

Development of an Isotach Simulation Tool for use in Performance Analysis

A Thesis
in TCC 402

Presented to

The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

by

Consuela Y. Toye

March 24, 1997

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in Humanities Courses.

Author Signature: Consuela Yvonne Toye

Approved: _____
Technical Advisor: Paul F. Reynolds, Jr.

Approved: _____
TCC 402 Advisor: Roseanne Welker

Table of Contents

	<u>Page</u>
Table of Figures.....	ii.
Table of Tables.....	iii.
Glossary of Terms.....	iv.
Abstract.....	vi.
1.0 Introduction.....	1
1.1 Project Summary.....	1
1.2 Project Context.....	1
1.3 Project Rationale.....	2
1.4 Thesis Report Overview.....	2
2.0 The Isotach Project.....	3
2.1 Parallel Computing.....	3
2.2 Isotach Logical Time.....	4
2.3 Isotach Networks.....	4
2.4 Current Isotach Projects.....	5
3.0 The Isotach Simulation Tool.....	6
3.1 Specification.....	6
3.1.1 SES/Workbench.....	6
3.1.2 Model.....	6
3.2 Implementation.....	8
3.2.1 Definitions & Flow Control.....	8
3.2.2 The Network Components.....	8
3.2.3 The Component Palette.....	14
4.0 Results & Recommendations.....	16
4.1 The Isotach Simulation Tool.....	16
4.2 User's Manual.....	16
4.3 Overall Assessment.....	16
4.3 Recommendations for Future Work.....	17
Bibliography.....	18
Appendix A -- User's Manual.....	19

Table of Figures

	<u>Page</u>
Figure 1: Token Distribution Process.....	7
Figure 2: Token Generator in SES/Workbench.....	8
Figure 3: Token Manager Implementation in SES/Workbench.....	9
Figure 4: Switch Implementation in SES/Workbench.....	11
Figure 5: Host Generator in SES/Worbench.....	12
Figure 6: Host Implementation in SES/Workbench.....	13
Figure 7: The Component Palette-- Isotach Simulation Tool.....	15

Table of Tables

	<u>Page</u>
Table 1: Port Assignments at Token_Manager_Input.....	10
Table 2: Determination of Array Assignment in Block Node.....	11
Table 3: Node Assignments at Switch_Delay[] Node of Size N.....	12

Glossary of Terms

Algorithm: A systematic procedure guaranteed to produce a result after a finite number of steps.

Bandwidth: The amount of data that can be sent through a given communication circuit per second.

Deadlock: A situation wherein two or more processors are unable to proceed because each is waiting for one of the others to do something.

FIFO: A hardware buffer from which items are taken out in the same order they were put in, i.e. First In, First Out.

Livelock: This differs from deadlock in that the process is not blocked or waiting for anything, but has a virtually infinite amount of work to do and can never catch up.

Local Area Network(LAN): A data communications network which is geographically limited (typically to a 1km radius) allowing easy interconnection of terminals, microprocessors, and computers within adjacent buildings. Ethernet and FDDI are examples of standard LANs.

Logical Time System: A set of constraints on the way in which events are ordered, i.e. assigned logical times.

Model: A description of observed behavior. Models allow complex systems to be understood and their behavior predicted within the scope of the model. A model may be used as the basis for simulation.

Network: Hardware and software data communication systems.

Network Topology: A network topology shows the hosts and the links between them. A network

layer must stay abreast of the current network topology to be able to route packets to their final destination.

Palette: A set of icons that represent something (a file, directory or action) in a graphical user interface. These icons can then be included in models by dragging and dropping them onto a workspace.

Parallel Computing: The simultaneous use of more than one computer to solve a problem. There are many different kinds of parallel computer (or “parallel processor”).

Prototype: A model of a product.

Switch Interface Unit(SIU): An interface between a switch and a host.

Abstract

The Isotach project is a research project in the Computer Science Department at the University of Virginia. Work on the Isotach project began in the late '80s. Since then, the Isotach project has made much progress due to funding from government agencies, such as the Defense Advanced Research Projects Agency (DARPA) and National Science Foundation (NSF). Recently, the Isotach research group has received funding to implement a hardware prototype of Isotach networks. With this new project, the urgency arose to collect performance statistics on Isotach networks. This data would be used to create and implement an effective hardware prototype.

This thesis provides the researchers working on the Isotach project with a tool to graphically create, animate, and analyze Isotach networks. The Isotach Simulation Tool developed in this thesis allows a research to collect performance data with ease. This tool eliminates the need for the researcher to implement the individual components of a network and design the network as well. The research only needs to design the network topology. The Isotach Simulation Tool provides all of the necessary components to design the network. This allows the researcher to collect performance data in a shorter amount of time.

The Isotach Simulation Tool was developed in SES/Workbench, a design specification, modeling, and simulation tool. This tool was chosen because of its design and animation capabilities. The Isotach Simulation Tool is a component palette containing all of the necessary components to create an Isotach network. These components include a switch, token manager, and host. These components when interconnected to create a network will implement the isonet algorithm that defines an Isotach network. The complete product presented in this thesis includes a component palette and user's manual. The user's manual was developed to guide the user in the development of an Isotach network utilizing the component palette.

1.0. Introduction

1.1 Project Summary

The Isotach project is a research project in the Computer Science Department at the University of Virginia. This thesis provides the researchers working on the Isotach project with a tool to graphically create, animate, and analyze Isotach networks. The complete product presented in this thesis includes a component palette and user's manual. The component palette contains model representations of all components essential in creating an Isotach network of arbitrary topology. To supplement the component palette, the author published an user's manual to guide the user in the development of an Isotach network utilizing the component palette.

1.2 Project Context

This project can be considered one of many stages in the research and development of the Isotach project. Ms Craig Williams began work on Isotach in her dissertation work with the help of Professor Paul F. Reynolds in the late '80s. Funding for the Isotach project came from many government agencies, such as the Defense Advanced Research Projects Agency (DARPA) and National Science Foundation (NSF). This funding allowed for more research into how Isotach networks behave and its applications. Isotach is currently running in software form over a Myrinet, a gigabit-per-second local area network, connected network of Intel PC's and is being installed at several facilities. In addition, the Isotach research group has received funding to implement a hardware prototype of Isotach networks. Many more research opportunities are expected in the Isotach project.

1.3 Project Rationale

Isotach networks was created as a solution to the problem of synchronization in parallel systems. It proposes to reduce the costs of synchronization in parallel computations. "Few existing networks offer ordering guarantees stronger than FIFO [First In, First Out] delivery order among messages with the same source and destination"(Reynolds, "Isotach Networks"). As a result, a process must use delays, locks, or some other synchronization mechanism to ensure that execution is consistent with the program's constraints. Isotach networks intend to offer a stronger guarantee about ordering by implementing a logical time system. This logical time system allows processes to predict and control the logical times at which processors receive their messages. This method of synchronization improves processor performance, because the delay associated is decreased.

To provide evidence of Isotach networks' capabilities, an extensive performance analysis must be conducted. This thesis focuses on the development of a simulation model of Isotach networks that facilitates the collection of performance data. This project does not provide such data, but provides a tool allowing such data to be collected with ease. With this product, more detailed information about Isotach networks can be gathered. Thus, a more accurate and effective hardware prototype of Isotach networks can be created.

1.4 Thesis Report Overview

The remainder of this report begins by presenting background information on the Isotach project. This information leaves the reader with a basic understanding of what Isotach networks are and how they behave. The paper will then proceed to describe the finished product in detail. Discussion of the specification and implementation of the components contained in the component palette will be given in detail. The next chapter presents a final summary of the component palette and user's manual. Finally, the author states recommendations for future work.

2.0 The Isotach Project

2.1 Parallel Computing

“Parallelism has been part of the computing scene from the beginning, but in the second half-century of modern computing, it appears destined to become the central architectural feature.”(Kuck, 1996)

For over three decades, parallel computing has offered the highest performance levels available in computing. Parallel processors can speed up computations by simultaneously performing operations that a sequential computer would perform serially. In the 1960s, parallel systems used the highest level of technology available. Therefore, the systems were difficult to manufacture, expensive, and hence rare. In the decades since, the technology used in parallel systems has gradually simplified. This has opened the door for more parallel systems to be built. Yet, even with all of these advancements, parallel systems have not achieved their full potential.

The failure of parallel systems to achieve their full potential is typically associated with memory bandwidth or interprocessor communication.(Kuck, 1996) The inability to achieve their full processing power lies in the overhead caused by waiting for data to be retrieved from memory or for synchronization of signals. The Isotach project focuses on interprocessor communication and the synchronization of signals. In most parallel systems, the synchronization mechanism of choice is a lock. However, locks include delays incurred for lock maintenance, unnecessarily restricted access to variables, and the care that must be taken to avoid deadlock and livelock(Reynolds, “Isotach Networks”). Instead of such synchronization techniques as locks, Isotach networks implement a logical time system to control the synchronization of signals between processors.

2.2 Isotach Logical Time System

A logical time system is a set of constraints on the way in which things are ordered or assigned logical times. The logical time system assigns times to messages detailing when to send and receive them. The assignment of these times is related to the distance that the message must travel to reach its destination. The logical time system allows processes to control the logical time at which their messages are received. A process knows the distance the message must travel and can therefore determine the logical times to send and receive the message. This system does not incur the delays involved with conventional methods of synchronization.

The Isotach logical time system is based on the logical time system defined by Lamport in a paper on ordering events in a distributed system. Lamport’s system is concerned with the sending and receiving of messages. Times are assigned to events based on the *happened before* relation.

Event A happened before event B if it one of three conditions is satisfied: 1) *A* and *B* occurred in the same process and *A* occurs before *B*; 2) *A* is the event of sending a message *M* and *B* is the event of receiving message *M*; 3) there exists some event *C* where *A* *happened before C* and *C* *happened before B*. In Lamport’s system if *A* *happened before B*, the logical time at *A* must be less than the logical time at *B*. The Isotach logical time system extends Lamport’s system in that the times must be consistent with the Isotach invariant as well as the *happened before* relation.

The Isotach invariant states that number of pulses or signals that it takes for a message to be

received must be equivalent to the distance the message must travel to reach its final destination.

2.3 Isotach Networks

An Isotach network is defined as a network that implements the Isotach logical time system. In other words, it is a network that delivers messages in an order that allows sending and receiving events to be assigned logical times consistent with the *happened before* relation and the Isotach invariant. The creators of Isotach networks have developed an algorithm for implementing isotach logical time on a network of arbitrary topology. This algorithm is known as the isonet algorithm. Below is a description of the isonet algorithm:

Each SIU and switch maintain a logical clock, a counter initialized to zero, that gives the pulse component of the local logical time. The clocks are kept loosely synchronized by the exchange of tokens. A token is a signal sent to mark the end of one pulse of logical time and the beginning of the next. Initially each switch sends a token wave, i.e. it sends a token on each output, including the outputs to adjacent SIU's, if any. Thereafter, each switch sends a token wave $i + 1$ upon receiving the i th token on each input, including the inputs from adjacent SIU's. Each time a switch sends a token wave, it increments its clock. When an SIU receives a token, the SIU increments its clock and returns the token to the switch.(emphasis in original) (Reynolds, "Isotach Networks")

2.4 Current Isotach Projects

The Isotach project has made much progress over the years and has many opportunities for further research. An Isotach network is currently operating in software form on a Myrinet, a gigabit-per-second local area network, connected network of Intel PC's. In addition, the system is being installed at several facilities. The largest project to date is the prototyping effort being conducted in conjunction with the Electrical Engineering Department at the University of Virginia. This effort is to develop a hardware implementation of Isotach networks. This thesis project will aid in the research and development of this prototype. For the future, research may be conducted into cache coherence, distributed simulations, and a possible implementation of Isotach networks closer to the processor.

3.0 The Isotach Simulation Tool

3.1 Specification

3.1.1 SES/Workbench

SES/Workbench is a design specification, modeling, and simulation tool that is useful in constructing and evaluating system designs. SES/Workbench was chosen for the simulation tool for several reasons. First and foremost for its availability and the developers familiarity with it. In addition, SES/Workbench works well for evaluating complex systems involving a high degree of concurrent processing. The most important aspect in the decision to use SES/Workbench was its animation capabilities. SES/Workbench allows for the design and simulation of a model to occur in one space almost simultaneously. The user of the model would be able to create their model, collect data on their model, and animate their model using one program.

3.1.2 Model

The model to be created would represent a network of arbitrary topology consisting of interconnected nodes in which each node is either a switch or a host. Adjacent nodes communicate over FIFO links. Each host has a network interface called a switch interface unit (SIU); and each switch has a token manager to handle the flow of tokens through the network. A token manager will send a wave of tokens to all of the adjacent nodes. It will then wait until it receives tokens from all of its adjacent nodes before releasing another wave of tokens. *Figure 1* on the following page illustrates this property.

The components necessary in a model of an Isotach network consist of the following: a switch, token manager, and a host. For future research, we limited the sizes of a switch to be 4-, 8-, 16-, and 32-ports. A token manager will always occupy one port on a switch with the remaining ports being occupied by either another switch or a host.

3.2 Implementation

3.2.1 Definitions and Flow Control

In SES/Workbench, transactions are the basic entities representing execution processes. For this model, a transaction would represent either a token or a message. Every transaction has four inherent, accessible characteristics in its state. These characteristics are its identifier, category, phase, and port. For the remainder of this discussion, a transaction with category token will be referred to as a token, and transaction's with a category message will be referred to as a message. This model will use the characteristics category, phase, and port of a transaction's current state to direct its flow throughout the model.

3.2.2. The Network Components

Token Generator

Purpose: To generate tokens at the switch

General Algorithm:

```
if <port_number i> connected to HOST
    generate token with phase HOST_i, port H_i
else if <port_number i> connected to SWITCH
    generate token with phase my_switch, port s_i;
```

Actual Implementation:

Node Name: TokenX_generator

Node Type: Source Node

Figure 2: Token Generator in SES/Workbench

A source node was used to represent the generation of tokens from the token manager. A source node in SES/Workbench creates transactions in a model periodically according to an arbitrary user-defined algorithm. The code placed in the method body of this node implements the general algorithm previously described. The user is asked to enter parameter values that will give the model information about the topology. The node will then generate the necessary tokens and send them to the token manager. These tokens are generated only once at the start of the simulation and are recycled for the duration of the simulation. The number of tokens generated varies based on the size of the switch for which the generator is associated.

Token Manager

Purpose: To send and receive tokens

General Algorithm:

```
if <phase_number i>
    set <port_number i>;
fixed_delay;
wait_until (next_queue empty);
move into next queue;
wait_until (queue is full);
release all tokens in queue;
```

Actual Implementation:

Submodel Name: TokenX_Manager

Node Types: Enter Node, Delay Node, arrayed Block Nodes, Interrupt Node, Exit Node

Figure 3: Token Manager Implementation in SES/Workbench

A submodel was designated to represent the token manager because of the complexity involved with its implementation. The above algorithm is implemented in this submodel. A token would enter the submodel from either a token_generator or a switch. The method of the enter node, labeled token_manager_input, would assign a port number to a token based on its current phase, with the exception of tokens whose destination is a switch (*see Table 1*). Those tokens whose destination is a switch are designated with phase my_switch or foreign. A token with phase my_switch is not affected at the node token_manager_input, however a token with phase foreign has its phase changed to my_switch at the node token_manager_input. These phase designations become important when directing control from the switch.

Table 1: Port Assignment at Token_Manager_Input

Token's Current Phase	Port Assignment
Host_1	H1
Host_2	H2
Host_3	H3

The token then moves on to the delay node, token_delay. This node represents the inherent physical delay that would occur. A delay node was chosen here to ensure that each token would incur a delay that was independent of the other tokens in the submodel. After the delay, the token moves to the first block node, block_for_next_wave[]. Block_for_next_wave is an array of block nodes for independent processing of tokens. This is essential to ensure that the algorithm is implemented correctly. At this node, the token will wait in its designated block node until the next block node, block_for_current_wave[] is empty. Also, this node is an array of block nodes. Tokens are assigned to an element of the arrayed block node based on their port numbers, *see Table 2*. This method of assignment is used for both arrayed block nodes. When arriving at block_for_current_wave, the token will wait until all elements of the arrayed block node are occupied. At that point, the last token to enter the block node will move on to the interrupt node, release_current_wave. At this node, all tokens in the node block_for_current_wave are released. The tokens then exit the submodel through the exit node, token_manager_output, to move to the switch. The only difference in the token manager for different size switches in the component palette is the comparison done in the method body of the node, block_for_current_wave and the

method body of the node, `release_current_wave`, because they are dependent on switch size.

Table 2: Determination of Array Assignment in Block Node

Port	Block Node Array Index
HXor sX	X

Switch

Purpose: To direct messages and tokens throughout the network

General Algorithm:

fixed_delay;
release token on designated <port_number_i>;

Actual Implementation:

Node Name: Switch_XxX

Node Types: Enter Node, arrayed Service Node, Exit Node

Figure 4: Switch Implementation in SES/Workbench

As with the token manager, a submodel was chosen to represent a switch. A token would enter a switch from a host or a token manager. Upon entering the switch, the token moves to the service node, `Switch_Delay[]`. The service node represents the physical delay incurred at a switch. The token then waits in its designated service node for a specific time interval. A token is assigned a designated node in the array based on its phase and port assignments, *see Table 3*.

Table 3: Node Assignments at Switch_Delay[] Node of Size N

Phase	Port	Switch_Delay [N] node index
Host_X	HX	X-1
my_switch	sY	Y-1
foreign		N-1
	token_return	N-1

Once its service time has expired, the token will exit the switch to return to the token manager or to a host.

Host Generator

Purpose: To generate tokens and messages at the host

General Algorithm:

generate token;
exponentially generate messages;

Actual Implementation

Node Name: Host_generator

Node Type: Source Node

Figure 5: Host Generator in SES/Workbench

A source node was used to represent the generation of tokens from a host as in the token generator described previously. The node will generate one token and send it to the host. This token is generated only once at the start of the simulation and is recycled along with the other tokens for the duration of the simulation. In addition to tokens, this node also generates messages to emulate messages traveling through an actual network. These messages are generated at an exponential rate and are destroyed when they reach their final destination.

Host

Purpose: To represent hosts in a network

General Algorithm:

fixed delay;
if token
 release token;
else if message
 if sending
 release message;
 else
 process message;

Actual Implementation

Submodel Name: Host

Node Types: Enter Node, Delay Node, Exit Node, Sink Node

Figure 6: Host Implementation in SES/Workbench

As with the switch and the token manager, a submodel was chosen to represent a host. Upon entering a host from a switch or the host_generator, the method body of the enter node, host_input, is executed and the phase and port values for a token are set. These values are dependent on the port at which the host is attached to a switch. If the host is attached to port X of a switch, then the token is assigned phase host_X and port HX. A message is not altered at the node, host_input. Both hosts and messages move on to the delay node, siu_turnaround, at which they incur a delay representing the physical delay incurred at a host. A token then exits the host. At host_output, its port is changed to token_return. A message, if its port is send, exits the host.

generate token;
exponentially generate messages;

Actual Implementation

Node Name: Host_generator

Node Type: Source Node

Figure 5: Host Generator in SES/Workbench

A source node was used to represent the generation of tokens from a host as in the token generator described previously. The node will generate one token and send it to the host. This token is generated only once at the start of the simulation and is recycled along with the other tokens for the duration of the simulation. In addition to tokens, this node also generates messages to emulate messages traveling through an actual network. These messages are generated at an exponential rate and are destroyed when they reach their final destination.

Host

Purpose: To represent hosts in a network

General Algorithm:

```

fixed delay;
if token
    release token;
else if message
    if sending
        release message;
    else
        process message;

```

Actual Implementation

Submodel Name: Host

Node Types: Enter Node, Delay Node, Exit Node, Sink Node

Figure 6: Host Implementation in SES/Workbench

As with the switch and the token manager, a submodel was chosen to represent a host. Upon entering a host from a switch or the host_generator, the method body of the enter node, host_input, is executed and the phase and port values for a token are set. These values are dependent on the port at which the host is attached to a switch. If the host is attached to port X of a switch, then the token is assigned phase host_X and port HX. A message is not altered at the node, host_input. Both hosts and messages move on to the delay node, siu_turnaround, at which they incur a delay representing the physical delay incurred at a host. A token then exits the host. At host_output, its port is changed to token_return. A message, if its port is send, exits the host.

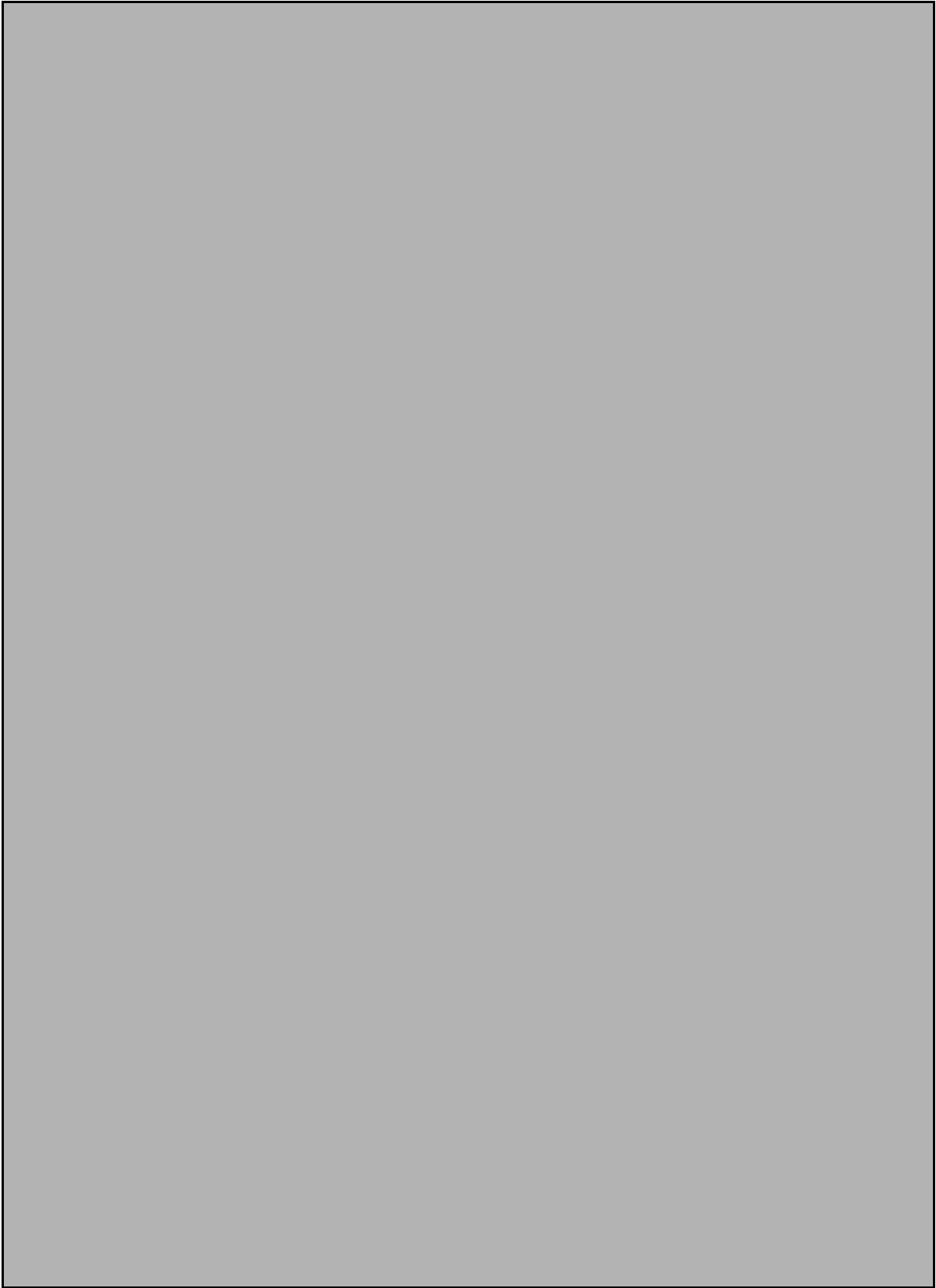
At host_output, its port is changed to received. A message, if its port is received, moves to the sink node, processor. Since messages are not recycled in this model, a sink node was used to destroy or process the message.

3.2.3 The Components Palette

The components palette is the primary product resulting from this project. The component palette consists of all the components described above with variations. On the palette, the components are grouped together by the size of the switch. All components that are needed to accurately represent a 4-port switch are included in the palette and grouped together. The development of a palette allows users to select various components from the palette and include them in their model. This saves researchers time by eliminating the need to implement individual components. This time may then be used in gathering important performance data.

To aid in the collection of important performance data, various size switches were implemented. A switch size is defined as the number of input/output ports of a switch. This palette includes components to implement 4-port, 8-port, 16-port, and 32-port switches. These variations will allow the user to create a wide range of network topologies. Ultimately, this flexibility will allow for extensive performance data to be collected. The component palette is displayed in *Figure 7* on the following page.

Figure 7: The Component Palette -- Isotach Simulation Tool



4.0 Results & Recommendations

4.1 The Isotach Simulation Tool

The Isotach Simulation Tool consists of a component palette and graphical user interface. The graphical user interface is inherent in SES/Workbench. The component palette consists of all of the components needed to accurately create and animate an Isotach network of any topology. This palette gives users a graphical means for creating an Isotach network and viewing its behavior. At the same time, it eliminates the need for the researcher to implement the components necessary to create an Isotach network. Therefore, this palette is an essential tool for researching Isotach networks. It facilitates the gathering of research data on Isotach networks with ease and accuracy.

4.2 The User's Manual

The user's manual is a supplement to the Isotach Simulation Tool. This manual serves as a guide on how to successfully create an Isotach network using the component palette in SES/Workbench. Step-by-step instructions on what is necessary to create an Isotach network are provided in this document. In creating this manual, the author assumed the primary user of this product would be individuals familiar with Isotach networks and SES/Workbench. Therefore, the focus was placed on the technical aspects of the usage and interactions of the components in the palette. The complete user's manual is located in *Appendix A*.

4.3 Overall Assessment

The product resulting from this thesis serves its intended purpose. It is a well-developed product that will prove useful for future projects. It gives the user the capability to create a complex system in very little time. This will give the user more time to focus on the actual use of the system rather than the design of the system. Even with all of the advantages made available by with this product, there is still room for improvement.

The product was designed with limitations. Some of these limitations include the elimination of some physical properties of hosts and/or switches that were not implemented in this system. These properties were not essential in the development of this system or for the gathering of necessary data. However, they may become important in the future. Also, assumptions were made as to the individuals that would use this product. The users were assumed to be researchers, preferably graduate students, who had a basic understanding of Isotach networks and SES/Workbench. These assumptions and limitations are some of the disadvantages of this product.

4.4 Recommendations for Future Work

Currently, this model serves its intended purpose. Yet, there is still some more work to be done. Possible projects include conducting extensive user testing to determine the usability of the model. Also, an evaluation or examination of the model to determine if changes may be made to its implementation to make the model more usable for individuals who may not have a competent background in SES/Workbench. As with the actual model, the user's manual should undergo sufficient user testing. This testing would show how helpful and accurate the user's manual is to an individual working with the component palette. This information could be very useful to future users of the model or individuals maintaining the model.

The model itself is designed to be used in further research of the Isotach networks. Possible projects include the evaluation of the model with respect to the ease in collecting statistical infor-

mation, i.e. how the model aids or hinders the researcher in the collection of performance data. Also, evaluating how aesthetically appealing the model is to a user or how the model gives the user a better understanding of the behavior of Isotach networks. These projects may provide information on the effectiveness of using SES/Workbench to model Isotach networks. As usage of the model increases, other areas for improvement are destined to arise.

Bibliography

Boden, N.J., Cohen, R.E. Felderman, Kulawik, A.E., Seitz, C.L., Seizovic, J. ng, and Su, W. (1995, February). "Myrinet-- Gigabit-per-Second Local Area Network". IEEE Micro 15, 1, pp. 29-36.

Kuck, David J. (1996). High Performance Computing: Challenges for Future Systems. New York: Oxford Press.

Lamport, L. (1978, July). "Time, Clocks, and Ordering of Events in a Distributed System". Communications of the Association for Computing Machinery. pp. 558-563.

Lewis, Ted G. and El-Rewini, Hesham (1992). Introduction to Parallel Computing. New Jersey, Siman and Schuster.

Reynolds, Paul F. Jr., Williams, C., and Wagner, R.R. (1997, April). "Isotach Networks". IEEE Transactions on Parallel Distributed Systems

SES/Workbench (1995). Austin, TX: Scientific and Engineering Software, Inc.

Appendix A The Isotach Simulation Tool User's Manual

by
Consuela Toye
version 1.0