Final report

Study of Network Performance Analysis Tools in Simulations including Real-Time Constraints

Jean Paul Meynard

LiTH-IDA-Ex-97/43

1997-06-02

Acknowledgements

First of all, I would like to thanks my supervisor Anders Törne for having initiated and supported this work. I am very grateful also to Leif Finno for his precious advice and his patience during the modelling of IDA’s network. Thanks also to Nahid Shahmehri for having allowed me to take advantage of this Erasmus program and attend some courses at Linköping University.

A special thought for Pablo Gruer for his comments and his news from France.

I could not forget Todd Wiseman at AST and Minghua, Fred and David at OPNET, who kept answering to my tricky questions in spite of everything.

A few other special thanks to the innebandy players for keeping me in great shape and the RTSLAB team members for the Wednesdays coffee breaks.

ABSTRACT

This project concerns a study of performance analysis tools dedicated to computer networking. This investigation was carried out in order to assess the abilities of these tools to represent and simulate models under real-time constraints.

Two software packages have been tested and evaluated QASE RT, an information system simulator, and OPNET, a powerful communication system simulator.

The method employed consisted in four steps: theoretical studies, simulation tool evaluation, simulation and reporting. Two networks have been used to illustrate the tools' functionalities: the actual network at the Department of Computer and Information Science (IDA) and a client/server-oriented network.

The results of the simulations applied on these models are discussed. The advantages and disadvantages of the evaluated tools are presented at the end of this document

Keywords:

simulation, network, real-time, performance analysis, modelling

TABLE OF CONTENTS

FIGURES

Tables

Chapter 1

INTRODUCTION

This chapter is a brief description to the subject matter and the background of the thesis is explained. This parts also deals with the method followed and the limitations and scope of this work. At the end of the chapter are some short instructions on how to read this thesis.

1.1 Project Background

The thesis was carried out within the Real-Time Systems Laboratory of Linköping University, under the supervision of Anders Törne.

The project aims at investigate two kinds of network performance analysis simulators in order to evaluate their abilities to assist the user in the modelling, simulation and analysis phases of a distributed system on which real-time constraints are applied.

1.2 Problem Introduction

This study raises several sub-problems:

- which suitable tool for network simulation has to be considered
- what real-time constraints it could be useful to model,
- what size the models should have
- what level of complexity the modelled system should represent

After having dealt with those points, the remaining questions concern the results of the simulation, which ones are relevant and the way they can be deemed correct. Another aspect assessed in this study is the use of the two tools in terms of, for example, flexibility, feature richness and object reusability.

1.3 Project Objectives

The purpose of this project was to investigate the abilities of network simulators as a tool in network design. More specifically, the project involved investigating simulation tools and developing models within a specific framework with the help of this tool. After this evaluation, a simulation study was carried out and the outcome presented after analysis.

In doing this, we hope to learn a lot about simulation methodology, the actual simulation tools and their applicability for systems involving real-time constraints.

1.4 Method

The method employed in this project can be divided into four steps. They are successively:

1. Knowledge acquisition and problem formulation
2. Simulation tool evaluation
3. Simulation study
4. Evaluation and report writing

These four steps are discussed in detail below.

1.4.1 Knowledge acquisition and problem formulation

The first step in this project was to gain knowledge of the simulation theory and get well acquainted with the simulation methodology. The main sources of information concerning these mentioned points were the book of Law and Kelton [Law91] and a thesis report about a specific performance analysis simulation [Hol95].

Regarding computer networks, the information used come from a book written by Andrew Tanenbaum [Tan96] and from several on-line articles and reports [Wrl88], [Wit96], [Gra97].

The literature from the university course *Embedded Systems, Real-Time and Concurrent Programming* [Tör96] and a book about the distributed systems [Mul93] were used to provide a better comprehension of the real-time aspects of the models to be built.

Two sorts of model have been chosen for this study.

- The first one represents a client-server based application whose workload depends on an algorithm including contention for resources, scheduled actions, process synchronization, concurrent access and inter-process communication. The aim of this model is to offer a wide range of the most common real-time constraints through a protocol, often referred as the Dining Philosophers.
- The second one represents the network at the Department of Computer Science of Linköping University (IDA). It is made up of about four hundred workstations and offers a good example to test the scalability of each simulation tool. As no physical model of IDA network was available, a few interviews with Leif Finno (technical support) were necessary to get a good comprehension of its physical implementation.

1.4.2 Simulation tool evaluation

This study was initiated by an evaluation copy of a simulation tool, later upgraded with the fully commercial version. This first tool is marketed under the name QASE RT and it is supported by a Colorado-based company, Advanced System Technology. OPNET from MIL 3 Inc. was the other company to accept to participate in this evaluation. Both softwares include a tutorial session highlighting the basic functions and the scope of each product.

1.4.3 Simulation study

The two tools were used to try to simulate both models. Regarding the validation of the models the conceptual part was checked implicitly or with the model verification features. In addition the built-in trace functions gave good indications about the communication over the network. In the case of the client-server application, it clearly indicates if the system's behaviour was compliant with the algorithm.

It has to be mentioned that neither model were validated by physical measurements and the obtained data do not necessary match real world values. This point will be discussed in the *Limitations* section.

1.4.4 Evaluation and report writing

The final step was the evaluation and reporting. The conclusions presented at the end of this document are drawn both from the simulation studies and the thesis project itself.

A paper copy of this document is available at IDA and an on-line version is available on IDA web site at the following address: http://www.ida.liu.se/~x97jeame/report

1.5 Limitations

Both simulations were made using discrete-event simulation(1) tools as a method to investigate system performance and limitations follow:

1. It is very difficult to construct an accurate model which behaves as the real system would do. In the case where the real system does not even exist such as the first model (client-server application), it is almost impossible.
2. Model validation is difficult and differences between values obtained from the simulation and the physical measurements always exist. As no relevant measurements are available about IDA network, the model reliability is entirely based on the accuracy of the provided network components.

However, the main purpose of this study was to assess the use of simulation tools and not to design an error-free representation of a given model. A lot of attention have been paid, however, on the tool abilities to tend to this goal.

(1) This term will be explained in the section SIMULATION

1.6 Report disposition

This section gives additional information about the contents of this document and can help the reader to find an explanation of the technical terms used in this introduction.

- The first chapter, INTRODUCTION, gives a brief introduction to the thesis and the methodology followed and the limitations considered.
- Chapter two, SIMULATION, presents advanced simulation theory.
- Chapter three, DISTRIBUTED SYSTEM, defines basic terms in computer networking, the real-time aspects and the communication protocol used within the scope of this study.
- Chapter four, MODELLING, shows how the conceptual modelling was made and how it was transferred into an implementation. The different model's components are discussed as well.
- Chapter five, QASE RT EVALUATION, describes the capabilities of QASE RT and its modeling procedures. Then both models are simulated and the results presented.
- Chapter six, OPNET EVALUATION, describes the capabilities of OPNET and its modeling procedures. Then both models are also simulated and the results presented.

Chapter 2

SIMULATION

In this chapter, some relevant aspects of the theory of simulation are discussed.

2.1 Simulation

In this section, simulation is introduced as a useful problem solving tool, why and when simulation should be used are also discussed.

2.1.1 Simulation definition

What is a simulation?

A brief answer is that a simulation means to imitate the structure and behaviour of something in certain aspects. In the technical sense of the word it is a method of using models for the study of actual or postulated systems. The model may be anything from a set of equations to a physical miniature. What is suitable depends on how natural the representation is and the pre-requisite and goals of the simulation. By observing the model and how it reacts, conclusions are drawn on how the real system could behave. This is illustrated in figure 1.

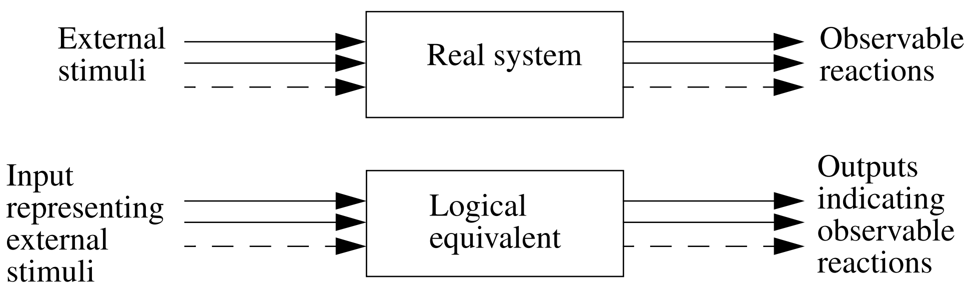


FIGURE 1. A system and its model

Given a model, experiments can be conducted in which simulation data are collected. These data can later be analysed and used to answer questions regarding the system under investigation.

2.1.2 Simulation motivation

There are a number of basic reasons to use a simulation [Hol95]:

- to examine the behaviour of a system where it is difficult, costly or impossible to do so (e.g. if the system does not exist)
- when it may be hazardous or expensive to observe the real system (e.g. a nuclear power plant)
- when the time scale of the real system is too small or too large
- to see the inner behavior of a system and know what causes a system to react in a certain way.

There are, of course, other methods competing with simulation as a means of solving these kinds of problem. The most common one is probably the analytical models. These describe systems as a set of different inputs and equations, and the result can be extracted directly from the equations. One disadvantage, which is often regarded as an eliminating factor, is that the modeller may have to make some simplifying assumptions to make it mathematically trackable. These assumptions may be quite unrealistic. The complex description of the system makes very hard to formulate, understand, and change. It can also lead to equations that defy analytical solution. These things make analytical models best suited for simple systems.

Simulation, on the other hand, can handle very complex systems. It also provide a safe way of training system operators. It is more realistic than an analytical models and the model representation is easier to explain and understand. A major advantage of simulation, is its usefulness throughout a complete project. During a network design project different network structures and protocols can be tested and evaluated together. Once it is done, a network component evaluation can be done to decide which particular devices to use. The simulation can thereafter be used to determine network performance and identify possible problems. During system testing, the model can be used to fine tune the system parameters. In the operational phase it can be used to evaluate the effects of the incoming upgrades.

Unfortunately, there are some negative sides that have to be mentioned. First of all, the main drawback is that it normally works statistically. This means that it never provides exact answers and never guarantees an optimal solution. In addition, you can never say that the real system will work just because the model does. The main reason is that a certain number of simplifications have been made to avoid add too many details to the system description. As a result simulation never can be exhaustive in practice.

Simulation has many other applications than computer networks, it is used as well for constructing production lines, professional training, e.g.airplane pilots, queuing systems, strategic decision making, e.g. business managements, and communications, e.g. telephone networks.

In the tables 1 and 2, there is a summary of the advantages and disadvantages of both methods.

Simulation	Analytical solution
handles complex system	concise model
natural model representation	gives optimum, if any
flexible model	
useful throughout entire project	
facilitates training	

TABLE 1. Methods advantages

Simulation	Analytical solution
no exact answer	needs simplifying assumptions
optimum solution not guaranteed	complex model representation
difficult model validation	long development time
computationally expensive	static model
facilitates training	has difficulties handling details

TABLE 2. Methods disadvantages

2.1.3 Simulation model types

Depending on how simulation time is incremented, simulation models are further divided into:

- continuous time
- discrete-time
- continuous-time/discrete-event

In continuous time models, time is considered to flow continuously, which is very common for models in, for example, automatic control. There are a set of differential equations governing how the system reacts at all times. For computer-based models, this can not be implemented due to the discrete nature of the computer. In the second type of model, time proceeds in steady discrete steps. By making the ticks small enough, a continuous model may be approximated. The third kind of model also use discrete time steps. The difference lies in what makes the clock tick. Since it is not necessary to imitate a real clock, time can advance directly to the next instant when something happens in the system, i.e. an event. The method is called next-step time-advance [Law91].

In a next-step time-advance method a simulation moves the system model through a sequence of states in order to mimic a real system. A variable called simulation time is maintained to model the progress of time as the model state changes. Event may cluster at the same simulation time. Multiple events can occur simultaneously, see figure 2.

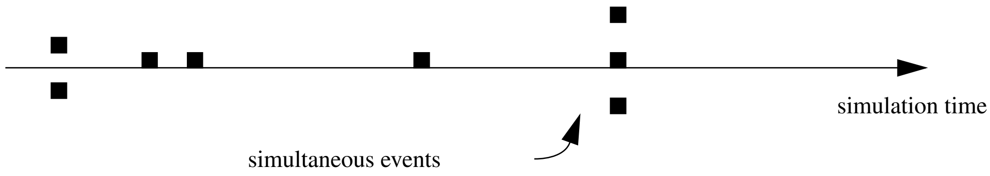


FIGURE 2. Event distribution on a simulation time line

Simulation time changes only when new events occur; during a single event, time cannot change. The current simulation time is defined to be the time of the currently executing event. The simulation time does not necessary advance by equal amounts at each event, but jumps to the next event's time.

The events are managed with an event list. The purpose of this is to make sure that all events are executed in the proper time order. Event may be scheduled for the current simulation time or for any future time but never for past times. The earliest remaining event is referred as the head of the event list, and the latest event is called the tail. An event is automatically removed from the event list after its execution. This modeling technique has been used in this project since this is the most common one in the actual network simulation tools.

2.2 Simulators

A simulator, also called simulation tool or simulation engine, is the program or system which runs the simulation. In this part some aspects of this tool are discussed.

Of the simulators which exist today, three major classes can be distinguished according to their real-world representation.

These classes are:

- **event-scheduling**, also called control-driven, all events scheduled to happen are ordered in a list according to their event times. This is a typical implementation of the next-step time-advance philosophy described in the previous section.
- **process-interaction**, this has been popularized by the object-oriented *Simula* programming language. A process is a sequence of events ordered by time.
- **activity-scanning,** this is a data-driven approach, activities are effected if some conditions, usually involving the simulation clock are satisfied.

The simulators investigated in this study belong to the event-scheduling class, which is nowadays the most common method implemented in simulation tools.

Chapter 3

DISTRIBUTED SYSTEM

This chapter aims at giving basic knowledge of the terms and keywords mentioned in this document and more specifically concerning the computer network and the communication.

3.1 Computer network

In this part, the modelling of computer network, especially local area networks, is discussed.

3.1.1 Network architectures

A computer network consists of a collection of independant computers which communicate with one other over a shared network media. The network media consists of two types of components: a transmission line and an inter-connection device. Local Area Network (LAN) are the networks confined to a geographic area such as a building or a campus.

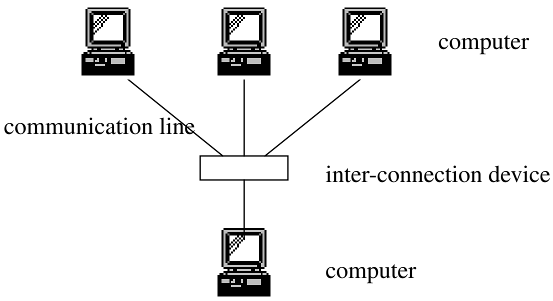


FIGURE 3. Basic LAN architecture

The inter-connection devices are the key components of the network, they allow the communication on the one hand between the LAN's computers and on the other hand with others networks or subnetworks. Three of them are going to be mentioned in the project's models and are defined below:

- ***Hub***: this device is used to connect a certain number of segments (communication link having one or more devices plugged in) according to the amount of ports it offers (4,8,16,32). It takes any incoming signal and repeats it out to all ports. This implies that the receivers have to check if they are the recipient upon reception of a message. In addition all the segments must have the same technical characteristics (throughput, message type).
- ***Bridge***: the function of a bridge is to connect two separate networks, it acts as a gate between them, adapting the incoming messages to the message type supported by the recipient.

- ***Router***: they are complex devices used to connect two or more networks together. A router can read the information contained in the header of messages and determines its correct destination. In addition, it can operate with networks running different protocols.
- ***Switch***: it is an extension of the concepts of bridging since more than two networks can be linked through this device. Two different architectures are now well spread, they are called cut-through and store-and-forward, they respectively allow to speed up the message propagation and to catch certain errors in the messages.

3.1.2 Network topologies

The topology defines the physical layout of the network, the most common are shown below in figure 4.

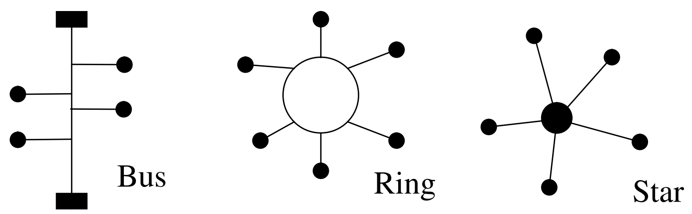


FIGURE 4. Common topologies

To connect communication equipment or networks from different vendors is an perilous exercise without standards. Because of computer network inherent complexity, the architecture of such systems is often defined through hierarchical layers (or levels), each of them having its own standard. The OSI-model states that communication architectures should be designed with seven-layers ranging from the physical level to the user application. Familiarity of the OSI model is assumed afterwards in this work. To get more acquainted with this model, see [Tan96].

In a local computer network some conditions affect the OSI-model. To use a network, OSI states that level 1 (physical) through level 3 (network) are necessary. A physical and logical link is always needed, but the Network layer has very little function in a LAN, since there is no routing. Furthermore addressing, sequencing, error control,. are taken care of in layer 2. The result is that the LAN Reference Model handles a three-layer functionality in two layers, see figure 5.

OSI model	LAN reference model
presentation	
application	
session	
transport	upper layers
network	logical link control
data link	medium access control
physical	physical

FIGURE 5. OSI model and LAN reference Model

3.1.3 Ethernet network

Ethernet is a bus or star-based local area network technology that is in common use today. As the bus topology will not be used in this project, the star topology will be solely described.

The star topology uses a hub as the central connection to networked devices and offers a 10 Mbits/second transfer speed.

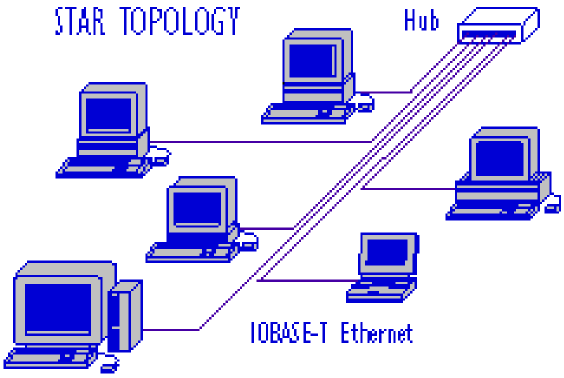


FIGURE 6. Ethernet star topology

The following paragraph deepens the characteristics of this particular LAN, a reader who are not familiar with this technology can find more accessible explanations in [Tan96].

The Ethernet standard is defined by the IEEE as IEEE standard 802.3. This standard defines rules for the configuration of an Ethernet as well as specifying how elements in an Ethernet network interact with one another.

<u>Layers description:</u>

- **Physical**
  The function of this layer is simply to transmit the frames from the MAC layer on the network and do the same to the MAC layer for the network's incoming frames.

- **Medium Access Control (MAC)**
  The role of this MAC protocol is to provide efficient and fair sharing of the communication channel connecting the stations of the LAN.

  The Ethernet MAC accepts data packets from a higher layer protocol (such as the Internet Protocol IP) and attempts to transmit them at appropriate times to the others stations. The MAC layer is in charge of constructing the frames upon transmission and disassembles the frames with address and error control upon reception.
- **Logical Link Control (LLC)**
  The LLC provides services for data transfer and connection handling. The LLC should also perform error and flow control and provide different types of service.

3.2 Packet switching

There are many forms of communication that can be supported by network communication links (e.g. packet, cell, frame, datagram,...). One fundamental structure, called packet, will be used in the models of this project and it is therefore explained below.

Packets are objects that contain formatted information that can change dynamically, be stored and be collected to be transmitted in group. Each packet is identified by a unique key and is destroyed upon reception by the recipient.

The fields include usually: the source address, the destination address, a time stamp, and the data to be transmitted.

Chapter 4

MODELLING

In order to evaluate the behavior of a network model under real-time constraints, two models have been built. This section describes the design of each of them.

4.1 IDA network

In this section, the model presentation, description and representation are successively discussed.

4.1.1 Model presentation

A natural idea to evaluate the tools was to consider an existing network, model it and reuse the model of this network as a support for incoming upgrades. IDA network was a suitable candidate and brought several advantages.

1. It is made up with many types of network entities and includes some subnetworks.
2. The number of connected devices is close to four hundred and it was interested to see how easy it is possible to represent so many devices with these tools.
3. Upgrades are periodically realized on this network and to foresee the behaviour of the network after some changes was relevant.

As no model of this particular network was available, a few interviews with Leif Finno (technical support) have been necessary. The main problem raised in this approach was to fix the level of complexity of the model to be represented. Adding more and more details makes the system more and more complex to picture. The level of details of this the network has been limited to the key-components, which form its backplane. It could have been also possible to add more components, but the amount of information required would have been difficult to handle.

4.1.2 Model description

Two subnetworks can be highlighted, one is dedicated to the research activities and the other is used by the students. The two networks are only connected together by a router, which is also the only link to the university's network.

- The research network is certainly the most complex one, it relies on three Ethernet switches and allows the communication between about 220 devices located in four different buildings. The file servers regroup storage devices, backup-drives and workstations acting as servers. In this architecture, two NFS servers are working to support the connections. A workstation is attached to one of the two servers to limit the consequences of network breakdowns.

- The undergraduate network connects about 180 devices located in four buildings, but only one switch is used to connect them together. The file server architecture is about the same than the research one. Two NFS servers are used to handle the students requests.

The communication over the network uses either a 10 Mbits or a 100 Mbits connection according to the user needs and the switches physical capabilities.

4.1.3 Model representation

The IDA network connects together several types of workstations, located in different buildings and at different throughputs. This increases the number of possible representations. In the following drawings, the research and undergraduate network are described separately and the different devices have been grouped by ‘building location’. This organization is illustrated through figure 7 to 11.

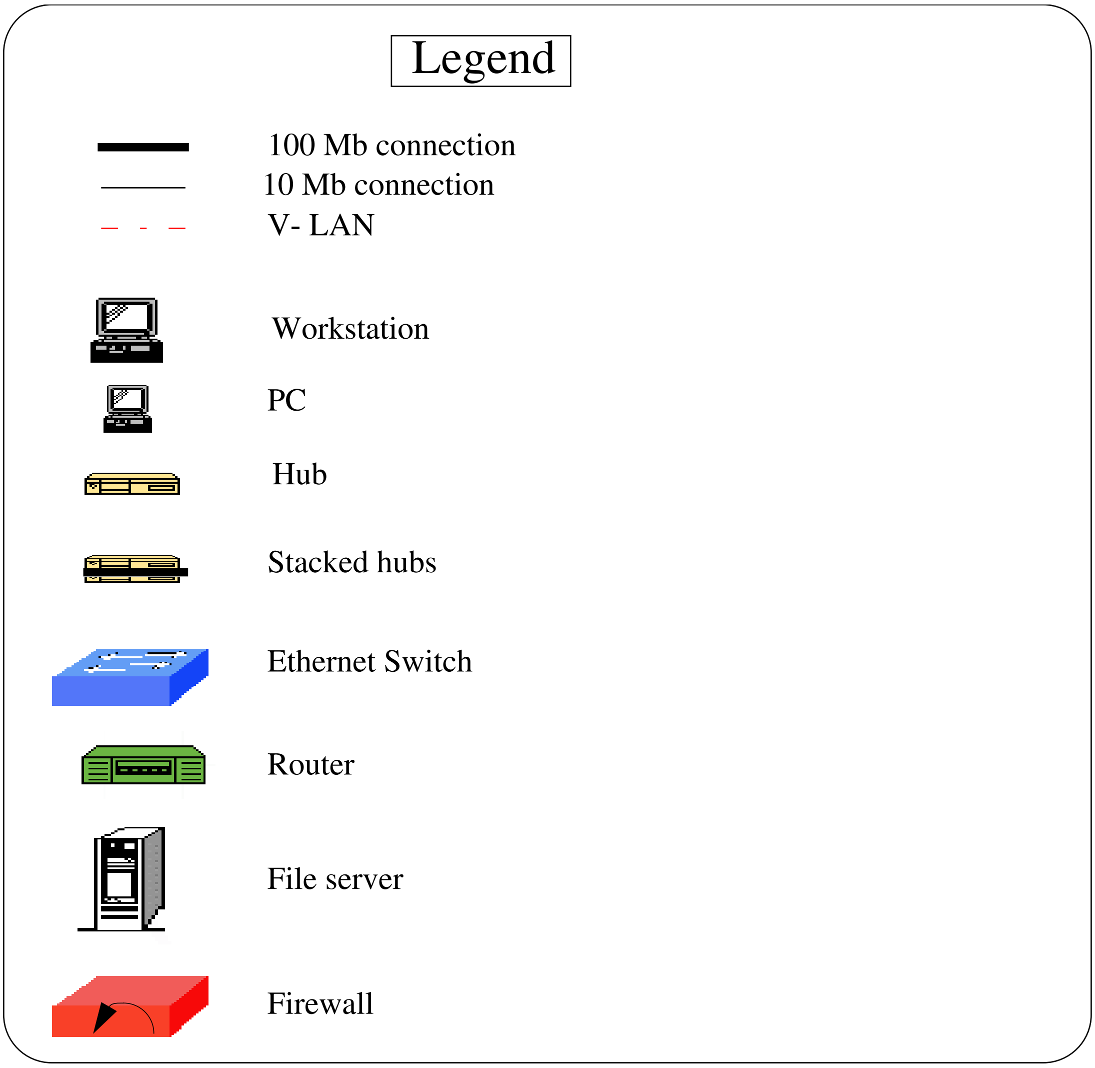


FIGURE 7. IDA network - legend

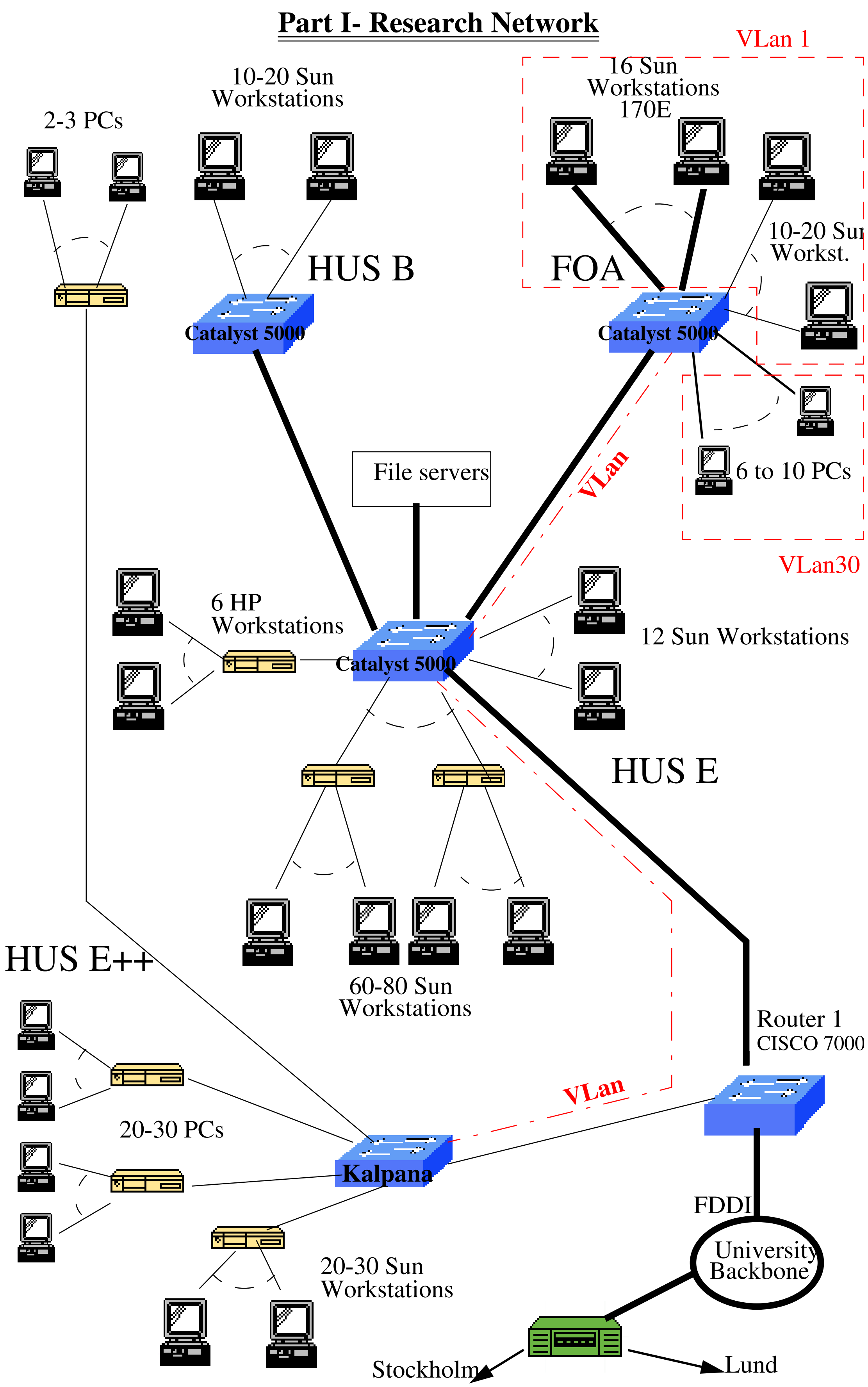


FIGURE 8. IDA network - research network

Part I- Research Network
File servers description

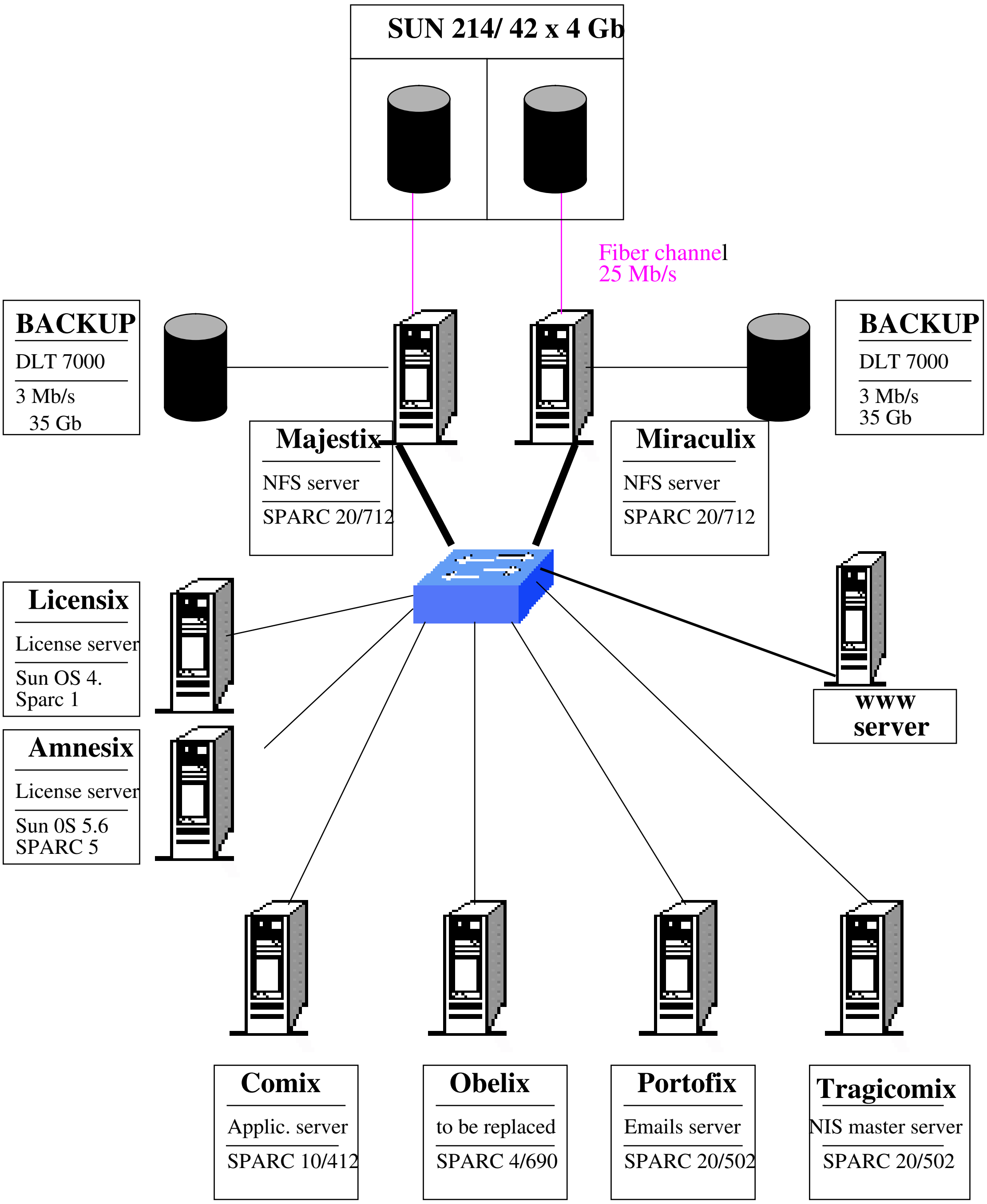


FIGURE 9. IDA network - research network - file servers

Part II- Undergraduate Network

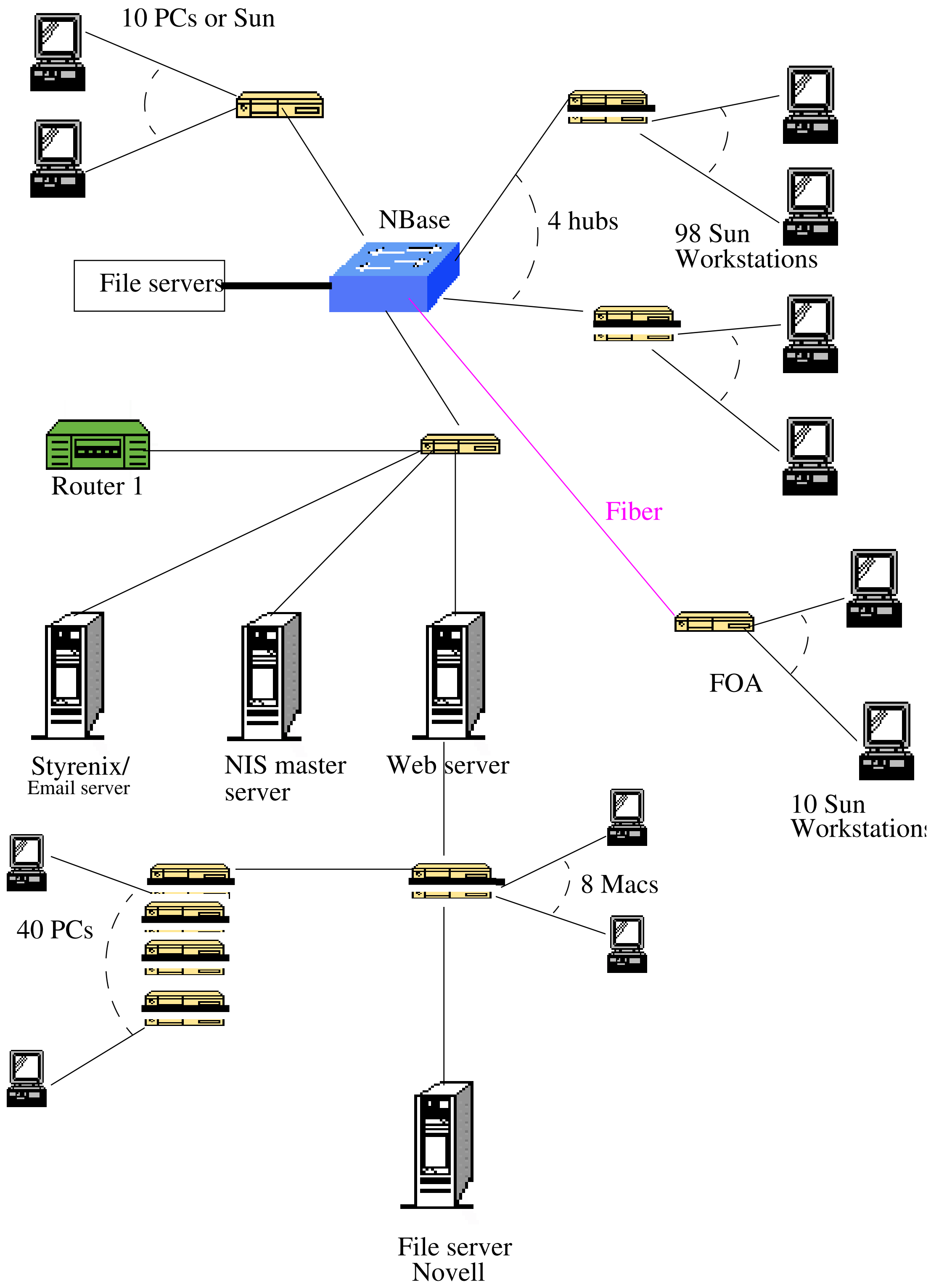


FIGURE 10. IDA network - undergraduate network

Part II- Undergraduate Network
File servers description

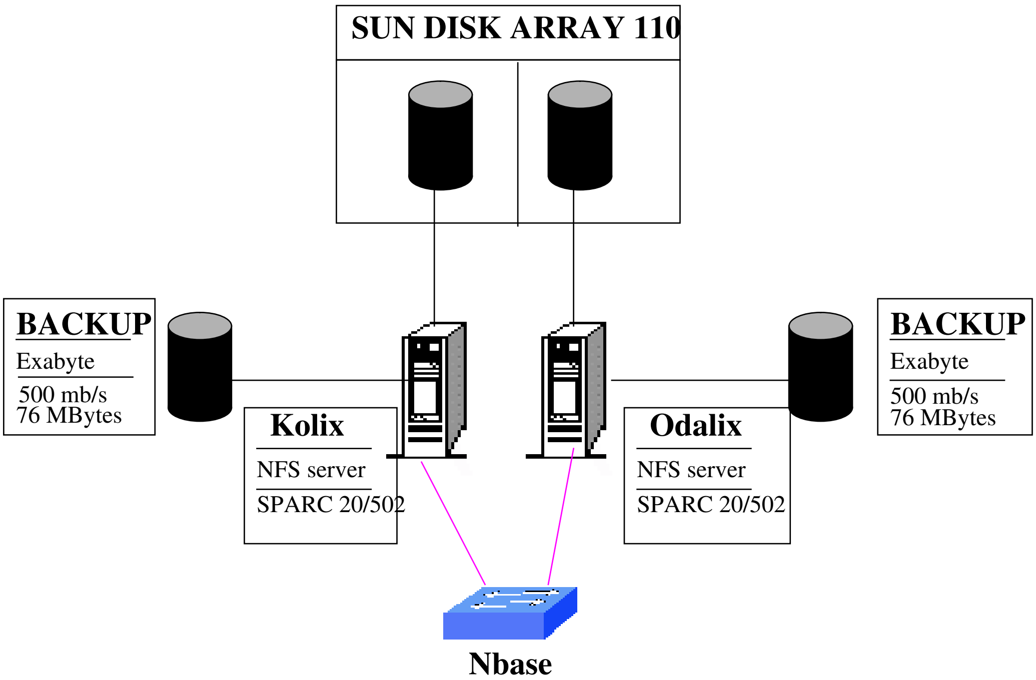


FIGURE 11. IDA network - undergraduate network - file servers

4.2 A simplified model: A Client-server Application

To model and track easily a simulation, it was decided to focus the investigation on a small system, featuring most of the usual aspects of concurrent access and shared resources situation. In the following section, the justification of this choice and its implementation are successively presented.

4.2.1 Model presentation

1. Model justification

Using a smaller network than IDA's was relevant since the latter is very complete, but at least three disadvantages can be pointed out:

- because of its size and organization (multi-layer architecture) this model is difficult to handle for this kind of simulation
- the representativeness of real-time constraints is not fully achieved in the daily workload
- too many parameters can interfere. Many side-effects (corrupted results, irrelevant information) can generate an inaccurate or untraceable simulation

2. Choice of the application

The ‘Dining Philosophers’ model, whose description appears in the next section, has been chosen for many considerations:

- It is well-known and the different solving strategies of this problem have been frequently treated in the literature.
- It involves the use of many real-time aspects (limited resources, concurrent access, time constraints,...) and provides an overall assessment of the software’s abilities through the modeling process.
- If the simulation tools appear to be effective, they could be directly used as a support in undergraduate classes to illustrate graphically the effects of the various implementations of this problem.
- It can physically be modelled using various sorts of network devices.

3. The classic problem of the dining philosophers

Many versions of the dining philosophers have been developed and discussed. The version selected for this study will be explained in this section and later modelled and evaluated.

Described below is one of the descriptions of the problem, first investigated by Dijkstra:

> In the old times there lived people, who had time to think over principles of the world, who were wise and who consequently influenced politics and morality and daily affairs: the philosophers.
>
> But there was one thing they depended on: they were steadily affected by hunger. But eating sometimes caused big problems. It said in the old tales that a group of starving philosophers died - although they had full plates.
>
> Supposing they are Chinese philosophers and dining is organized as follows:
>
> In the dining room, there is a large table (see figure 12) and five chairs. Five bowls and only five chopsticks are available on the table. In the middle of the table there is a main noodle bowl being filled frequently. For eating noodle one needs two chopsticks, and as the table is quite large one can take only the chopstick to one’s left and to one’s right. Then the philosopher can start eating. Then he puts the chopstick back on the table at the same place and starts meditating again. If it turns out that a philosopher dies, he releases the chopsticks he may have in his hands before heaving his last sigh...

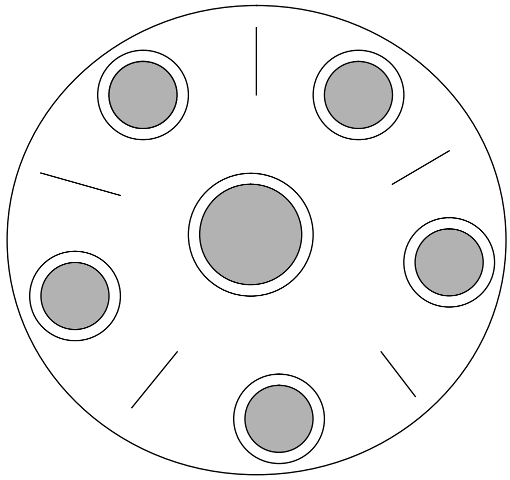


FIGURE 12. Table description

4. The refinement

The final version is derived from the aforementioned rules with a few additional characteristics:

In fact, it may happen that every philosopher becomes hungry at the same time and decides to take one of the chopstick and keep it no matter if they cannot use it to eat (deadlock problem). It is also possible that some of them make a conspiracy against one to let him starve in order to eat more often (starvation problem).

The strategy I followed prevents the deadlock case, but not starvation to avoid making the application too complex. It also introduces two new characters: the cook, who is in charge of refilling the noodle bowl as long as some noodles are available in the food storage (or pantry) and the waiter, who handles the chopsticks contention.

5. Parameters

Different parameters have been introduced to simulate the system under various configurations.

- Philosophers' parameters

  - *Maximum_waited_time_before_eating*: after this duration, the philosopher is starving and eventually dies.

  - *Eating_time*: pre-determined period allowing the philosopher to finish his noodle bowl.

  - *Ponder_time*: the philosopher has to think during this amount of time before trying to take his dinner.

  - *Rest_time* (optional): break after having eaten and before starting to think.

  - *Waiting_right_stick*: sampling interval to get a right stick

  - *Waiting_for_noodle*: sampling interval to get a noodle bowl

- Resource parameters:

  - *Number_Noodle_Parts*: the amount of noodle bowls that can be filled up when the cook refills the main noodle bowl.

  - *Number_Noodle_Bowl*: the number of times the main bowl can be refilled. This is equivalent to the amount of noodles stored in the pantry.

  - *Noodle_cooking_time*: after being removed from the pantry, the noodles are not available on the table, they have to be cooked for this period of time.

  - *Check_enough_noodle_time*: the sampling interval to check the amount of noodles in the main bowl

4.2.2 Simulation goals

Several goals have been defined in order to get an overall assessment of the simulation abilities of the tools. They have classified into three classes:

- **Time Statistical Analysis**

  The time statistics give relevant values about the time spent in different parts of the program. This can also be used to check if the deadline are met. In this model it could be interesting to know:

  - How long must a philosopher wait before getting a noodle bowl?
  - How long must a philosopher wait before getting the two chopsticks?
  - How long does it take for a philosopher to eat. i.e. getting a full noodle bowl and two chopsticks?

- **Real-time constraints**

  Four real-time constraints have been chosen and applied on the model. The constraints considered are:

  - Resource sharing
  - Concurrent access
  - Scheduling abilities
  - Process synchronization

- **Network constraints**

  Some aspects of the communication over a network had to be controlled. This concerns:

  - The network's behaviour when it is overloaded.
  - The available indicators (queue management, network quality,...).
  - The detection of the bottlenecks.

4.2.3 The implementation of this problem

After discussing the Dining Philosophers problem in the previous section the model's design can now be introduced.The different entities which make up this problem are successively described below:

- **The philosophers**

From a functional point of view, each philosopher undertake the following actions:

1. Wait until the dinner is ready.
2. Start thinking one more time before trying to eat.
3. Take a bowl and fill it up.
4. Take the chopstick on his left, if it is available, otherwise wait for it.
5. Take the chopstick on his right, if it is available, otherwise release his left chopstick (to avoid deadlock situations).
6. Eat for a lapse of time (i.e.the time to finish his noodle bowl).
7. Put back the chopsticks on the table.
8. Start thinking again, and return to step 3.

By analogy, we can consider that a philosopher can be modelled by a processor, having several successive actions to handle.

- **The cook**

The cook also has many tasks to cope with, he must check regularly if there is enough noodles in the main bowl. If it is not the case, he has to take more noodles from the food storage, cook them and then pour them into the main bowl located on the table. He must also decide to go back home when there is not any noodles left.
Therefore he can be represented by another processor, whose function would be to execute the previously described actions.

- **The food storage**

The noodles are kept in the pantry and the cook has to know how many noodles are left in his kitchen. The cook is only person allowed to get access to this place.

The food storage can regarded as data storage device whose data would be the amount of noodles not yet eaten.

- **The table**

The table contains the chopsticks, the five noodle bowls and the main bowl

It could be seen as a data storage device whose data will be the amount of noodles left in the main bowl, the state of the chopsticks and the noodle bowls.

- **The chopsticks**

A chopstick can only be used by the two philosophers next to it and only one of them can use it at a time. The state of this data will be stored in the 'table'.

- **The main bowl**

The main bowl can be refilled a certain number of times by the cook, the quantity of noodles is determined by him as well. The philosophers can fill their bowl provided that there is enough noodle in the main bowl. Otherwise they have to wait.
It could be represented by a data, meaning the quantity of noodles. Updated either by the cook, who adds a predetermined quantity, or by the philosophers, who withdraw a unit of noodles each time they fill their bowl. To limit the access to this data, the waiter will be the only person to handle this item directly

- **The noodle bowls**

Each philosopher has his own bowl, and has to fill it up before starting eating, nevertheless the bowl will not be a relevant data in itself, thus it will not be represented in the model.

- **The waiter**

The waiter handles the requests from the philosophers and the cook. He is the only one to access the data stored in the table and to execute the updating. Therefore he can be represented by a processor, whose function would be to execute the previously described actions.

- **The communication between the different entities**

As previously stated, the philosophers are sat around a large table and the cook is working in the kitchen, that is why they communicate only with the waiter. He has an overall view of the situation and can perform any authorized action claimed by the philosophers or the cook. We could extend the analogy in saying that the different entities are acting in an distributed system. The requests are performed through messages driven by a network linking the philosophers, the cook and the waiter.

- **To sum up**

The restaurant	distributed system
The orders	messages to the waiter
The philosophers	processors
The cook	processor
The waiter	processor
The table	storage device
The food storage	storage device
The chopsticks	data
The main bowl	data
The noodle bowls	not physically represented in the implementation

4.2.4 Hardware aspects

The network components characteristics have to be carefully specified to ensure the consistency of the results during the simulation if the model is tested under different configurations.

The attributes listed have been singled to parametrize the model

- The speed of the processors: Waiter, Cook, Philosophers
- The operating system policy of those processors
- The network type and protocol
- The transfer rate of the storage devices

4.2.5 The network aspects

The communication between the different entities is based on messages. In this section, the different messages exchanged between on the one hand, the philosophers and the waiter, and on the other hand, the cook and the waiter are briefly listed in the tables below. In addition, figure 13 gives a graphical representation of the whole system.

1. **Data flows Philosophers to and from Waiter**

Request: Philosopher to Waiter	Reply: Waiter to Philosopher
message type: Noodle_bowl.Get	message type: Noodle_bowl.Ack
message size: GET_BOWL_DATA_SIZE	message size: ACK_DATA_SIZE

TABLE 3. Get noodle bowl

Request: Philosopher to Waiter	Reply: Waiter to Philosopher
message type: Left_Chopstick.Get	message type: Left_Chopstick.Ack
message size: GET_DATA_SIZE	message size: ACK_DATA_SIZE

TABLE 4. Get left chopstick

Request: Philosopher to Waiter	Reply: Waiter to Philosopher
message type: Right_Chopstick.Get	message type: Right_Chopstick.Ack
message size: GET_DATA_SIZE	message size: ACK_DATA_SIZE

TABLE 5. Get right chopstick

Request: Philosopher to Waiter	Reply: Waiter to Philosopher
message type: Left_Chopstick.Release	message type: Left_Chopstick.Ack
message size: RELEASE_DATA_SIZE	message size: ACK_DATA_SIZE

TABLE 6. Release left chopstick

Request: Philosopher to Waiter	Reply: Waiter to Philosopher
message type: Right_Chopstick.Release	message type: Right_Chopstick.Ack
message size: RELEASE_DATA_SIZE	message size: ACK_DATA_SIZE

TABLE 7. Release right chopstick

2. Data flows Cook from and to Waiter

Request: Cook to Waiter	Reply: Waiter to Cook
message type: Set_up_Table.Get	message type: Set_up_Table.Ack
message size: GET_SET_UP_TABLE__DATA_SIZE	message size: ACK__DATA_SIZE

TABLE 8. Set up the table

Request: Cook to Waiter	Reply: Waiter to Cook
message type: Amount_Noodle.Get	message type: Amount_Noodle.Value
message size: GET_AMT_NOODLE__DATA_SIZE	message size: GET_AMT_NOODLE__DATA_SIZE

TABLE 9. Check the amount of noodles in the main bowl

Request: Cook to Waiter	Reply: Waiter to Cook
message type: Refill_Main_Bowl.Value	message type: Refill_Main_Bowl.Ack
message size: REFILL_MAIN_BOWL__DATA_SIZE	message size: ACK_DATA_SIZE

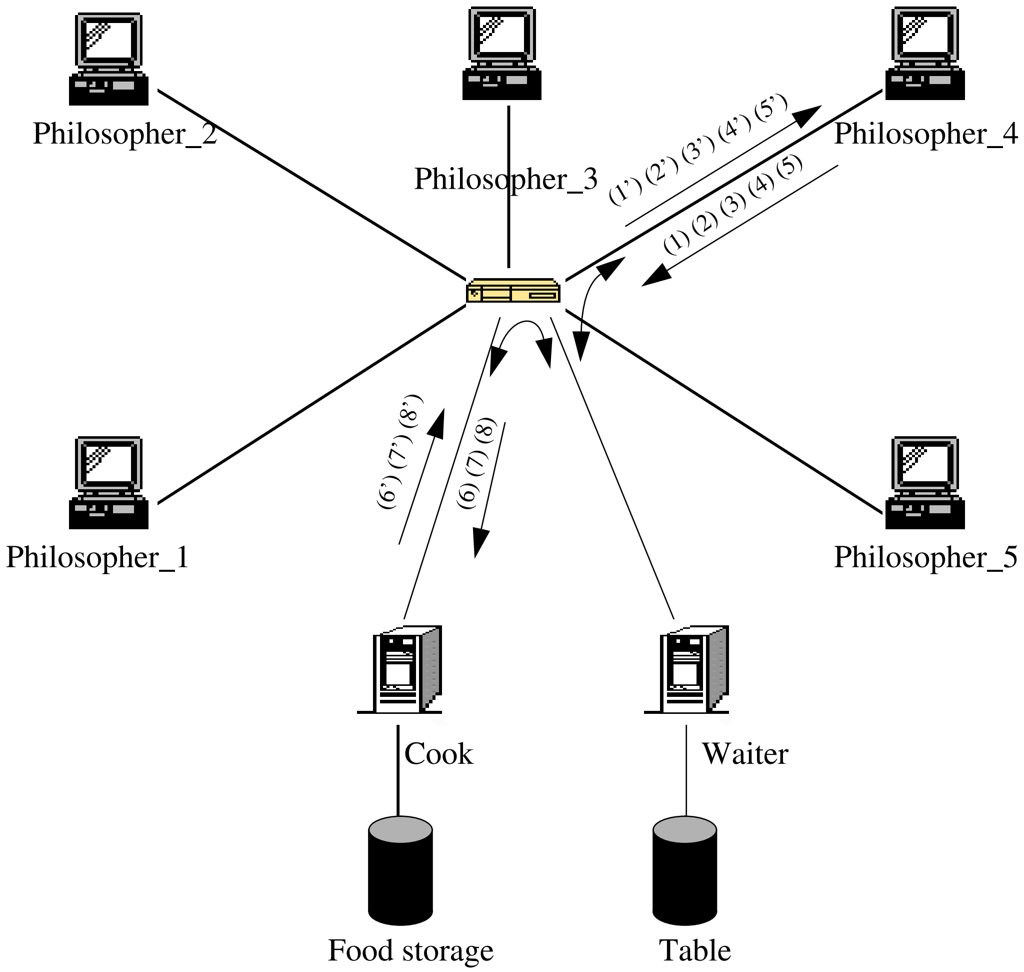
TABLE 10. Refill the main bowl

3.Data storage access and resource management

The two data storages are seen as directly connected to a processor and any communication device between the data storage and its processor should not be considered.

Various operations can be undertaken on the data storage device in order to allocate, deallocate data or update values.

- Remove one instance of noodles from the food storage.
- Remove a portion of noodles from the main bowl.
- Refill the main bowl with ‘Number_Noodle_Parts’ portions of noodles.
- Get the amount of noodles in the main bowl.
- Get the status of a chopstick.
- Update the status of a chopstick.



Get left chopstick:	(1).Get	(1').Ack
Get right chopstick	(2).Get	(2').Ack
Get noodle bowl	(3).Get	(3').Ack
Release left chopstick	(4).Release	(4').Ack
Release both chopsticks	(5).Release	(5').Ack
Check the amt of noodle	(6).Get	(6').Value
Refill the main bowl	(7).Value	(7').Ack
Set up the table	(8).Get	(8').Ack

FIGURE 13. Message-oriented communication

Chapter 5

QASE RT EVALUATION

QASE RT from Advanced System Technologies, is the first system performance analysis tool to be investigated. Two studies are carried out. First of all IDA network is modelled to assess QASE ‘s modeling abilities, then the Dining Philosophers problem is implemented and tested in different configurations.

5.1 QASE RT. presentation

This product is made by a Colorado-based company, AST Engineering Services, Inc. and its particularly well suited to model distributed client/server applications.This tool aims at evaluating information systems before their implementation or expansion.

QASE has been developed to allow the user to describe his system using graphical components without having to construct a large program.

It also offers the possibility to perform two kinds of simulations:

- **Discrete-event simulation** for detailed performance evaluation under dynamic conditions.
- **Analytics simulation** (mathematical evaluation) provide a rapid, best case evaluation of performance for steady state conditions.

5.2 QASE system description

QASE uses an object-oriented approach to represent the different entities of the system. This is a hierarchical entity-attribute specification, where each object contents hardware, software and data descriptions.

5.2.1 Model building and Configuration

Five steps are necessary to describe a model through its physical implementation, the traffic definition, and the simulation parameters declaration. The different steps are listed below and afterwards explained.

- Information flow description
- Software description
- Data description
- Hardware description
- System parameters

1. Information flows categories

This section allow the user to represent graphically the flow of information through the system.

- **Workload**: external events, e.g entry of data, or internal, e.g query on a database, that cause the system (the Execution flows) to react, to do something.
- **Execution flows** (EF): control flow diagrams that describe functional processing threads in response to system workload stimuli (see figure 14).
- **Subflows**: reusable subsets of Execution flows. They cannot be triggered by a stimulus. They act as macros and allow the user to break up a complex system into smaller sub-systems.

Each control flow diagram can be made up of several components:

- **Primitive function** (node): represents the work that the system has to do in response to a stimulus from the flow connector. A Program Instance (see Software categories) must always be linked to this node.
- **Flow connector** (link): indicates the information flow between two components, only one flow is allowed between them and it is necessarily oriented. A condition or a probability can be associated with the transition, as well as the size of the exchanged data.
- **Delay node**: causes a delay in the information flow, this must be considered as a delay caused by events or objects outside the scope of the system description. It does not affect utilization statistics of any processor or storage device.
- **Initiator**: receives the workload stimulus and start the evaluation of the Execution Flow.
- **Terminator**: is the completion point of an information flow. Several end points can be used in the same Execution Flows
- **Hierarchical function**: inserts a subflows in the current Execution Flow description. It must refer to an existing subflow.
- **Response meter**: calculates the response time of a path between two points on a EF. They do not affect the behaviour of the system and can be seen as a non-intrusive measurements probe.

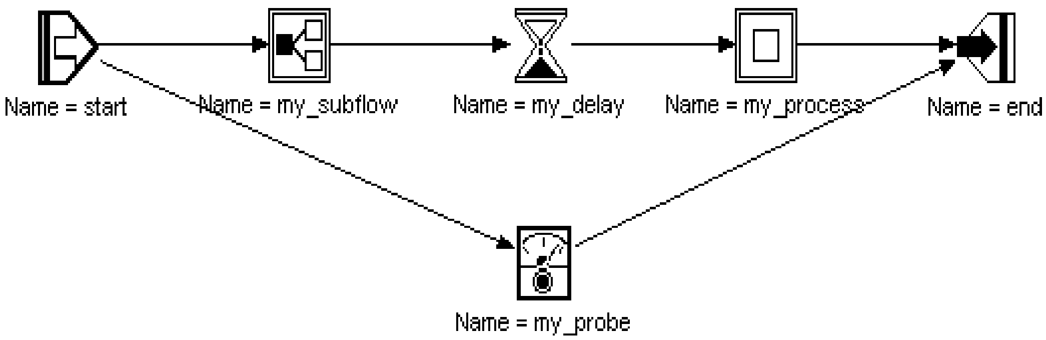


FIGURE 14. Control flow diagram

2. Software categories

In this section, the user describes the instructions that must be executed in every primitive function. A Program Content can be reused in any Program Instance. The latter has two arguments: a Program Content and a task.

- **Program Contents** ordered list of pseudo-code statements that specify resource demands on the computer system ‘s hardware (processing time, resource sharing, data access, synchronization), see figure 15.
- **Program Instance** describes the work done by primitive functions on EF and Subflows. This is an instance of a logical Program Contents
- **Operating system** defines the task management and scheduling policies, e.g.: SOLARIS, DOS
- **Tasks** are collections of Program Instances for system processing, data and communication resources.

```
Put 1 instances of <Main_noodle_bowl>
Allocate 1 instances of Data Element <Main_noodle_bowl>
Delay 1 seconds
Unlock <lock_main_noodle_bowl>
```

FIGURE 15. Program Content

3. Data and Synchronization categories

This part deals with the system’s data architecture and the synchronization resources.

- **Data elements** define the data quantity, which is transferred between processing and storage devices.
- **Data stores** logical groups of data elements. A data store belongs to at most one Storage Device.
- **Allocated resources** can be used for synchronizing Program Instance execution based on data availability.
- **Locks** synchronize the execution of resource demands between competing Program Instances.

4. Hardware categories

This section deals with the physical hardware implementations.

- **Hardware Architecture Diagrams** are hierarchic specifications that describe the characteristics and interconnections of all processing, storage, and communication devices in the system, see figure 16.
- **Processor Device Type** is a reusable definition of the instruction processing capacity (speed) of a hardware processor, e.g IBM PC, Sun Sparc.
- **Storage Device Types** describe the physical characteristics of storage devices such as the transfer rate, the latency and the capacity.
- **Communication Device Types** describe the physical characteristics of a hardware communication device such as Ethernet.
- **Communication Device Protocols** describe the logical data transmission characteristics of the communication device (packet size, overhead sizes), e.g. TCP/IP.
- **Data Paths** define the route data takes through the hardware architecture in response to inter-task communication between processing devices or getting and putting data between a processing device and a storage device.

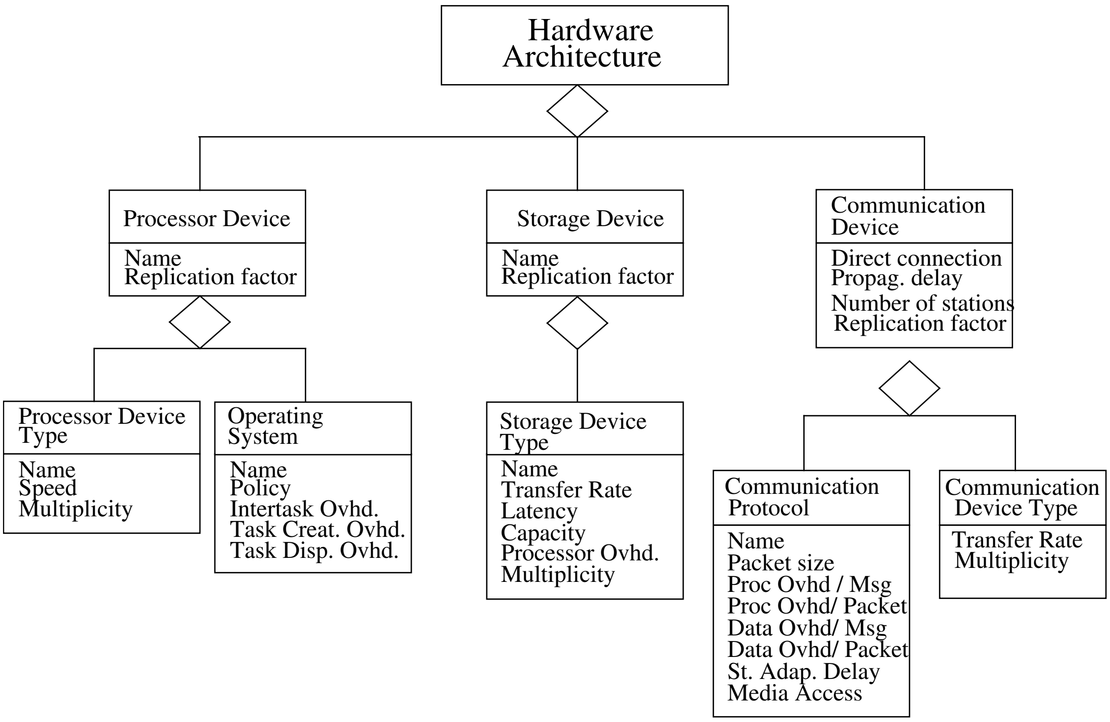


FIGURE 16. Hardware architecture diagram

Note: in order to keep a easy-maintainable diagram, it is possible to associate a multiplicity or a replication factor to create several processors having the same characteristics represented by only one icon in the drawing.

5. Data management

A few commands are provided to make possible the transfer of data between a Data Storage to the Processor Device.

- **Get** indicates the transfer of a data element from its storage device to the Program Content requesting the data. This statement only specifies the transfer load over the network and does not concern the storage capacity of the storage device.
- **Put** indicates the transfer of a data element from the Program Content of a processor to its storage device. This statement only specifies the transfer loading over the network and does not concern the storage capacity of the storage device.
- **Delay** is used to delay the execution of a Program Instance. No load will be placed on the processor executing this task by the delay.
- **Allocate data** removes a data element from its storage device
- **Deallocate data** replace a data element into its storage device

6. Operating system policy

Four policies can be selected, especially in the case of multi-tasking application:

- FCFS: First Come, First Served. The task priority is ignored
- Priority Non Pre-emptive: a task is completed once it is started, the next task to be executed will be the one which has the highest priority.
- Priority Pre-emptive: a task can be interrupted because another one with a higher priority is requesting the CPU. It can be destructive or not.

  - Priority Pre-emptive Destructive: oblige the task which has been interrupted to be restarted from the beginning.

  - Priority Pre-emptive Non Destructive: the interrupted task will continue from the point of interruption.

7. System Parameters

The user can specify global and local variables and constants to implement the different simulation parameters. The only accepted types are real or integer. Those values are accessible from any part of the system description.

- **global or local variables**

  The identificator of the global variables is g.<variable name>
  The identificator of local variables is as.<variable name>

- **constants**

5.2.2 Performing simulations

In this section, the available statistics and simulation parameters are presented.

1. **Simulation statistics**

A response meter can be added to an execution flow in order to compute the passed time between two points. The attributes described below are automatically available when one of the objects are created.

Object / Annotations	Flow Connector	Delay & Function Node	Response Meter	Processor	Storage Device	Comm. Device
Arrivals/Departures	Simulation	Simulation	Simulation	Simulation	Simulation	Simulation
In-process	Simulation	Simulation	Simulation			
Min,Max,Mean response time	Simulation	Simulation	Simulation			
Service time	Analytics	Analytics	Analytics			
Offered load				Analytics	Analytics	Analytics
Number in queue				Simulation	Simulation	Simulation
Min,Max,Mean Queue time				Simulation	Simulation	Simulation
Capacity utilization					Simulation	
Transfer utilization				Simulation	Simulation	Simulation

TABLE 11. Simulation statistics

2. **Simulation set-up**

The information resented above can be displayed during or after the simulation. Bar and XY charts are available to display the statistical values collected during a simulation.

Some simulation features are only useable for a complete simulation:

- Step-by-step simulation

  The user can pause the simulation and decide to go on event by event. Associated with a slow running mode it is also possible to follow graphically the evolution of the system.

- Random number

  Before each simulation the user can choose to reset the random number generator in order to get the same random values as at the previous simulation.

- Number of simulation events between GUI inputs/outputs

  In increasing those values the user makes the simulation go faster. On the other hand if he/she needs to watch the evolution of the system (outputs) or open other QASE windows (inputs), he/she has to lower them.

5.2.3 Analyzing results

Two kind of files can be used to track the simulation and the overall behaviour of the system.

- *The log file*

  This ASCII file contains all the system evolution such as the stimulus arrivals in the execution flows, the resource utilization for the processor, the storage device and communication device.

```
66.000  AllocRsrc       AllocRsrc       Msg_waiter_cook Utilization     100.000 %
66.901  Eating_process4 RespMeter       time_eating4    Completion      28.3980 seconds
67.001  Workload        Workload        Start_waiter    Arrival 1       Arrivals
67.000  Implementation  Disk    Food_Stock (capacity)   Utilization     0.200   %
67.000  Implementation  CommDev Network Utilization     20.100  %
67.000  Implementation  Processor       Waiter  Utilization     100.000 %
67.000  Implementation  Disk    Table   Utilization     0.702   %
67.000  Implementation  Disk    Table (capacity)        Utilization     0.761   %
67.000  AllocRsrc       AllocRsrc       Chopstick_12    Utilization     100.000 %
67.000  AllocRsrc       AllocRsrc       Chopstick_23    Utilization     100.000 %
67.000  AllocRsrc       AllocRsrc       Chopstick_34    Utilization     100.000 %
67.000  AllocRsrc       AllocRsrc       Chopstick_45    Utilization     20.200  %
67.000  AllocRsrc       AllocRsrc       Chopstick_51    Utilization     40.100  %
```

FIGURE 17. Extract of a log file

- The trace file

  This is another ASCII file which contains the information displayed in the trace window during the simulation. These information are provided by the trace functions, which allow the user to print out sentences, comments or values when a transition flow is triggered. It improves the visibility for model validation, but they have to be customized by the user.

```
Cook>There is enough noodle, time=60.901005
5>No right stick available
2>*******The philosopher get a right chopstick, time=61.502007
1>The philosopher has got a full bowl, time=61.703003
5>The philosopher is releasing left chopstick, time=62.702000
3>The philosopher has got a full bowl, time=64.701988
5>No right stick available
Waiter>Chopsticks for philo 4 released
4>Philosopher has released the chopsticks, time=66.900986
4>That was a good meal, time=66.900986
4>Philosopher needs a rest now,time=66.900986
4>Philosopher needs to think now,time=66.900986
5>The philosopher is releasing left chopstick, time=67.300980
Cook>There is enough noodle, time=67.500977
Waiter>Chopsticks for philo 2 released
2>Philosopher has released the chopsticks, time=67.902969
```

FIGURE 18. Extract of a trace file

In addition, two sorts of graphics can be obtained:

- A bar chart can be drawn through the Online graph window after the simulation in order to display the simulation statistics.

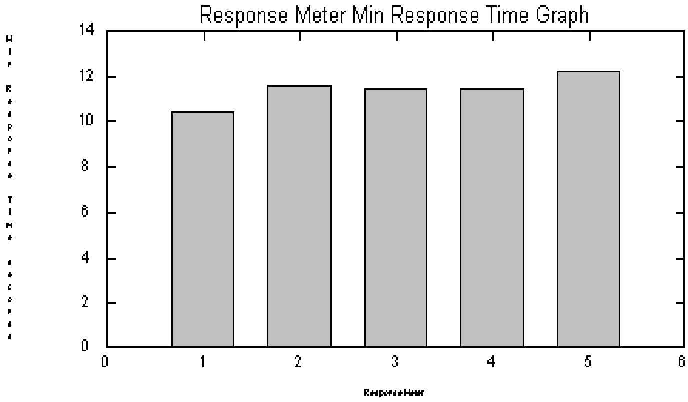


FIGURE 19. Bar chart

- A histogram or XY line chart can be used to see the evolution of objects utilization versus the simulation time. This kind of graphic is produced using the log file.

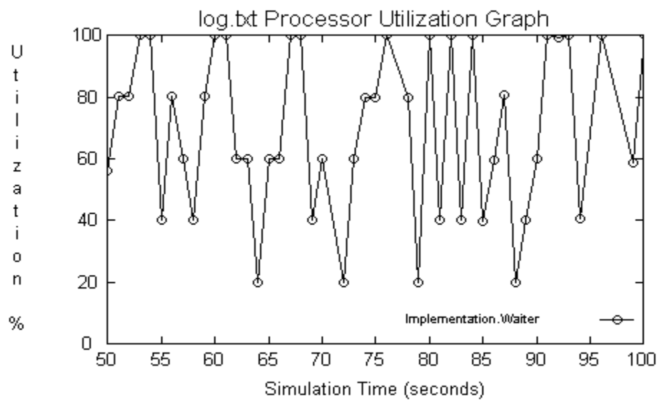


FIGURE 20. XY line chart

Various annotations, see figure 21, can be added on the drawing. They contain the value of the previous simulation results and can indicate, for example, how many times a flow was triggered and many other statistical values belonging to a component.

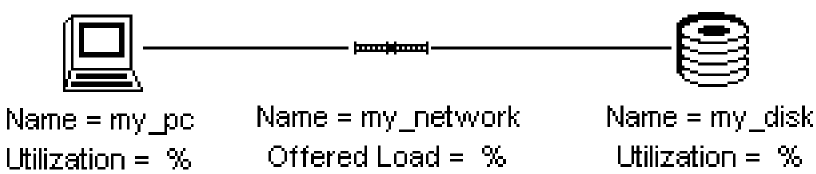


FIGURE 21. Annotations

5.2.4 Simulation parameters

A lot of pre-defined dynamic attributes are available during the simulation. Most of them are associated to a particular object (device, lock, resource,.) and keep track of its current performance estimates.

Some attributes are:

- the current value of the simulation clock
- the min, max, mean time that stimuli have spent in queue
- the number of requests currently in process at a device, task or allocated resource (useful for testing a monitor's value)
- the number of requests currently queued at a device, task or allocated resource

5.2.5 Resource management and contention

There are two ways to coordinate the data access: the monitor and the semaphores. Both are no significance to any specific data store or storage device, they are virtual entities, which are used during a simulation. The analytical analysis ignores those entities.

- **Monitors**

  The *allocate* statement reserves a certain quantity of resource. If that quantity is unavailable, this request is queued and the next instructions are blocked until the requested quantity becomes available.

  At the opposite, the *deallocate* statement releases a certain quantity of resource, allowing the blocked process to continue.

- **Semaphores**

  A lock can be applied in a Program Content to access a particular data, but the user must be aware that a lock does not belong to a data. Also no check is performed by the simulator to prevent the user from deadlocks due to lock mismanagement. Combined with a global variable, it is possible to create a semaphore and define the common functions P() and V().

5.3 Ida Network Model Evaluation

The main purpose of this model was to evaluate the ability of QASE RT to represent a large network with several layers of decomposition. As this software is dedicated to analyze a network from a high-level of abstraction, very few physical characteristics have been taken into account. In addition the usual traffic is by essence bursty and physical measures are gained with difficulty. Therefore, no real simulation has been made on this network. Nevertheless a good assessment of QASE's scalability has been obtained from the modeling phase. In this part, a condensed graphical representation is given, a more detailed and readable one appears in the appendix.

Three hierarchical levels have been spotted, the first one presented below, shows the separation between the two main network and the link to the university backbone.

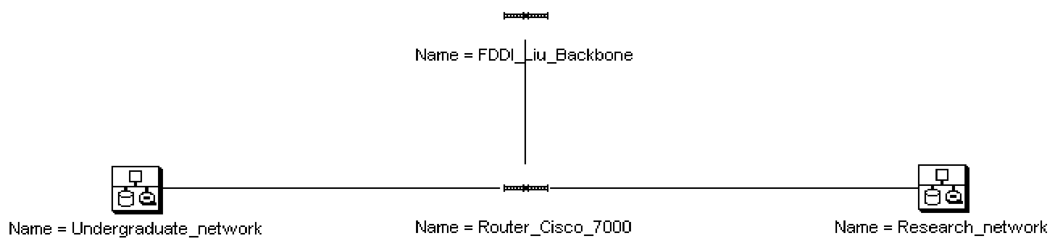


FIGURE 22. QASE RT/ IDA network / overview

1. Research network

The research network is conform to the physical model previously described. The three switches support the connections from the various workstations. A feature of QASE RT that allows to use one icon to represent a higher number of devices having the same behaviour and being connected to the same network device.

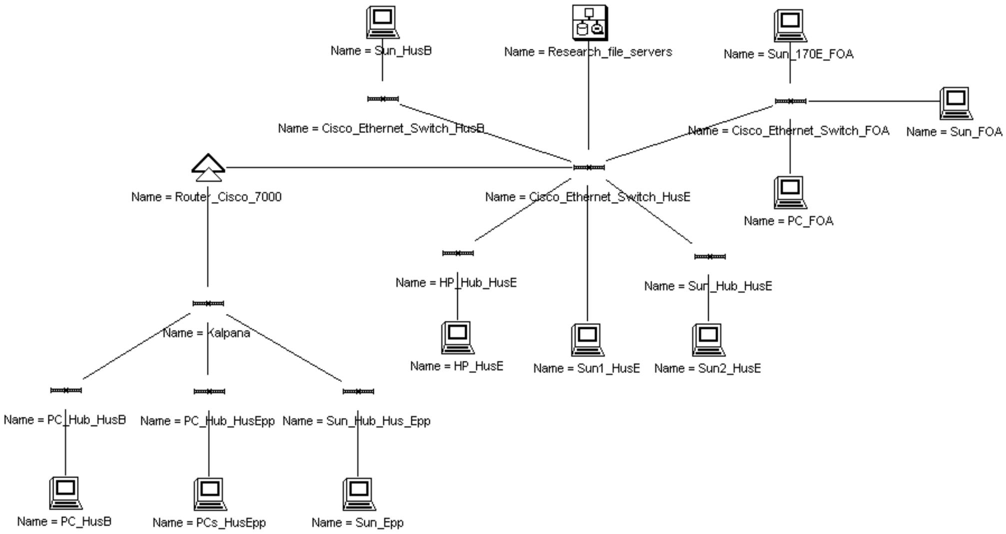


FIGURE 23. QASE RT/ IDA network / research network / overview

The third level of abstraction, illustrated by figure 24, is the file servers model.

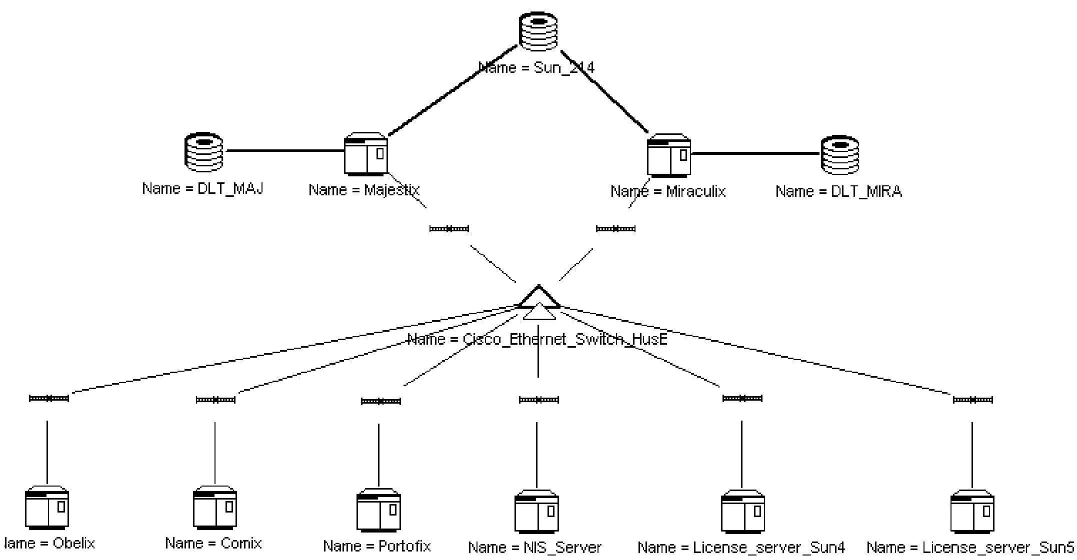


FIGURE 24. QASE RT/ IDA network / research network / file servers

2. Undergraduate network

The undergraduate network is also based on an Ethernet switch called *Nbase*, pictured in the middle of the drawing, figure 25.

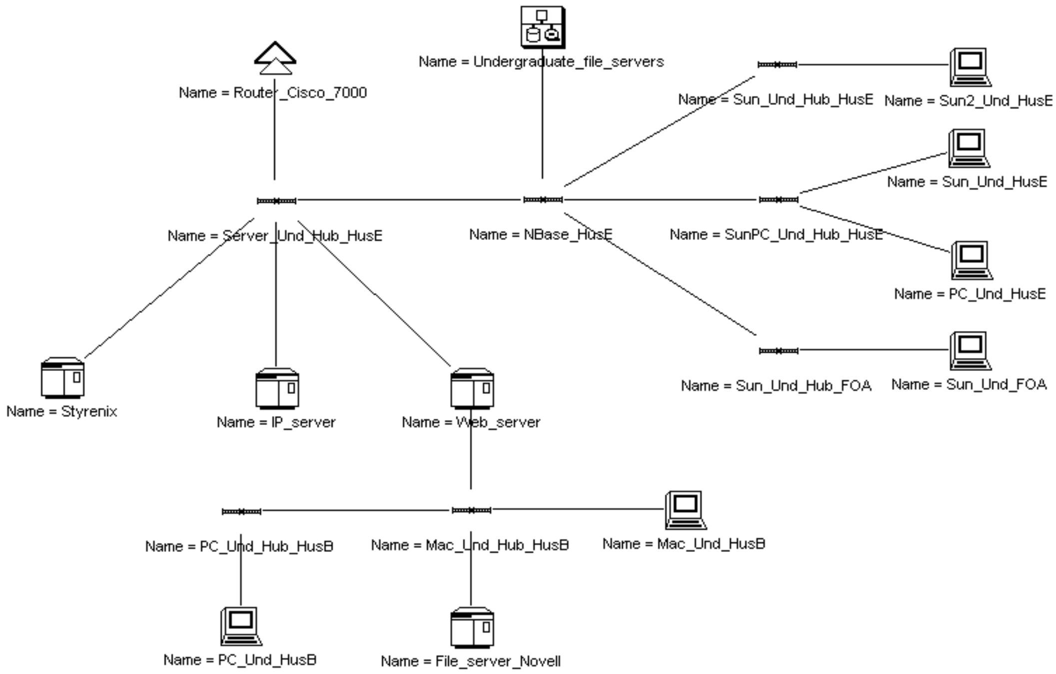


FIGURE 25. QASE RT / IDA network / undergraduate network / overview

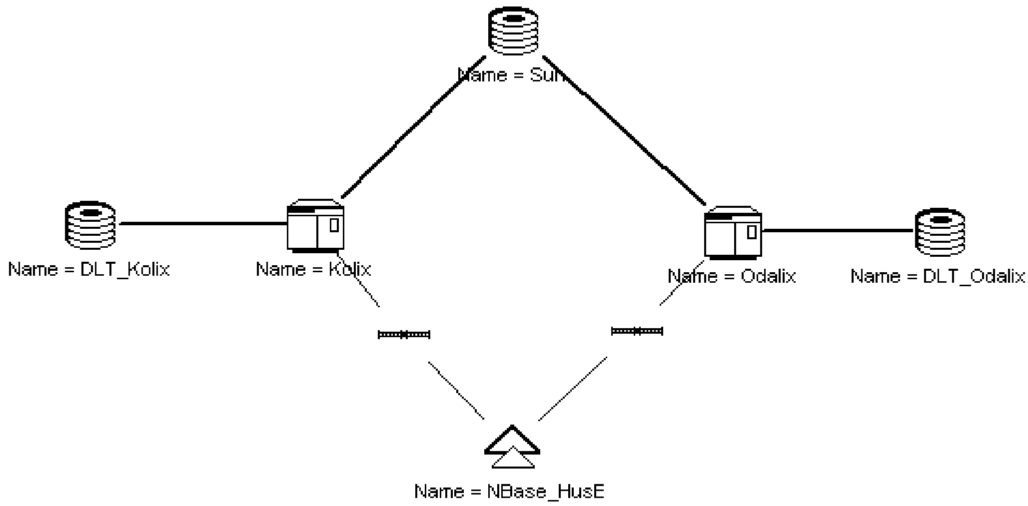


FIGURE 26. QASE RT / IDA network / undergraduate network / file server

5.4 Client-server Application Evaluation

QASE has shown its ability to represent a multi-level network architecture and the possibility to represent several identical components. This is a very convenient feature when an huge number of devices are considered, this drastically increases the readability of the diagrams and make corrections and updates really easy. At that point, the scalability of the tool has been tested only. A more complete evaluation is realized in the Client-server application model.

5.4.1 Model architecture

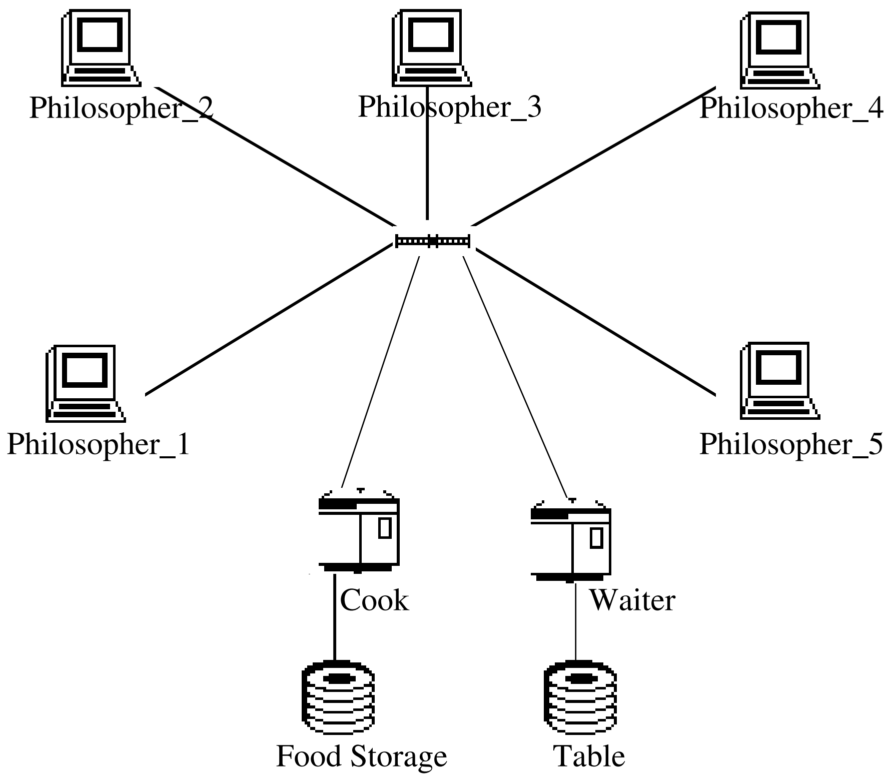


FIGURE 27. QASE RT / Client-server model / overview

This model is made up of five workstations, one for each philosopher, two processors, one for the cook and one for the waiter. A network interconnects the different components and finally a storage device (Food storage) is linked to the cook and another one (Table) to the waiter.

1. Hardware considerations

Most of the hardware characteristics have been entered using constants. Therefore it was easy to change the various parameters such as the transfer rate or the processor speed to perform different simulations. To simplify the model, every processor or workstation has the same technical characteristics as the others. The waiter has to handle six tasks in parallel, i.e. one for the cook's requests and one for each philosopher.

2. Software considerations

The code to be executed was entered in the Program Instance and Program Content The commands offered are limited to data access instructions, network loading and monitors positioning. As each philosopher requires distinct chopsticks (=resources), it was inadequate to call the same Program Content (block of instructions) in the philosophers' Program Instance. There is no instruction allowing the user to specify dynamically which resource has to be locked or unlocked.

3.Workload considerations

Many subflows have been created in order to improve the visibility of the global model. To describe the philosophers' behaviour, four level of abstraction were in turn considered:

- 1st level: [**Overall view]** Test cyclically the end (philosopher's death) of the task
- 2nd level: **[Everyday life]** Run ponder -, eating process and starvation timer
- 3rd level: **[Eating process]** Run noodle waiting and eating noodle process
- 4th level: **[Eating noodle process]** Run left, and then right chopstick waiting

The trace functions, contained in the transition flow's formula, turned out to be very convenient. It allows to scrutinize each important transition (more convenient than each simulation step), get the value of a variable, a system parameter or the current simulation time. To avoid overloading the drawing with annotations the only information displayed were the number of stimulus arrivals on a transition flow.

4. Inter-process communication

- Philosophers and Cook to Waiter

  The requests from the philosophers and the cook are sent to the waiter through the network according the following protocol:

  - The waiter has 6 tasks, running in parallel. Each of them is initially in idle state waiting for a request from its customer. This is realized by using a monitor (MSG_Cook or MSG_PhilosopherX, with X=1,2,3,4,5).

  - A philosopher or the cook send a request to the waiter, the command is executed (Put statement) to simulate the network communication. Then the same monitor is used to signal the waiter that it can execute the instruction requested by the message.

  This synchronization by a monitor is not optimal, but there is no other way to signal that a message was sent and arrived to the recipient. In addition, the monitor manipulations are 'logical entities' that do not cost anything in terms of CPU, resource utilization or network load.To simplify the message communication, the size of each message is fixed and set to MSG_DATA_SIZE.

  The waiter takes into account a message every second, whereas a message can be sent from the cook or philosophers at any time

- Waiter to Cook and Philosophers

  The same principle is employed during the communication from the waiter to the cook and the philosophers. Another kind of monitors (Msg_Waiter_Cook, Msg_Waiter_PhilosopherX, with X=1,2,3,4,5) allows the synchronization between the sender and the recipient.

  The waiter can send a message at any time and this message is taken into account by the cook or philosophers at any time

Important note:

As it is not possible to simulate data exchanges directly between two processors, since the *Put* or *Get* statement refer to a data stored in Storage Device, the solution was to use a data called MSG, stored in the Table Storage Device and then make a call to the *Put* and *Get* statements in each communication. The main drawback of this is that the utilization rate of this device will be affected artificially (but not the capacity utilization). This feature is missing because QASE RT is primarily Client/Server oriented more than network communication oriented.

5.4.2 Simulation preparation

The simulations were performed only with the Simulation command. The analytic simulation could be used because it does not take into account the resource contention (locks, allocated resources, tasks buffer).

1. **Time statistical analysis**

Three response meters were used for each philosopher to probe timing issues and answer the following questions required in the simulation protocol:

- How long must a philosopher wait before getting a noodle bowl?
- How long must a philosopher wait before getting the two chopsticks?
- How long does it take for a philosopher to eat?

In all cases three values were obtained: minimum, maximum and mean time

2. **Real-Time constraints**

- Chopsticks

  The five chopsticks are represented by five distinct semaphores, using the following semantic: the left chopstick of the philosopher 1 was named Chopstick_12, since it is located between Philosopher 1 and Philosopher 2. The operations on the semaphore were *allocate* (equivalent to the wait() function) and *deallocate* (equivalent to the signal() function). In addition the variable ar.Chopstick_12.nIP (property of the semaphore) contains the number of the current requested access to this semaphore.

- Main bowl

  The synchronization was made using a lock named *Lock_Main_Bowl.* It aims at preventing other processes from accessing the global variable, *g.Main_Bowl*, which contains the current value of the semaphore. The lock and unlock statement allows the user to define the critical section in the Program Contents.

- Noodles

  The contention access to the food storage is not so relevant since there is only one task allowed to perform operations on it. Nevertheless, in case of further extension to this model, a semaphore, named *Noodle*, is used to restrict the access to the number of noodles available in the pantry.

3. **Network constraints**

The main attributes of a communication device are the transfer rate, the packet size and the media access (for an overview of all the attributes, see figure 16). It was assumed that the packet size and media access belongs to a given protocol. Therefore the easiest means to change the network behaviour, simulate bottle-neck and act on the messages communication remains the transfer rate. This was done with the Transfer_rate_network parameter.

5.4.3 Experiments

In this section, three QASE experiments are discussed.

The first simulation was made under normal conditions, no constraints were applied on the network, i.e. infinite transfer rate. The aim of the simulation was to evaluate the QASE abilities as a discrete-event scheduler under real-time constraints. This means using the messages communication and the data concurrent access management.

The second simulation aimed at proving the effect of the simulation time on the results.

Then the model was tested with network constraints to create queues, bottle-necks and find out how they can be detected with the software's features.The network is supposed to be error-free, that implies that no message is lost, corrupted or altered. Nevertheless it can be delayed according to network throughput.

1. Hardware description

The experimentations have been undertaken using the following configuration:

- Software version: QASE RT v. 3.2.1 for Windows NT
- PC: Compaq Workstation Pentium Pro, with 64 Mbytes in RAM
- O.S: Windows NT 4.0
- Model file size: about 4 Mbytes.

2. Normal conditions

The first simulation aimed at showing the QASE abilities as a pure discrete-event scheduler. In order to do so, the data size value was set to zero and the transfer rates to the highest value. This tool provided two ways to track the data flow, the resource and processor utilization.

- The log file can be useful to obtain utilization percentage event by event. But the size and complexity of this model dilute the information.
- The trace file contains the kinds of information the user decides to put into it before the simulation. This was particularly useful to notice when a philosopher got both chopsticks, released them or was waiting for a bowl. These was only some of the numerous printout the user could choose to implement. It is also possible to add time stamps to each printout using the system variable system._simTime (or s._sTime).

Qase is particularly well suited to implement multi-tasking programs. Indeed it is possible to define different O.S. policies, inter-task and task creation overhead.

This first example illustrates the effect of the O.S policy on the model behaviour. The simulation was performed for 600 seconds and the results were compared to a 2000 second simulation. It turned out that only a few values were slightly affected. Therefore the first one will be solely presented. The philosophers and the cook run only one task at a time, that is why they were not concerned by the O.S. policy changes. On the other hand, the waiter handles six tasks running concurrently.

In this example a priority was assigned to each task handled by the waiter, a low ranking implies a high priority. The cook received the highest priority of the example(0). The statistical measures and additional graphics are presented in the appendix.

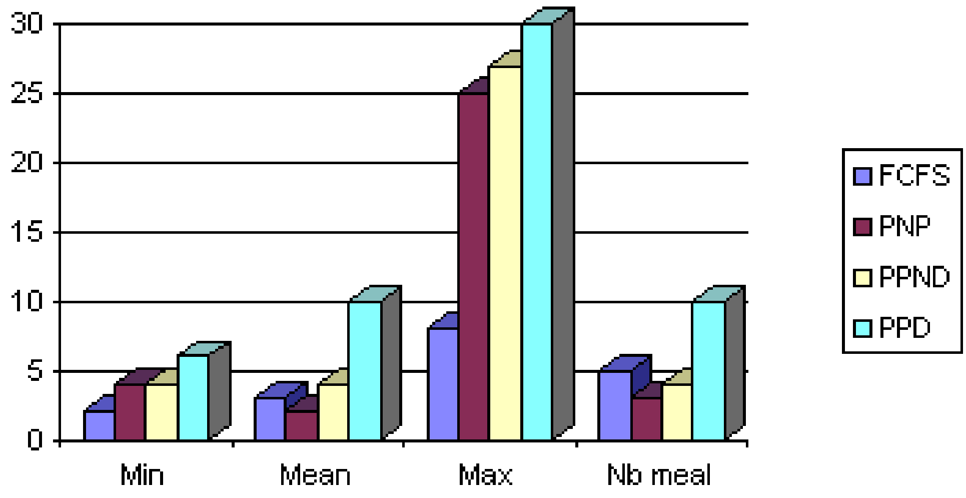


FIGURE 28. QASE RT / Client-server application / normal conditions

Legend: FCFS: First come, first served
PNP: Priority Non Pre-emptive
PPND: Priority Pre-emptive non destructive
PPD: Priority Pre-emptive destructive

Results analysis:

The priority with pre-emptive destructive policy gives the most important timing differences between the tasks. This can be explained by the fact that a interrupted task has to restart from the beginning to be completed, but when this one is about to be executed again, another task with higher priority can stop it again.

The FCFS policy is the fairest since it implies the lowest timing differences. This is due to the implementation of the problem, in which all the philosophers have to execute the same amount of actions and have the same rights.

The priority non pre-emptive and pre-emptive non destructive give about the same result, but this is due to the small execution time associated to the tasks. When this one is increased, the maximum eating time for PPND is 60% more important than PNP (the minimum eating time is about 30% bigger).

3. Influence of the time on the results

The following simulations aimed at proving the effect of the simulation time on the results and bring a few directions regarding the choice of the optimal simulation time for a given model.

Description of the experiments:

The next figures below have been obtained in using a discontinued simulation process in fact every 30 mn the simulation was stopped manually the statistical values were written down and then, the process was resumed. These information are available in the log file, but the size of this file is very huge after a 3-hour simulation,. Nevertheless it remains an alternative possibility.

- The warm-up time (or start-up time)

  The warm-up time indicates the first simulation period when the system steady state is not reached. In our case, this involves various setting times such as the cooking of the first batch of noodles, the contention for the first noodle bowl and chopsticks. On the first diagram, which represents the difference between the highest and lowest mean eating time of all the philosophers, it can be noticed that from 10 mn to 1 h, the mean value can give important figures especially if the degree of fairness is low (PPD policy).

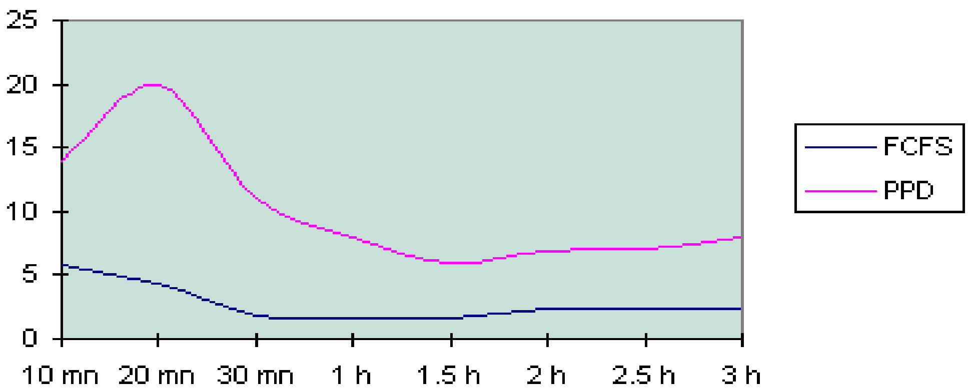


FIGURE 29. Difference between the highest and lowest mean eating time of all the philosophers vs time simulation

  Not considering this time period could lead to unexpected statistical results. That is why the data collected during the warm-up time have to be discarded. This can be easily done with QASE RT in setting the probe’s starting time after a pre-determined warming-up time. However, this operation has to be carefully made in relation to the simulation goal. If the goal aims at, for instance, reckoning the network load, the warming-up values have to be taken into account. In addition, sweeping up those values as stated before let assume that the system travels monotonously towards its steady state.

- How long should a simulation last?

  The answer to this is a compromise based on three considerations:

  1- The simulation time has to be greater than the warming-up time to avoid the side-effects or start after it.

  2- The simulation time should not be too short otherwise not enough relevant data will be gathered.

  3- The simulation could run for hours until the computer suddenly breaks down and erases the statistical counters' values.

In the following figure, the maximum eating time of the philosophers were compared under two configurations: in the first case, First Come First Served, the fairest policy under a normal load and in the second case Priority Pre-emptive Destructive the less fair one.

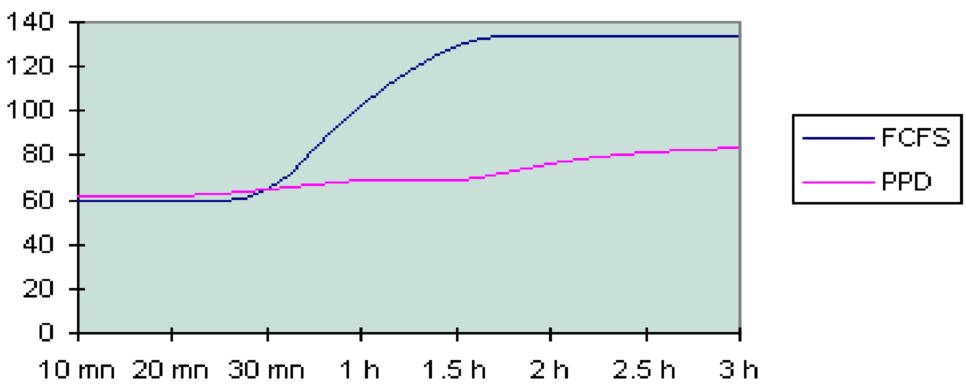


FIGURE 30. Maximum eating time of all the philosophers according the simulation time

The longer the simulation lasts, the more numerous the executed combinations are and the more unexpected cases can be unveiled. With a PPD policy, a 3-hour simulation only proves that the maximum increases steadily and a longer simulation is undoubtedly required. On the other hand with a FCFS policy a peak was reached rapidly and an analysis of the time stamps components can indicate if its a worst case or not. In that case more combinations have to be performed.

4. Network overload

Description of the experiments:

The next step is this study was to evaluate the network when it is overloaded.

As QASE software is not strictly based on the physical aspects of a network, parameters such as the number of collisions and others protocol statistics are not available.

Nevertheless it offers the possibility to test the model in varying one or several model's parameters. In the following examples, the size of the messages exchanged through the network is ranged from 0 to 200 bytes, whereas the transfer rate stay the same. The simulation lasts one hour using the *incremental variation* command. Knowing the maximum message size can be very relevant if you want to avoid saturating the network and slowering its global behaviour and exceeding the time constraints.

The two following diagrams show the network bottle-neck and the communication delay generated

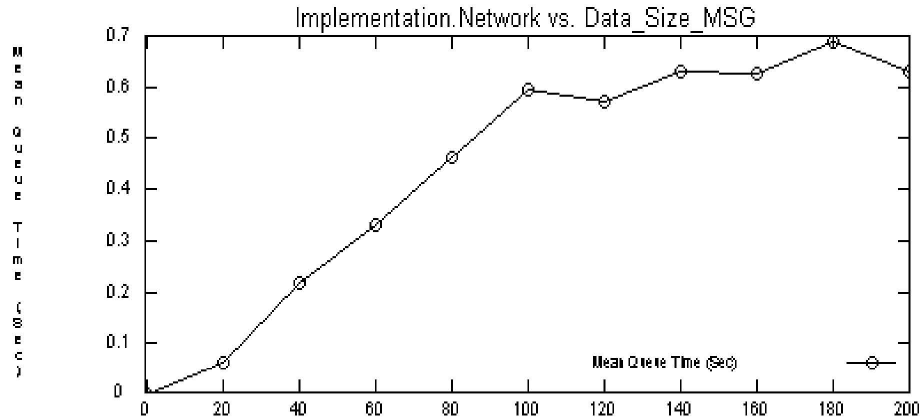


FIGURE 31. Mean Queue Time vs the message size

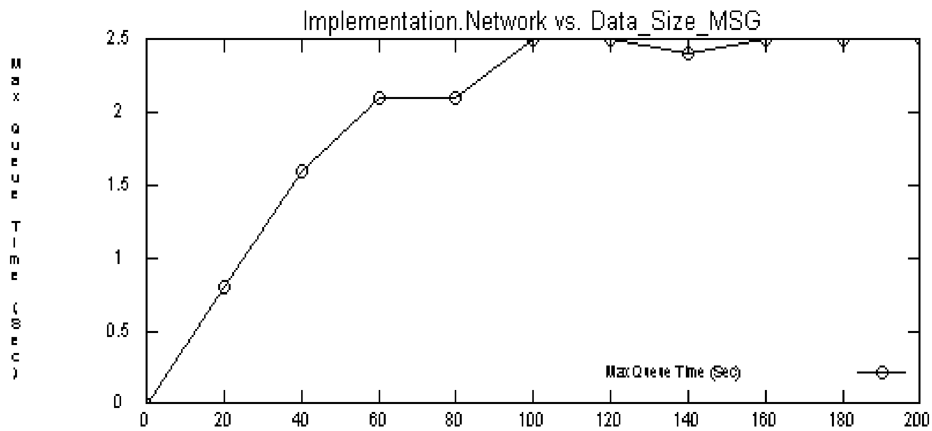


FIGURE 32. Max Queue Time vs the message size

5.5 Overall assessment

QASE RT has shown a strong ability to model both kind of networks in the system architecture as well as the system workload. The system description provides enough features to model a network properly from a high level of abstraction, i.e. without paying too much attention to the physical characteristics of the connected devices. In addition, its graphical user-interface makes easy the modelling. Other positive aspects of QASE are the hardware description editor which allows model breakdown into subnetworks and the workload description which offers a smart interface to specify the traffic over the network. The ability to simulate different operating systems give the opportunity to carry out studies about the event scheduling.

Regarding the simulation, all the useful features have been gathered in this software to provide efficient statistic data collections. The built-in trace tools allow to control very easily the workload during the model, highlight the bottleneck and the devices utilization.

Some negative points must be mentioned as well.

Firstly, the user has to keep in mind that in considering the network without emphasizing the physical attributes of the devices or the communication, a bigger independence is given to the model. Nevertheless the obtained values is obviously less accurate. This implies that the user has to be very cautious when the characteristics of the hardware is entered. He has to take into account the approximations to modulate the values of these parameters.

Secondly, QASE RT is dedicated to a particular kind of workload, this is mainly removing data from a storage device or adding data to this kind of device from a workstation. This is of course well suitable for most of client/server application. The application modelled in this project has shown, however, that it could become quite tough to model a workload whose execution depends on an elaborate algorithm, for example no value can be transmitted from a processor to another, which is a main drawback in a distributed system when you are trying to make process talk to each others.

Another weakness of QASE RT is the number of pre-defined components, which does not eliminate the guesswork of the hardware device, such as a workstation, a hub.

To sum up, QASE RT can be quite valuable in the first design phases of a network or to have a rough idea of the behavior of a network before realizing some changes. When it comes to have precise values within the scope of a performance analysis or in the case of a dynamic network workload, QASE RT turns out to be unsuitable for this kind of tasks.

Chapter 6

OPNET EVALUATION

6.1 OPNET presentation

OPNET is a tool for design and analysis of information systems, communication network and distributed systems through simulation. It provides a graphical environment for model specification, simulation, data collection, and data analysis. The tasks of the OPNET users is divided into two categories: the technical problem simulations and the preparation of the customized models environment. OPNET domain of application ranges from standard LAN (Local Area Network) to Mobil radio network through satellite network.

OPNET's main features are:

- an object-oriented approach
- hierarchical models
- development of detailed custom models
- performance statistics
- post-simulation analysis tool

This software is based on a discrete-event modeling approach. The events can be triggered according to the user specification (ex: scheduling event) or be the result of an action. This is for example the receipt of a message, the expiration of a timer, the indication of a resource availability, the indication of a completion of a task by a resource, the failure of device, a change in a statistic that is monitored by a process.The event management is made using the Event package. This is a collection of functions that masks the internal data structure of the event-list. It also provides information about the scheduled events such as the id and type of the current event, the next and pending event.

6.2 OPNET's system description

The tool assists the user in the three following phases of the system description:

- Model building and configuration
- Performing simulations
- Analyzing results

In this section, each phase will be successively described and application examples will be provided later through the study of two models.

6.2.1 Model building and Configuration.

The model building consists of specifying how the network is physically implemented, how it works and what kind of traffic flows in it. OPNET offers three features to achieve this task:

- Network editor: choice of a network topology
- Node editor: definition of the data flow models
- Process editor: definition of the control flow models (behavioural logic)

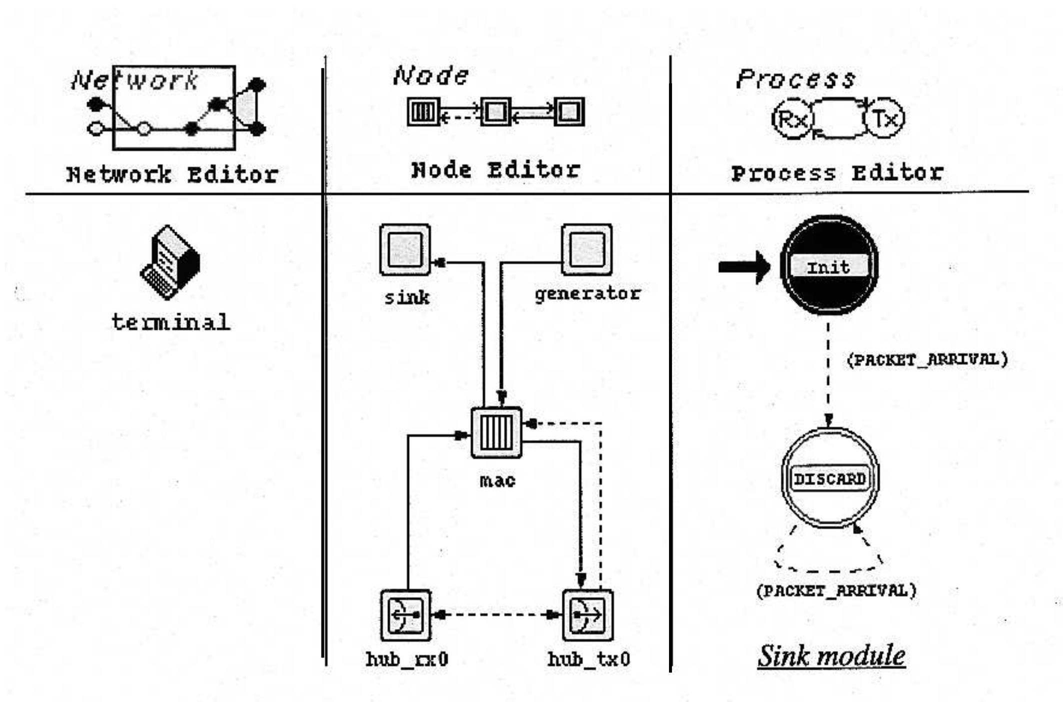


FIGURE 33. The three hierarchical description level

Major elements of network design which can be modelled include: network topologies, network components and various communication links. The behavior of each of them can be realized using finite state machines described by a state-transition diagram.

To facilitate the modeling process, pre-defined objects of the model library can be used directly or edited and modified to suit the application.

Each of the editors give access to a particular domain, respectively Network Domain, Node Domain and Process Domain, which can be seen as an abstraction level of the model.

1. Network domain definition

This section deals with the overall scope of a system to be simulated. It's the highest level description of the objects contained in the system. It specifies the objects in the system as well as their physical locations, interconnections and configurations. Three classes of objects can be used: a subnetwork, a communication node and a communication link.

- **subnetwork** is a container that abstracts the network components specified in it. It can contain other subnetworks. This provides an mechanism to represent complex networks and to break down the system complexity.
- **communication node** represents a network device which could be a predefined workstation, hub, server, bridge or terminal as well as a user-defined node whose behavior is described in the Node Model domain
- **communication link** allows communication between nodes in the form of structured messages, e.g. point-to-point, bus, radio.

The diagram below represents three network components communicating together with a duplex point-to-point link.

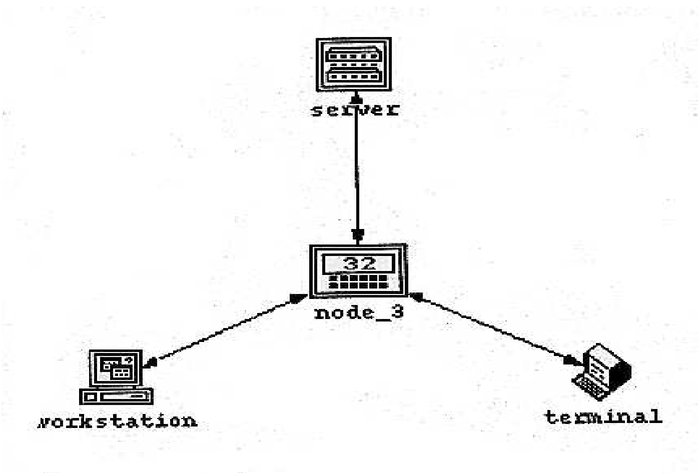


FIGURE 34. Network domain

2. <u>Node domain definition</u>

A node consists of one or several modules implementing the functionality of the node. The following types of modules are available:

- **processor module** is used to execute general process of data packets A typical processor can receive a packet on an input stream, perform processing and send the packet on an output stream. The actions to be performed by this module can be detailed in the Process domain.
- **queue** is a processor module containing internal resources called subqueues. The purpose of those subqueues is to facilitate buffering and managing a collection of data packets. The number of subqueues can be defined by the user and the capacity of each of them is unlimited, if not specified.
- **transmitter** (point-to-point, bus, radio) serves as the outbound interface between packet streams inside a node and communication links outside the node. It acts as an output port to which a communication link of the corresponding type is connected.
- **receiver** (point-to-point, bus, radio) serves as the inbound interface between communication links outside the node and packet streams inside a node. From the point of view of a network model, it acts as an input port.
- **generator:** is a traffic source offering a mechanism to model a probabilistic packet source. The time between successive packets is controlled by a probability density function (PDF). Two varieties of generators coexist: asynchronous and synchronous.

The following example describes the work of a particular terminal (see figure 34), the sink and the generator represent two processor modules, which respectively discard and send packets. A transmitter and a receiver are used for the outside word communication and a queue module, MAC, queues the incoming or outgoing packets.

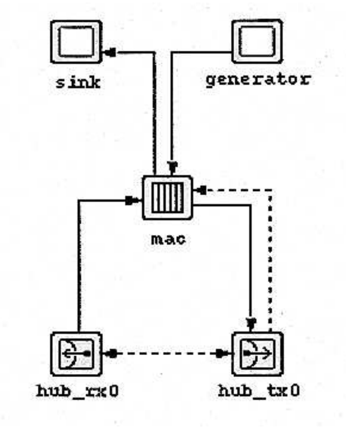


FIGURE 35. Node Domain

3. Process domain definition

In the two previous steps, the models of the components library have been used and customized a system. The process description can be viewed as the highest level of detail. The data specification, the functions to be performed, and the general behavior of a node has now to be modelled.

The general behavior of a node can be model through a state-transition diagram also referred as finite state machine. This kind of representation consists of two basic component types: states and transitions. States are used to represent the actions at a given time or according a given order. Transitions specify the changes in state that are possible for the process. The models have to be deterministic, i.e. only one transition from a state can be started. In addition, the states within a model are mutually exclusive. As a result, a process is always in exactly one state.

States and transitions define the progression of a process in response to simulated events. Actions can be implemented in each state using one of the 350 pre-defined functions. The embedded Proto-C featuring C-ANSI data structures and functionalities.

The pre-defined functions, called Kernel procedures, are classified in various packages ranging from the visual representation of a simulation to the interruption analysis.

Traditional state-transition diagrams represent the state of a system by identifying which state is occupied at a given time. This could be a drawback for systems with dynamically expanding states. Proto-C offers the possibility to declare dynamically new processes and allocated memory.

The user is offered several ways to declare data structures and variables:

- Constants set static values of the process or define macros testing, for example, the arrival of an event using Kernel procedures.

ex: #define PACKET_ARRIVAL (op_intrpt_type() == OPC_INTRPT_STRM)

- Temporary variables can be declared,. The values which has been assigned to them is lost when the state-transition diagram evolves from a state to another one. This is particularly well suitable for the data that do not require persistence such as the loop indexes.

- State variables are persistent. This implies that they retain their values over time. The state variables are private for each process. In other words, changes made to a state variables in one process do not affect the values of variables held by another.

- Global variables provide a way for multiple processes in different modules to store information in a common location. This feature comes from the C language. External declarations are identified with the keyword *extern*. As such a data access is not necessary a realistic representations of the actual system, it has to be used with caution.

- Process model attributes increase the reusability and flexibility of a given process model introducing changeable parameters to implement variable behavior such as the application start time and time-out duration.

- Simulation attributes introduce parameters in the model which will be only taken into account at the simulation level. Another advantage of this kind of attribute is that many values may be specified for each new simulation. This provides a mechanism for iterating the system model through a series of experiments This point will be re-explained in the section dealing with the model simulation.

As illustrated in figure 36 below, which represents the sink module (see previous figure), a state-transition diagrams always involves a initial state, here called ‘INIT’. The user-defined event ‘PACKET_ARRIVAL’ allows a transition from the two states. A state can have a looping transition.

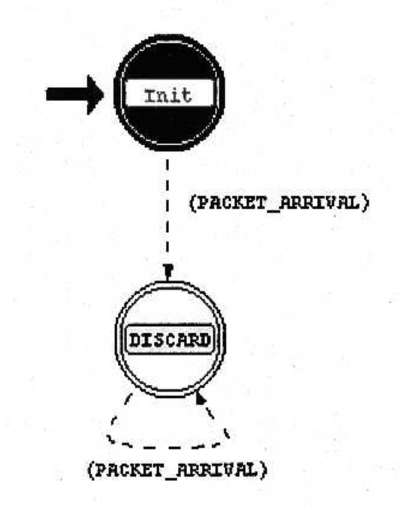


FIGURE 36. Process Domain

6.2.2 Performing simulations

Three tools are available for performing simulations:

- The Simulation tool customizes the simulation of the model.
- The Debugging tool gives information during the simulation.
- The Probe specifications editor allows the user to select or customize the probes used to collect statistical simulation values.

6.2.3 Analyzing results

Two tools are provided to analyze the simulation results:

- An analysis tool allows the user to view the statistics collected during the simulation in the form of graphs.
- An Animation viewer can show the data flows or a finite state machine evolution during a simulation.

6.2.4 Simulation parameters

A process model may define parameters called attributes, that are used once it is instantiated. Two kinds of attributes have to be differentiated: the model attributes and the simulation attributes.

- A model attribute is used to make a process reusable in another context. The value of this attribute can be provided at the Process Domain level or be promoted to a higher one (Node and Network Domain). This allows the user to control the behavior of such a process from a higher abstraction level and without any ‘hardwiring’. This increases its reusability and maintainability. In addition, type specification and range values can be set for this attribute in order to protect it against misuse.
  Example: initial value of a semaphore.
- A simulation attribute is related to the simulation itself. This could be a parameter whose value could eventually change according to the desired simulation. Using simulation attribute allows to assign either a value or a range of values to it and then, run a simulation without having to modify and recompile the models.
  Example: a delay for a given event or a packet size

6.2.5 Resource management and contention

1. Process synchronization

It is possible to check that a processor or a queue is busy and transmit this piece of information to another processor. This kind of operation can be realized using a ‘Statistic Wire’. This can be seen as a specific communication link between two processors (see picture 37).

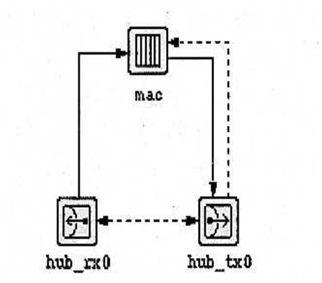


FIGURE 37. Statistic wire

2. Semaphores

OPNET does have any pre-defined model, that could be compared to a semaphore. Nevertheless the most common strategy for modeling resource management is to create a queue module in the node domain to represent each resource. Then it is possible to reproduce the inner-behavior of a semaphore. The execution of the instructions within a state is atomic from the first instructions to the last one, except if one of them specifically forces an event.

In the semaphore model used in this thesis the basic semaphore's properties have been implemented.

- One request can be handle at a time.
- A private variable is updated according to the request type value (release or take).
- If a take-request arrives and if there is no more resource available, the request is queued.
- When a resource is released, the queue is checked and a queued request can then be satisfied if there was one.

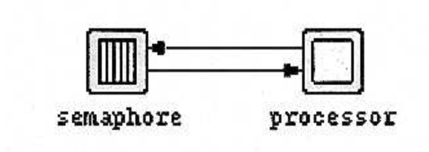


FIGURE 38. Own-defined semaphore / Node Domain

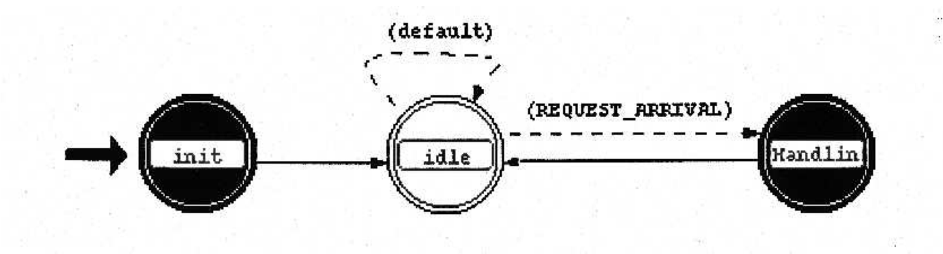


FIGURE 39. Own-defined semaphore / Process Domain

6.3 Ida Network Model Evaluation

This task has not been accomplished yet. The main reason is that components, such as a Ethernet switch or a back-up device, are not provided by OPNET in the components library. This implies that these components have to be analyzed and then reproduced in new modules. Due to time constraints this modelling was not completed at the time of writing this document

6.4 Client-server Application Evaluation

This model was the mean to evaluate the building, modeling and simulating capabilities of OPNET. That is why an extensive use of the various built-in features of this software has been made. In the next section, a description of this model is presented implemented

6.4.1 Model architecture

In accordance with the conceptual model previously described, the architecture of the system involves:

- <u>from the processors side</u>: five terminals, which were preferred to the pre-defined workstations to an easier model adaptation, and two servers. The two data storage devices have been placed in the server.
- <u>from the communication side</u>: a 8-port hub, which will obviously offer enough ports to handle the communication between the processors. The connections are realized using an duplex 10baseT cable supporting an Ethernet protocol.

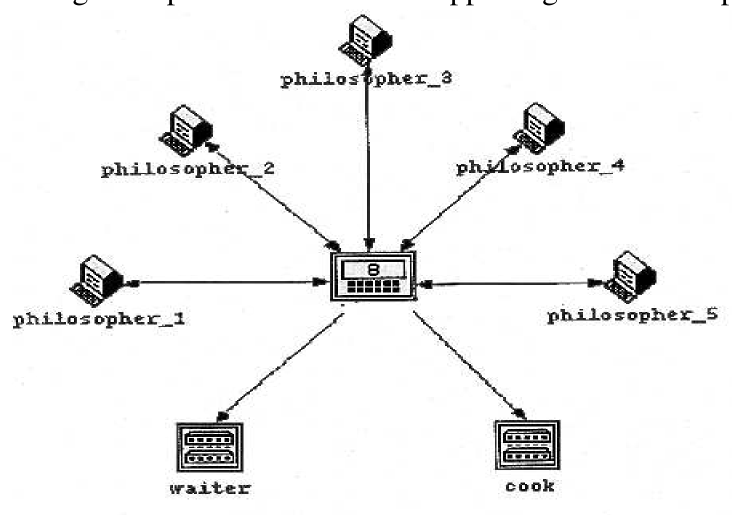


FIGURE 40. Client-server network representation

In the next sections, the top-down approach used to described this model is detailed, some printouts of this model have been put in the appendix.

1. <u>Hardware considerations</u>

The next diagram (see figure 41) was entered in the Network Editor and represents the highest level of abstraction that can be reached through the description process. It is also referred as the ‘top level’. The five terminals (philosopher_1 to philosopher_5) have the same characteristics.

As it is illustrated on the following figure, each terminal owns several attributes, the *model* attribute directly refers to the Node component, here, which describes the general behavior of this terminal and the *application.start time* attribute is a property of the module *application* of the node *arel_nd_philo*, whose value has been promoted to this level of abstraction. Assigning the value *promoted* at this level implies that a value for this attribute will have to be provided in the simulation parameters.

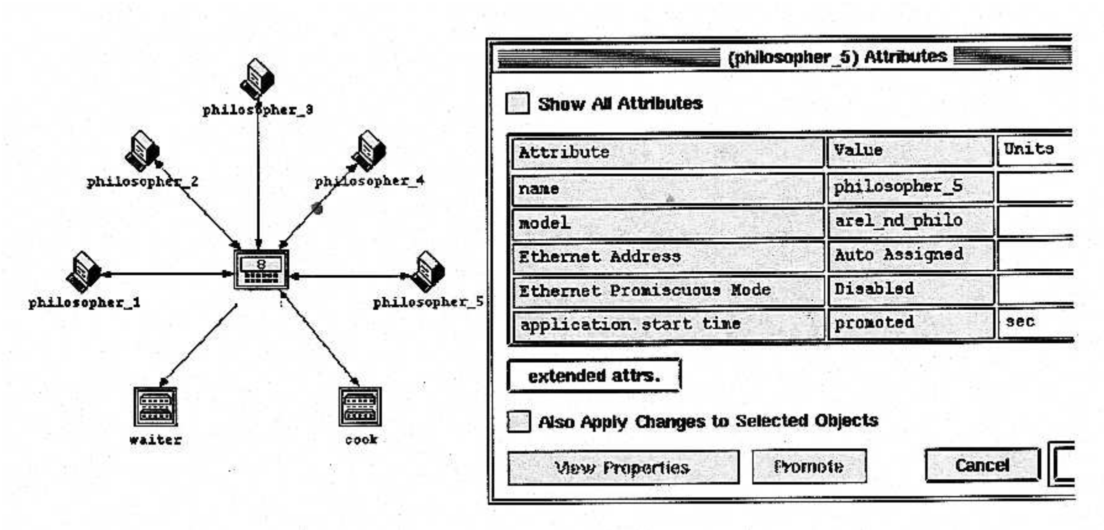


FIGURE 41. Network Component Attributes

The waiter and cook processes have to model the access to a data storage device. As this device is not explicitly provided by OPNET component library, it has been modelled in the Node Domain.

Finally, to ensure the compatibility of combinations of links and network components, a built-in feature can check the completeness and consistency of the connections, warning the user if the data rates of a channel between two linked elements is different or if there is no supported packet formats in common.

2. Node considerations

After defining the network topology, each node has to be defined. Note that this is not a requirement to proceed that way, since each node is considered as an independent object and can be reused without any limitation. This is a way to achieve this model in a top-down approach.

As each node is associated to a network component linked to other nodes through an Ethernet protocol (IEEE 802.3 standard), it has to be compliant with this norm In the following figure, three different module layers, which have to be view as the implementation of the LAN Reference Model. The receiver and transmitters have modelled the working of the physical layer and a queue module is used to represent the inner-behavior of the MAC layer. The higher level simply supports the application level, which actually differentiates the three kind of nodes of this system: the cook, the waiter and the philosophers.

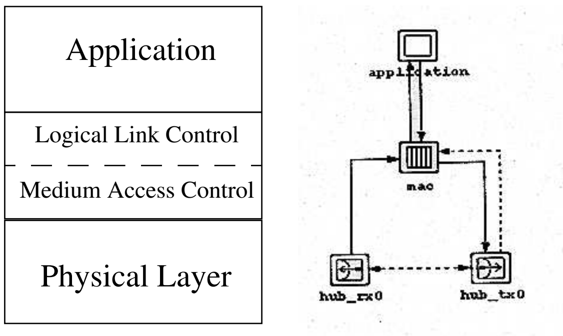


FIGURE 42. Node's general structure

Note: The application layer can be represented by more that one module, examples are given in the Appendix. It is not possible to define multi-level node specifications. There is unfortunately only one, which can lead to dense diagrams. A hub with 32 ports implies one transmitter and one receiver per port, plus a queue module to handle the queries. As a result there are 65 modules on the same drawing. This situation is rare and the colour attribute of the links can help the user to make his model clearer.

3. Process considerations

No dynamic process creations have been made in this model. The reason is the complexity of each process, which makes less difficult a representation by state-transition diagrams. In addition the model's behavior is rather static.

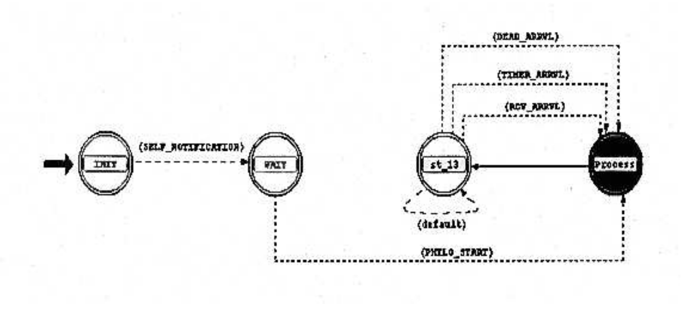


FIGURE 43. Process Dispatcher

4. Inter-process communication

The only dynamic entity that circulates among the components of the system is called a packet. This is a data structure that is organized into fields of information according to a user-defined format. They are considered as dynamic because they are created and destroyed as a simulation progresses. For example, a packet sent from a station A over a network will be destroyed upon reception and analysis by the recipient B, if the station has to reply to A, it will create a new packet, put the right information in it and send it to the station A.

Three sorts of packet have been used in this model. The first one concerns the communication over the network. The format of the packet has been provided by the OPNET components library (see table 13). The creation and reception of those packets are realized by the MAC layer, that is why the structure of this packet should remain transparent to the user. The only data communicated between the MAC interface and the application are the destination address and data whose size is not fixed. Those values are assigned respectively to the *dst_addr* and *data* fields.

Field Name	Type	Size (bits)
preamble	information	64
dst_addr	integer	48
src_addr	integer	48
type	integer	16
data	packet	undefined
fcs	integer	32
last_sent_MAC_ID	integer	0

TABLE 12. Ethernet packet format

The second packet type comports the information used in the communication between the waiter process and its subprocesses. The field *msg_code* indicates the type of the message and the field *data* provides an extra-field to associate a value to the message type. The size of each value is set to zero. This implies that the propagation delays are not taken into account in this communication. As all the processes are running on the same machine, these information are deemed irrelevant in this model.

Field Name	Type	Size (bits)
dest_address	integer	0
src_address	integer	0
msg_code	integer	0
data	integer	0

TABLE 13. Inter-process communication packet format

The third packet is used for the communication between a process and a semaphore and consists of only two field. The field sema_value indicates the type of operation to perform such as to allocate a resource or to deallocate a resource. Id_proc is the identificator of the process having made a request.

Field Name	Type	Size (bits)
sema_value	integer	0
id_proc	integer	0

TABLE 14. process-semaphore communication packet format

5. Simulation parameters

Here is the list of the parameters that have to be updated before running a simulation.

Field Name	Type	Unit	Size (bits)
Eating_time	integer	sec	to be specified (2)
Maximum_waiting_time_before_eating	integer	sec	to be specified
Ponder_time	integer	sec	to be specified
Rest_time	integer	sec	to be specified
Waiting_for_noodle	integer	sec	to be specified
Waiting_right_stick	integer	sec	to be specified
Number_noodle_bowl	integer		to be specified
Number_noodle_parts	integer		to be specified
Check_enough_noodle_time	integer	sec	to be specified
Noodle_cooking_time	integer	sec	to be specified
Cook.application.start_time	integer	sec	to be specified
Philosopher_1.application.start_time	integer	sec	to be specified
Philosopher_2.application.start_time	integer	sec	to be specified
Philosopher_3.application.start_time	integer	sec	to be specified
Philosopher_4.application.start_time	integer	sec	to be specified
Philosopher_5.application.start_time	integer	sec	to be specified
Philosopher_1.application.start_time	integer	sec	to be specified
Waiter.proc.start_time	integer	sec	to be specified
Cook.resource.initial_value_sema	integer		Number_noodle_bowl
Cook.database.Access Database	integer		to be specified

Field Name	Type	Unit	Size (bits)
Waiter.database.*.Access Database (1)	integer		to be specified
Waiter.*.initial_value_sema (1)	integer		1 (mutex)

(1) The asterix is used by OPNET to group the modules having the same parameter in a given node

(2) This value has to be specified according to the desired simulation.

TABLE 15. Simulation parameters

The parameters being declared as a Model Attribute instead of Simulation Attribute appear in this table with the following notation: <Node>.<Module>.parameter_name.

6.4.2 Simulation preparation

As three levels of specification are used. OPNET offers the possibility to simulate a layer independently from its higher layers. In this way, it is possible to simulate only a state finite machine, regardless of its physical implementation.

OPNET does not give the choice of the operating system policy. The only one available is a priority pre-emptive non-destructive policy, But it is possible to simulate other policies, adapting the transitions in the process diagrams, disabling the interruptions at the beginning of a task is a way to model other policies.

From the numerous simulations performed on this model either under normal or extreme condition, the flexibility of OPNET has been demonstrated. The results of one of them is presented below and show the high level of features assisting the user during the simulation process.

The statistical values are defined at the process domain and are collected during the simulation. In figure 44, three statistics for each philosopher are expected, in addition another has been defined to measure the throughput over the network and two other probes record the evolution of two attributes to generate animations after the simulation.

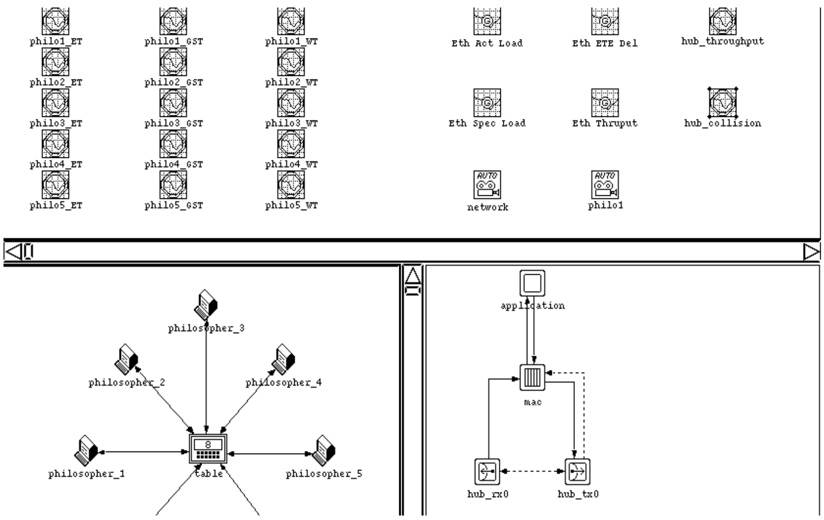


FIGURE 44. Probe Editor

6. Trace file

Figure 45 is a part of the trace file recording the values given by a probe. In this case the probe tracks the Eating_time parameter of philosopher 5. This file is then used to produce the graphics shown in the next section.

```
----------------------------------------------
trace:                  'top.philosopher_5.application.Philo ET'
length:                  12
number of values:        12
abscissa, min:           0
          max:           200
ordinate, min:           5.00101760000007
          max:           40.0045576236001
initial value:           undefined
final value:             40.0045576236001
expected value:          16.3612991079622
sample mean:             16.2024265038476
variance:                153.777733769389
standard deviation:      12.4007150507295
** conf. intrvls valid if entries are independent samples
80% conf interval:     [    10.4856968654613, 21.9191561422339    ]
90% conf interval:     [     8.6255896078519, 23.7792633998433    ]
95% conf interval:     [    6.85228735559758, 25.5525656520976    ]
98% conf interval:     [    4.54162078447832, 27.8632322232169    ]
99% conf interval:     [       2.768318532224, 29.6365344754712    ]
** Operations List **
Original vector: top.philosopher_5.application.Philo ET
---------
```

FIGURE 45. Trace file

6.4.3 Experiment

Experiments have been performed on this model. The results presented below come from a simulation similar to the one made with QASE RT under ‘normal conditions’. For further information about the simulation parameters, please refer to the paragraph Experiments (paragraph. 5.4.3).

The two graphics below show the possible representation for respectively discrete and continuous statistical variables.

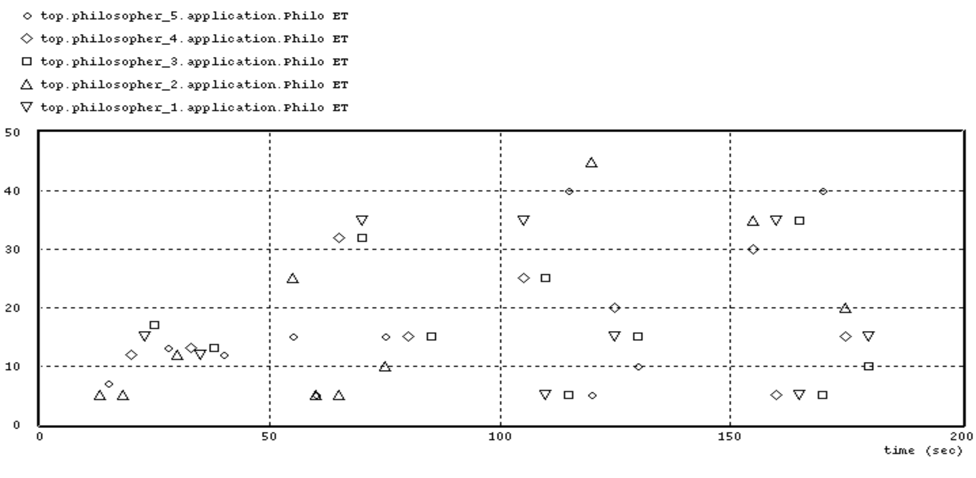


FIGURE 46. Eating time values per philosopher

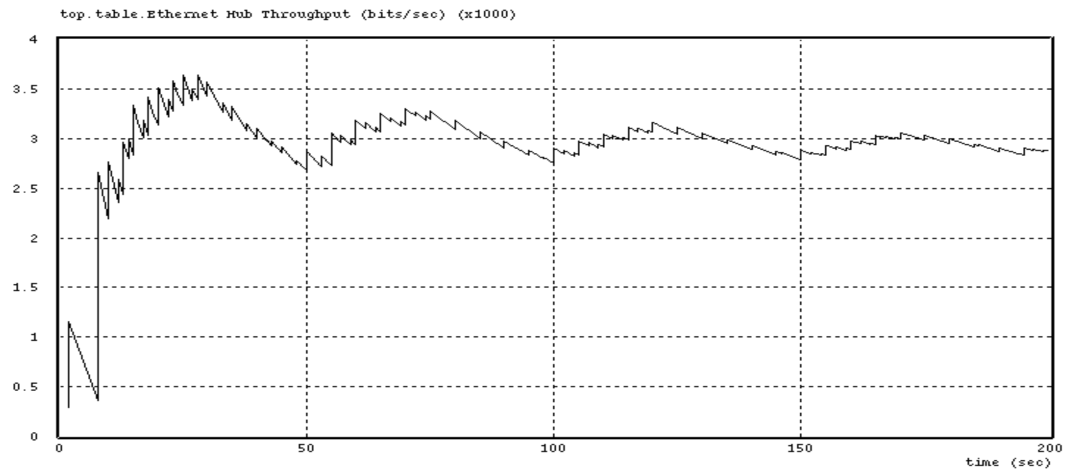


FIGURE 47. Ethernet throughput in the hub

6.5 Overall assessment

The client/server modeling has shown the richness of features and the power of the description language of OPNET. In addition the built-in functions assist the user in the model specifications through a very well-designed user-interface. This user-interface is based on graphical editors which provide an intuitive mapping from the modelled system to the OPNET model specification. The pre-defined functions are very useful to control precisely the event management. They also help the user to specify the packet communication and the queue management. The object orientation supports the definition of new classes and objects. The reusability of those objects is emphasized by their set of attributes.OPNET also provides additional tools to analyze and graphically represent the simulation results.

All these functionalities make OPNET a perfect tool to model any kind of distributed and communication systems.

The main disadvantage is its relative complexity to model a given system. The time required to achieve the modelling of a system can be very high, especially when some network components have to be defined as it was the case in this study. This implies notably that the user must also have a very good knowledge of computer networking and network protocols. To describe the behaviour of each network components the user has to be familiar with the state-transition diagrams and the event scheduling. In addition, to take advantage of the C functions provided by OPNET a good knowledge of C programming is necessary.

CONCLUSION

During this project the simulation tools, QASERT and OPNET, have been investigated and it turns out that each of them presents several assets. These assets make them extremely valuable if their use is adapted to the system to be modelled.

In the early stages of a network design, QASE RT offers a straightforward way to test several network types. It provides a user-friendly interface to enter the expected work load and to select basic network components. Thanks to this model, a simulation can give useful information regarding the throughput of communication devices and where the bottle-necks are. OPNET can also be used if the network components are available in the model library and if the user is already familiar with this tool.

When accurate data are required for a performance analysis or when the network type has been singled out, using OPNET could be very valuable. The modelled system can indeed be described very precisely to approach a realistic network. This tool can measure with a high precision the utilization of the devices connected to the network. User-defined statistical values can be also set in various parts of the system and be automatically collected during the simulations. These data can then be analysed to find out if they meet the project requirements. QASE RT does not appear to be suitable for this kind of performance analysis because of its high level of abstraction.

On the whole, these tools give the user a real opportunity to foresee what will be the effects of changes and expansions on a given network prior to hardware investments.

BIBLIOGRAPHY

Simulation

[Law91] Law, Averill M., Kelton, David D.(1991)
Simulation, Modeling, and Analysis
2nd ed. McGraw-Hill, New-York

Distributed and Real-time System

[Mul93] Mullender, Sape (1993)
Distributed Systems
2nd ed. ACM Press New York, New York

[Tör96] Törne, Anders (1996)
Embedded Systems, Real-Time and Concurrent Programming
Compendium, IDA, Institute of Technology, Linköping University

Computer network

[Tan96] Tanenbaum, Andrew S. (1996)
Computer Networks
3rd Edition, Prentice Hall,

[Wrl88] Boggs, David R., Mogul, Jeffrey C., Kent, Christopher A.
Measured Capacity of an Ethernet: Myths and Reality
Western Research Laboratory, Research Report 88/4, Digital Inc.

[Wit96] Wittman Art (1996)
Interactive Network Design Manual
Network Computing On-line, CMP Media Inc.

[Gra97] Grand Junction Inc. (1997)
Switched Ethernet: An Evolutionary Alternative for High Speed Networking
Grand Junction Web Site, Product Information (1)

Simulation tools

[Mil96] Mil 3, Inc (1996)
OPNET 3.Modeler User‘s Guides
Mil 3, Inc Washington DC, USA

[Ast96] AST Engineering Services (1996)
QASE RT User’s Guide v. 3.2.1
Advanced System Technologies, Inc., Englewood, Colorado

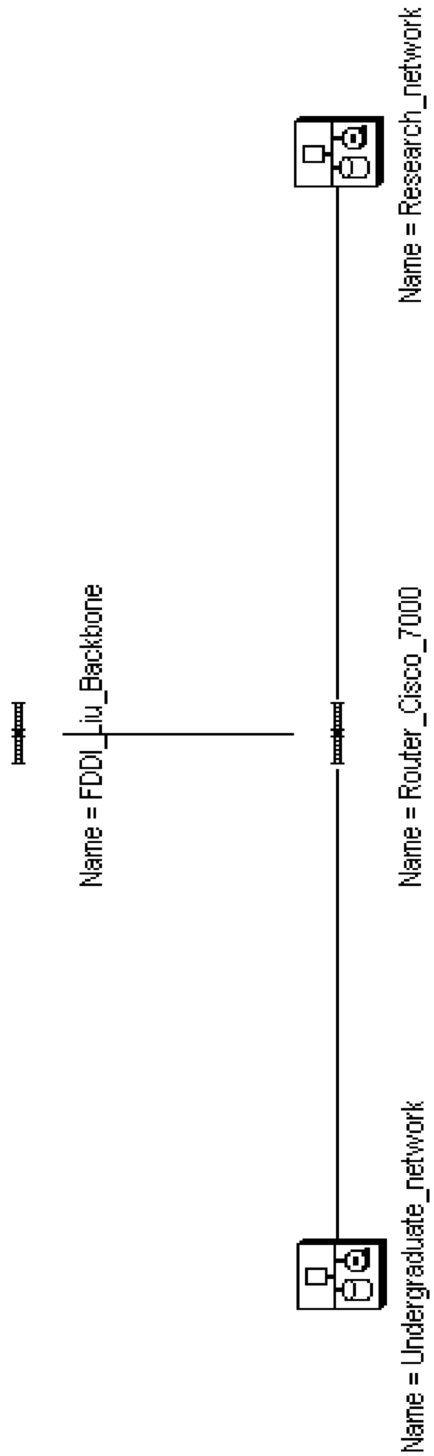
Reports & studies

[Hol95] Holm, Joakim (1995)
A performance analysis simulation of a PACS using BONeS Designer ™
Final report, IDA, Institute of Technology, Linköping University

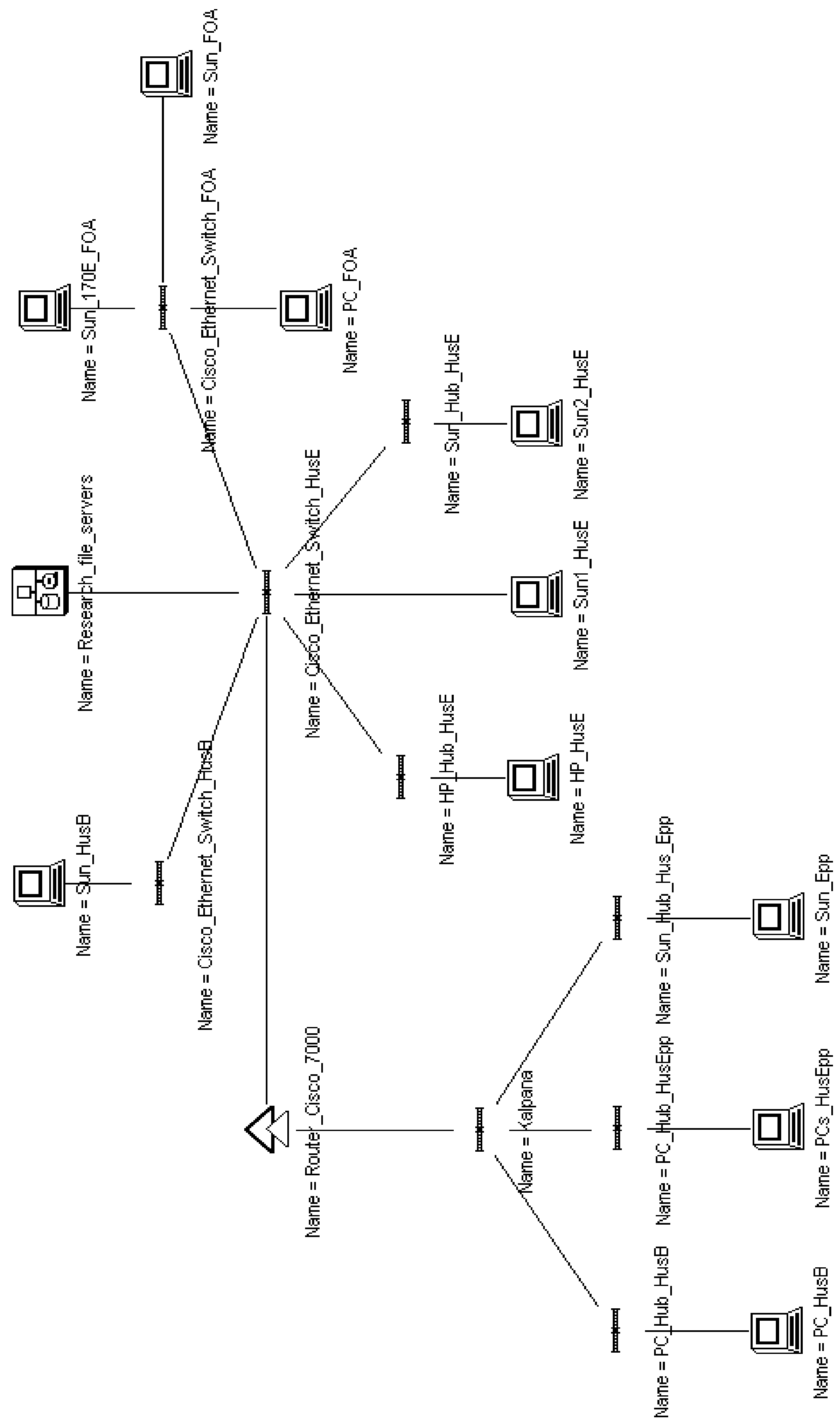
(1) After Grand Junction’s buy-out by Cisco Inc, this document is not available any more.

APPENDIX

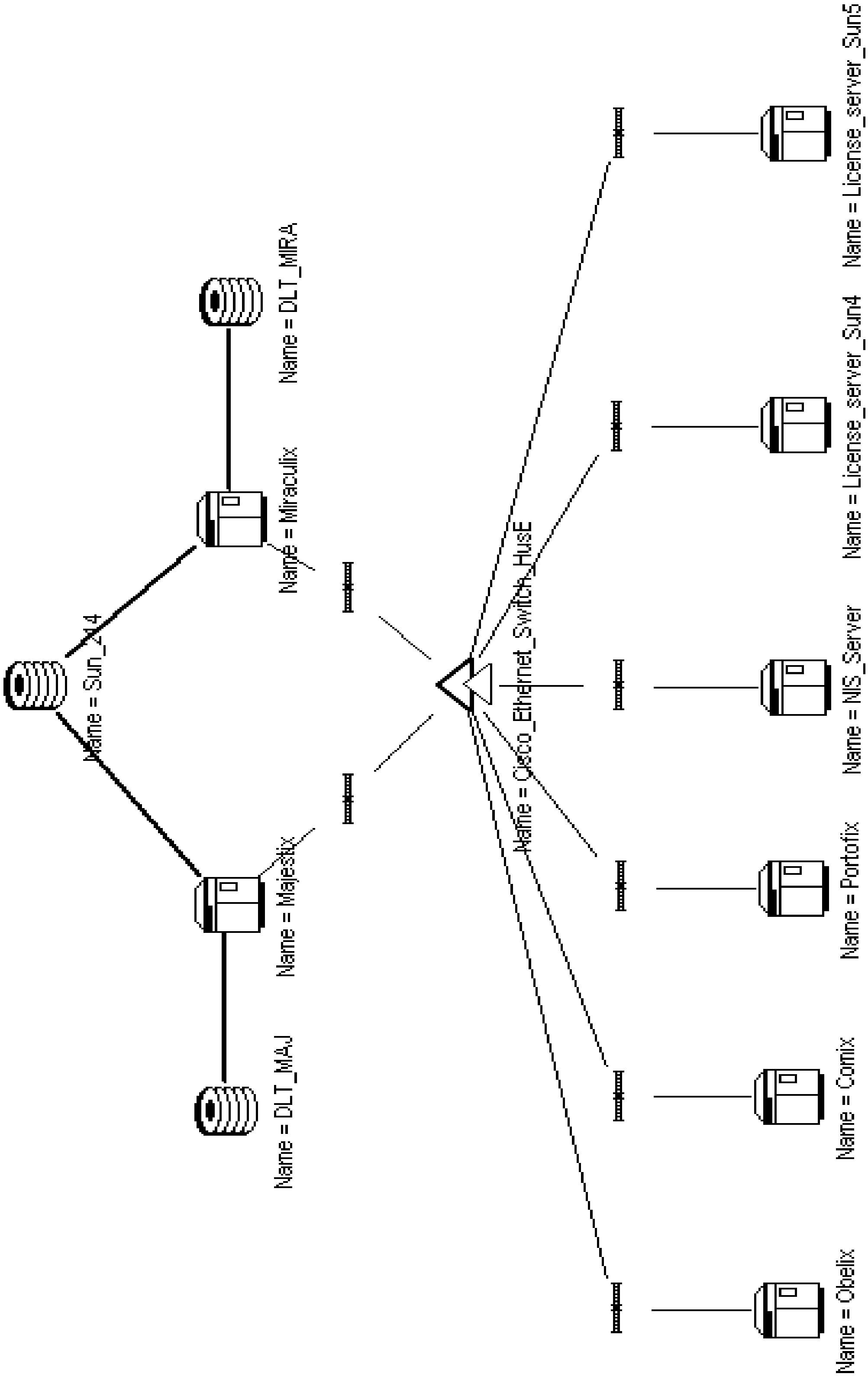
QASE RT - IDA network model, part 1/5



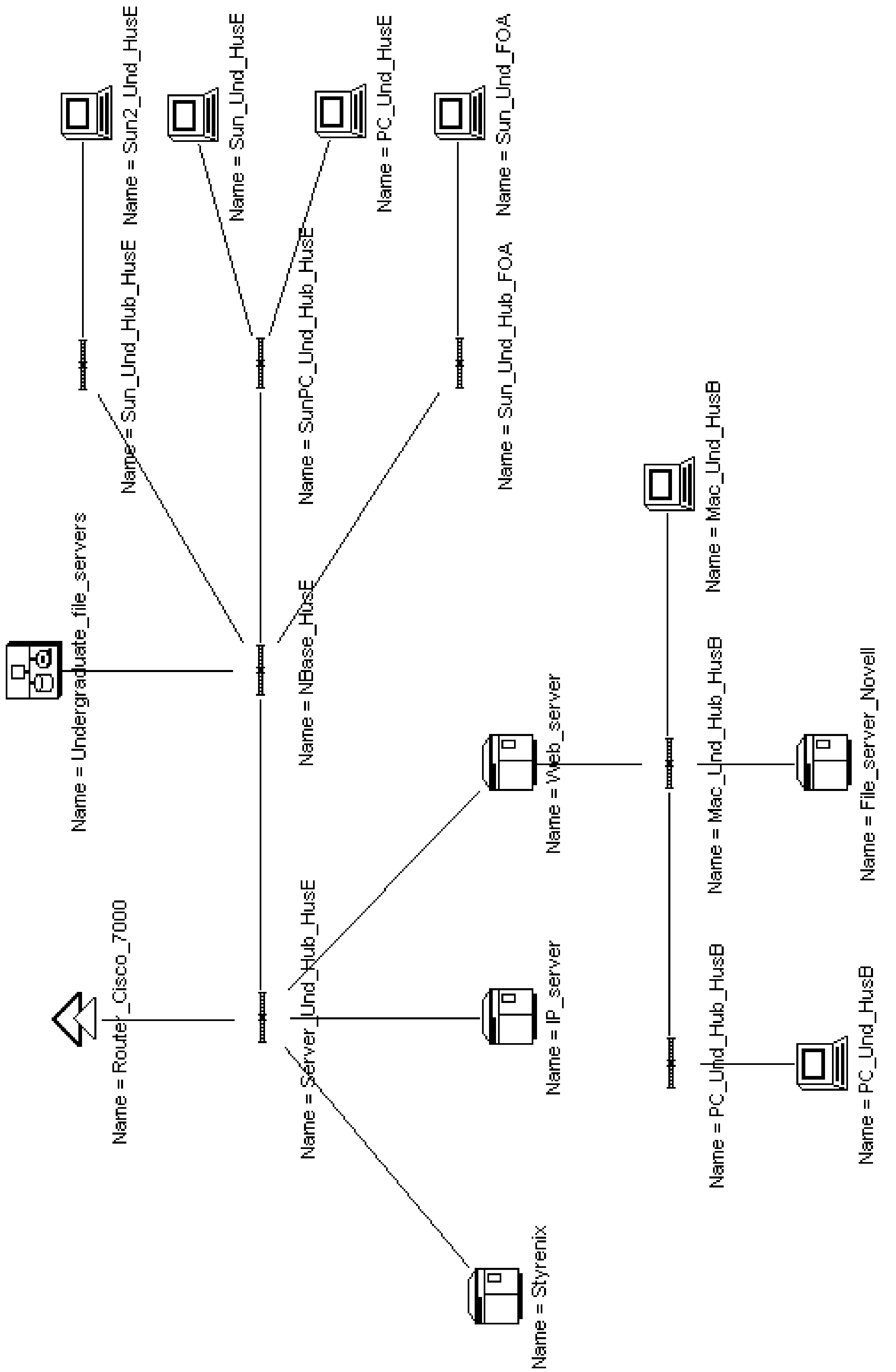
QASE RT - IDA network model, part 2/5



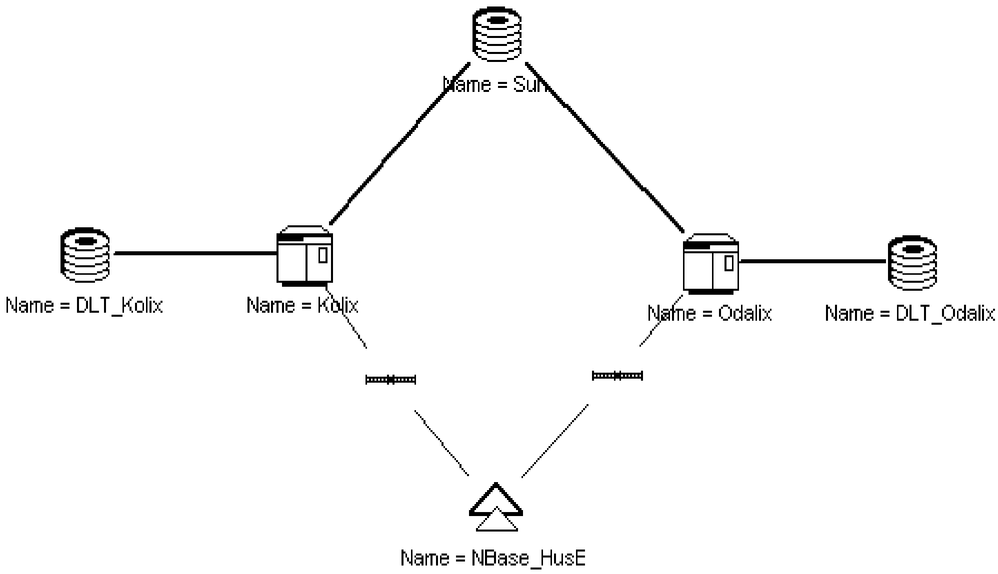
QASE RT - IDA network model, part 3/5



QASE RT - IDA network model, part 4/5



QASE RT - IDA network model, part 5/5



Data storage access and resource management

- To remove 1 instance of Noodle from the food stock

  Data type: Noodle
  Data size: DATA_SIZE_NOODLE
  Storage device: Food_stock
  Required by: Cook
  Action: Get_Noodle

- To remove a portion of noodle from the main bowl

  Data type: Main_bowl
  Data size: DATA_SIZE_MAIN_BOWL
  Storage device: Table
  Required by: Waiter
  Action: Get_Portion_Main_Bowl

- To refill the main bowl with 'Number_Noodle_Parts' portions of noodle

  Data type: Main_bowl
  Data size: DATA_SIZE_MAIN_BOWL
  Storage device: Table
  Required by: Waiter
  Action: Refill_Main_Bowl

- To get the amount of noodle in the main bowl

  Data type: Main_bowl
  Data size: DATA_SIZE_MAIN_BOWL
  Storage device: Table
  Required by: Waiter
  Action: Get_Value_Main_Bowl

- To get the value of a chopstick

  Data type: Chopstick
  Data size: DATA_SIZE_CHOPSTICK
  Storage device: Table
  Required by: Waiter
  Action: Get_Value_Chopstick

- To update the value of a chopstick

  Data type: Chopstick
  Data size: DATA_SIZE_CHOPSTICK
  Storage device: Table
  Required by: Waiter
  Action: Put_Value_Chopstick

QASE RT / SIMULATION

Normal conditions

Results of a 600 s simulation using the four scheduling policies.

TABLE 1. FCFS policy

Philosophers	Priority	Nb meal	min eating time	mean eating time	max eating time
1	5	27	10	19	44
2	4	26	12	20	32
3	3	28	11	18	38
4	2	27	11	18	37
5	1	23	12	22	43

TABLE 2. Priority pre-emptive destructive policy (PPD)

Philosophers	Priority	Nb meal	min eating time	mean eating time	max eating time
1	5	20	13	26	60
2	4	22	16	24	36
3	3	28	12	17	39
4	2	29	13	17	33
5	1	30	10	16	30

TABLE 3. Priority non pre-emptive policy (PNP)

Philosophers	Priority	Nb meal	min eating time	mean eating time	max eating time
1	5	24	11	21	53
2	4	25	12	21	53
3	3	26	11	19	39
4	2	23	14	21	43
5	1	26	10	20	28

TABLE 4. Priority Pre-emptive non destructive policy (PPND)

Philosophers	Priority	Nb meal	min eating time	mean eating time	max eating time
1	5	25	13	21	53
2	4	25	14	20	35
3	3	27	13	18	34
4	2	28	13	18	27
5	1	29	10	17	26

Normal conditions (cont')

Parameter values:

Waiter_processor_speed	10	
Philosopher_processor_speed	10	
Cook_processor_speed	10	
Waiter_inst_task	2	
Data_Size_MSG	0	/* no network load */
Eating_time	5	
Ponder_time	3	
Rest_time	0	
Waiting_right_stick	1	
Waiting_for_noodle	2	
Noodle_cooking_time	4	
Number_Noodle_Parts	10	
Number_Noodle_Bowl	100	/* enough noodle available for the sim. */
Check_enough_noodle_time	5	

Influence of the time on the results

Parameter values:

Waiter_processor_speed	10 inst./s	
Philosopher_processor_speed	10 inst./s	
Cook_processor_speed	10	
Waiter_inst_task	2	
Data_Size_MSG	10	
Transfer rate Network	100	
Eating_time	5	
Ponder_time	3	
Rest_time	0	
Waiting_right_stick	1	
Waiting_for_noodle	2	
Noodle_cooking_time	4	
Number_Noodle_Parts	10	
Number_Noodle_Bowl	100	/* enough noodle available for the sim. */
Check_enough_noodle_time	5	

Network overload

Parameter values:

Waiter_processor_speed	10 inst./s	
Philosopher_processor_speed	10 inst./s	
Cook_processor_speed	10	
Waiter_inst_task	2	
Data_Size_MSG	0 to 200	
Transfer rate Network	100	
Eating_time	5	
Ponder_time	3	
Rest_time	0	
Waiting_right_stick	1	
Waiting_for_noodle	2	
Noodle_cooking_time	4	
Number_Noodle_Parts	10	
Number_Noodle_Bowl	100	/* enough noodle available for the sim. */
Check_enough_noodle_time	5	
Operating system:	FCFS	

OPNET / SIMULATION

Client-server model, Node Domain

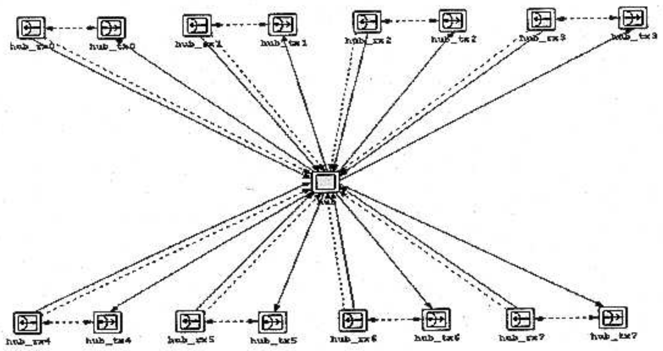


FIGURE 1. 8-port hub

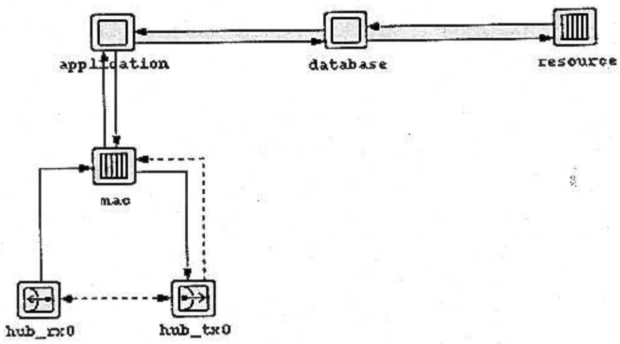


FIGURE 2. Cook node

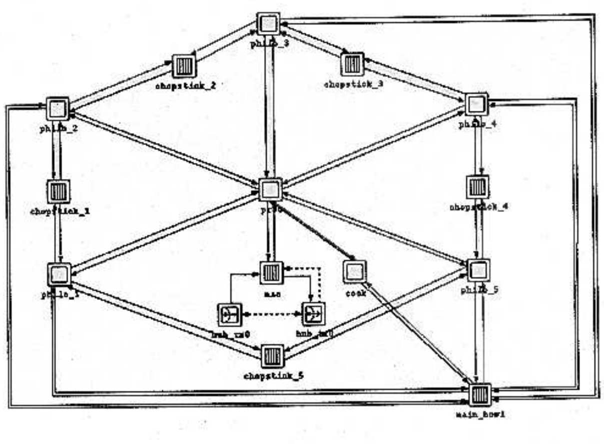


FIGURE 3. Waiter node

Client-server model, process Domain

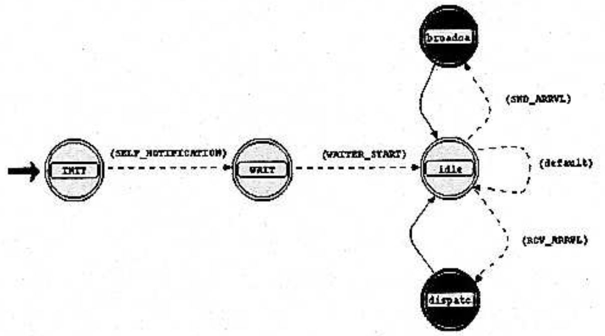


FIGURE 4. Waiter dispatching process

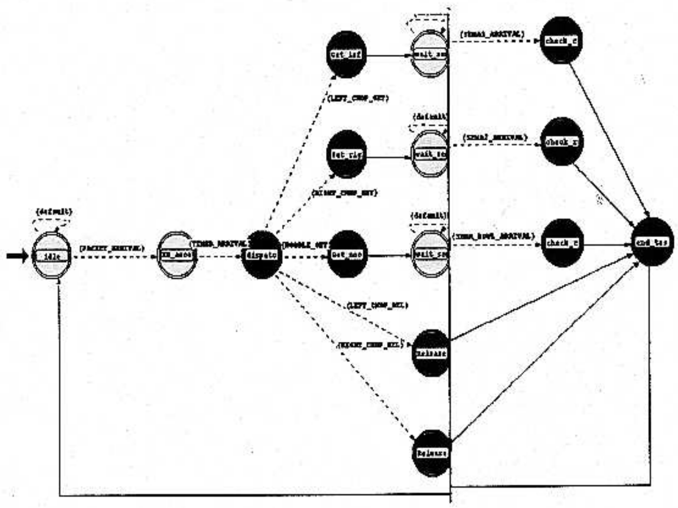


FIGURE 5. Waiter/Philo subprocess

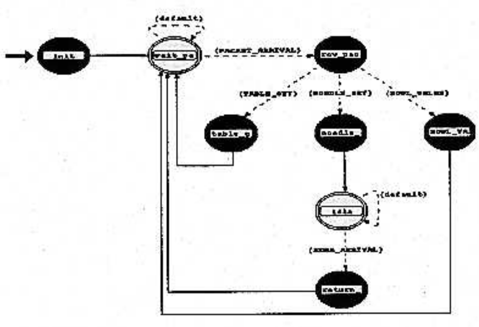


FIGURE 6. Waiter/Cook subprocess

Client-server model, process Domain

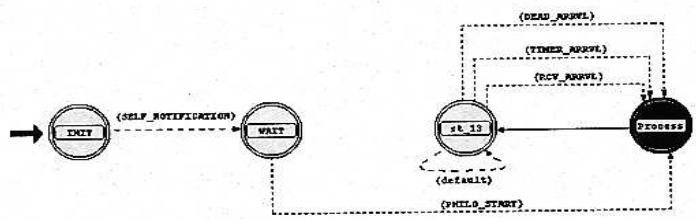


FIGURE 7. Philosopher process

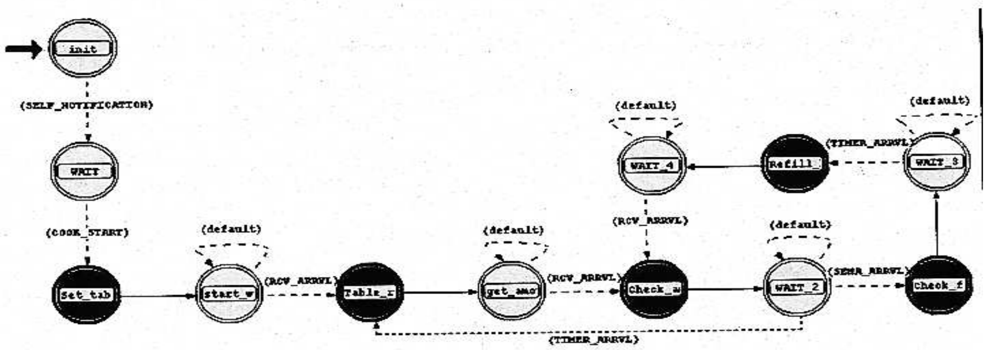


FIGURE 8. Cook process

Avdelning, Institution
Division, department

Institutionen för datavetenskap

Department of Computer and
Information Science

Datum
Date

1997/06/09

Språk
Language

- [ ] Svenska/Swedish
- [x] Engelska/English
- [ ] ____________

Rapporttyp
Report category

- [ ] Licentiatavhandling
- [x] Examensarbete
- [ ] C-uppsats
- [ ] D-uppsats
- [ ] Övrig rapport
- [ ] ____________

ISBN

ISRN

Serietitel och serienummer
Title of series, numbering

ISSN

LiTH-IDA-Ex- 97/43

URL för elektronisk version

http://www.ida.liu.se/~x97jeame/report

Titel
Title

Network Performance Analysis Tools on Simulations including Real-Time constraints

Författare
Author

Jean-Paul Meynard

Sammanfattning
Abstract

This project concerns a study of performance analysis tools dedicated to computer networking. This investigation was carried out in order to assess the abilities of these tools to represent and simulate models under real-time constraints.
Two software packages have been tested and evaluated QASE RT, an information system simulator, and OPNET, a communication system simulator.
The method employed consisted in four steps: theoretical studies, simulation tool evaluation, simulation and reporting. Two networks have been used to illustrate the tools' functionalities: the actual network at the Department of Computer and Information Science (IDA) and a client/server-oriented network.
The results of the simulations applied on these models are discussed. The advantages and disadvantages of the evaluated tools are presented at the end of this document

Nyckelord
Keywords

simulation, network, real-time, performance analysis, modelling