3.4-Debugging Workload

There were many software models coded to debug the hardware models as they were being built. These software models include a Packet Walk model, a Congestion to Host model, and a Two Node Dialogue software model. The Packet Walk model transmits a single message from one node to the next in order to test simple transmission and system setup. The Congestion to Host model makes all nodes transmit after a small random delay to the host node. This model is to test congestion control and handling of the system. Finally, the Two Node Dialogue model tests the ability of repeated messages between two nodes for general debugging. The text of these models can be found in Appendix C.

Chapter 3

3 – Results and Analysis

3.1 – Simulation Results

All simulations were performed using the same simulation parameters and simulation files as described in section 2.3. The Reference hardware model using the Myrinet network simulated the easiest as it took minimal debugging work to update the model to a newly updated software model interface and compiler. As I was working others were improving the Reference model and software simulation schema. The Star Optical Network hardware model took a large amount of the time to debug. In the end, after multiple revisions, the star network hardware model described above worked well over each software model. The real disappointment was that the design for the Ring Optical Network hardware model failed.

The ring network failed because the design did not take into account the immense amount of data and different data types that would need to pass on the single incoming channel of a node at any given time. It was setup such that only one node was allowed to transmit data on a channel at a time, however there were also Request To Send packets being sent to see if the channel was free that would corrupt the data transmission. Additionally, at the same time the receiving node could be sending data to a third node. The third node would then generate acknowledgements onto the receiving channel of the receiving node. This would corrupt any data on the channel as well as the acknowledgment. To try and solve these issues, many different protocol alterations were made to the ring network system including message transmission retry and dynamic retransmission times. Finally, a set of control channels was added for the acknowledgement and other control functions. In the end the degradation incurred by these additions and the added weight and power that would be required by a second set of channels and transmitters and receivers was seen as too far from the system goals, and

work on the ring network system was halted. Additionally, the small amount of data that could be taken showed a distinct loss in throughput and performance, making the system unable to be compared with either the Reference or Star network systems. If work on the ring network is to proceed an alternative collision avoidance scheme will have to be introduced.

3.1.1 – Space-Based Radar Corner Turn Results.

Below are the general system results for the final simulations running the Space-Based Radar Corner Turn software model as described in section 2.3.1. This simulation was run twice to view the randomness involved in it. In general the same amount of data was transferred on the network in each simulation, and there are only slight variations in the timing and bandwidth utilization. This slight variation is due to small randomness added to the amount of processing done at each node and in a small random delay at the start of the simulation. Because there are only slight variations we will use the first run of the Space-Based Radar Corner-Turn model on each network model as our guide.

Table 3-1: *Space-Based Radar Corner- Turn results for the Reference Network Model*

Simulation Attribute:	**First Run**	**Second Run**
Number of messages:	72	72
Number of bytes sent:	67108896	67108896
Total run time: (seconds)	1.264178	1.264612
Avg. message size: (Kbytes)	910.2	910.2
Avg. message latency: (ms)	17.558024	17.56406
Avg. message bandwidth: (Mbyte/s)	53.085018	53.066775

Table 3-2: *Space-Based Radar Corner-Turn results for the Star Optical Network Model*

Simulation Attribute:	**First Run**	**Second Run**
Number of messages:	72	72
Number of bytes sent:	67108896	67108896
Total run time: (seconds)	0.386004	0.385561
Avg. message size: (Kbytes)	910.2	910.2
Avg. message latency: (ms)	5.361173	5.355014
Avg. message bandwidth: (Mbyte/s)	173.855222	174.055184

3.1.2 – Synthetic Aperture Radar, Backend Processing

Below are the simulation results from the simulation runs of the Synthetic Aperture Radar back end processing as described in section 2.3.2. As this model is identical every time it is run only one simulation run was performed on each hardware network model.

Table 3-3: *Synthetic Aperture Radar processing results from the Reference Network Model*

Simulation Attribute:	
Number of messages:	224
Number of bytes sent:	45863424
Total run time: (seconds)	2.68871
Avg. message size: (Kbytes)	199.9
Avg. message latency: (ms)	12.003169
Avg. message bandwidth: (Mbyte/s)	17.057781

Table 3-4: *Synthetic Aperture Radar processing results from the Star Optical Network Model*

Simulation Attribute:	
Number of messages:	224
Number of bytes sent:	45863424
Total run time: (seconds)	0.570536
Avg. message size: (Kbytes)	199.9
Avg. message latency: (ms)	2.547035
Avg. message bandwidth: (Mbyte/s)	80.386585

3.1.3 – Random Workload

The results from the Random Workload are slightly different in nature to the prior two models. We are averaging the results from four simulations using the random software model described in section 2.3.3 to give general numbers for the results of a

random workload. The average, standard deviation, and deviation percentages were then calculated. As can be seen the individual simulations produced relatively similar numbers for both the Reference Network and Star Optical Network; as given by the normal work and communication distributions. In general the standard deviation is good, however the Reference model produced higher deviations between its simulations. This is not an issue as the deviations follow from the number of messages sent, which deviated wider in the Reference model.

Table 3-5: *Random Workload results from the Reference Network model*

Simulation Attribute:	**First Run**	**Second Run**	**Third Run**	**Fourth Run**
Number of messages:	197	227	248	223
Number of bytes sent:	1922615	2245368	2412632	2135162
Total run time: (seconds)	0.056013	0.073658	0.076962	0.074021
Avg. message size: (Kbytes)	9.5	9.7	9.5	9.4
Avg. message latency: (ms)	0.284331	0.324484	0.310332	0.331932
Avg. message bandwidth: (Mbyte/s)	34.32432	30.483794	31.348203	28.845408

Simulation Attribute:	**Average**	**St. Dev.**	**Dev Pct**
Number of messages:	223.75	20.9344214	2.3
Number of bytes sent:	2178944.25	205461.796	2.4
Total run time: (seconds)	0.0701635	0.00954896	3.4
Avg. message size: (Kbytes)	9.525	0.12583057	0.3
Avg. message latency: (ms)	0.31276975	0.02096919	1.7
Avg. message bandwidth: (Mbyte/s)	31.2504313	2.29711565	1.8

Table 3-6: *Random Workload results for the Star Optical Network Model*

Simulation Attribute:	**First Run**	**Second Run**	**Third Run**	**Fourth Run**
Number of messages:	239	249	221	231
Number of bytes sent:	2444577	2420871	2225783	2218388
Total run time: (seconds)	0.019341	0.019207	0.015874	0.018242
Avg. message size: (Kbytes)	10	9.5	9.8	9.4
Avg. message latency: (ms)	0.080926	0.077138	0.071826	0.07897
Avg. message bandwidth: (Mbyte/s)	126.391052	126.038727	140.2198	121.60791

Simulation Attribute:	**Average**	**St. Dev**	**Dev Pct**
Number of messages:	235	11.8883697	1.3
Number of bytes sent:	2327404.75	122034.022	1.3

Total run time: (seconds)	0.018166	0.00160451	2.2
Avg. message size: (Kbytes)	9.675	0.27537853	0.7
Avg. message latency: (ms)	0.077215	0.00391147	1.3
Avg. message bandwidth: (Mbyte/s)	128.564372	8.06935605	1.6

3.2 – System Network Analysis:

In general both the Reference Myrinet Network System and the Star Optical Network System are unique. The nodes of the Reference system have an internal bus 32bits wide at a speed of 33MHz. This totals to a maximum throughput of approximately 1Gbps. The Myrinet network of the Reference system has a throughput of 1.28Gbps in each direction. The throughput of the network is exactly that required by the node. Alternately, the network sends data at a similar rate so only small amounts of receiving buffer are required for small burst overflows.

The nodes of the star optical system have an internal bus 64 bits wide at a speed of 66 MHz. This internal bus then has a throughput of 4.2Gbps. The star optical network has a throughput of 4.2 Gbps per channel. This totals to a maximum throughput of 37.8Gbps in each direction, assuming there are nine channels entering and leaving each node. The throughput of the node is thus much less than the total possible data that can be sent or received through the network. The consequences of this are analyzed in the next few subsections. In general the results showed that the star optical network is a good start to development of future networks and distributed computing systems.

3.2.1 – Simulation Run Time Analysis

There is no distinct way to relate the Reference Network timing characteristics to the Star Optical Network timing characteristics. This is because so many upgrades were made to the Star Optical Network hardware at the same time. First, the node data processing speed was increased from 500 MHz to 750 MHz. This would then give an average decrease in the processing time of a node by 33%. Next, the creation/consumption ability of a node was increased by 400%. The networks were increased in the same manner to approximately match the bandwidth of the nodes. The software models were chosen such that the node network bandwidth increase is the dominant factor in the increase in speed of the system. Additional factors that reduce the

overall system speed and network bandwidth increases of the Star Optical Network model are the addition of serialization delay, and error correction delay and redundancy. At these speeds the delay is negligible, however the increase in the size of data transmitted due to redundancy is 25% and can not be ignored.

The network speedup is reflected well in the running times of the Space-Based Radar Corner Turn and Random Workload software models. In each of the software model simulations it can be seen that the increase in speed is slightly below four times that of the Reference model. The Space-Based Radar Corner-Turn software model sees an increase of 330%. The Random Workload software model sees an increase of 380%. It is surprising to note however, that the Synthetic Aperture Radar software model saw a 470% increase in simulation speed over the Reference Network model. This higher increase in system speed-up is probably due to the unique pipelined design of the backend processing for Synthetic Aperture Radar as compared to the other software models. Additionally, the Star Optical Network is better designed for the pipeline of the Synthetic Aperture Radar processing software. This is because within the pipeline there is a distinct order of passing of data as each node performs its assigned task. In the Reference Network the Myrinet fabric forced contention between nodes passing to one another. The Star Optical network on the other hand has a guaranteed path between each pair of nodes allowing contention free passing of the data along the processing pipe. In this case the Star Optical network could be built such that only the required channels for the pipeline are present, removing excess weight and power needs.

3.2.2 – Message and Packet Latency

Message and packet timing is recorded to a log file, similar to other System data. What might be unique is that a message is marked as sent when it is first given to the Node Controller of a node to be transmitted. It may then be held up in a queue while other prior messages are sent. When the last of a message's packets are received it is logged as finished. This method of message latency logging is used to view node as well as network congestion. Packets, in comparison, are marked as sent when a node first attempts to transmit that packet. When the packet is received at a node it is then logged

as received. In this way packet latency helps to show network as well as individual channel congestion.

The average latencies from the three software model simulations can be seen in Table 3-7 below.

Table 3-7: *All Software Model Simulation latency results.*

Latency Attributes	**Reference Network**	**Star Network**
Space-Based Radar Corner Turn software		
Avg. message latency: (ms)	17.558024	5.361173
Avg. packet latency: (ms)	0.204808	0.005244
Synthetic Aperture Radar processing software		
Avg. message latency: (ms)	12.003169	2.547035
Avg. packet latency: (ms)	0.193584	0.005095
Random Workload software		
Avg. message latency: (ms)	0.284331	0.080926
Avg. packet latency: (ms)	0.122793	0.003614

As can be seen the average message latencies correlate to the increase in node and network bandwidth. However, these average packet latencies are not a good judge of the network. Beside the fact that each model uses a different maximum packet size because of the link protocol they are using, the average packet latency includes the latency for very small packets that are the last part of a message. In the Star Optical Network there are smaller packets, thus average latency is much lower then expected.

A better way to analyze packet latencies is to view packet latencies over time. Figures 3-1, 3-2, and 3-3 are graphical representations of the packet latencies vs. simulation time for the Space-Based Radar Corner Turn, Synthetic Aperture Radar processing, and Random Workload software model simulation runs, respectively.

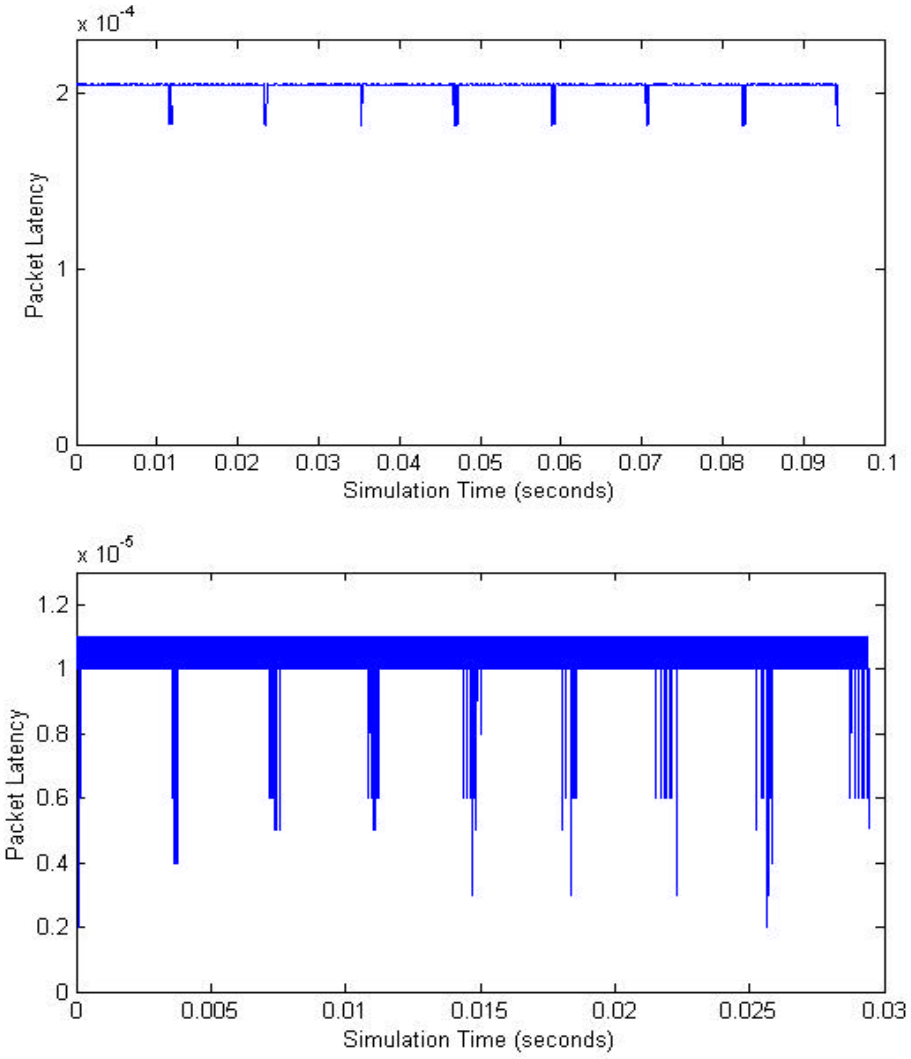


Figure 3-1: *Data packet latencies vs. Simulation Time for simulations of the* Space-Based Radar *Corner Turn software model. Reference Network (Top). Star Optical Network (Bottom)*

Packet latencies for the Space-Based Radar Corner-Turn model are very good. In the Reference Network, latencies during the main section of a send are mostly uniform. At the end of each message, the packet latency drops dramatically as there is only a small amount of data to send to complete the message. In the Star Optical network, packet latency holds very steady at 0.01 ms. There are much smaller latencies corresponding to the acknowledgement packets. If viewed on a smaller time scale, data packet latency is uniform over simulation time as is acknowledgement packet latency. These packet

latencies are expected from the Space-Based Radar Corner Turn software. In each phase, data is transmitted between unique sets of nodes. Thus, overall network congestion should be uniform.

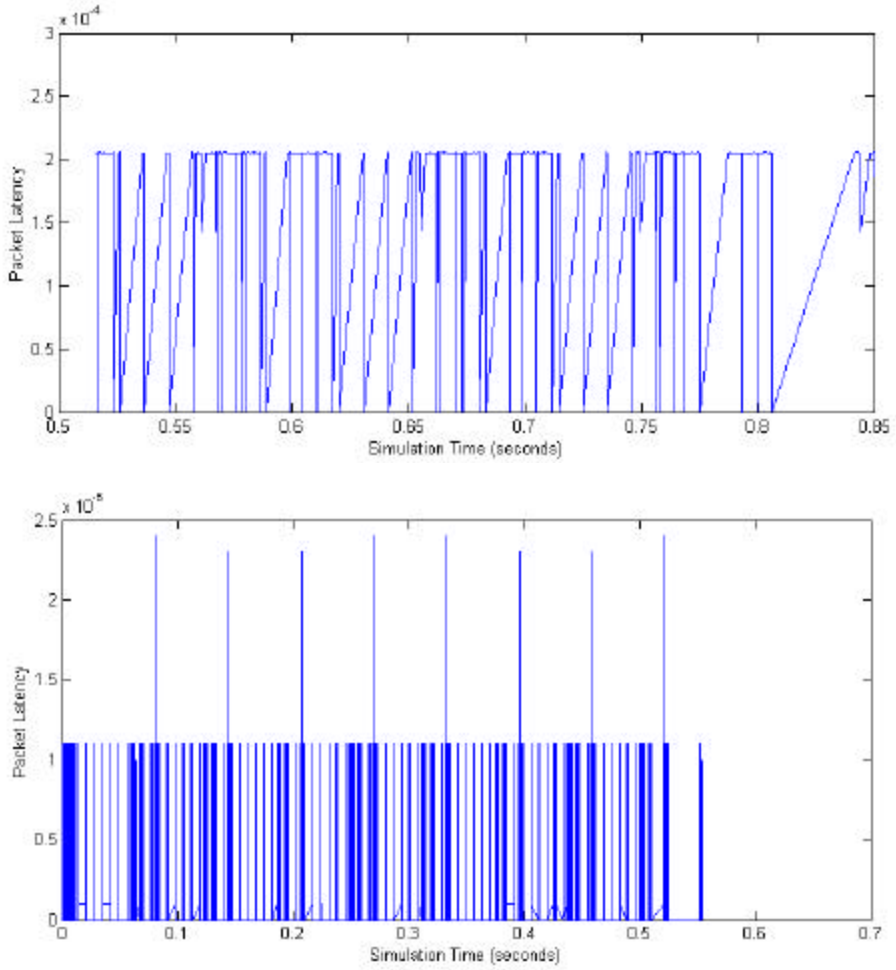


Figure 3-2: *Packet latencies vs. Simulation Time for simulations of the Synthetic Aperture software model. Reference Network (Top). Star Optical Network (Bottom)*

Synthetic Aperture Radar processing has the best packet latency. Both the Reference Network and the Star Optical Network have very consistent latencies with very little deviation. Of course this could mean that each packet is being delayed an equal amount, but it is actually due to the pipelining of the system. The software model was built in such a way to maintain a predictable flow of information that does not interfere with itself. One aspect that is puzzling is the large spike in latency in the Star Optical

Network latencies. This spike is a result of packet transmission from the pipeline at the end of each frame's processing. It is possible that there is a bottleneck from the last processing stage to the host node. This is most likely due to the fact that the host node receives the original frame data from the sensors and then performs large amounts of processing on said data.

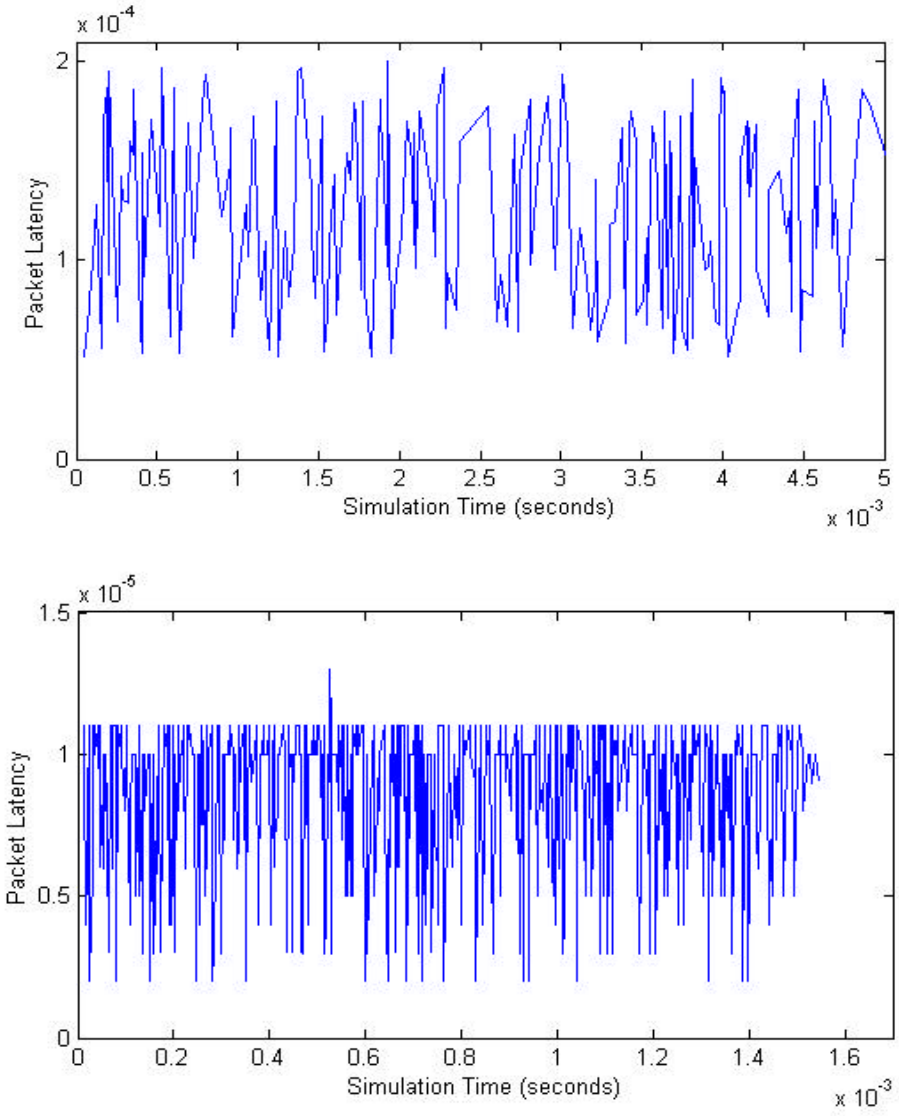


Figure 3-3: *Packet latencies vs. Simulation Time for simulations of the Random Workload software model. Reference Network(Top). Star Optical Network(Bottom)*

Packet latencies for the Random Workload software model are as expected. Due to the random nature of the size of messages and when messages are sent, the overall average packet latency does not show packet latency well. For the most part maximum sized packets average to a consistent latency in both of the network models. Interestingly there is a single spike of a set of larger latency packets. These packets were most likely the victims of temporary network congestion.

3.2.3 - Network Bandwidth Utilization

Network Bandwidth utilization is recorded in link utilization in the same way that memory usage and packet and message latencies are recorded. A network link in either the Reference Network or the star optical network is the physical connection between two points. Figure 3-4 below is an example of a graphical link utilization analysis showing the link utilization as a function of time for the star optical network model running the random workload software. Each gray horizontal band represents the state of a channel over time. When the band turns black, it is being used to transmit data. In the Reference model it was possible to that the network would be in contention for resources. If this were to happen the band would turn white until the contention was resolved. In general it is not efficient to examine this graphical link, instead we will analyze the utilization percentages.

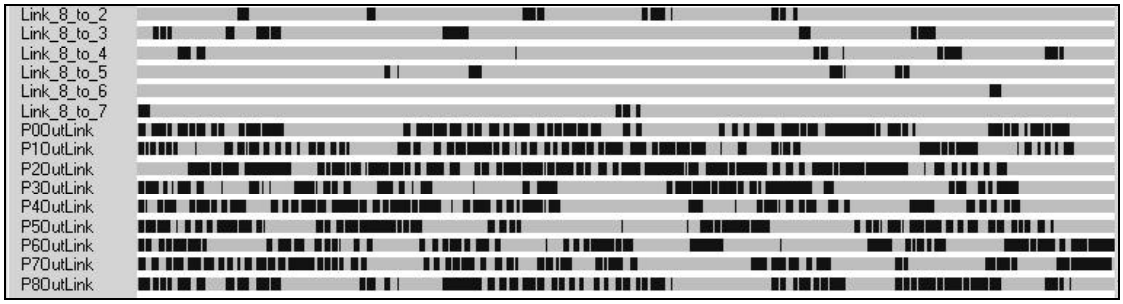


Figure 3-4: *Partial Timing diagram for a simulation of the random workload software on the star system. The link lines at the bottom are the combined output of each node. For example, P6Outlink is the total output of the node at full network bandwidth. Above the combined links are the individual channels from one node to the next. There is a unique link from each node to every other node.*

Network bandwidth utilization during simulations running the Space-Based Radar Corner-Turn software model can be best seen in Table 3-8. A node's outlink is the output of the Node Controller before it enters a network. In the case of the Star Optical Network the outlink is the connection between the Node Controller and the transmitters on the individual optical channels. This outlink is the only connection to the network and subsequently is a good measure of the utilization of the bandwidth of the node. As can be seen, in the Reference model network utilization is almost 100%. The Star Optical Network, in contrast, only uses 66% of its node network bandwidth. However, a lower usage is not necessarily bad.

Table 3-8: *Node output link utilization for simulations using the* Space-Based Radar *Corner-Turn Software model*

Reference Network Utilization	
Network Link	**Utilization**
Host Node outlink	0.984345915
Node 1 outlink	0.984345915
Node 2 outlink	0.984345631
Node 3 outlink	0.984345991
Node 4 outlink	0.984345803
Node 5 outlink	0.984345933
Node 6 outlink	0.98434581
Node 7 outlink	0.984346063
Node 8 outlink	0.984345747

Star Optical Network Utilization	
Network Link	**Utilization**
Host Node outlink	0.646857614
Node 1 outlink	0.64678989
Node 2 outlink	0.646829247
Node 3 outlink	0.646843446
Node 4 outlink	0.646843101
Node 5 outlink	0.646811113
Node 6 outlink	0.646858213
Node 7 outlink	0.646873968
Node 8 outlink	0.646821366

Similar data was obtained for simulations run on the Synthetic Aperture Radar backend processing software model and the Random Workload software model. The data for Synthetic Aperture Radar software simulations and Random Workload software simulations are in Table 3-9 and 3-10, respectively.

Table 3-9: *Node output link utilization for simulations using the Synthetic Aperture Radar Software model*

Reference Network Utilization	
Network Link	**Utilization**
Host Node outlink	0.069290154
Node 1 outlink	0.123072238
Node 2 outlink	0.123077157
Node 3 outlink	0.123064273
Node 4 outlink	0.123067787
Node 5 outlink	0.024413499
Node 6 outlink	0.024418184
Node 7 outlink	0.062575769
Node 8 outlink	0.062569342

Star Optical Network Utilization	
Network Link	**Utilization**
Host Node outlink	0.021808375
Node 1 outlink	0.038500425
Node 2 outlink	0.038524468
Node 3 outlink	0.038740131
Node 4 outlink	0.038761382
Node 5 outlink	0.007937666
Node 6 outlink	0.007938746
Node 7 outlink	0.000127867
Node 8 outlink	0.01955228

Table 3-10: *Node output link utilization for simulations using the Random Workload Software model*

Reference Network Utilization	
Network Link	**Utilization**
Host Node outlink	0.389802874
Node 1 outlink	0.538584948
Node 2 outlink	0.519053627
Node 3 outlink	0.699036908
Node 4 outlink	0.531393922
Node 5 outlink	0.441882961
Node 6 outlink	0.514080218
Node 7 outlink	0.522561017
Node 8 outlink	0.50308883

Star Optical Network Utilization	
Network Link	**Utilization**
Host Node outlink	0.49111572
Node 1 outlink	0.466611278
Node 2 outlink	0.609606163
Node 3 outlink	0.353853087
Node 4 outlink	0.431387718
Node 5 outlink	0.411944896
Node 6 outlink	0.437053923
Node 7 outlink	0.442546034
Node 8 outlink	0.457219244

Overall, the bandwidth utilization data obtained from the three software simulation models shows that although the Star Optical Network System is much faster and has a much higher bandwidth, it is more wasteful of that bandwidth. This result is evident before analysis of the individual channel utilizations of the Star Optical Network. The utilization of all of the channels from a node add up to the outlink utilization for that node. I believe that the Star Optical Network does not work as efficiently as the Reference model because, as with the ring network, it is interfered with by control

packets. There is much room for improvement because each of the individual channels has much more ability and is used sparingly. One such needed improvement is the alteration of the automatic repeat protocol. Currently the protocol sends a single packet and then waits for an acknowledgement. If a node sent a set of N packets it could then receive acknowledgements as it transmitted more data.

3.2.4 – Memory Utilization/Requirements

An aspect of the Star Optical Network system of interest to examine is the input/output memory requirements for a node. For a given node the network is effected by message receive memory, packet receive memory, and message send memory. If there are N nodes in the system, then there may be, at the maximum, N-1 messages in the process of being sent to a given node. The current node model waits for the entire message to be received before the message is processed by the node. The message is stored in main memory, but these messages might be very large and thus consume large amounts of memory. In the Reference network model, there was only one message being received at any time due to the Myrinet network specifications. The receive message memory consumption for each of the three software models can be found in figures 3-5, 3-6, and 3-7 below. In each diagram, a step shows the allocation of memory to contain the entire message. When the message arrives that memory is deallocated as the message is processed into the node.

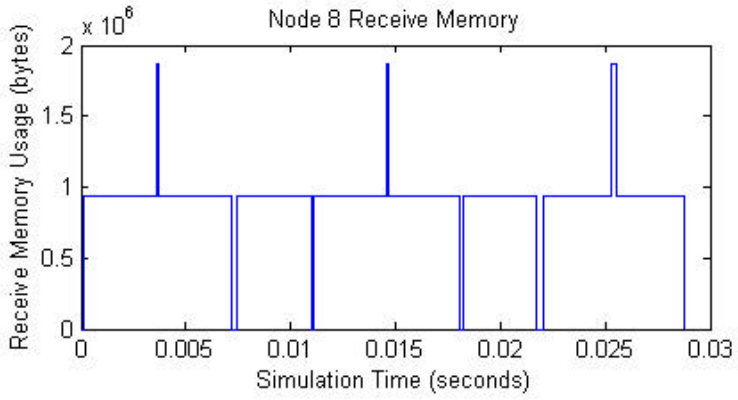


Figure 3-5: *Receive memory usage for the* Space-Based Radar *Corner Turn software model. Node is representative of the receive memories for all nodes in the Star Optical network model. There is little variation of the receive memories of the other nodes.*

Figure 3-5 diagrams receive memory usage for node 8 of the Star Optical system running the Space-Based Radar Corner Turn software model. All of the other nodes have similar receive memory usage. This pattern arises because of the communication pattern of this software model. Each node received some data from another node and then from a second node. Each node then processes that data, splits the processed data in two and then sends the data out to two other nodes. Except for the fact that there is an immense amount of data being received at any given time, this model is very predictable and thus a static amount of receive memory may be allocated from the system memory at all times.

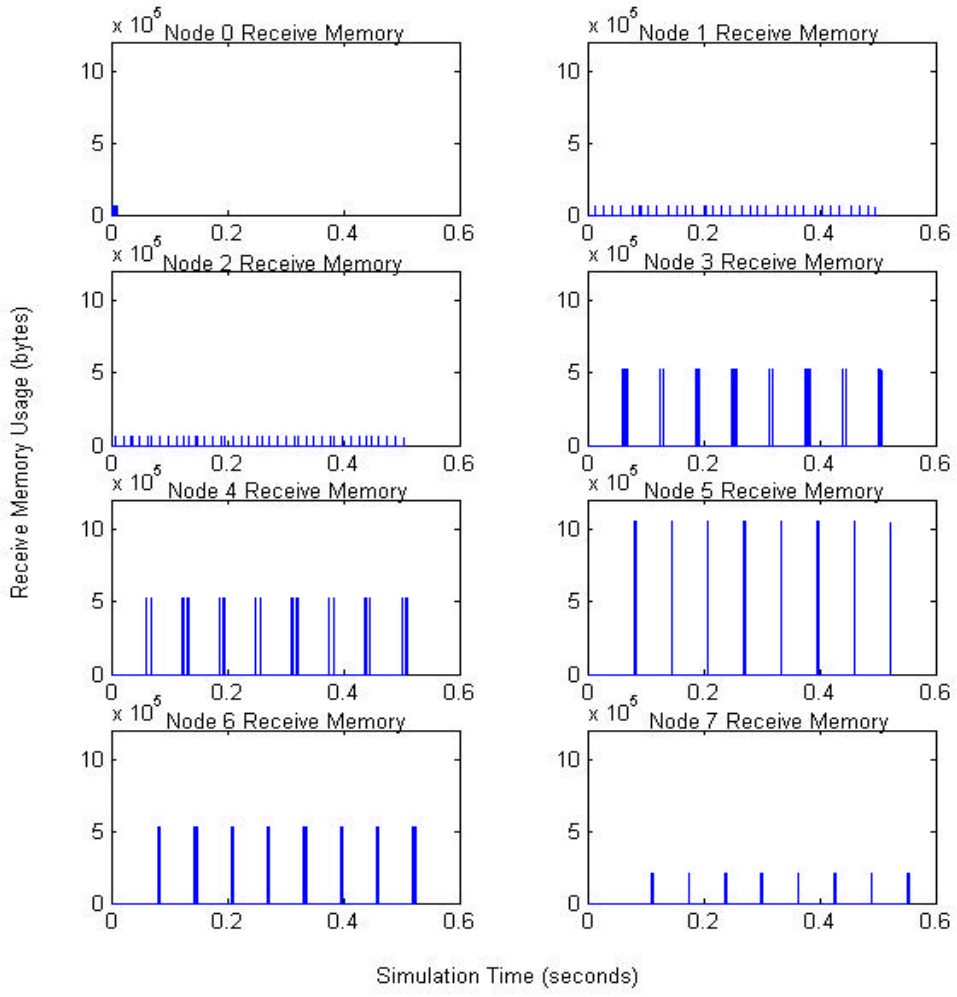


Figure 3-6: *Receive memory usage for the Synthetic Aperture Radar software model. Node 8 is used as the sensor node and thus only is an input for data to the system, thus omitted.*

Figure 3-6 diagrams receive memory usage for all nodes of the Star Optical system running the Synthetic Aperture Radar software model. In this model each node has a different processing assignment in the pipelined processing of radar images. As can be seen in the diagram, each node then requires a different amount of receive memory. This is due to processing expansion and collection of the data at first, and then processing and condensing of the data to form the final image. From this model we can see that a system may be built in such a way that reduces cost, weight, and power. We can design the specifications of each node to contain only the amount of receive memory that that stage of the pipeline requires. Such a design however will reduce the flexibility of those components For example, nodes 1 and 2 will need only small amounts of receive memory at any one time and in contrast node 5 will need large amounts of receive memory at sporadic times. Node 5 may be better off allocating memory dynamically, while nodes 1 and 2 may work better with independent static receive memory.

Finally, figure 3-7 diagrams receive memory usage for all nodes of the Star Optical system running the Random Workload software model. This model shows the general amount of receive memory required by a general system. As can be seen in the diagrams for each node, it cannot be determined ahead of time what the requirements of a node will be. Some nodes require little receive memory sporadically while some nodes require larger amounts all of the time. In general this pattern will change at each running of this software model. We cannot make distinct statements about the total amount of memory required as this is determined by the amount of data being processed by the system. However, from this simulation we can say that given a uniform random workload of this nature that the total amount of receive memory required for a given node is approximately four times the maximum message size. This means that in the worst case four other nodes are sending data to a given node. The other three runs of the Random Workload software model corroborated this outcome.

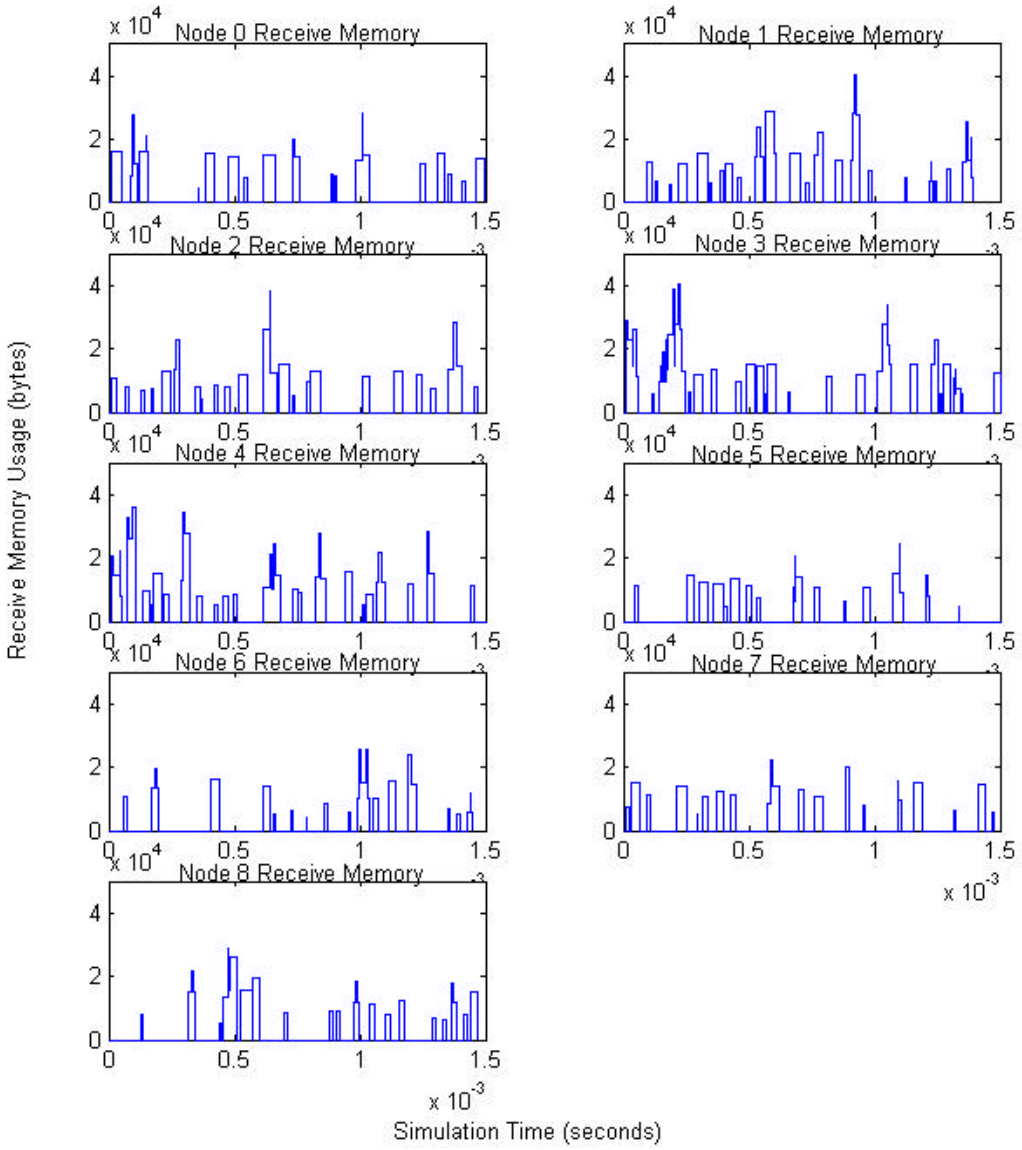


Figure 3-7: *Receive memory usage for the Random Workload software model.*

A similar memory concern is the buffering of just received packets before they can be processed and added to a waiting partial message. At the Star Optical Network interface of a node, there can be up to N-1 packets being received at a time. The consumption of packets is then slower then the reception of packets. In the three software models simulated on the Star Optical Network system there is no excess use of the receive packet buffers. The software models do not make more then a few nodes

send to a single node at any time. The consumption of packets by the receiving node is fast enough that packets do not back up in the FIFO buffer. A diagram of the size of the incoming packet buffer is not included because it only varies between one and zero. At no time does the packet buffer ever begin to backup. From these software models and their requirements on the system the resulting data indicates that no packet buffer is required.

Finally, memory utilization for the sending of messages is only an issue in this system if it is allowed to be. If there is not enough message memory then the node will have to wait for messages to be sent before it can queue additional messages. It is possible that at this time the node is blocked and the system will then be slowed. Similarly, if there is too much message memory then that memory will be wasted and the system resource will be wasted. The full usage of the send memory for all nodes for all three software models can be found in appendix D. Figures 3-8 and 3-9 are representative diagrams of the send memory usage for the Space-Based Radar Corner Turn and Random Workload software models. The Synthetic Aperture Software model is slightly different.

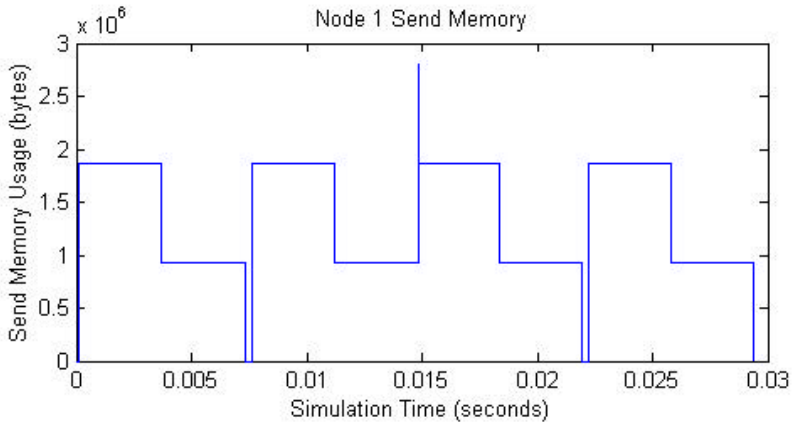


Figure 3-8: *Send Memory for simulation of the* Space-Based Radar *Corner Turn software model.*

The diagram above is an example of the send memory usage of the Space-Based Radar Corner-Turn software model. This diagram looks like the receive memory usage diagram above as there is a very even structured communication pattern in this model.

Similarly, a large amount of memory is needed, but this is a variable of the data being passed. In general the send memory required is bounded and will not exceed three messages queued to be sent at any given time.

The Synthetic Aperture Radar software model send memory requirement is similar to the Space-Based Radar Corner Turn send memory in that it mirrors the receive memory requirements. In general each node requires a different amount. There is no representative diagram as each node performs a different section of the SAR processing. Additionally, this memory requirement is bounded and can be built into the unique specifications for each node.

The Random Workload software model send memory usage is random as expected from analysis of the receive memory usage. The diagram below is one of the most active send memory usage diagrams. At its peak the node has nine messages queued for transmission to other nodes. The memory required is a function of the size of the messages.

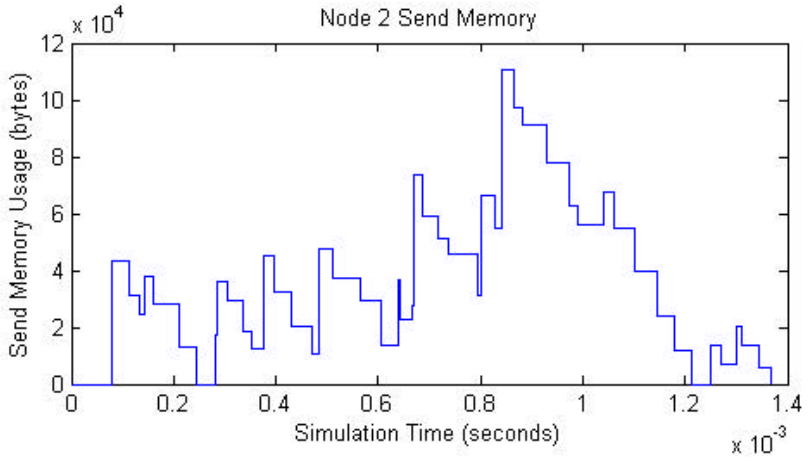


Figure 3-9: *Send Memory for simulation of the Random Workload software model.*

3.3 – Project Results Overview Table

Project Issues	Ring Optical Network	Star Optical Network
Message and Packet Latency	N/A	Latency increased with the upgrade on the node speed. Network congestion was all but nonexistent.
Network Analysis	N/A	Bandwidth Utilization is lower then the Original Networks. The network has room for growth.
Memory Utilization	N/A	Utilization is constant and predictable for all software models.
Memory Requirements	N/A	Not analyzed well in this project, but the need is predictable and allocation could be performed to maximize a node.
Network Topology	Good. Implementation of a better protocol and possible redundancy through more channels will be needed.	Great. Wavelength channels could help reduce weight and power.
Network Protocol	Needs massive collision avoidance design and implementation work.	Good. Node development will clear delay issues.
Node design	Needs to be upgraded with an alternate packet transmission protocol.	Needs to be upgraded with an alternate packet transmission protocol.
Technology Status	"Channel Add Drop Multiplexors" are not available and need to be designed and built.	Optical Switch is available, but needs to be integrated with the system. Node interfaces need to be designed and built.

Chapter 4

4 – Conclusion

4.1 – Network Systems

4.1.1 – Star Optical Network

In general the Star Optical Network system performed better then the Reference Network system. This result is dependent on many factors, but we believe that the Star Optical System is the correct next step towards an optical network for a high-performance distributed computer. The high points of this new network design are network utilization and its room for growth, memory utilization, and speed. However, there are many aspects of the systems that need to be looked into further like specific system bottlenecks, transmission protocols, and the failure of the ring optical network design.

The Reference Network has a network utilization that on average was higher then the Star Optical Network utilization. However, the Star Optical Network has much more room for growth. The Star Optical Network can grow because of its lower link utilization. The lower link utilization is due to the speed increase of the network itself, the added serialization, error correction latency, and data redundancy. Basically, the design of the computing node itself, i.e. transmitting only one message at a time, is the bottleneck that keeps the Star Optical Network utilization low. The current node design is based on the Reference Myrinet Network, and for this reason the computing nodes will need to be redesigned to work well with the Star Optical network.

Memory utilization is also a high point of the Star Optical Network design.
In general the send and receive memory usage of the Star Optical network system over the three software models is good. However, we have not determined what the effects of limitations on memory would be. Additionally, initial results from the three software models indicate that the current setup of the Star Optical Network does not need any sort

of input packet buffer. This has the potential to reduce the amount of weight, cost, and power required in the computing system.

Another high point of the Star Optical Network system is that it can handle the increase in speed required by a new generation of computing nodes. In fact it has more bandwidth capabilities than the nodes are able to use efficiently. These increases are evident in the timing analysis in chapter 3. Overall, the speed of the Star Optical network system is sufficient to meet the desired speed increase of the upgraded node, even though there were many new aspects of a system added that can slow the system down.

Unfortunately, although the Star Optical Network system proved itself in many areas, there are also many concerns that need to be more fully addressed. First, the Star Optical Network system is bottlenecked not by its network, but by the nodes to which it connects. The nodes can handle only one message and one packet transmission at a time, which reduces the bandwidth utilization of the network. Related to this issue is the automatic request retransmission protocol. As mentioned only one packet is transmitted at a time and then an acknowledgement is waited for. It is preferable to change this protocol to a sliding window protocol where a number of packets are transmitted and then only those that were not received correctly get retransmitted. It is not certain that the addition of the ability to send more than one message at a time is a good idea because of the design of the node. This capability would then alter the network memory requirements of a node. Send and receive memory usage will increase and the need for an input packet buffer might arise. We recommend incorporation of a sliding window packet transmission protocol, but do not recommend allowing the transmission of more then one message at a time.

Another problem with the Star Optical Network system is the same as that which degraded the performance and design of the Ring Optical Network system to the point where it failed as a good design for this project. The inefficiency of bandwidth usage is due to acknowledgements. Often the link utilization would be degraded due to a node waiting for the acknowledgement of a packet while the node sending that acknowledgement is busy sending a packet to another node. The maximum degradation a node sees is when the sending node has to wait the entire send time of a maximum sized packet before it can receive an acknowledgement. Alterations to the system to fix this

could be to modify the acknowledgement protocol to a sliding window protocol, some sort of alternate network path for control signals as attempted in the Ring Optical Network system, or possibly a smaller maximum packet size or alternate network protocol.

4.1.2 – Ring Optical Network

As mentioned earlier in this paper, the Ring Optical Network system design did not work. Basically, collisions became a large problem for the reference design and the updated design that worked slightly better was no longer in the scope of this project. The addition of a second network to handle control communications is too large and would be too power hungry to be of much use. This is not to say that a Ring Optical Network system could not be used in a distributed computer. Originally, it was favored over the Star Optical Network system. One of two things needs to be performed to make a ring network a reality; either the network architecture needs to be drastically, or a more complex communications protocol needs to be used to meter transmissions on each channel. This protocol may produce similar results to the Star Optical system, but will require more development.

4.2 – Future development

Many aspects for possible future development with these network models have been described above. From the analysis of the Star Optical Network we recommend a new node controller design. In this design the node will have the ability to send multiple packets at a time and possibly to allow more then one message to be sent at a time. Of course the optimal way to send more then one message would use the novel structure of the Star Optical Network to send to two different nodes. A variable that would need to be decided on then is how many messages can each node send at any given time before the nodes get overwhelmed with data being received on multiple channels at full data rates.

Another future research area concerning this project is looking at the effects of memory limitation on the overall efficiency of the Star Optical Network system. During the simulations performed on the three software models the processing nodes were allocated a large amount of memory for receiving and sending messages. This memory

was never exceeded and it is a good possibility that limitations on this memory would produce lag in the network system. It is possible that a message not be accepted at a receiving node because of lack of memory, thus forcing the system to wait until memory is available.

Additionally, we believe more research should be performed on the Ring Optical Network system. This model does have promise, however there are multiple problems with its current design. Redesigning of the node can help many of these problems. Mainly the Ring Optical Network needs a more complex packet control protocol. The ring is quite useful for many reasons. It does not rely on one central device for all communication and it intrinsically has more fault tolerance capability. If a ring is cut, then the network may use the other direction ring to bypass the fault. However, from general past experience in the networking field and from this research, we understand that this ring design does not work. In the future a ring-based system could work given more development.

Finally, there are many research possibilities that derive from the optical technology used as the base for these networks. One aspect of both of these models that needs research is of course the hardware devices described above. The Star Optical Network optical switch and node interfaces can be built today, but there is development required before any devices could be used. Similarly, the Ring Optical Network node interface optical multiplexor could be built with today's technology, however they do not exist in this form as available products. Another aspect of optical technology that would need research are the transmitters and receivers used at each node. These devices are currently available, but an integrated Demo-2 device is not available for production. Additionally, an alternative to redundancy by channel, using parallel VCSELs, detectors, and optical fibers, in these Optical Networks is redundancy by wavelength, using a single optical fiber and multiple different wavelength VCSELs. Research in this area would be most helpful to reduce weight and overall connection complexity. However, additional failure mode analysis is required to determine if there needs to be additional physical redundancy. If the sustem were dependant on a single optical fiber and this fiber was cut then the system would be fully interrupted.

In conclusion, this research has shown that for a next generation High Performance Distributed Computer system a solution to its future data distribution fabric could be a Star topology based Optical Network. The speed increases from such a network provide the bandwidth needed by a next generation computing node. The Ring Optical Network probably would not be a good idea because of the additional research and development needed to make it a viable solution. However, there are still many areas of the Star Optical system that need to be looked at like memory usage and node design. This research concludes that a Star Optical Network is a viable future network for a High Performance Distributed Computer.

References:

- Agilent Photonic Switching Platform: N3566A Dual 16x32 Photonic Switch White Paper. Agilent Technologies.

- Requirements Review Documentation. December 15-16, 1998. Sanders: A Lockheed Martin Company.

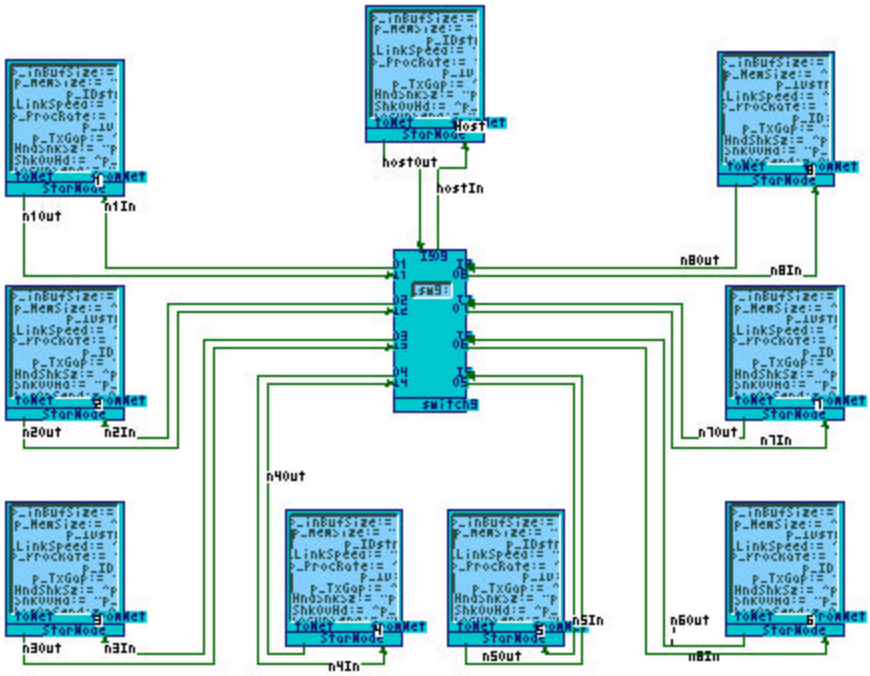
- Myrinet-on-VME Protocol Specification Draft Standard, VITA 26-199x Draft 1.1. VITA Standards Organization. August 31st, 1998.

- Ngo, David. A Reliable Infrastructure Based on COTS Technology For Affordable Space Applications. February 1 2001. IEEE.

- WaveStar LambdaRouter Technical Bulletin. Lucent Technologies Inc. May 3, 2001.

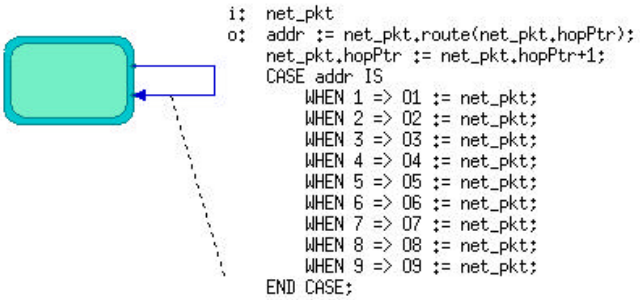
Appendix A: Star Optical Network System Diagrams

Top-level Star Optical Network System Diagram:



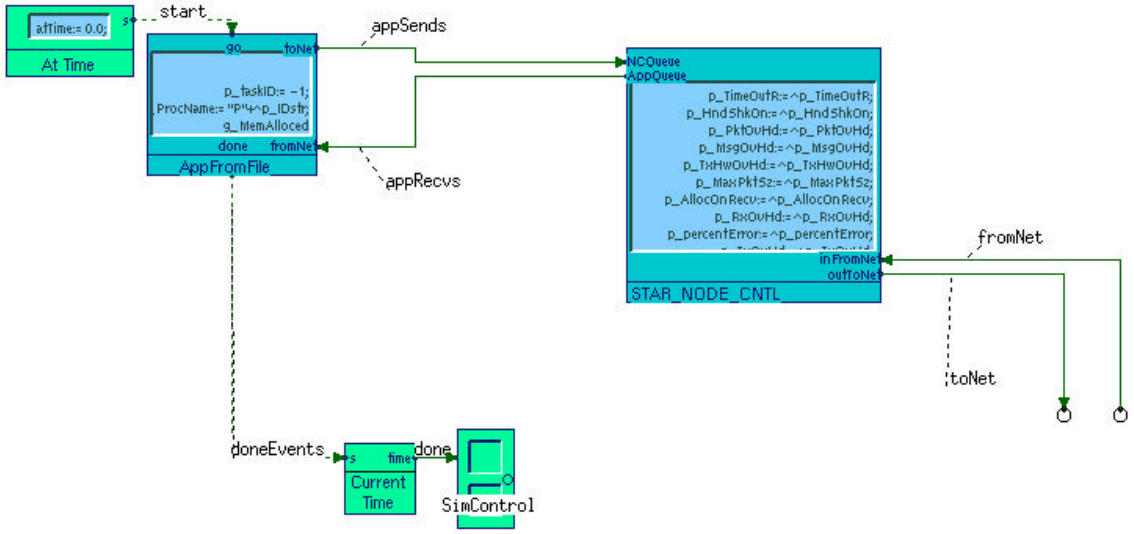
Central Optical Switch (switch9) Finite State Machine, all inputs are net_pkt:

```
Inputs:  net_pkt;
Outputs: 09, 08, 07, 06, 05, 04, 03, 02, 01;
Locals: addr = MSinteger;

Initialize:
BEGIN
    NULL;
END;
```

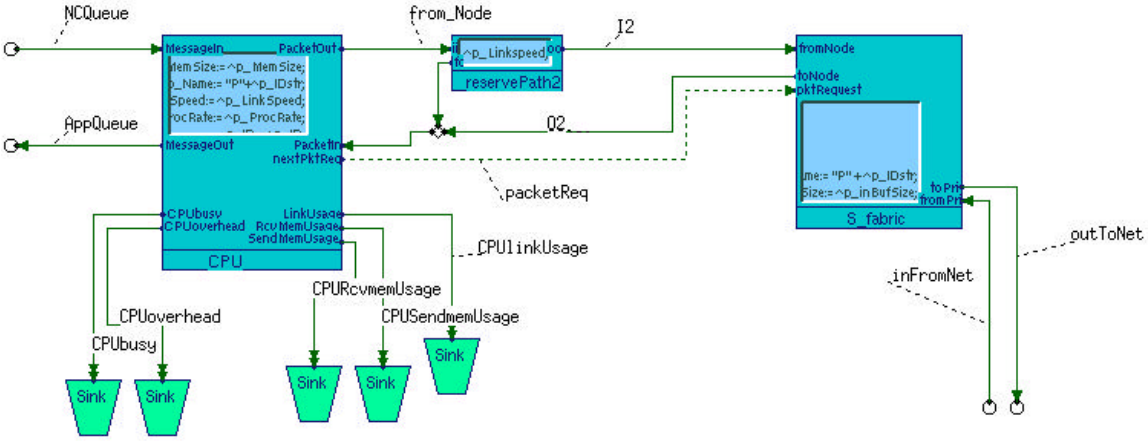


```
i:  net_pkt
o:  addr := net_pkt.route(net_pkt.hopPtr);
    net_pkt.hopPtr := net_pkt.hopPtr+1;
    CASE addr IS
        WHEN 1 => 01 := net_pkt;
        WHEN 2 => 02 := net_pkt;
        WHEN 3 => 03 := net_pkt;
        WHEN 4 => 04 := net_pkt;
        WHEN 5 => 05 := net_pkt;
        WHEN 6 => 06 := net_pkt;
        WHEN 7 => 07 := net_pkt;
        WHEN 8 => 08 := net_pkt;
        WHEN 9 => 09 := net_pkt;
    END CASE;
```

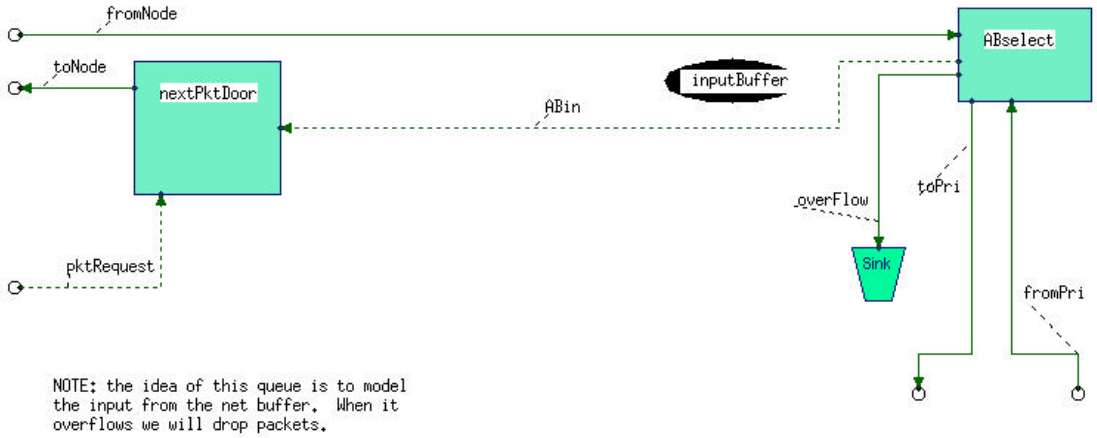
Star Optical Network Node (StarNode) Module Diagram:



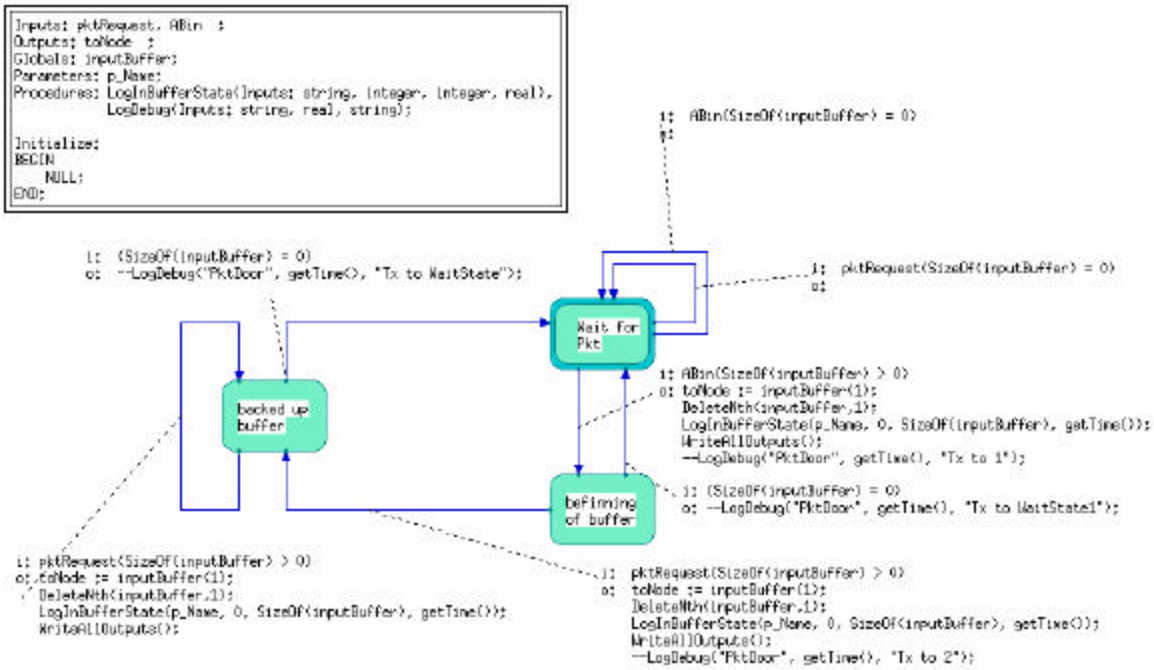
Star Optical Network Node Controller (STAR_NODE_CNTL) Module Diagram



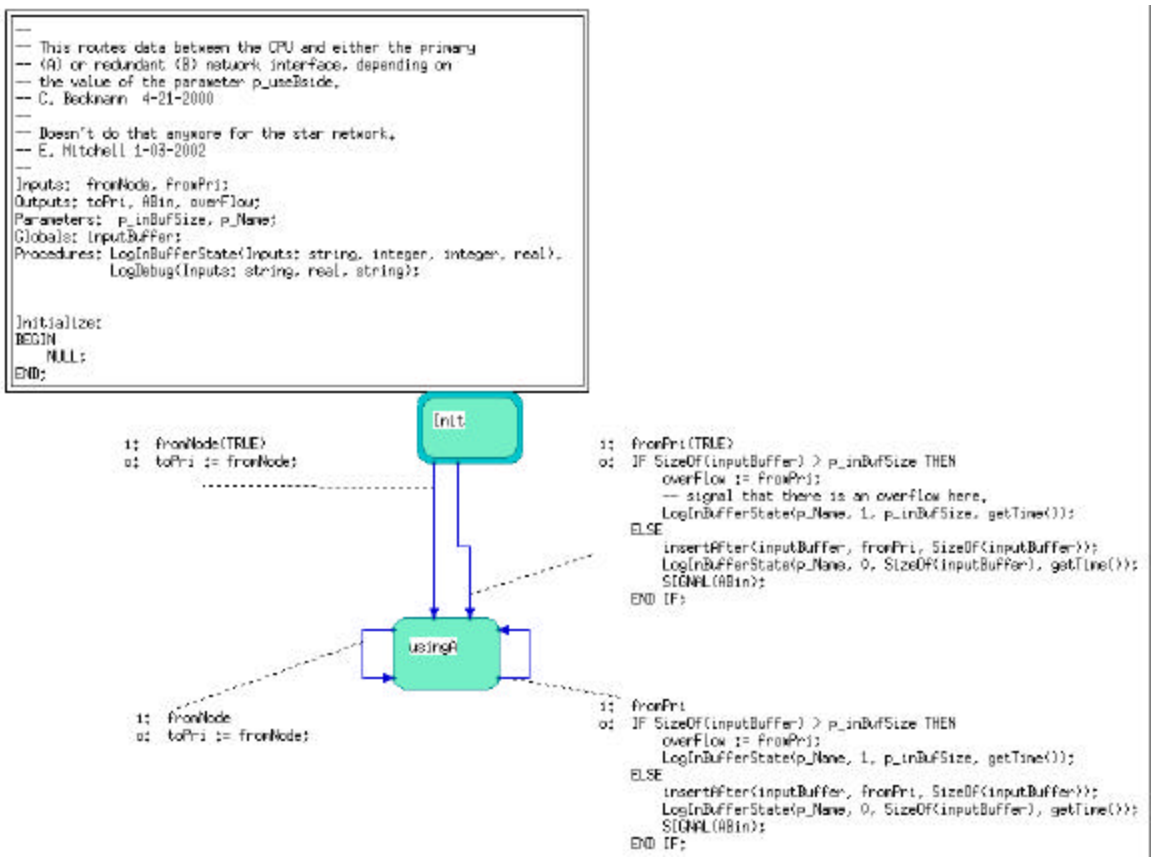
Star Optical Network Fabric (S_Fabric) Module Diagram:



Star Optical Network nextPktDoor Finite State Machine:



Star Optical Network ABselect Finite State Machine:



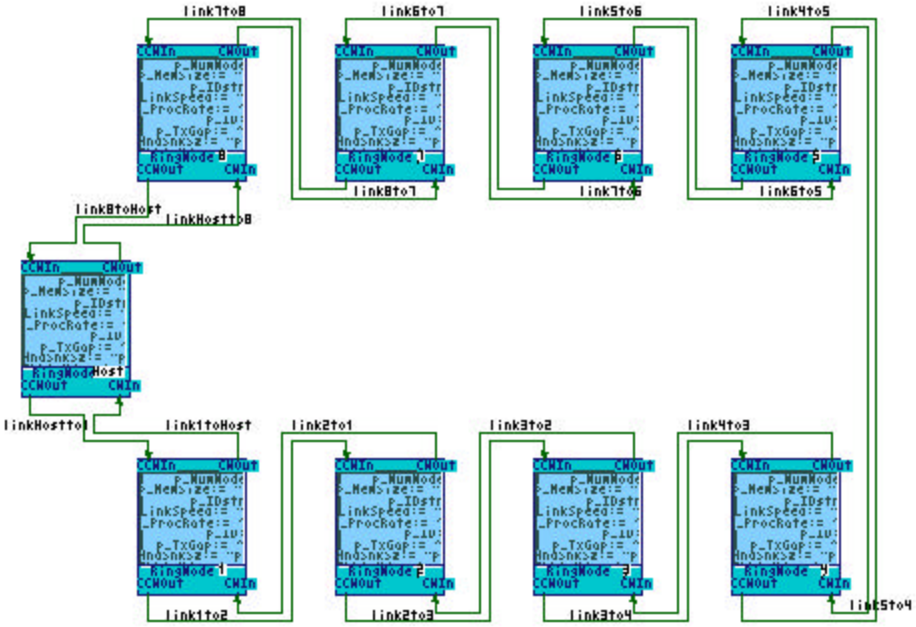
Memory State and Usage Logging Function callable from anywhere in the Hardware Model:

```
-- This logs memory data to the log file.
--
--Eric J. Mitchell 01/29/02
--

Inputs: Name, curTime, value ;
Globals: g_LogFile;
Locals: fstat;


Procedure:
BEGIN
    fstat := put(g_LogFile, curTime);
    fstat := put(g_LogFile, " Memory: ");
    fstat := put(g_LogFile, Name);
    fstat := put(g_LogFile, " ");
    fstat := put(g_LogFile, value);
    fstat := putline(g_LogFile, "");
END;
```

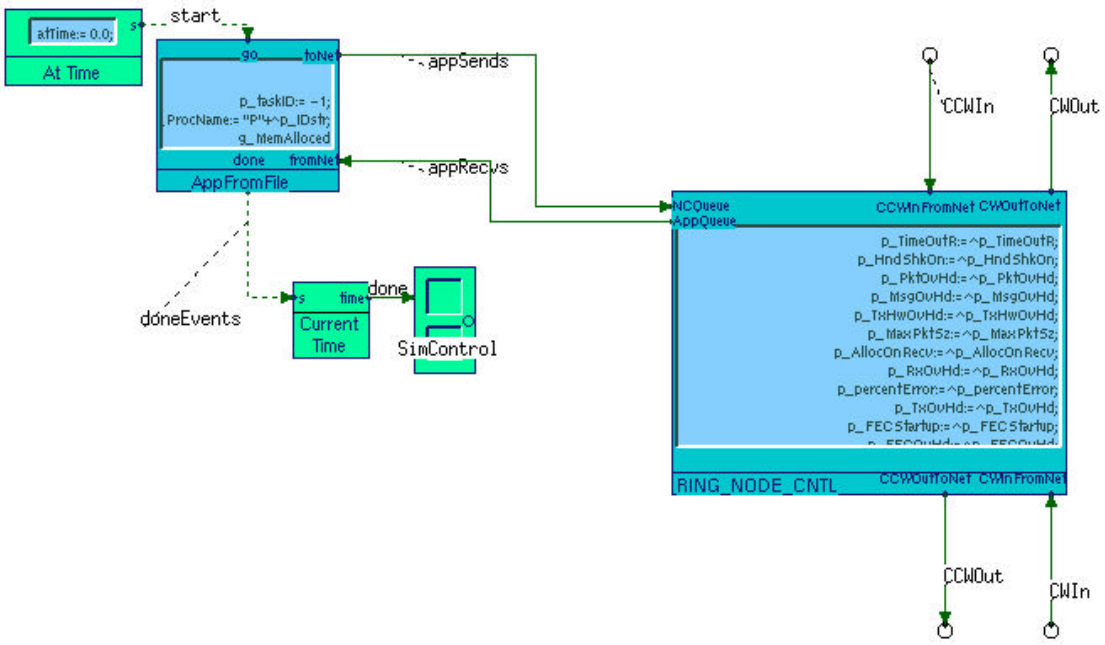
All other logging functions for message, packet, and link state have a similar form to this function.

Appendix B: Ring Optical Network System Diagrams

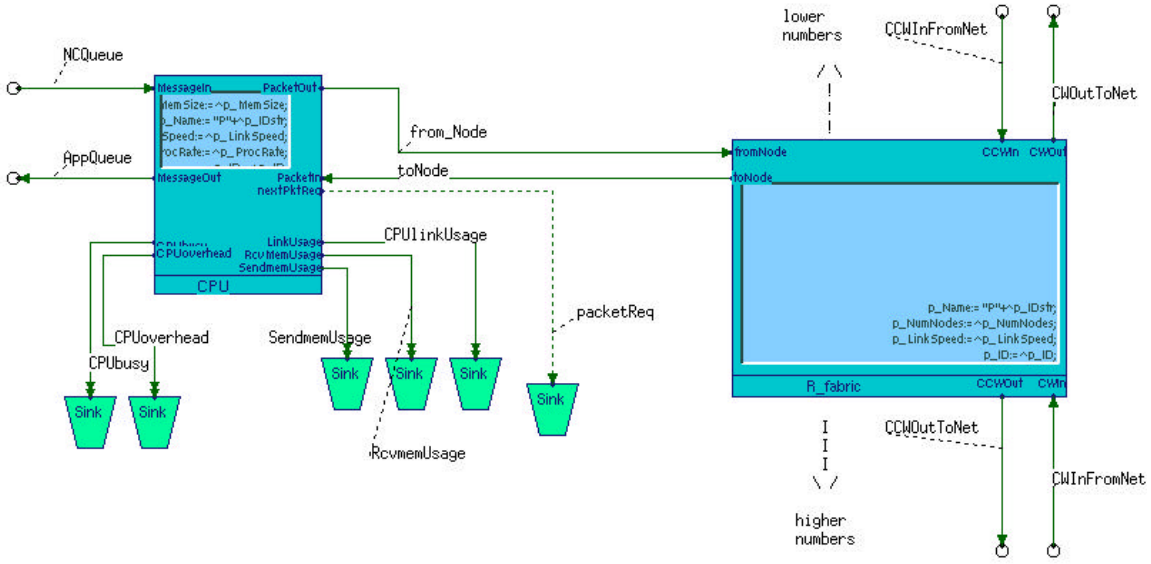
Top-Level Ring Optical Network System Diagram:



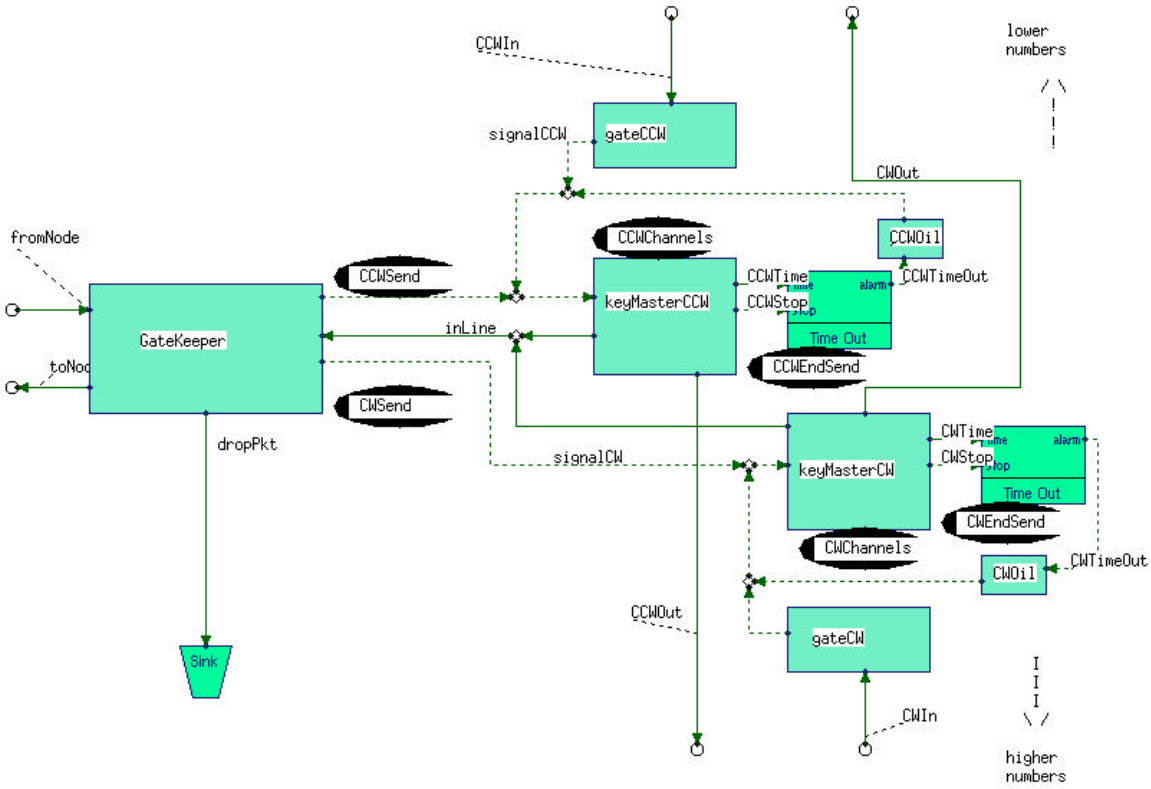
Ring Optical Network Node (RingNode) Diagram:



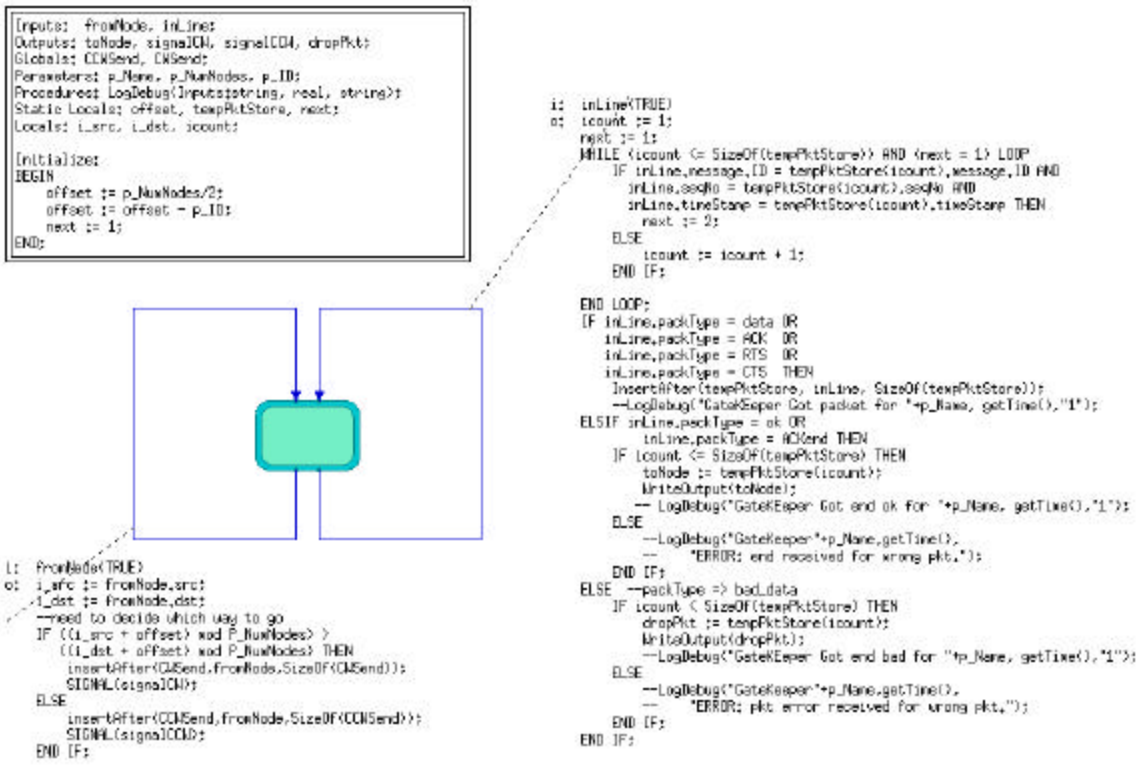
Ring Optical Network Node Control (RING_NODE_CNTL) Module Diagram:



Ring Optical Network Fabric (R_fabric) Moduel Diagram:



Ring Optical Network Gatekeeper Finite State Machine Diagram:



Ring Optical Network gateCCW Module function: (gateCW is the same in the opposite direction)

```
Inputs:   CCWIn;
Outputs:  signalCCW;
Globals:  CCWChannels;
Procedures: LogDebug(Inputs:string, real, string);
Parameters: p_Name;

Initialize:
BEGIN
    NULL;
END;

Procedure:
BEGIN
    insertAfter(CCWChannels,CCWin ,SizeOf(CCWChannels));
    SIGNAL(signalCCW);
    --LogDebug("gateCCW" + p_Name + " timeout ", getTime(), "2");
END;
```

Ring Optical Network keyMasterCCW Module function: (keyMasterCW is the same in the opposite direction)

```
--------------------------------------------------
-- This spec processes packets from both the
-- network and the node.  It controls collision
-- processing.  In addition this process controls link
-- usage.
--
-- CWW means the ring in the counterclockwise direction.  This also means the nodes
-- are numbered higher in the next nodes until it jumps to the lowest numbered node
-- The nodes must be based at node 0 up to NumNodes-1.
--
--  CCWUsage sets:
--      0 is array for each channel for passthrough errors, 0 ok, 1 error
--      1 is array for send is progress on that channel, 0 no send, 1 in progress
--      2 is array for send error, 0 ok, 1 error
--      3 is array for passthrough in progress, 0 no, 1 in progress
--
-- Eric J. Mitchell  01-15-02
--------------------------------------------------


Inputs:   signalCCW;
Outputs:  InLine, CCWOut, CCWTime, CCWStop;
Parameters:  p_NumNodes, p_LinkSpeed, p_ID, p_Name;
Globals:  CCWSend, CCWChannels, CCWEndSend;
Locals:  i_src, i_dst, pktTemp, s_src, s_dst, s_tst;
Static Locals: CCWEnd, sendTime, s_me, s_next, CCWUsage, s_sendDst;
Procedures: LogLinkState(Inputs: string, integer, real ),   --name, state, time
            LogDebug(Inputs: string, real, string),         --name, time, message
            integer_to_string(Inputs: integer; Outputs: string);

Initialize:
BEGIN
    integer_to_string(p_ID, s_me);
    integer_to_string((p_ID + 1) mod (p_NumNodes), s_next);
END;

Procedure:
BEGIN
--fired when either the node has a new packet, the network sends us a new packet or when the timer
-- runs out on a send from this node.

--First, check if there is a packet to send.  Look out for collisions.
IF SizeOf(CCWSend) > 0 THEN
    pktTemp := CCWSend(1);
    CCWEnd := pktTemp;  --store the last sent packet
    DeleteNth(CCWSend,1);
    i_src := pktTemp.src;
    i_dst := pktTemp.dst;
```

```
    integer_to_string(i_src, s_src);
    integer_to_string(i_dst, s_dst);
    s_sendDst := s_dst;
    CCWOut := pktTemp;
    CCWTime := ((8.0 * pktTemp.size)/p_LinkSpeed);
    WriteOutput(CCWOut);
    WriteOutput(CCWTime);
    sendTime := getTime() + ((8.0 * pktTemp.size)/p_LinkSpeed);

    --rebuilding a quick fix to make it seem as if control packets are on a seperate set of rings.  This
    --alteration is forced because there were major collision issues due with crossing messages.
    IF pktTemp.packtype = ACK THEN
        --LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 1, getTime());
    ELSE
        CCWUsage(1,i_dst) := 1;  --set node send to active
        IF CCWUsage(3, i_dst) >= 1 THEN  --check if there is a passthrough active
            --collision set all to collided.
            CCWUsage(0, i_dst) := 1;
            CCWUsage(2, i_dst) := 1;
            LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 4, getTime());
            LogLinkState(p_Name+"OutLink", 4, getTime());
           IF pktTemp.packType = bad_data THEN
               s_tst := "bad Data";
           ELSIF pktTemp.packType = ACK THEN
                s_tst := "ACK";
            ELSE
               s_tst := "other";
           END IF;
           LogDebug("Collision by a send pkt:", getTime(),
                       "From: "+s_src+" TO: "+s_dst+" Type: " +s_tst);
      ELSE
          LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 1, getTime());
      END IF;
    END IF;
END IF;


-- Next check to see that sent packet did not end.
WHILE SizeOf(CCWEndSend) > 0 LOOP
    DeleteNth(CCWEndSend, 1);

    --addition to fix ACKing issua
    IF CCWEnd.packtype = ACK THEN
        CCWEnd.packtype := ACKend;
        CCWOut := CCWEnd;
        WriteOutput(CCWOut);
         CCWUsage(1, CCWEnd.dst) := 0;
         LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_sendDst, 0, getTime());
    END IF;
    END IF;
END LOOP;


-- lastly, check the network for packets to forward.
WHILE SizeOf(CCWChannels) > 0 LOOP
    pktTemp := CCWChannels(1);
    DeleteNth(CCWChannels,1);
    i_src := pktTemp.src;
    i_dst := pktTemp.dst;
    integer_to_string(i_src, s_src);
    integer_to_string(i_dst, s_dst);
    --delay for the link difference
    delay(1.0/p_LinkSpeed);

    IF pktTemp.packType = data
    OR pktTemp.packType = RTS
    OR pktTemp.packType = CTS THEN
        If pktTemp.dst = p_ID THEN
            --its our packet
            InLine := pktTemp;
            WriteOutput(InLine);
```

```
    ELSE
        --pass the packet through, we are not sending.
        CCWOut := pktTemp;
        WriteOutput(CCWOut);
        CCWUsage(3,i_dst) := CCWUsage(3,i_dst)+1;   -- signal we are passing a pkt through
        IF CCWUsage(1,i_dst) = 1 THEN
            -- collision has occured between passing through packet and our send of a packet.
            -- Set link to 4 = error.
            CCWUsage(0,i_dst) := 1;
            CCWUsage(2,i_dst) := 1;
            LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 4, getTime());
        ELSE
             LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 1, getTime());
        END IF;
    END IF;
ELSIF pktTemp.packType = ACK THEN
    If pktTemp.dst = p_ID THEN
        --its our packet
        InLine := pktTemp;
        WriteOutput(InLine);
    ELSE
        CCWOut := pktTemp;
        WriteOutput(CCWOut);
        -- LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 1, getTime());
    END IF;
ELSIF pktTemp.packType = ok THEN    --this is where we handle end of Tx packets ok => good Tx.
    IF CCWUsage(0,i_dst) = 1 THEN    --collision here so change the token to bad_data => bad Tx.
        pktTemp.packType := bad_data;
        IF CCWUsage(1, i_dst) = 0 AND CCWUsage(3, i_dst) = 1 THEN
            CCWUsage(0, i_dst) := 0;
        END IF;
    END IF;
    IF pktTemp.dst = p_ID THEN
        InLine := pktTemp;
        WriteOutput(InLine);
    ELSE
        CCWOut := pktTemp;
        WriteOutput(CCWOut);
        CCWUsage(3, i_dst) := CCWUsage(3, i_dst)-1;
        IF CCWUsage(1, i_dst) = 1 THEN  --send is still in progress
            --leave in current state
        ELSE
            IF (CCWUsage(3,i_dst) = 0) THEN
                LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 0, getTime());
            ELSE
                --leave in current state
            END IF;
        END IF;
    END IF;
 ELSIF pktTemp.packtype = ACKend THEN
     IF pktTemp.dst = p_ID THEN
         InLine := pktTemp;
         WriteOutput(InLine);
     ELSE
         CCWOut := pktTemp;
         WriteOutput(CCWOut);
         --IF CCWUsage(1, i_dst) = 1 OR CCWUsage(3, i_dst) >= 1 THEN
         --    --leave channel as it currently is
         --ELSE
         --   --set the channel back to 0.
         --    LogLinkState("Link_"+s_me+"_to_"+s_next+":"+s_dst, 0, getTime());
         --END IF;
     END IF;
 ELSE --> pktTemp.packType = bad_data  --there was already a collision
     IF pktTemp.dst = p_ID THEN
         InLine := pktTemp;
         WriteOutput(InLine);
```

```
        ELSE
            CCWOut := pktTemp;
            WriteOutput(CCWOut);
            CCWUsage(3, i_dst) := CCWUsage(3, i_dst)-1;
            IF CCWUsage(1, i_dst) = 1 THEN  --send is still in progress
                --leave in current state
            ELSE
                IF (CCWUsage(3,i_dst) = 0) THEN
                    LogLinkState("Link_"+s_me+"_to_"+s_next+";"+s_dst, 0, getTime());
                ELSE
                    --leave in current state
                END IF;
            END IF;
        END IF;
    END IF;
END LOOP;
END;
```

Ring Optical Network CCWoil Moduel function: (Cwoil is the same in the opposite direction)

```
Inputs:   CCWTimeOut;
Outputs:  signalCCW;
Globals: CCWEndSend;
locals: filler = s_packet;
Procedures: LogDebug(Inputs:string, real, string);
Parameters: p_Name;

Initialize:
BEGIN
    NULL;
END;

Procedure:
BEGIN
    insertAfter(CCWEndSend,filler,SizeOf(CCWEndSend));
    SIGNAL(signalCCW);
    --LogDebug("CCWOil "+p_Name+" timeout", getTime(), "1");
END;
```

Appendix C: Debugging Software Model PERL Files

Packet Walk Debugging Software Model:

```
#
# Template for a pseudo-code software model
# that generates a software event trace when executed.

use GenTracev2;

my $paramFileName = ($#ARGV >= 0) ? $ARGV[0] : 'swmodel.swp';
my $outFileName   = ($#ARGV >= 1) ? $ARGV[1] : 'SWmodel.trc';
GenTraceSetup($paramFileName, $outFileName, $#ARGV >= 1);


#
# This one simply hops a packet around the nodes.  Send a message from
# 0 -> 1,
# then 1 -> 2, 2 -> 3 etc.
#
# BTW for the star, 0 = host, then there are 8 processing nodes 1
# through 8
#

## usage:      RECEIVE(destinationTaskID, sourceTaskID, msgnum);
## usage:      SEND(SourceTaskID, destinationTaskID, , msgnum,
## sizeInBytes);
## usage:      PROCESS(taskid, numberOfCycles);

      # Host node list
      SEND(0, 1, 100,15000);

      # Processor 1 list
      RECEIVE(1, 0, 100);
      PROCESS(1, rand(200));
      SEND(1, 2, 101, 15000);

      # Processor 2 list
      RECEIVE(2, 1, 101);
      PROCESS(2, rand(200));
      SEND(2, 3, 102, 15000);

      # Processor 3 list
      RECEIVE(3, 2, 102);
      PROCESS(3, rand(200));
      SEND(3, 4, 103, 15000);

      # Processor 4 list
      RECEIVE(4, 3, 103);
      PROCESS(4, rand(200));
      SEND(4, 5, 104, 15000);

      # Processor 5 list
      RECEIVE(5, 4, 104);
      PROCESS(5, rand(200));
```

```
SEND(5, 6, 105, 15000);

# Processor 6 list
RECEIVE(6, 5, 105);
PROCESS(6, rand(200));
SEND(6, 7, 106, 15000);

# Processor 7 list
RECEIVE(7, 6, 106);
PROCESS(7, rand(200));
SEND(7, 8, 107, 15000);

# Processor 8 list
RECEIVE(8, 7, 107);
PROCESS(8, rand(200));

#
# Stop all tasks

STOP(0);
STOP(1);
STOP(2);
STOP(3);
STOP(4);
STOP(5);
STOP(6);
STOP(7);
STOP(8);
```

Congestion to Host Debugging Software Model:

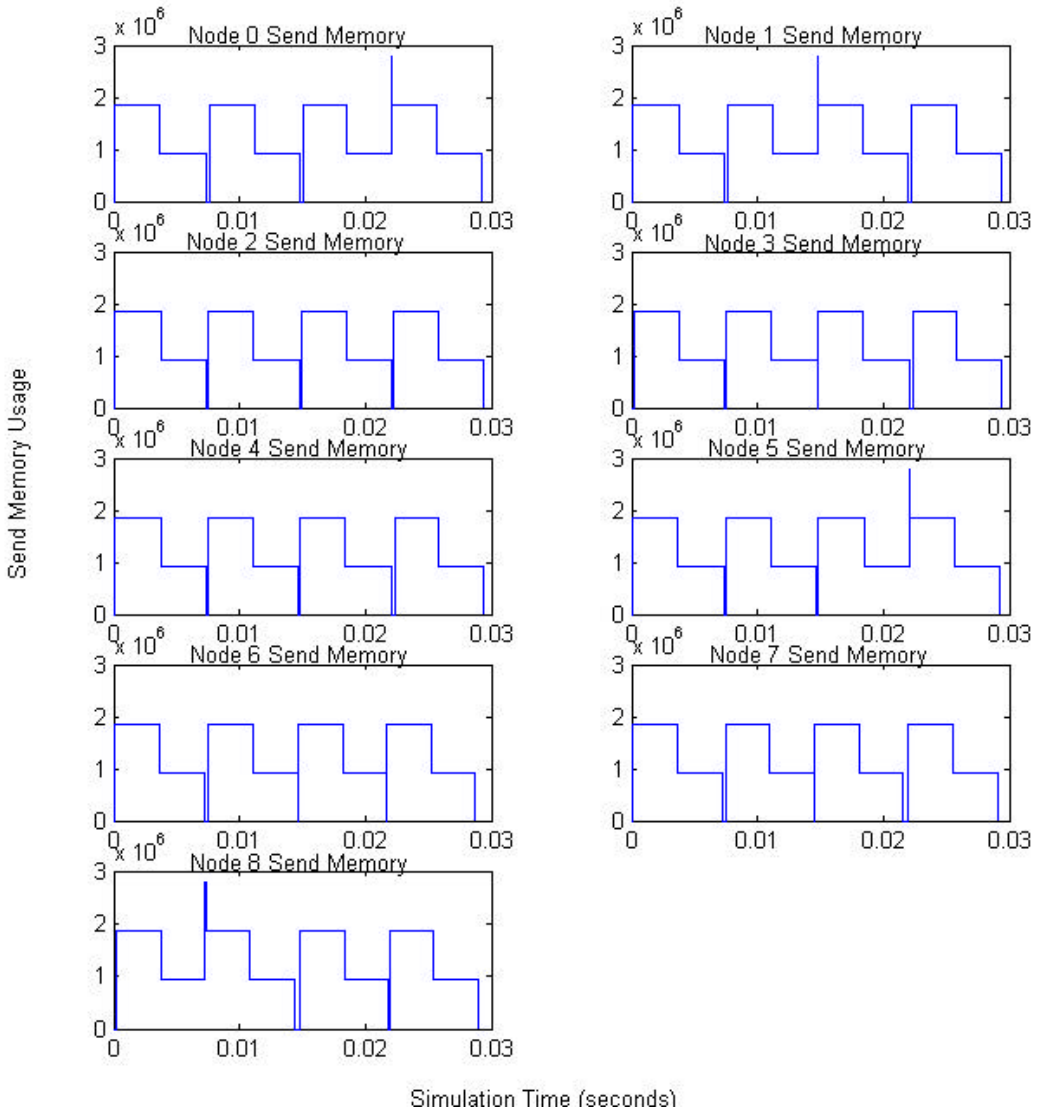
```
#
# Template for a pseudo-code software model
# that generates a software event trace when executed.


use GenTracev2;

my $paramFileName = ($#ARGV >= 0) ? $ARGV[0] : 'swmodel.swp'
my $outFileName   = ($#ARGV >= 1) ? $ARGV[1] : 'SWmodel.trc';
GenTraceSetup($paramFileName, $outFileName, $#ARGV >= 1);


## usage:      RECEIVE(destinationTaskID, sourceTaskID, msgnum);
## usage:      SEND(SourceTaskID, destinationTaskID, , msgnum,
## sizeInBytes);
## usage:      PROCESS(taskid, numberOfCycles);



      # Host node list
      RECEIVE(1, 0, 100);
      RECEIVE(1, 2, 200);
      RECEIVE(1, 3, 300);
      RECEIVE(1, 4, 400);
      RECEIVE(1, 5, 500);
      RECEIVE(1, 6, 600);
      RECEIVE(1, 7, 700);
      RECEIVE(1, 8, 800);

      #Other, all sends to 0
      SEND(0, 1, 100, 15000);
      SEND(2, 1, 200, 15000);
      SEND(3, 1, 300, 15000);
      SEND(4, 1, 400, 15000);
      SEND(5, 1, 500, 15000);
      SEND(6, 1, 600, 15000);
      SEND(7, 1, 700, 15000);
      SEND(8, 1, 800, 15000);


      #
      # Stop all tasks

      STOP(0);
      STOP(1);
      STOP(2);
      STOP(3);
      STOP(4);
      STOP(5);
      STOP(6);
      STOP(7);
      STOP(8);
```

Two Node Dialogue Debugging Software Model:

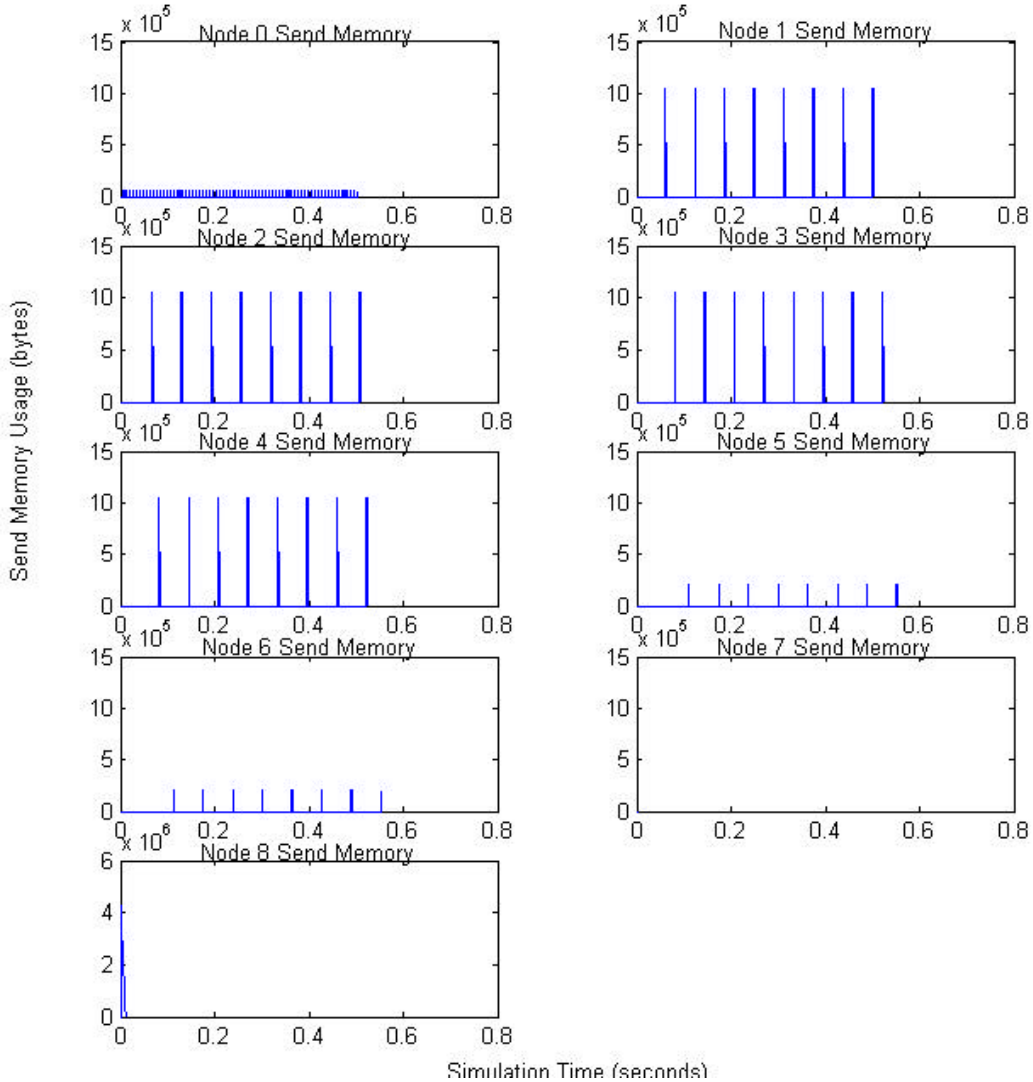
```
#
# Template for a pseudo-code software model
# that generates a software event trace when executed.


use GenTracev2;

my $paramFileName = ($#ARGV >= 0) ? $ARGV[0] : 'swmodel.swp';
my  $outFileName      =  ($#ARGV  >=  1)  ?  $ARGV[1]  :  'SWmodel.trc';
GenTraceSetup($paramFileName, $outFileName, $#ARGV >= 1);


## usage:      RECEIVE(destinationTaskID, sourceTaskID, msgnum);
## usage:      SEND(SourceTaskID, destinationTaskID, , msgnum,
## sizeInBytes);
## usage:      PROCESS(taskid, numberOfCycles);

      # Host node list
      SEND(0, 1, 100,15000);
      SEND(0, 1, 200, 17000);

      # Processor 1 list
      RECEIVE(1, 0, 100);
      PROCESS(1, rand(200));
      RECEIVE(1,0,200);
      #
      # Stop all tasks

      STOP(0);
      STOP(1);
      STOP(2);
      STOP(3);
      STOP(4);
      STOP(5);
      STOP(6);
      STOP(7);
      STOP(8);
```

Appendix D: Send Memory Usage Diagrams for the three Software models.

Space-Based Radar Corner Turn Send Memory Usage for all nodes



Synthetic Aperture Radar Send Memory usage for all nodes



Random Workload Send Memory usage for all nodes