



NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

**SIMULATION OF DISTRIBUTED OBJECT-ORIENTED
SERVERS**

by

Chee Khan, KWOK

December 2003

Thesis Advisor:
Second Reader:

Bill Ray
Shing Man Tak

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 2003	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE: Simulation of Distributed Object Oriented Servers			5. FUNDING NUMBERS	
6. AUTHOR(S) Kwok Chee Khan				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) Distributed object oriented (OO) computing such as RMI, COBRA and SOAP etc is fast becoming the de-facto standard for software development. The aim of the system designer is to determine the optimal deployment strategy for the system to perform efficiently. This is an enormous task especially when multiple object servers are fielded on hardware of different specifications. The number of possible deployment strategy of object servers to hardware grows exponentially with increase number of object server and machine. For example, with 3 machines and 10 object servers there are 59049 possible deployment patterns. Eventually, the number of possible deployment makes it impossible for system designer to setup test bed to determine the optimal deployment strategy. The main goal of the simulation model is to analyze the object server deployment, verify an existing optimization model and to determine the optimal deployment strategy that will reduce the client response time. In one of the experiment conducted with the simulation model, in an environment with 3 machine and 10 object servers, it will take 53 years to attempt all deployment patterns in the lab environment. The simulation model will take only 13 days, which is an improvement of 1480%.				
14. SUBJECT TERMS Distributed Object Oriented Architecture, Simulation, OMNet++, Optimization.			15. NUMBER OF PAGES 217	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

SIMULATION OF DISTRIBUTED OBJECT ORIENTED SERVERS

Chee Khan, KWOK
Civilian, Ministry of Defense Singapore
B.S., National University of Singapore, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
December 2003**

Author: Chee Khan, KWOK

Approved by: Bill Ray
Thesis Advisor

Shing Man-tak
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Distributed object oriented (OO) computing such as RMI, COBRA and SOAP etc is fast becoming the de-facto standard for software development. Distributed OO systems can consist of multiple object servers and client application on a network computer, as oppose to a single large centralized object server.

The aim of the system designer is to determine the optimal deployment strategy for the system to perform efficiently. This is an enormous task especially when multiple object servers are fielded on hardware of different specifications. The number of possible deployment strategy of object servers to hardware grows exponentially with increase number of object server and machine. For example, with 3 machines and 10 object servers there are 59049 possible deployment patterns. Eventually, the number of possible deployment makes it impossible for system designer to setup test bed to determine the optimal deployment strategy.

The main goal of the simulation model is to analyze the object server deployment, verify an existing optimization model and to determine the optimal deployment strategy that will reduce the client response time. In one of the experiment conducted with the simulation model, in an environment with 3 machine and 10 object servers, it will take 53 years to attempt all deployment patterns in the lab environment. The simulation model will take only 13 days, which is an improvement of 1480%.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	BACKGROUND.....	5
A.	INTRODUCTION.....	5
B.	PROFILES	5
1.	Hardware Profiles	6
2.	Object Server Profiles	6
3.	Client Application Profiles	7
4.	User Profiles.....	8
C.	OPTIMIZATION MODEL.....	9
1.	Objective Function	9
2.	Processing Speed Term	10
3.	Network Speed Term	11
4.	RAM Limits	11
5.	CPU Limits.....	11
D.	VALIDATION OF OPTIMIZATION MODEL	11
E.	CURRENT RESEARCH APPROACH	12
III.	OVERVIEW OF OMNET++ AND NETWORK SIMULATION	13
A.	BACKGROUND	13
B.	MODELING CONCEPTS	13
1.	Hierarchical Nested Modules.....	14
2.	Communicate with Message through Channel	14
3.	Parameters	16
4.	Topology Description Method	16
C.	PROGRAMMING THE ALGORITHM.....	16
D.	RUNNING THE SIMULATION	16
1.	Running the Simulation.....	17
2.	Analyzing The Result	19
IV.	OBJECT ORIENTED SERVER SIMULATION MODEL	21
A.	INTRODUCTION	21
B.	REQUIREMENT MODELING.....	21
1.	Use Case Diagram	21
2.	Actor Description.....	22
3.	Use Case Description	22
C.	CLASS DESCRIPTION	24
1.	Class Inheritance	24
2.	Association Diagram	25
D.	NETWORK DESCRIPTION (NED) FILE.....	26
E.	MESSAGE DESCRIPTION	29
F.	IMPLEMENTATION DETAIL	30
1.	Computation of Response Time	30

2.	Random Role Execution.....	31
3.	RAM Limit.....	32
4.	Output Result.....	32
G.	CONFIGURATION FILE	33
H.	PATTERN GENERATOR.....	34
1.	Deployment Configuration File.....	34
2.	Role Configuration File	35
V.	VERIFICATION EXPERIMENT 1.....	37
A.	INTRODUCTION.....	37
B.	SCENARIO PROFILE	37
C.	COMPARING SIMULATION RESULT WITH TEST BED.....	40
1.	Result for Test Case S1.....	43
2.	Result for Test Case S2.....	45
3.	Result for Test Case S3.....	47
4.	Result for Test Case S4.....	49
5.	Result for Test Case S5.....	51
6.	Result for Test Case S6.....	53
7.	Result for Test Case S7.....	55
8.	Result for Test Case S8.....	57
9.	Result for Test Case S9.....	59
D.	COMPARING SIMULATION RESULT WITH TIME SLICE	61
E.	STUDY OF GROWTH POTENTIAL.....	62
F.	COMPARING SIMULATION RESULT WITH LINGO MODEL	64
G.	CONCLUSION	64
VI.	VERIFICATION EXPERIMENT 2.....	67
A.	INTRODUCTION.....	67
B.	SCENARIO PROFILE	67
C.	MODEL SETUP.....	69
1.	Simulation Model Setup	69
2.	Lingo Model Setup.....	70
D.	WEIGHED MODEL	70
1.	Ranking	70
2.	Highest Priority	70
3.	Weighted Model	70
E.	COMPARING SIMULATION RESULT WITH LINGO MODEL	71
1.	Equal Weigh Model.....	71
2.	Effect of Weigh Model	73
F.	CONCLUSION	74
VII.	RECOMMENDATION AND FUTURE WORKS	75
A.	MODELING OF SYSTEM USING UML	75
B.	IMPROVED NETWORK MODELING.....	75
C.	REFINING THE RESULT TO ROLE/METHOD CALL LEVEL	76
D.	ENTERPRISE RESOURCE PLANNING (ERP).....	77
E.	CONSTRAINTING THE DEPLOYMENT PATTERNS	77

VII. CONCLUSION	79
APPENDIX A - SOURCE CODE OF THE SIMULATION MODEL	81
A. ADDRESS.CPP	81
B. BTNLIST.CPP	81
C. DNSSVC.CPP	84
D. INTERACTLIST.CPP	86
E. METHOD.CPP	89
F. OBJ.CPP	90
G. OBJLIST.CPP	91
H. SIMEVENT.CPP	95
I. SMACHINE.CPP	96
J. SOBJSVR.CPP	116
K. SROLE.CPP	125
L. SSWITCH.CPP	132
M. STATLOG.CPP	138
N. DISOBSVRSIM.NED NETWORK DESCRIPTION FILE	141
APPENDIX B - SOURCE CODE OF PATTERN GENERATOR.....	145
A. PATTERNGENERATOR.CPP	145
APPENDIX C –VERIFICATION EXPERIMENT 1.....	157
A. CONFIGURATION FILES	157
B. DETAILED RESULT	172
1. Simulation Run 1 (1 role 1, Ram Limit 0.0)	172
2. Simulation Run 2 (1 role 2, Ram Limit 0.0)	173
3. Simulation Run 3 (1 role 3, Ram Limit 0.0)	174
4. Simulation Run 4 (4 role 2, Ram Limit 0.0)	175
5. Simulation Run 5 (3 role 3, Ram Limit 0.0)	176
6. Simulation Run 6 (28 role 1, Ram Limit 1.5)	177
7. Simulation Run 7 (1 role 1, Ram Limit 1.5)	178
8. Simulation Run 8 (1 role 2, Ram Limit 1.5)	179
9. Simulation Run 9 (1 role 3, Ram Limit 1.5)	180
10. Simulation Run 5 with Time Slice 0.3 sec.....	181
11. Simulation Run 5 with Time Slice 0.5 sec.....	182
12. Simulation Run 5 with Time Slice 1.0 sec.....	183
13. Simulation Run 5 with Time Slice 2.0 sec.....	184
APPENDIX D – VERIFICATION EXPERIMENT 2.....	185
A. CONFIGURATION FILES	185
B. LINGO MODEL	192
C. DETAILED RESULT	194
BIBLIOGRAPHY	215
INITIAL DISTRIBUTION LIST	217

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	Simple and Compound Modules	14
Figure 2.	Connecting modules.....	15
Figure 3.	Demonstration user interface	18
Figure 4.	Debugging window	18
Figure 5.	Plove view of response time output file.	19
Figure 6.	User Case Diagram	21
Figure 7.	Class inheritance diagram	24
Figure 8.	Class diagram	26
Figure 9.	Simulation Model for S5	39
Figure 10.	Graphical result of S1	43
Figure 11.	CPU Utilization for Optimal Deployment Strategy Pattern 1	44
Figure 12.	Role Response Time for Optimal Deployment Pattern 1	44
Figure 13.	Graphical result of S2	45
Figure 14.	CPU Utilization for S2.....	46
Figure 15.	Role Response Time for S2	46
Figure 16.	Graphical result of S3.....	47
Figure 17.	CPU Utilization for S3.....	48
Figure 18.	Response Time for S3.....	48
Figure 19.	Graphical result of S4	49
Figure 20.	CPU Utilization for S4.....	50
Figure 21.	Response Time for S4.....	50
Figure 22.	Graphical result of S5.....	51
Figure 23.	CPU Utilization for S5.....	52
Figure 24.	Response Time for S5.....	52
Figure 25.	Graphical result of S6.....	53
Figure 26.	CPU Utilization for S6.....	54
Figure 27.	Graphical result of S7	55
Figure 28.	CPU Utilization for S7	56
Figure 29.	Response Time for S7	56
Figure 30.	Graphical result of S8.....	57
Figure 31.	CPU Utilization for S8.....	58
Figure 32.	Response Time for S8.....	58
Figure 33.	Graphical result of S9.....	59
Figure 34.	Optimal Deployment Pattern S9	59
Figure 35.	CPU Utilization for S9.....	60
Figure 36.	Response Time for S9.....	60
Figure 37.	Graphical result of Time Slice.....	61
Figure 38.	Effect of growth on response time	63
Figure 39.	Role Response Time	72
Figure 40.	Result wrt to overall response time.....	73

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	Computation of deployment pattern.....	2
Table 2.	Example of hardware profile.....	6
Table 3.	Example of object server profile	7
Table 4.	Example of client application profile	8
Table 5.	Example of user profile.....	8
Table 6.	Actor description.....	22
Table 7.	Network Description File Parameters	27
Table 8.	Simple Module Parameters	29
Table 9.	Message Description Table	30
Table 10.	Configuration file description	34
Table 11.	Sample bit arithmetic operation	35
Table 12.	Machine Profile.....	37
Table 13.	Object Server Performance Profile	37
Table 14.	Complex Object Server Profile	37
Table 15.	Role Call Pattern	38
Table 16.	User Interface Call Chart.....	38
Table 17.	Test Case	39
Table 18.	Test Result of Test Case S1 – S4	41
Table 19.	Test Result of Test Case S5 – S9	42
Table 20.	Optimal Deployment Pattern S1	43
Table 21.	Optimal Deployment Pattern S2	45
Table 22.	Optimal Deployment Pattern S3	47
Table 23.	Optimal Deployment Pattern S4	49
Table 24.	Optimal Deployment Pattern S5	51
Table 25.	Optimal Deployment Pattern S6	54
Table 26.	Optimal Deployment Pattern S7	55
Table 27.	Optimal Deployment Pattern S8	57
Table 28.	Test Result with Time Slice	62
Table 29.	Effect of growth on deployment pattern	63
Table 30.	Result for Test Case S1-S3.....	64
Table 31.	Result for Test Case S4-S6.....	64
Table 32.	Result for Test Case S7-S9.....	64
Table 33.	Deployment Strategy Result.....	64
Table 34.	Machine Profile.....	67
Table 35.	Object Server Performance Profile	68
Table 36.	Complex Object Server Call Profile	68
Table 37.	Role Call Pattern	69
Table 38.	User Interface Call Chart.....	69
Table 39.	Example of Weighted Model.....	71
Table 40.	Deployment Pattern.....	72
Table 41.	Weighted model	73

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Professor Bill Ray for his support and guidance throughout the course of my thesis work. His keen interest and knowledge in the subject area and technical expertise has been a great source of help. I thank Andras Varga, the developer of OMNet++, for the support provided in answering some of my queries related to the product. I would like to thank all the other 6 Singaporean students in the Master Defense Technology System (MDTS) Information Assurance Track for the wonderful time in learning and playing together. The time spent here will be remembered. I'm grateful for my wonderful wife Wendy and daughter Charis who provided me with their constant support, love, and understanding throughout this research. Lastly, I would like to thank God for providing me this opportunity to learn and improved myself.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

With the massive reduction in the cost of microprocessors and the large increase in microprocessor performance over the past few years, distributed microcomputer based systems are becoming a cost effective solution to many problems. Coupled with the advancements of object-oriented technology in the past decade, object oriented architecture is often used for the building of large scale computer based systems.

Distributed object oriented (OO) computing such as Remote Method Invocation (RMI), Common Object Request Broker Architecture (COBRA) and Simple Object Access Protocol (SOAP) etc is fast becoming the de-facto standard for software development. Distributed OO systems can consist of multiple object servers and client applications on a network of computers, as oppose to a single large centralized object server. Consequently, information processing is distributed over several computers rather than confined to a single machine.

In military environment, a set of distributed object servers could be used to support many applications aboard a ship. The distributed object servers could serve track object, intel object, logistic object and target object to a host of client applications. These applications could handle such tasks as Anti-Submarine Warfare (ASW), Anti-Surface Warfare (ASUW), Anti-Air Warfare (AAW), Electronic Warfare (EW), humanitarian missions and rescue missions.

A user's network of computers will change frequently. Object servers, applications, hardware and user preferences will be in a constant state of change. The distribution of object servers across different hardware platform will result in multiple deployment strategies. The number of deployment strategy = m^n , which grows exponentially with an increase in either the number of object servers (n) or hardware platform (m).

Object Server	Computer	Number of pattern
3	3	27
3	5	243
10	3	59049
3	10	1000

Table 1. Computation of deployment pattern

As the number of deployment pattern increases, it becomes a tedious task for system engineers to setup all deployment patterns in order to find the deployment strategy that will provide the optimal system efficiency. The difficulty in selecting the best pattern increases with the need to deploy different object servers on computers with different CPU and memory capacity. Lack of better tools, the deployment strategy is based on best-guess or experience of the system engineer. The resulting deployment strategy may/may not be optimal.

As a system evolves, common hardware changes consist of adding new computers, removing old computers, upgrading CPUs, modifying RAM and modifying network bandwidth capacity. Each of these hardware changes will produce an event that would trigger the system to re-evaluate its deployment strategy. The intent of system engineers is always to find a deployment strategy that performs at peak efficiency. Some measurement of efficiency can be based on CPU resource utilization, bandwidth utilization and user response time. The user response time is affected by the following:

- The number of clients. The greater the number of client, the more object server calls will be executed and the longer is the response time.
- The frequency of function called invoked by each client. With an increase in the frequency of function call, more requests will be queued and therefore the longer will be the response time.

- The number of instruction cycle in the object method invoked. The more complex the object method, the number of instruction cycle will increase thus increasing the response time.
- The speed of the microprocessor. The faster the processor, the faster is the response time.
- The scheduling algorithm of the microprocessor. Depending on the time needed to complete an object server call, a non-preemptive and time slicing scheduling algorithm will result in different response time.
- The number of object server served by a single microprocessor. With more object server loaded in a computer, the higher the number request are queued thus increasing the response time. The number of object server to be loaded in a computer is also limited by the RAM capacity.
- The network delay due to transmission delay and data rate. The higher the network delay, the greater the response time.

By knowing ahead of time how many users will be accessing the system, each user interaction pattern and each computer configuration, a model can be developed to make recommendation of a possible optimal deployment strategy. Even a simplistic model may lead to large gains in performance.

The goal of this research is to develop a simulation model that models a distributed object-oriented environment. The model will allow system engineers to model the environment [i.e. the number of clients, the interaction patterns, computer configuration], and simulate the deployment of object server on different hardware platforms and derive the optimal deployment strategy. The criterion used to evaluate the optimal deployment strategy was to minimize the user response time.

This chapter gave a brief introduction to the problem and the motivation for the research. Chapter II gives an introduction to the dissertation work done

by Professor William J Ray [William 2001] that used a mathematical model to predicts the optimal deployment patterns. Chapter III introduces the discrete event simulation tool OMNet++ that is used to model the distributed object-oriented environment. Chapter IV documents the design of the distributed object oriented simulation model, supporting tool and how the result of the simulation run can be used. Chapter V contains the comparative study on the results of the simulation model with the experimental result from Professor William J Ray's dissertation. Chapter VI contains the comparative study of the mathematical and simulation model on a more complex scenario. Chapter VII discusses future refinement to the model. Chapter VIII contains the conclusions that can be drawn from this research.

The appendix contains the following information

- Appendix A – List of source code of the simulation model.
- Appendix B – List of source code of the supporting utility.
- Appendix C – Experiment described in Chapter V.
- Appendix D – Data collected from the test described in Chapter VI.
- Appendix E – The modified Lingo model.

II. BACKGROUND

A. INTRODUCTION

There has been little work on deployment strategies for distributed object servers. Some relevant research is in the area of load balancing and client/server performance.

The goal of load balancing is to balance the load across multiple machines. For a situation where inter object server call is prevalent, better response time may require having different object servers to run on a single machine. Therefore, to optimize the response time, previous load balancing strategies may not be useful.

The focus of current optimization technique for the client/server performance is to improve the performance of single server relationship with its clients. The need of analyzing distributed object servers architecture is to target performance when multiple servers are involved.

A research work by Professor William J. Ray describe a method that can generate distributed object oriented server deployment architectures to take advantage of hardware platform and network resources for the purpose of reducing average client response time. Average client response time was chosen over minimizing the maximum response time of one call because the method takes into account the entire usage profile. The proposed optimization model maps system characteristic (called profiles) into equations to minimize the average client response time.

The work done by Professor William J. Ray is used as a springboard for the development of this research work and the rest of this chapter will describe the approach he adopted.

B. PROFILES

A profile is an abstraction of a given characteristic of the system. The elements in the profile are the raw data that the model will use to reason about

the given characteristic. There are profiles for each machine, server, application and user type. There is also a profile that describes the network. The system then must map all of these profiles into equations to minimize the response time. The more complex the modeling of the hardware becomes the more computationally intensive the approach will become. Initially, the research explores an approach with simplistic profiles to demonstrate its capabilities.

1. Hardware Profiles

The hardware profiles describe the characteristic of the hardware platforms used in the deployment. The aspects being modeled in the hardware profiles include characteristics of each computer such as CPU speed, RAM size. The hardware profile also models the network speed between each pair of computers.

An example of hardware profile is:

Hardware	RAM	CPU speed
SIX	64MB	600MHz
BR733	128MB	733MHz
GIGA	128MB	1000MHz

Table 2. Example of hardware profile

2. Object Server Profiles

The object server profile describes the behavior and method supported by an object server. It models the method computational time, return message size and RAM utilization of the object server. The computational time can be collected easily with a small client application that exercises each method call and records the data. Thus, actual implementation code for the application isn't needed to estimate the object server profiles. In the optimization model, the computational time of each method call needs to be captured and normalized to specific hardware architecture. The return message size is used to compute the network

transmission time and therefore giving preference for the object server to be deployed on a hardware platform with a higher network speed. The object server RAM utilization is used to limit the number of object servers by the hardware RAM capacity.

An example of object server profile is:

Object Server	RAM	Method	Return message
A	44MB	1	100kbytes
		2	120kbytes
B	60MB	1	50 kbytes
		2	200 kbytes
		3	20 kbytes

Table 3. Example of object server profile

3. Client Application Profiles

The client application profiles describe the client interaction with the object servers. It includes modeling the object server method call and the estimated frequency of call. Profiling becomes more difficult if the application code is not available. When source code is not available, mechanism must be in place to record all the events that occur in the task. The system must allow a user to create typical scenarios and record the method calls that occur in the scenario. This could be done by simulation or monitoring calls to the object servers when the system is in a training mode.

An example of client application profile is:

Client Application	Button	Methods Called
Client 1	Button 1	A.1
	Button 2	A.2 + B.1
Client 2	Button 1	C.1 + D.3

	Button 2	B.1 + B.2
	Button 3	C.2 + D.4

Table 4. Example of client application profile

4. User Profiles

The user profile describes the way a user interacts with a client application. The way a user interacts can be characterized, but not precisely predicted. User interaction is based on the role the user played. Different role will execute different application that would interact differently with the distributed object servers. A more refined profile could include frequency information for the tasks and calls for each task and response time goals for each task. By profiling each role, the user could choose to re-optimize his deployment to decrease the response time when user chosen roles change. Multiple roles can exist for each user. The user could then select a set of roles and have the system come up with an optimal deployment strategy to meet these criteria.

An example of user profile is:

Role	Button called	Frequency over 20 mins
Role 1	Client 1. Button 1	50
	Client 2. Button 2	10
Role 2	Client 1. Button 2	2
	Client 2. Button 1	10
	Client 2. Button 3	5

Table 5. Example of user profile

C. OPTIMIZATION MODEL

1. Objective Function

The objective function that needs to be minimized is the sum of all the response times for a given call pattern over a given time interval.

$$\text{Minimize} \left[\sum_{n=0}^N \sum_{m=0}^M \frac{a_{nm} * R_n * S_{norm}}{S_m} + \sum_{i=0}^N \sum_{j=0}^N \frac{B_{ij}}{Q_{ij}} \right]$$

Note that the optimization process ranges over all possible combinations for a_{nm} and finds the minimum based on the above objective function and constraints.

The objective function is subject to the following four constraints:

- Object servers cannot be split across machine

$$\begin{aligned} \mathbf{A}_{nm} &= 1, \text{ iff server } n \text{ is running on machine } m. \\ &= 0, \text{ otherwise} \end{aligned}$$

- Each server can run on only one machine [no multiple instances of the same server.

$$\forall n \left[\sum_{m=0}^M a_{nm} \equiv 1 \right]$$

- RAM usage by the object servers cannot pass a set threshold on each machine.

$$\forall m \left[\sum_{n=0}^N a_{nm} * V_n \leq T_m * U \right]$$

- CPU time on a given machine cannot surpass the corresponding real time interval.

$$\forall m \left[\sum_{n=0}^N \frac{a_{nm} * R_n * S_{norm}}{S_m} \leq C \right]$$

where

N = Number of object servers

M = Number of physical machines

R_n = Normalized machine load of server n (seconds, s)

S_{norm} = Speed of the normalizing machine (MHz)

S_m = Speed of machine m (MHz)

B_{ij} = Data sent between server i to server j (bits, b)

Q_{ij} = Network Speed between server i to server j (bps)

T_m = Physical RAM on machine m (bits, b)

n = Memory allocated by server n (bits, b)

U = Multiple to limit RAM utilization [$0.1 < U < 3.0$]

C = Time Interval [seconds, s]

2. Processing Speed Term

This part of the function looks at all possible deployment patterns.

$$\left[\sum_{n=0}^N \sum_{m=0}^M \frac{a_{nm} * R_n * S_{norm}}{S_m} \right]$$

a_{nm} is used to keep track of the deployments. a_{nm} is zero if SERVER n is not located on MACHINE m . If SERVER n is location on MACHINE m , then a_{nm} is one.

3. Network Speed Term

The network speed term of the objective function is:

$$\text{Minimize} \left[\sum_{i=0}^N \sum_{\substack{j=0 \\ j \neq i}}^N \frac{B_{ij}}{Q_{ij}} \right]$$

The network speed term adds some time for each time a server-to-server method is called. The number of bits is divided by the rate of transmission.

4. RAM Limits

The RAM limit is to limit the amount of a machine RAM that can be used by the object servers. This constraint basically states that the total memory usage of all the object servers loaded on a machine will be less than a percentage of the memory on that machine.

5. CPU Limits

Since all of the processing measurements are averages and the user profiles are averages over time, we cannot exceed 100% CPU loading. Even though the CPU can queue tasks when overloads, it doesn't have the chance to catch up if the user profiles truly reflect the user requests.

D. VALIDATION OF OPTIMIZATION MODEL

The optimization model is validated by experimental measurement. A test bed was created with Windows 2000 machines that match the characteristics of the profiles. Servers were created using JDK 1.3 and RMI as the middleware.

Software to simulate the three different users was also created. This simulation software was instrumented to measure the actual time the software was blocked waiting for an object server method call to response. The experiment involves 3 hardware and 3 object servers, a total of 27 different deployment patterns. All 27 different configurations were established and the average response time for each configuration was measured and recorded.

All 27 configurations were tested twice. One tested the configuration with the object servers using much less than the stated memory needs. Another tested the configuration with the object servers using all of the stated memory needs. Some configurations strained the machines memory limit. These configurations resulted in system failures in the test results.

E. CURRENT RESEARCH APPROACH

This research aims to develop a simulation model for the distributed object-oriented environment. It uses the same profile representation as the optimization model. Deployment strategy with different profiles is build using a discrete event simulation tools OMNet++.

The model will simulate user and object server interaction and compute the average response time for each user. The average response time for each deployment strategy is compiled and the minimum user response time is chosen as the optimal deployment strategy.

The experimental and optimization model of 3 object servers and 3 machines is chosen to verify the accuracy of the simulation model to the real-world implementation.

III. OVERVIEW OF OMNET++ AND NETWORK SIMULATION

A. BACKGROUND

OMNet++ is an object oriented modular discrete event simulator. The name itself stands for Objective Modular Network Test-bed in C++. The development of OMNet++ was started at the Technical University of Budapest (BME), in 1992. It has been developed mostly by Andras Varga at the Department of Telecommunication (BME-HIT).

The simulation tools can be used for modeling:

- Communication protocols
- Computer networks and traffic modeling
- Multi-processor and distributed systems.
- Administrative systems
- Any other system where the discrete event approach is suitable.

A Discrete Event System is a system where state changes (events) happen at discrete points of time, and events take zero time to happen. It is assumed that nothing (i.e. nothing interesting) happens between two consecutive events, that is, no state change takes place in the system between the events (in contrast to continuous systems where state changes are continuous).

Those systems that can be viewed as Discrete Event Systems can be modeled using Discrete Event Simulation such as OMNet++. (Continuous systems are modeled using differential equations and suchlike.). Computer networks are usually viewed as Discrete Event Systems.

B. MODELING CONCEPTS

OMNet++ provides efficient tools for the developer to describe the structure of the actual system. Some of the main features are:

- Hierarchically nested modules.

- Modules communicate with message through channels.
- Flexible module parameters.
- Topology description language.

1. Hierarchical Nested Modules

An OMNet++ model consists of hierarchically nested modules. The depth of module nesting is not limited, which allows the user to reflect the logical structure of the actual system in the model structure. The model is often referred to as networks. The top level model is the system module. The system module contains sub-modules, which can also contain sub-modules themselves. Simple modules encapsulate C++ code that generate and react to events, in other words, implement the behavior of the model. Modules can have parameters which are used for three main purposes: to customize module behavior; to create flexible model topologies and for module communication as shared variables.

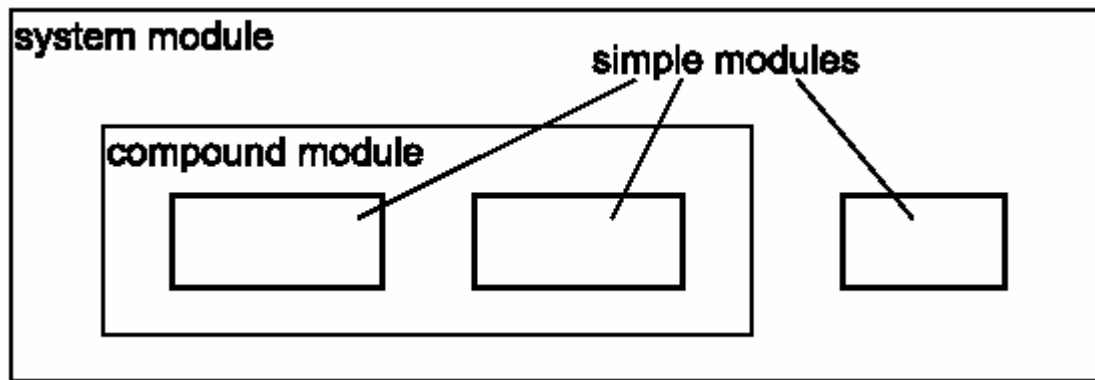


Figure 1. Simple and Compound Modules

Client computers, servers, and network devices can be modeled as sub-modules.

2. Communicate with Message through Channel

Gates are the input and output interfaces of the modules; messages are sent out through output gates and arrive through input gates. Each connection is created within a single level of the module hierarchy: within a compound module,

one can connect the corresponding gates of two modules, or a gate of one sub-module and a gate of the compound module.

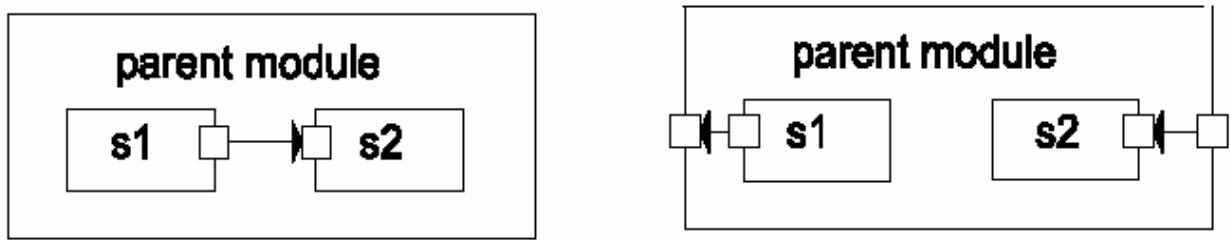


Figure 2. Connecting modules

Three attributes can be assigned values in the body of the connection declaration, all of them optional:

- Propagation delay (sec) is the amount of time the arrival of the message is delayed when it travels through the channel.
- Bit error rate (errors/bit) has influence on the transmission of messages through the channel.
- Data rate (bits/sec) is used for transmission delay calculation.

OMNeT++ uses messages to represent events. Messages are sent from one module to another – this means that the place where the “event will occur” is the message’s destination module, and the model time when the event occurs is the arrival time of the message. Modules can send messages directly to their destination or along a predefined path, through gates and connections. The “local simulation time” of a module advances when the module receives a message. The message can arrive from another module or from the same module.

Time within the model is often called simulation time, model time or virtual time as opposed to real time or CPU time or which refers to how long the simulation program has been running or how much CPU time it has consumed.

3. Parameters

Modules can have parameters that are used for three purposes:

- To parameterize module topology
- To customize simple module behavior
- For module communication, as shared variables.

Within a compound module, parameters can define the number of sub-modules, number of gates, and the way the internal connection are made. Compound modules can pass parameters or expressions of parameters to their sub-modules.

4. Topology Description Method

The developer defines the structure of the model in the NED (Network Description) language descriptions. The NED language supports modular description of a network. This means that a network description consists of a number of component descriptions (channels, simple/compound module types). The channels, simple modules and compound modules of one network description can be used in another network description.

C. PROGRAMMING THE ALGORITHM

The simple modules of a model contain the algorithms as C++ functions. The full flexibility and power of the programming language can be used, supported by the OMNet++ simulation class library. Elements of the simulation (messages, modules, queues etc.) are represented as objects and they are designed so that they can efficiently work together, creating a powerful framework for simulation programming.

D. RUNNING THE SIMULATION

An OMNet++ model consists of the following parts:

- NED language topology description(s) which describe the module structure with parameters, gates etc.
- Simple modules sources that implement the behavior of each modules.

The simulation system provides the following components:

- Simulation kernel. This contains the code that manages the simulation and the simulation class library.
- User interfaces. OMNet++ user interfaces are used with simulation execution, to facilitate debugging, demonstration, or batch execution of simulations.

1. Running the Simulation

The simulation executable is a standalone program, thus it can be run on other machines without OMNet++ or the model files being present. When the program is started, it reads in a configuration file (usually called `omnetpp.ini`); it contains setting that control how the simulation is run, values for model parameters etc. The configuration file can also prescribe several simulations runs; in the simplest case, they will be executed by the simulation program one after another.

OMNet++ simulations can feature different user interfaces for different purposes: debugging, demonstration and batch execution. Advanced user interfaces make the inside of the model visible to the user, allow him/her to start/stop simulation execution and to intervene by changing variables/objects inside the model. User interfaces also facilitate the demonstration on how a model worked internally.

The same simulation model can be executed with different user interface without any change in the model files themselves. The user would test and debug the simulation with a powerful graphical user interface, and finally run it with a simple and fast user interface that supports batch execution.

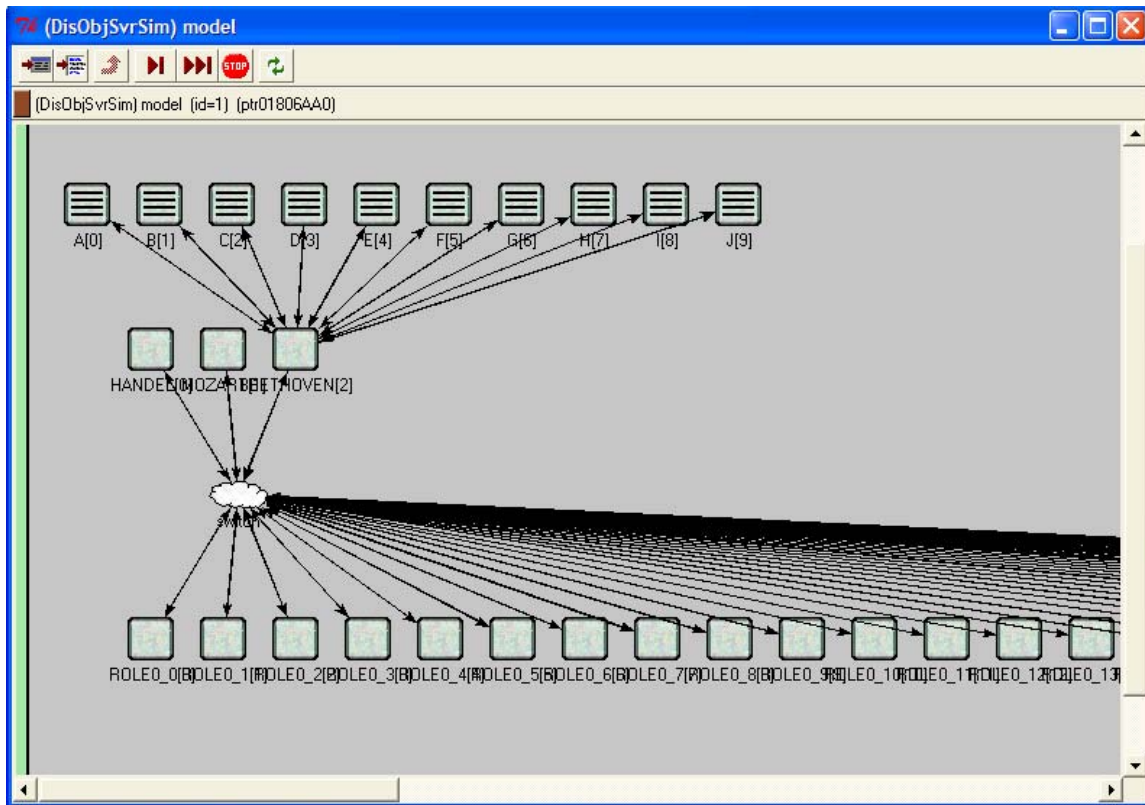


Figure 3. Demonstration user interface

Figure 4. Debugging window

2. Analyzing The Result

The output of the simulation is written into data files: output vector files, output scalar files, and possibly the user's own output files. Data such as resource utilization, response time can be written into the output files. OMNet++ provides a GUI tool names Plove to view and plot the contents of the output vector files. The output files are text files in a format which can be read into math packages like Matlab, or imported into spreadsheets like Excel.

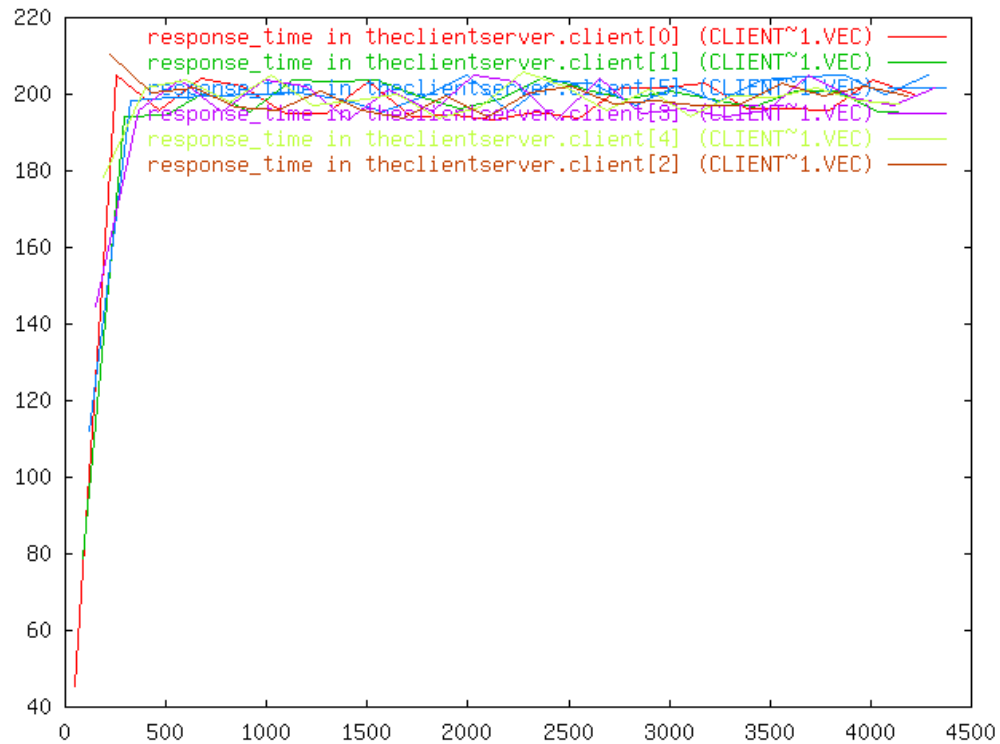


Figure 5. Plove view of response time output file.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. OBJECT ORIENTED SERVER SIMULATION MODEL

A. INTRODUCTION

This chapter will describe the requirement, design consideration and implementation of the simulation model.

B. REQUIREMENT MODELING

1. Use Case Diagram

The block diagram below describes the how the system designer interacts with the Distributed Object Oriented Server Simulation Model.

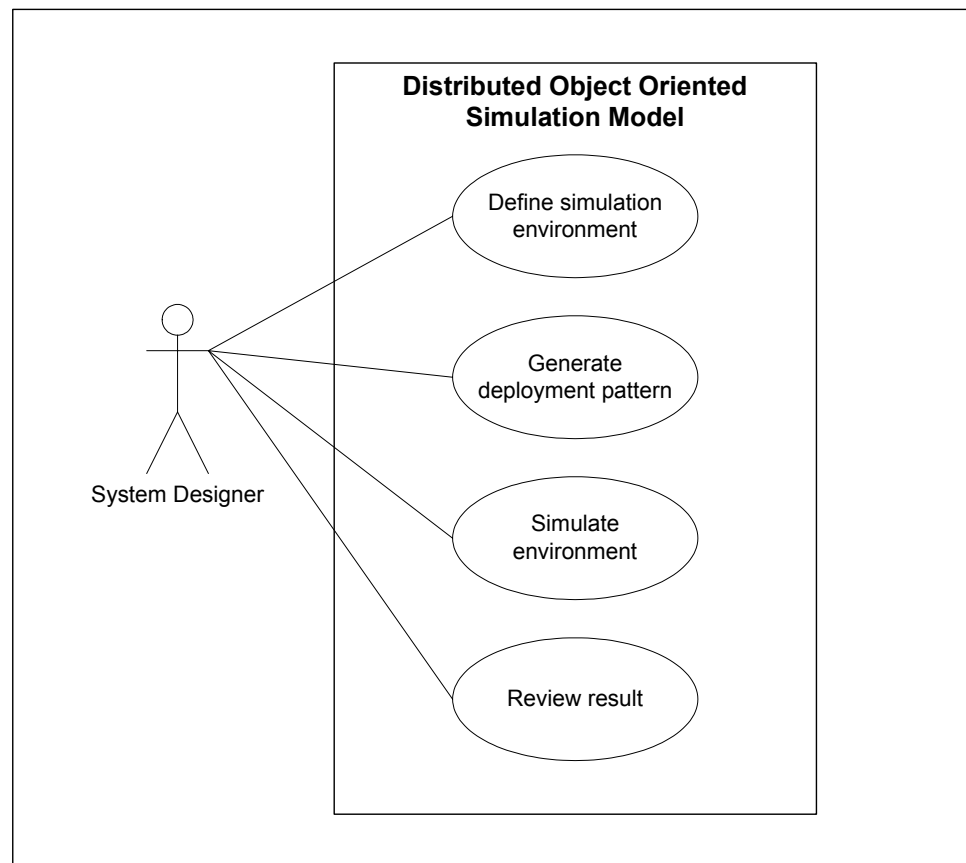


Figure 6. User Case Diagram

2. Actor Description

The actor that interacts with the simulation model is as follows:-

Actor	Description
System Designer	The actor is responsible for determining the optimal deployment pattern. The actor will describe the simulated environment with hardware, object server, application and role profiles. The actor will execute the simulation model and determine the optimal deployment pattern based on the simulation result.

Table 6. Actor description

3. Use Case Description

The following is the description of the use case.

Use Case : Define simulation environment

Brief Description :

To allow the user to define the environment to be simulated based on the machine, object server, role, and client application profile.

Special Requirements:

The simulation model should be a black box to the system designer. The redefinition of a new simulation environment should be achieved by configuration file. The definition of the simulation environment should not require the user to modify the source code of the simulation model

Pre-condition

The environment to be simulated must already be defined. That includes the hardware, object server configuration, and the role.

Use Case : Generate deployment pattern

Brief Description :

To allow the user to generate all the deployment pattern based on the number of machine and object server.

Special Requirements:

Nil

Pre-condition

The number of machine and object server must be known.

Use Case : Simulate environment**Brief Description :**

To allow the user to define the simulation duration and execute the simulation.

Special Requirements:

Nil

Pre-condition

The simulation environment must be defined.

Use Case : Review result**Brief Description :**

To allow the user to review and compare the simulation result of different deployment pattern.

Special Requirements:

Nil

Pre-condition

The simulation must be executed.

C. CLASS DESCRIPTION

1. Class Inheritance

The behaviour of each component in the simulated environment is modeled as an extension to the cSimpleModule class provided by the OMNet++ library. The class inheritance diagram is shown in the following figure:

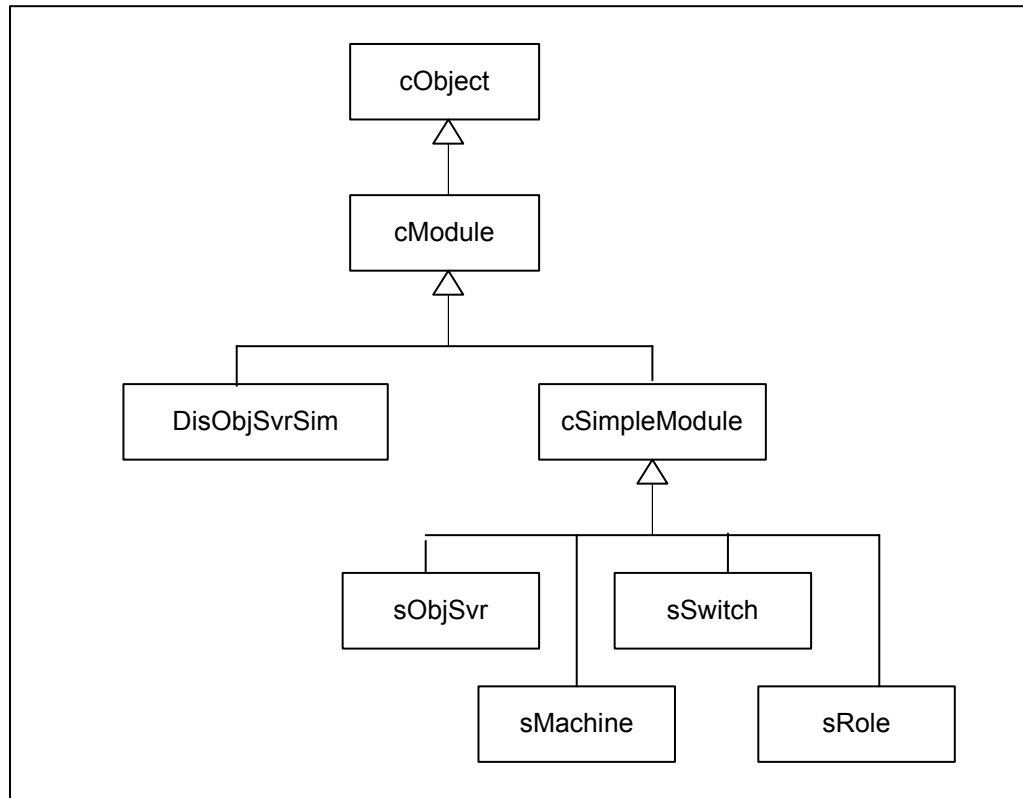


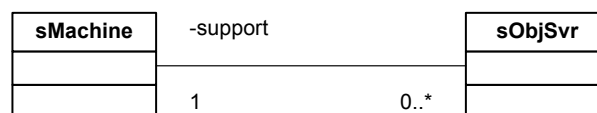
Figure 7. Class inheritance diagram

The purpose of each class is described in the following table:-

Class	Description	Type
sObjSvr	sObjSvr is used to model the behavior of an object server.	Control Object
sMachine	sMachine is used to model the behavior of a computer hardware.	Control Object
sSwitch	sSwitch is used to model the behavior of a network switch.	Control Object

sRole	sRole is used to model the behavior of a role.	Control Object
CAddress	CAddress is used to store the address of each machine and role that is connected with the switch. The address is used by the DNS Service to provide location independent name lookup to determine the gate to send a message.	Entity Object
CBtnList	CBtnList is used to store the list of client application button and the function called when the button is invoked.	Entity Object
CDNSSvc	CDNSSvc implement the necessary function needed to provide location independent lookup service.	Control Object
CInteractList	CInteractList is used to store the list of complex method call.	Entity Object
CObjList	CObjList is used to store the list of method supported by all object servers.	Entity Object
CSimEvent	CSimEvent is used to model the random invocation of application button by a role.	Control Object
CStatLog	CStatLog is used to write the simulation result in the output file stat.log and computed.log.	Interface Object

2. Association Diagram



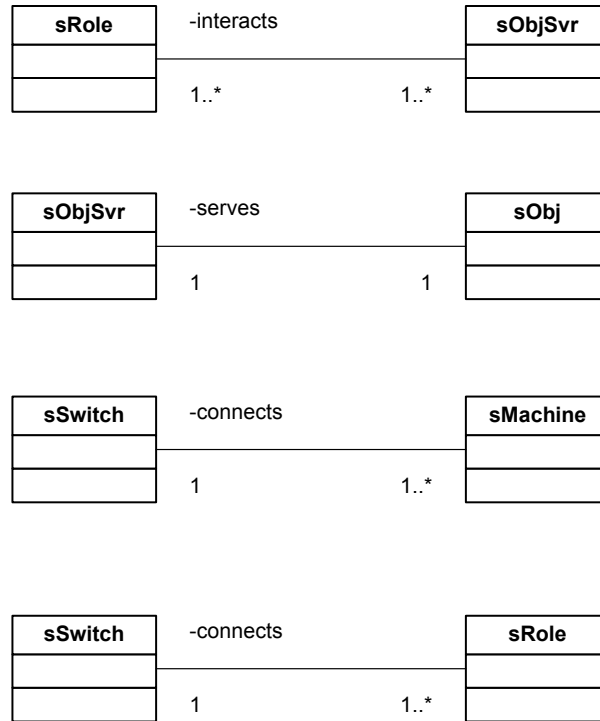


Figure 8. Class diagram

D. NETWORK DESCRIPTION (NED) FILE

The Network Description (NED) language describes the network environment. The NED description file will enable the simulation model to automatically build the distributed object server environment based on the parameters supplied by the designer. The parameters used in the defining the environment is shown in the following table:

Parameter	Description
run_name	The name of the simulation run. This is used to identify the output result of the different run in a batch execution. Typically the run_name is given value as run1, run2....runN.
num_machine	The number of machines deployed in the simulated environment. The number of machines will affect the number of connection created on the sSwitch to support switch-machine network communication.

num_role	The number of roles deployed in the simulated environment. The number of roles will affect the number of gates created on the sSwitch to model the machine-role network communication.
num_objsvr	The number of object servers deployed in the simulated environment.
max_objsvr	The maximum number of object servers deployed on any machine. This value is used to limit the number of gates created on each of the sMachine. sMachine that supports fewer than the max_objsvr will have unused gates.

Table 7. Network Description File Parameters

The Network Description (NED) language also defines the simple module used to model the object server, machine and the network switch. Each simple module definition includes parameters and gates that will be used to setup the environment and how each of the simple modules inter-connect. The actual value to be used is defines in the [Parameters] section of the omnet.ini file.

Modules	Parameters/Gates	Description
sRole	def_file	The filename of the configuration file used by the role.
	<i>r2sw_out</i>	The gate used for sending message from role to switch.
	<i>r2sw_in</i>	The gate used for receiving message from the switch.
sSwitch	<i>sw2r_out[]</i>	The gates used for sending message from switch to role. The number of gates is created dynamically based on the number of roles specified in num_role.
	<i>sw2r_in[]</i>	The gates used for receiving message from

		switch. The number of gates is created dynamically based on the number of roles specified in num_role.
	<i>sw2m_out[]</i>	The gates used for sending message from switch to role. The number of gates is created dynamically based on the number of roles specified in num_machine.
	<i>sw2m_in[]</i>	The gates used for receiving message from switch. The number of gates is created dynamically based on the number of machines specified in num_machine.
sMachine	num_objsvr	The number of object servers that was running on this machine.
	objsvridx	The starting index of the first object server running on this machine. This index is used to reference to the list of object server created.
	def_file	The filename of the configuration file used by the machine.
	<i>m2sw_out</i>	The gate used for sending message from machine to switch.
	<i>m2sw_in</i>	The gate used for receiving message from the machine.
	<i>m2os_out[]</i>	The gates used for sending message from machine to object server. The number of gate is created dynamically based on the number of object servers specified in num_objsvr.
	<i>m2os_in[]</i>	The gates used for receiving message from

		object server. The number of gates is created dynamically based on the number of machines specified in num_objsvr.
sObjSvr	cfg_file	The filename of the configuration file used by the machine.
	os2m_out	The gate used for sending message from object server to the machine.
	os2m_in	The gate used for receiving message from the machine.

Table 8. Simple Module Parameters

E. MESSAGE DESCRIPTION

Several messages are used in the simulation model. The following table is a description of the messages used.

Message	From/To	Description
NAME_REGISTER	Role/Switch Machine/Switch	Use by the machine and role to register the name with the switch.
OBJSVR_REGISTER	ObjSvr/Machine	Use by the object server to register itself with the machine.
INVOKE_OBJECT_SVR_CALL	Role/ObjSvr	Use by the role to invoke an object server call.
INVOKE_OS2OS_CALL	ObjSvr/ObjSvr	Use by an object server to call another object server. This is used to model complex method call.
MACHINE_COMPLETE_	Machine/ObjSvr	Use by the machine to inform

EXEC		the object server that the local method call has completed.
MACHINE_COMPLETE_EXEC_WITH_REMOTE	Machine/ObjSvr	Use by the machine to inform the object server that a remote method call has completed.
OBJSVR_EXECUTE_ON_CPU	ObjSvr/Machine	Use by the object server to send a request to the CPU to execute a method call. The number of instruction cycle is appended to the message.
OBJSVR_RESPONSE	ObjSvr/Role	Use by the object server to inform the role that the requested method call has completed.

Table 9. Message Description Table

F. IMPLEMENTATION DETAIL

1. Computation of Response Time

Before the role invokes a client application button, a timestamp will be recorded. The simulation will invoke the relevant object server method calls that are associated to the application button. If there are more than one object-server method calls, a new method call will only be invoked after the role has received the response from the current method call. For example, for the application button C2.B5 described in Chapter V, the role will invoke B.2 after the reply from A.1 is received. When the response from the last method call is received, a second timestamp is recorded and the difference in time is the response time.

The response time is dependent on the following:-

- The number and complexity of method calls. The higher number of method calls has a higher tendency for a longer response time. The more complex the method calls, the longer the response time.
- Network transmission time. This includes the total transmission time to and from the role to machine. The network transmission time is dependent on the message size, and the bandwidth.
- The time needed to execute the object server method call. This is dependent on the CPU utilization, the CPU speed, the number of instruction cycles and the scheduling term.

The response time is consolidated over the simulation run and the average response time is computed.

2. Random Role Execution

A few random feature is added to introduce randomness in the the execution of the simulation model. This is an attempt to model the random nature of the real operating environment. The random features are:-

- Random invocation of client application button by each role.
- Random allocations of CPU resource for the time slice scheduling option.

The probability of an invocation of client application button is assigned during the definition of the role profile. For example, during the training session, it is recorded there was 50 call to C1.B1, 10 call to C1.B2 and 30 call to C2.B2, the configuration file will be defined as:-

```
[call] C1.B1/50
[call] C1.B2/30
[call] C2.B2/10
```

An internal table will be set up with range from 0 to 89, and a random integer based on uniform distribution will be generated to determine which method to fire. For example, if the number drawn is 45, C1.B1 will be called and if the number drawn is 81, C2.B2 will be called. Due to this random implementation,

the simulation will model closer to the real world environment if the simulation is run over sufficient period of time.

For modeling random allocation of CPU resource for time slice scheduling, a maximum CPU allocation time is defined in the hardware profile. In real life, this value represents the longest time a process can execute on the CPU without being interrupted. When a process starts to consume CPU resource, the time slice scheduling will pick a random time between 0 and the maximum CPU allocation time, this random time will be the duration (sec) a process can execute on the CPU. After the duration has passed, the current process will be suspended, a new process selected for execution and another random CPU execution time will be generated.

3. RAM Limit

Each machine has limited RAM capacity, with virtual memory the number of applications that can execute simultaneously on a single machine is no longer limited by the RAM capacity. In virtual memory, when a reference memory is not in RAM, a page fault will occur and the reference memory will be moved from the virtual memory (in disk) to the main memory (RAM). This I/O operation is time consuming. As more application runs on a machine, the probability of page fault will increase and thus delaying the process execution time. In the simulation model, the RAM limit is used to limit the total amount of memory used by the object server that is running on the machine.

When a new object server registered to be run on a machine, the machine will monitor the total memory used. If the $(\text{total memory used}) > (\text{Ram limit} * \text{machine total RAM})$ then the simulation will terminate with a memory error. When the Ram limit is set to 0.0, the simulation will assumes that the machine has unlimited memory to support the object server.

4. Output Result

There are three output files that are generated after each of the simulation run. The three files are vector file, stat.log and computed.log.

The vector file is generated by OMNet++. The vector file can be read by Plove application which provides a graphical display of the output data. The strength of the vector file lies in the ease of reading the result. However, when the number of output data is large, then it is difficult to extract the result from the graphical display.

The stat.log and computed.log file are textual files that was formatted in a way that is easy to manipulate using third party spreadsheet application such as Excel. Stat.log file stores the run name, the role id, the number of client applications call invoked, the maximum and minimum response time and the average response time. The stat.log file is useful for comparing the boundary of the response time. The computed.log file stores the run name, the role id and the average response time. For an environment with 4 of role2, stat.log file will contain the result of all four role2 but the computed.log will contain the computed average response time of all role2.

G. CONFIGURATION FILE

A number of configuration file is used to define the simulation model. These configuration files are used to define parameters that are beyond the scope of the OMNet++ parameter files [ometpp.ini]. The list and description of the configuration files is shown in the following table:-

Configuration File	Description
Button.def	The file defines the list of button provided by the client application. The file also defines the list of object server method call when the application button is invoked.
Interact.def	The file defines the list of complex method call.
Obj.def	The file defines the list of method served by all object servers. It defines the list of method, the number of instruction cycles needed and the return message

	size.
[<i>Machine</i>].def	The file defines profile of a machine. It defines the machine name, CPU power, RAM size and limit, process and disk swap time, and the time slicing property.
[<i>ObjSvr</i>].def	The file defines profile of an object server. It defines the object server name, the object served, the ram utilization.
[<i>Role</i>].def	The file defines profile of a role. It defines the role name, the calling frequency, the object server method called and probability of call.

Table 10. Configuration file description

H. PATTERN GENERATOR

As described in Chapter 1, the number of deployment strategy = m^n (m – number of machine, n – number of object server). With 3 machine and 10 object server, there is a total of 59409 deployment strategy. Manual generation of such number of deployment strategy will be a laborious task – an automated deployment strategy generator is needed. The *PatternGenerator* is developed to automate the generation of configuration file. The two configuration files generated are:-

- Deployment strategy configuration file.
- Role configuration file.

1. Deployment Configuration File

The format of the deployment configuration file is shown in Appendix [William 2001]. The complexity lies in ensuring that the correct number of deployment patterns based on the environment is correctly generated. A set of bits (n bits) is used to represent the number of hardware platform such that $2^n >$ number of hardware. Multiple set of bits are used to represent each object

server. These bits are concatenated m times, where m is the number of object server and subsequent bit arithmetic will ensure that all patterns were represented. The resulting bit string will have $n * m$ bits.

For example, in an environment has 3 hardware platforms, [SIX, BR733, GIGA], and 3 object server [A, B, C]. The number of bits need to represent the 3 hardware platform is 2 i.e. $2^2 = 4$. The bit representations are:

[00 – SIX]; [01 – BR733]; [10 – GIGA]; [11 – Invalid]

A resultant bit string of $2*3 = 6$ bit length is needed to represent all the object servers. The arithmetic operation perform is shown in the following table.

Bit pattern	Hardware deployment	Arithmetic
10 10 10	A – GIGA, B – GIGA, C – GIGA,	Subtract bit string with 1
10 10 01	A – GIGA, B – GIGA, C – BR733	Subtract bit string with 1
10 10 00	A – GIGA, B – GIGA, C – SIX	Subtract bit string with 1
10 01 11	Invalid	Subtract bit string with 1
10 01 10	A – GIGA, B – BR733, C – GIGA	Subtract bit string with 1
10 01 01	A – GIGA, B – BR733, C – BR733	Subtract bit string with 1
10 01 00	A – GIGA, B – BR733, C – SIX	
...	...	...
00 00 00	A – SIX. B – SIX, C – SIX	Complete

Table 11. Sample bit arithmetic operation

2. Role Configuration File

In the simulation environment, there are multiple role types and a number of roles per type. For example, an environment may have 10 of Role1, and 5 of Role2. To support the simulation, each role must have a separate configuration file. The *PatternGenerator* will generate all the role configuration file needed based on input value of the role type, the number of role, and the based role configuration file. In the above example, to generate the configuration file for Role1, the input value are:

Role Type : Role1

Number of Role : 10

Based File : role1.cfg

The PatternGenerator will generate 10 role1 type namely role1_0 to role1_9 and create 10 configuration file from the based file. The configuration file will contain the respective role name. The example of the based file and generated role1_0 file is given below.

Based File *role1.def*

```
[name] ROLE1
[type] 1
[min_wait] 6
[max_wait] 7
[call] C1.B1/10
[call] C1.B2/40
[call] C3.B2/24
[dr_sw] 200000000
[btw_call] 0.05
```

Generated File *role1_0.def*

```
[name] ROLE1_0
[type] 1
[min_wait] 6
[max_wait] 7
[call] C1.B1/10
[call] C1.B2/40
[call] C3.B2/24
[dr_sw] 200000000
[btw_call] 0.05
```

V. VERIFICATION EXPERIMENT 1

A. INTRODUCTION

The first verification experiment is based on the scenario used by the William Ray dissertation. The objective of this verification exercise is to verify the accuracy of the simulation model as compared to the real-life testbed and mathematical model.

B. SCENARIO PROFILE

The various profiles used in defining the environment is shown in the following table:-

Machine	RAM (MB)	CPU Speed (MHz)
SIX	64	600
BR733	128	733
GIGA	128	1000

Table 12. Machine Profile

OBJECT SERVER	METHOD	CPU TIME (s)	Number of instruction cycles	Average Size of Message (bits)	RAM Size (MB)
A	1	0.5796	579600	11200	44
A	2	2.6203	2620300	18400	
A	3	1.18175	1181750	44800	
A	4	2.0264	2026400	17600	
B	1	1.76655	1766550	400000	60
B	2	3.70085	3700850	2720000	
C	1	3.0043	3004300	32000	66
C	2	4.804	4804000	4000000	
C	3	0.48815	488150	40000	

Table 13. Object Server Performance Profile

Primary Method	Secondary Method
B.2	C.1

Table 14. Complex Object Server Profile

Roles	Call Pattern
Role 1	50 C1.B1 + 1 C1.B2 + 1 C2.B1 + 1 C2.B6
Role 2	10 C1.B1 + 40 C1.B2 + 24 C3.B2
Role 3	50 C2.B5 + 10 C2.B9 + 30 C2.B3 + 1 C2.B2 + 1 C3.B2

Table 15. Role Call Pattern

Buttons	Method Call
C1.B1	A.1
C1.B2	A.2 + B.1
C1.B1	A.1
C1.B2	A.2 + B.1
C2.B1	C.1 + C.2
C2.B2	C.3
C2.B3	C.2
C2.B4	C.3
C2.B5	A.1 + B.2
C2.B6	B.2
C2.B7	A.4
C2.B8	C.3 + A.3
C2.B9	A.1 + A.2 + A.3 + B.2
C3.B1	C.1
C3.B2	B.1 + B.2
C3.B3	C.2

Table 16. User Interface Call Chart

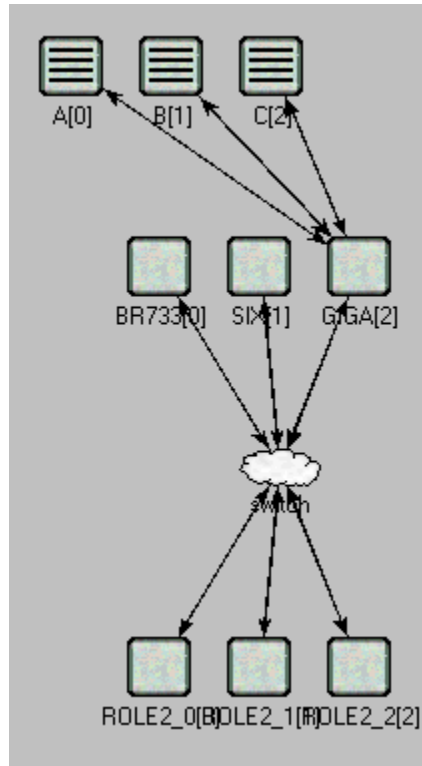


Figure 9. Simulation Model for S5

A total of 9 test cases is run with different number of role participation and ram limits. All test cases were simulated using the simulation model and the result is then compared with the result from the testbed and Lingo model. The 9 test cases are:-

Test Case	Scenario
S1	1 user (Role 1), Ram Limit 1.5
S2	1 user (Role 2) , Ram Limit 1.5
S3	1 user (Role 3) , Ram Limit 1.5
S4	4 user (Role 2) , Ram Limit 1.5
S5	3 user (Role 3) , Ram Limit 1.5
S6	28 user (Role 1) , Ram Limit 1.5
S7	1 user (Role 1), Ram Limit 1.0
S8	1 user (Role 2) , Ram Limit 1.0
S9	1 user (Role 3) , Ram Limit 1.0

Table 17. Test Case

The simulation model was run over a simulation time of 8hrs. In the actual testbed, the time needed to execute 27 patterns x 8hrs = 216hrs. For the

simulation model, execution time for 27 patterns is about 10mins. For the Lingo model, the optimal solution is found within 4 sec.

C. COMPARING SIMULATION RESULT WITH TEST BED

The simulation result of the S1-S9 is shown in the following two tables, highlighted cell represents the lowest average response time and thus the optimal deployment pattern.

Pat	A	B	C	Sim S1	Sim S2	Sim S3	Sim S4	TB S1	TB S2	TB S3	TB S4
01	GIGA	GIGA	GIGA	0.9558	5.2635	7.0909	15.0493	0.97633	5.15036	6.74195	14.60339
02	GIGA	GIGA	BR733	1.0962	5.7279	8.4143	11.67535	0.89934	5.53033	8.26652	11.7461
03	GIGA	BR733	GIGA	1.0558	6.371	7.9366	11.43463	0.96081	6.41717	7.80217	11.71142
04	GIGA	BR733	BR733	1.1033	6.7169	9.2392	15.53168	1.07964	6.68638	9.12494	14.33322
05	BR733	GIGA	GIGA	1.2533	5.8716	7.3726	11.43985	1.1408	5.95302	7.41334	11.3353
06	BR733	GIGA	BR733	1.2313	6.3193	8.6062	10.41733	1.21888	6.23306	8.50534	11.66662
07	BR733	BR733	GIGA	1.1837	6.7915	8.2699	17.17995	1.11909	6.87797	8.14272	17.06668
08	BR733	BR733	BR733	1.2751	7.107	9.5552	22.42225	1.18686	7.23888	9.42866	21.13467
09	GIGA	GIGA	SIX	1.1935	5.9121	9.4624	11.67603	0.99153	5.95855	9.25922	12.35508
10	GIGA	SIX	GIGA	1.0857	7.0883	8.7751	13.61095	0.87878	7.17686	8.62741	14.30257
11	GIGA	SIX	SIX	1.2209	7.6891	11.0043	19.2105	1.15777	7.8528	10.71298	18.37826
12	SIX	GIGA	GIGA	1.3226	6.3354	7.4474	12.10203	1.27438	6.37555	7.33272	12.0353
13	SIX	GIGA	SIX	1.4807	6.8569	9.9274	12.1953	1.40269	6.96919	9.83822	13.88488
14	SIX	SIX	GIGA	1.3782	8.1308	9.3435	22.0154	1.41398	8.21186	8.972	20.87854
15	SIX	SIX	SIX	1.4893	8.7288	11.5528	27.9355	1.64223	8.64436	12.13109	28.11943
16	BR733	BR733	SIX	1.3452	7.4566	10.5915	17.3523	1.19742	7.34209	10.38713	17.40676
17	BR733	SIX	BR733	1.2877	7.955	10.3251	15.1899	1.30637	7.86233	10.36099	15.65908
18	BR733	SIX	SIX	1.4664	8.141	11.3902	19.4705	1.3053	8.51408	11.06739	19.01137
19	SIX	BR733	BR733	1.5399	7.4566	9.7991	16.0319	1.29172	7.60183	9.59142	15.65258
20	SIX	BR733	SIX	1.5391	7.9663	10.9063	13.8823	1.46744	8.03317	10.59013	15.40732
21	SIX	SIX	BR733	1.5078	8.4218	10.6197	21.9925	1.44142	8.22203	10.18545	22.15056
22	GIGA	BR733	SIX	1.2014	7.0065	10.4098	11.1235	1.11434	6.98772	10.25939	11.52439
23	SIX	SIX	BR733	1.058	7.4031	10.0852	13.5477	1.06877	7.42305	9.83488	13.73901
24	BR733	GIGA	SIX	1.2905	6.4926	9.7467	9.1049	1.24636	6.51581	9.563	10.2016
25	BR733	SIX	GIGA	1.2798	7.681	9.0036	14.0829	1.3047	7.78317	8.74324	14.08931
26	SIX	GIGA	BR733	1.519	6.7215	8.8515	9.9500	1.35559	6.7525	8.62544	10.54422
27	SIX	BR733	GIGA	1.4169	7.318	8.4881	12.1508	1.30669	7.38083	8.25905	12.56952

Table 18. Test Result of Test Case S1 – S4.

Pat	A	B	C	Sim S5	Sim S6	Sim S7	Sim S8	Sim S9	TB S5	TB S6	TB S7	TB S8	TB S9
01	GIGA	GIGA	GIGA	16.6589	10.2673	1.0055	5.3012	6.9886	15.9786	9.30717	0.97734	5.12018	6.77685
02	GIGA	GIGA	BR733	11.9086	5.0118	1.0488	5.6899	8.3268	13.9259	4.96472	0.94298	5.58044	8.21316
03	GIGA	BR733	GIGA	12.8099	7.4687	1.0636	6.3843	8.0593	13.0662	4.33376	0.88703	6.34986	7.90056
04	GIGA	BR733	BR733	21.1609	3.1608	1.1247	6.611	9.2771	20.4154	3.78934	1.04139	6.69614	9.21795
05	BR733	GIGA	GIGA	14.9085	7.5642	1.1665	5.8862	7.3435	14.6145	7.00596	1.14467	5.87464	7.26764
06	BR733	GIGA	BR733	15.6307	15.3563	1.3483	6.2286	8.6711	15.7297	14.4355	1.28264	6.20492	8.51984
07	BR733	BR733	GIGA	12.6598	11.3845	1.2064	6.8876	8.1804	13.6160	10.8106	1.22803	6.838	8.23206
08	GIGA	GIGA	SIX	24.2953	18.7025	1.2656	7.2717	9.5299	23.3200	error	1.40952	7.21558	9.37386
09	GIGA	SIX	GIGA	14.9543	5.0028	1.1559	5.9421	9.4775	16.6374	error	1.0393	5.91619	9.46308
10	GIGA	SIX	SIX	14.4386	7.6163	1.0827	7.392	8.776	14.2478	error	0.96261	7.28895	8.53298
11	SIX	GIGA	GIGA	26.8708	3.5508	error	error	error	25.7966	error	error	error	Error
12	SIX	GIGA	SIX	15.2147	12.6736	1.4408	6.2603	7.5815	14.5530	error	1.34883	6.42448	7.34622
13	SIX	SIX	GIGA	19.7351	21.8104	error	error	error	19.0296	error	error	error	Error
14	SIX	SIX	SIX	15.0131	17.3469	error	error	error	15.8605	error	error	error	Error
15	BR733	BR733	SIX	30.6386	26.8308	error	error	error	30.3491	error	error	error	Error
16	BR733	SIX	BR733	15.9958	11.4163	1.4044	7.5513	10.6311	18.1436	error	1.2627	7.3226	10.5296
17	BR733	SIX	SIX	17.4819	15.2354	1.3635	8.145	10.1949	17.6795	error	1.43925	8.14897	10.1235
18	SIX	BR733	BR733	26.8316	7.6992	error	error	error	25.8905	error	error	error	Error
19	SIX	BR733	SIX	21.4192	12.7286	1.5526	7.7897	9.7489	20.7330	error	1.53566	7.74292	9.77058
20	SIX	SIX	BR733	19.6194	21.2661	error	error	error	19.8816	error	error	error	Error
21	GIGA	BR733	SIX	15.9954	17.1591	error	error	error	18.0537	error	error	error	Error
22	SIX	SIX	BR733	15.1154	3.0755	1.0286	6.968	10.3324	16.9338	3.54885	0.98269	6.96762	10.1936
23	BR733	GIGA	SIX	14.1918	2.9624	1.1604	7.4252	10.1292	15.9926	3.01411	1.13197	7.34378	9.80498
24	BR733	SIX	GIGA	14.8577	7.646	1.3054	6.4696	9.7671	16.0315	7.41357	1.31191	6.61303	9.6173
25	BR733	GIGA	SIX	13.0968	7.5482	1.2145	7.7743	9.1113	13.662	6.80710	1.18966	7.54856	8.86581
26	SIX	GIGA	BR733	12.3277	12.6422	1.3706	6.6922	8.8251	13.8393	11.1170	1.3903	6.77245	8.86009
27	SIX	BR733	GIGA	11.7289	12.4794	1.453	7.3095	8.4703	12.4880	12.0423	1.34461	7.45797	8.32806

Table 19. Test Result of Test Case S5 – S9

1. Result for Test Case S1

The graphical result of the simulation model and testbed for Test Case S1 is shown in the following figure. The graph show that there is a general trend of the two recorded response time is similar.

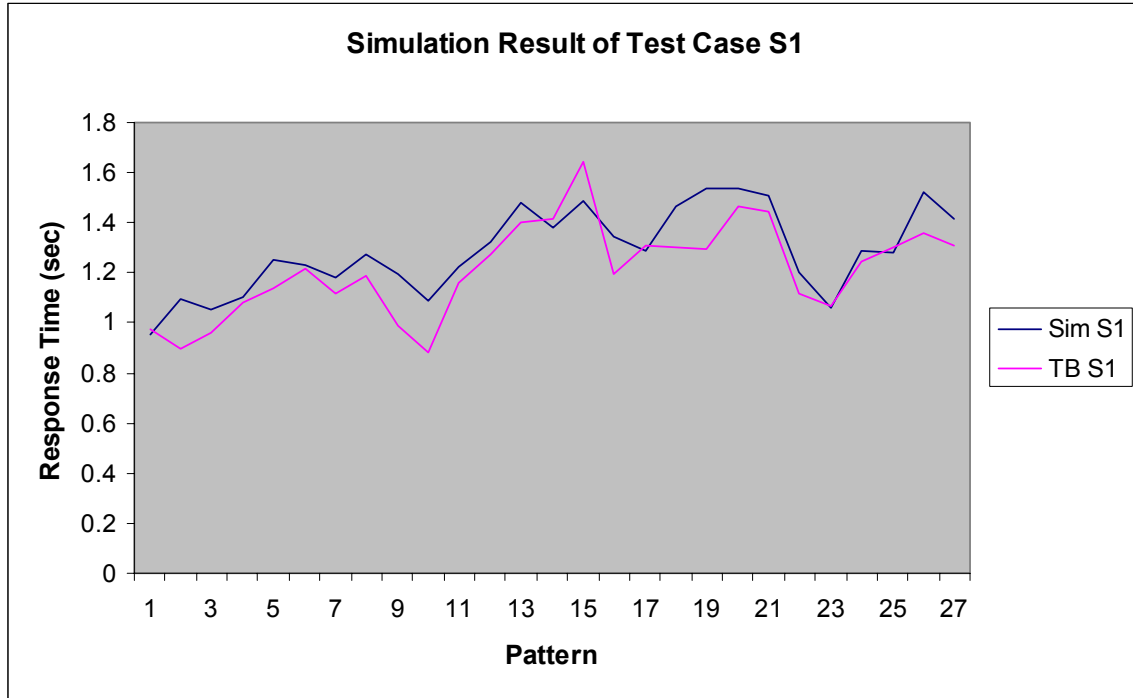


Figure 10. Graphical result of S1

The simulation model chose a pattern that is the 4th best result from the test bed. The difference in the timing selected is 0.098sec.

Roles	Role 1 (1 user), Ram Limit 1.5	
	Sim	Test bed
SIX	None	B
BR733	None	None
GIGA	A, B, C	A, C

Table 20. Optimal Deployment Pattern S1

The difference in the result could be attributed to the randomness of the test bed and simulation.

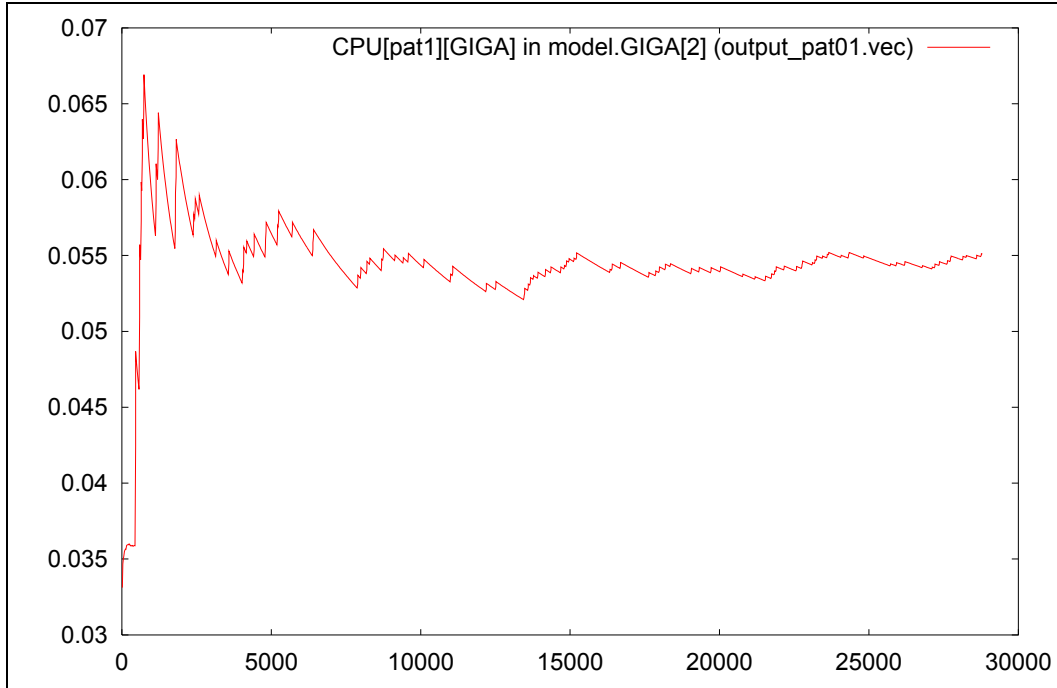


Figure 11. CPU Utilization for Optimal Deployment Strategy Pattern 1

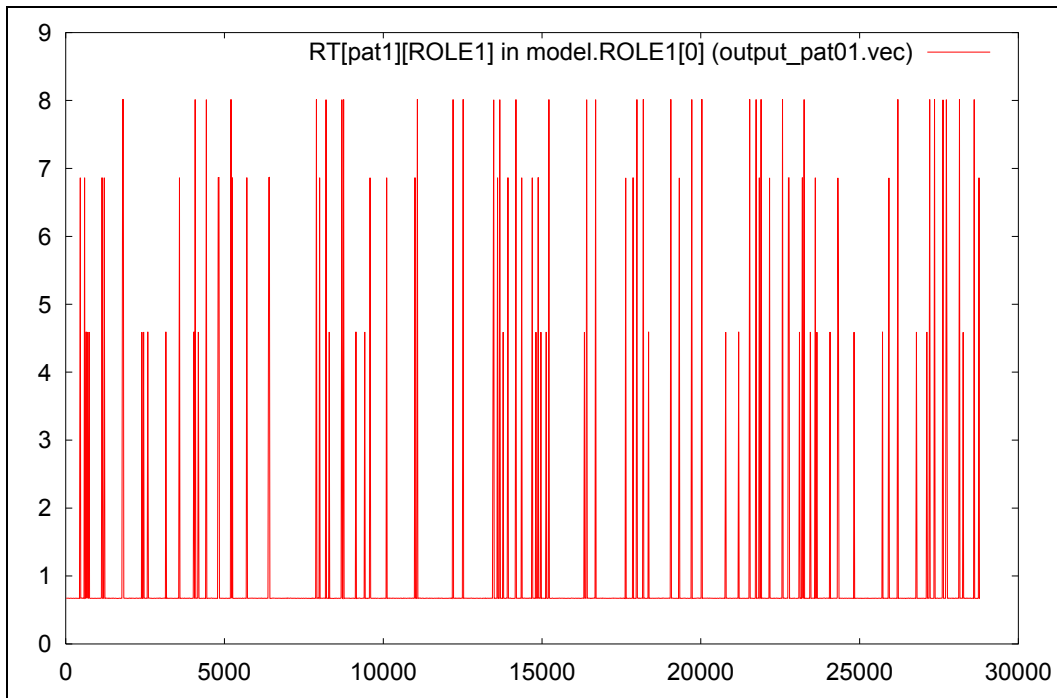


Figure 12. Role Response Time for Optimal Deployment Pattern 1

The CPU utilization diagram shows that the CPU utilization of GIGA is fairly low at (0.55%) and therefore the deployment pattern is able to support a

significant growth in the number of roles. The response time range is (8, 0.8) sec (max, min).

2. Result for Test Case S2

The graph shows that the simulation model modeled the test bed accurately. The highest difference between any deployment patterns is 0.2 sec.

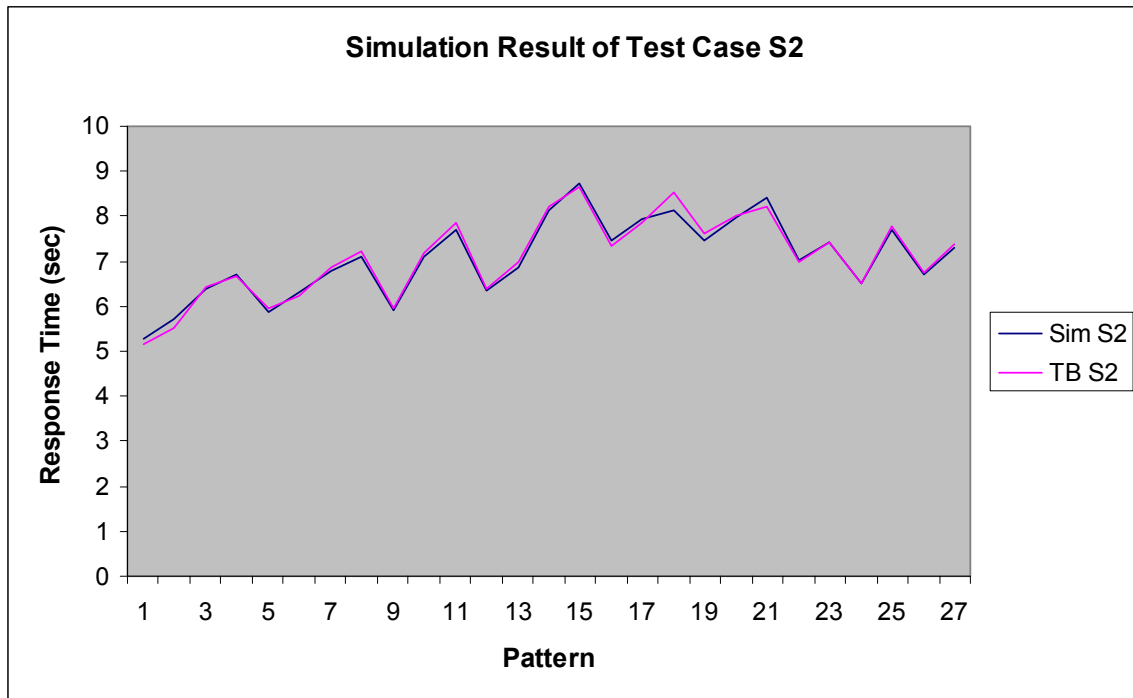


Figure 13. Graphical result of S2

For test case 2, the simulation model chose the same optimal deployment pattern as the test bed.

Roles	Role 2 (1 user) , Ram Limit 1.5	
	Sim	Testbed
SIX	None	None
BR733	None	None
GIGA	A, B, C	A, B, C

Table 21. Optimal Deployment Pattern S2

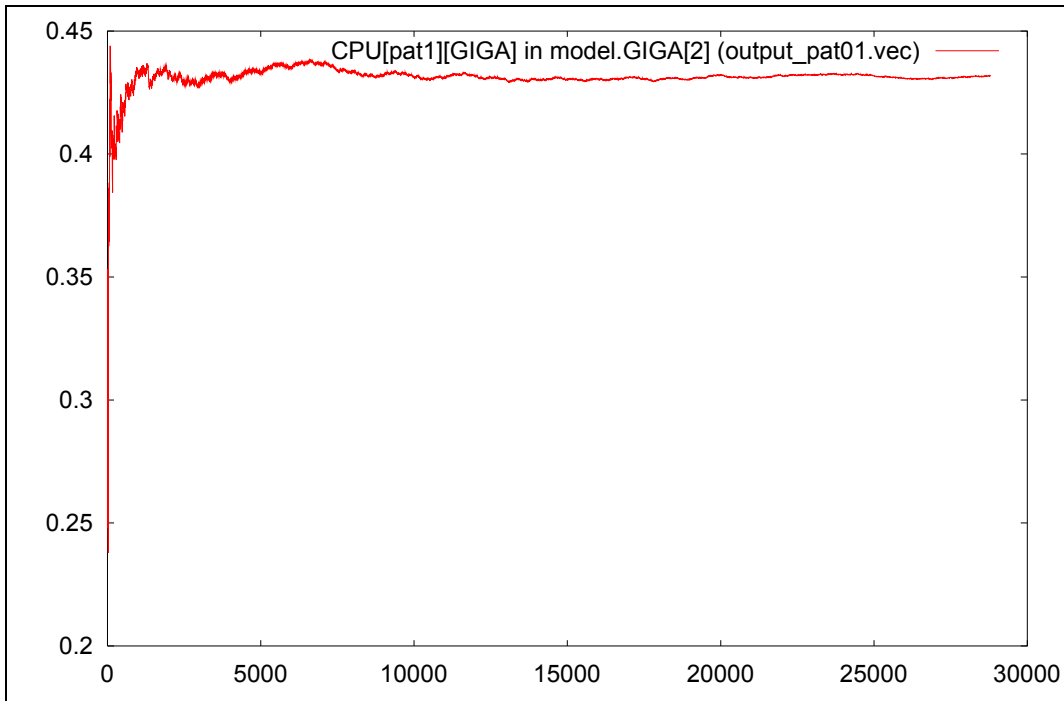


Figure 14. CPU Utilization for S2

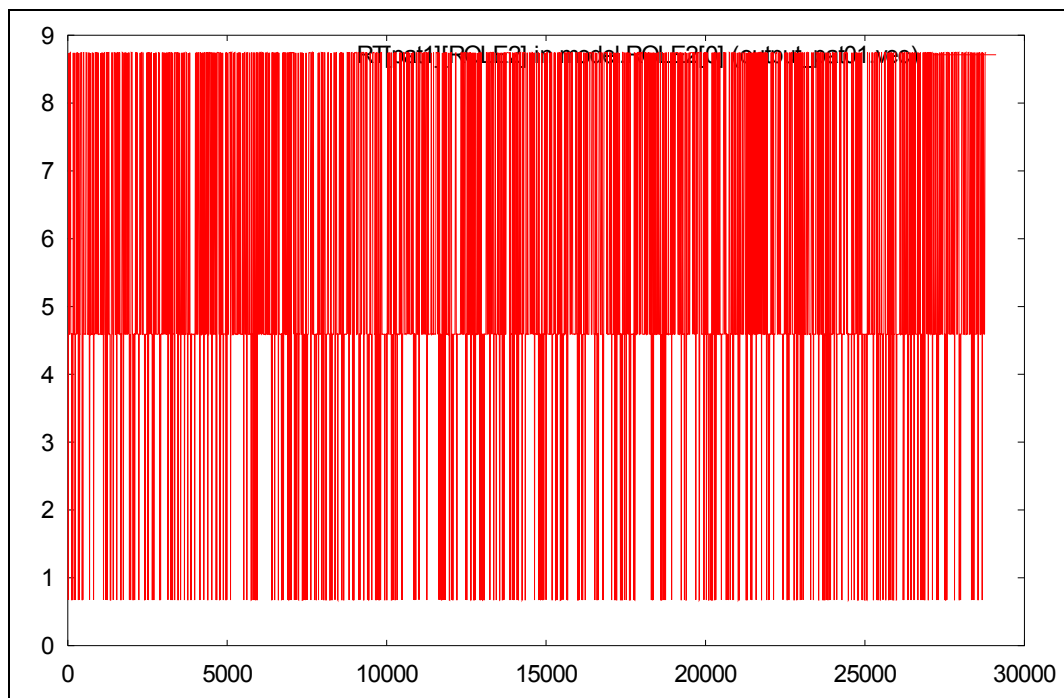


Figure 15. Role Response Time for S2

The CPU utilization diagram shows that the CPU utilization of GIGA is low at (43%) and therefore the deployment pattern is able to support a higher number of roles. The response time range is (8.8, 0.8) sec (max, min).

3. Result for Test Case S3

The graph shows that simulation model modeled the test bed accurately. The highest difference between any two patterns is 0.6 sec.

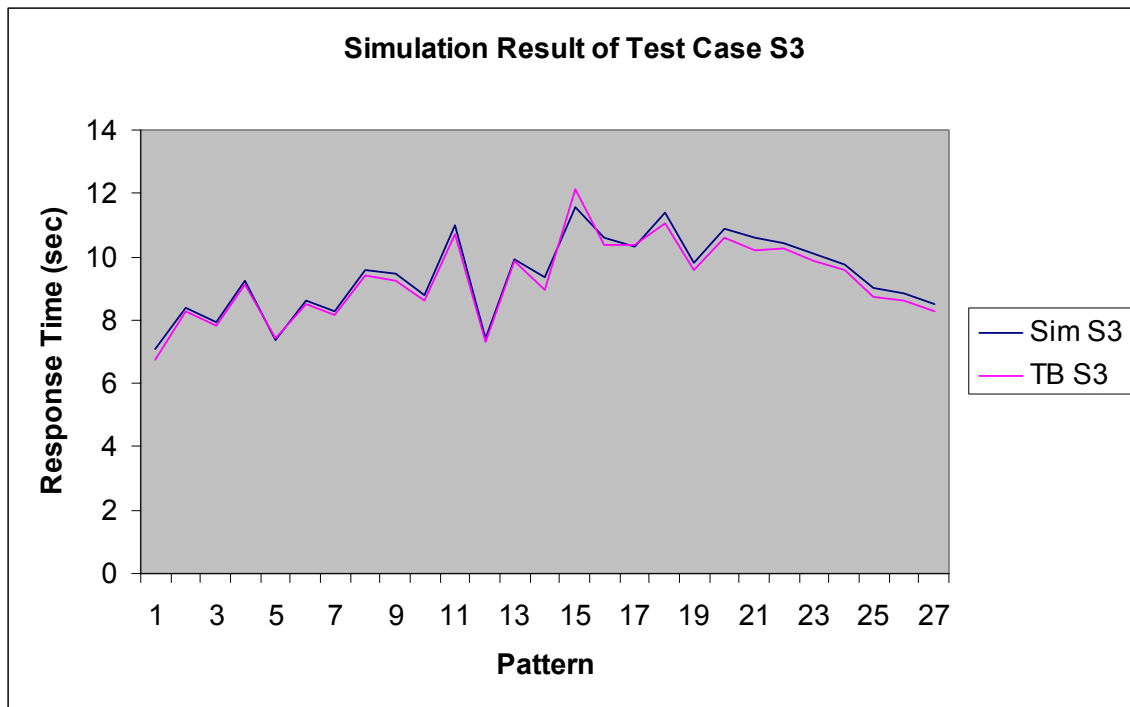


Figure 16. Graphical result of S3

For test case 3, the simulation model chose the same optimal deployment pattern as the test bed.

Roles	Role 3 (1 user) , Ram Limit 1.5	
	Sim	Testbed
SIX	None	None
BR733	None	None
GIGA	A, B, C	A, B, C

Table 22. Optimal Deployment Pattern S3

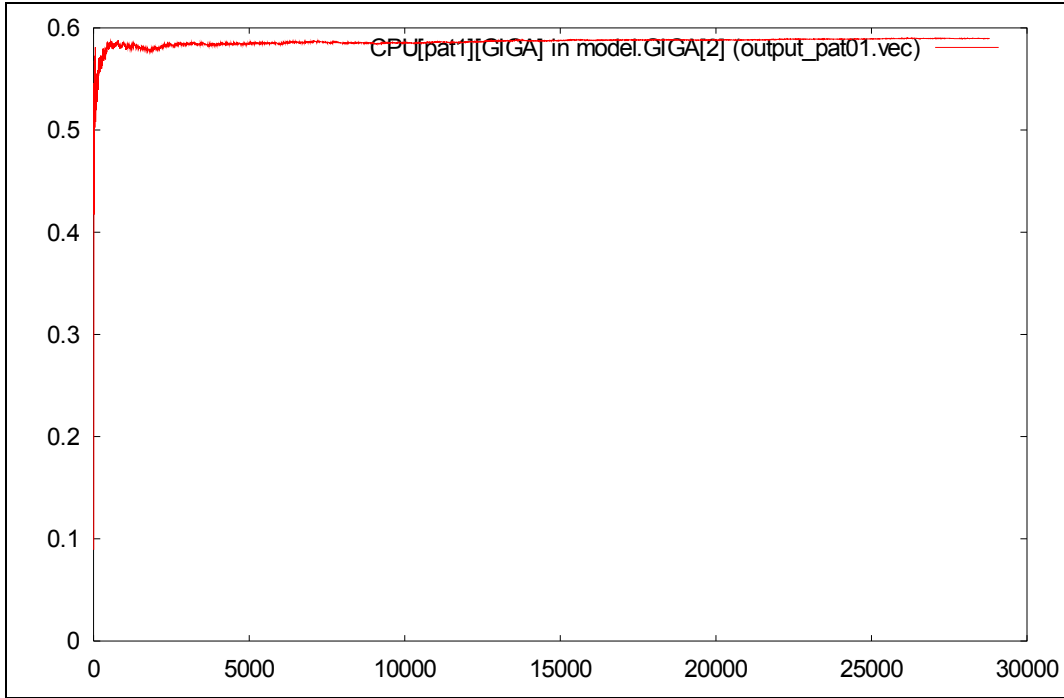


Figure 17. CPU Utilization for S3

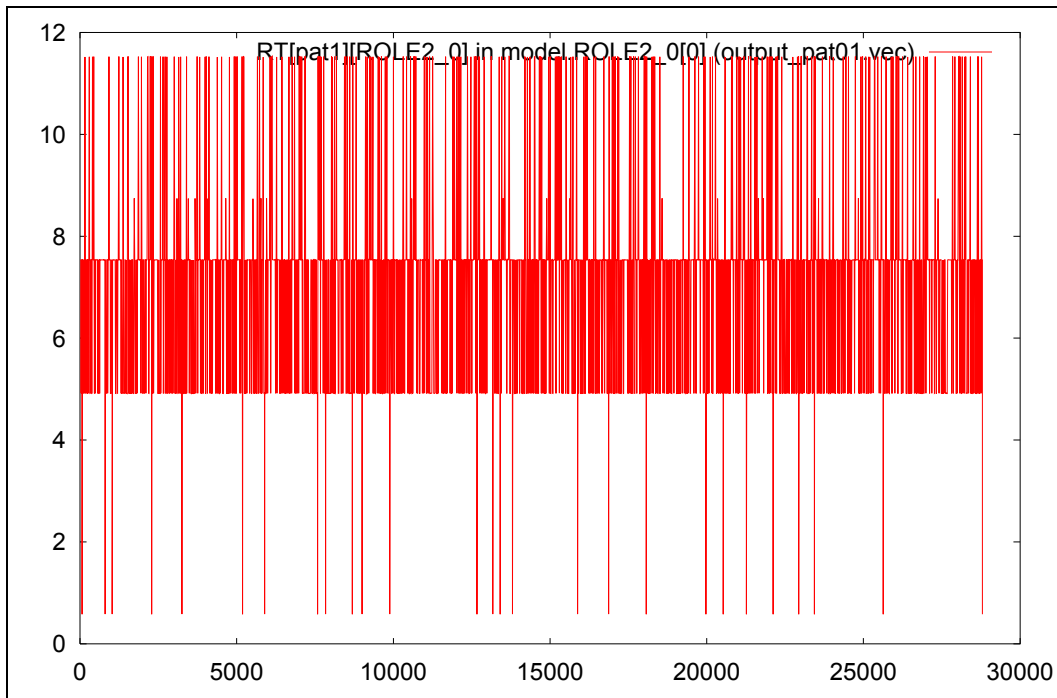


Figure 18. Response Time for S3

The CPU utilization diagram shows that the CPU utilization of GIGA is low at (59%) and therefore the deployment pattern is able to support a small increase in the number roles. The response time range is (11.8, 0.5) sec (max, min).

4. Result for Test Case S4

The graph shows that simulation model modeled the test bed accurately. The highest difference between any two patterns is 1 sec.

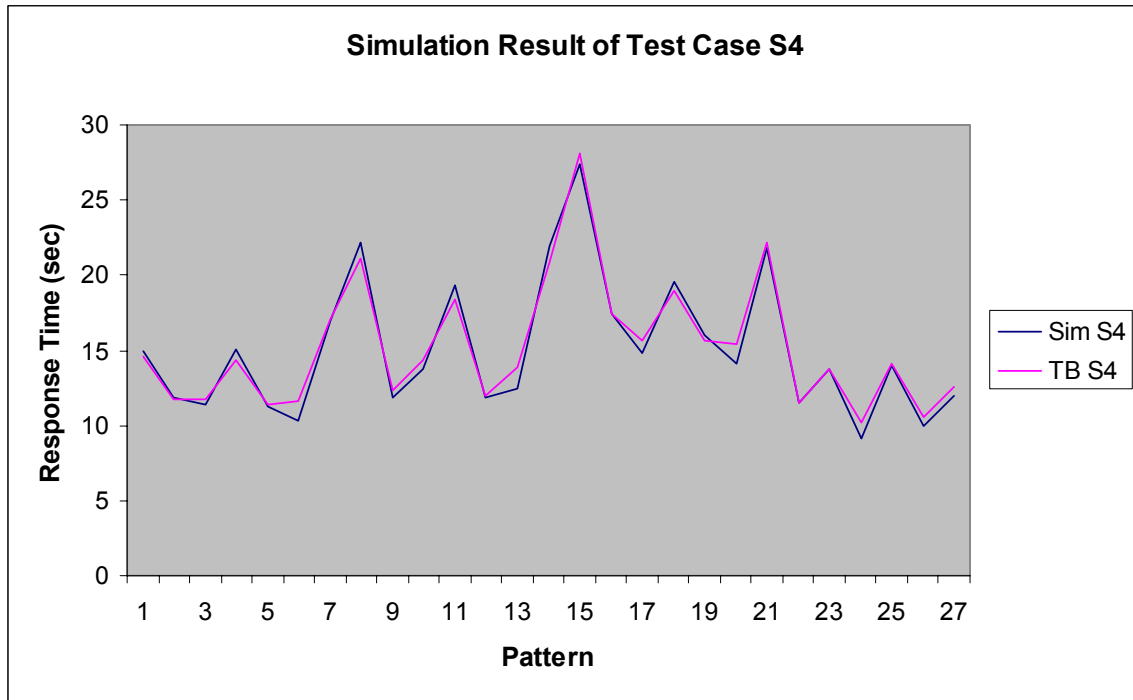


Figure 19. Graphical result of S4

For test case 4, the simulation model chose the same optimal deployment pattern as the test bed.

Roles	Role 2 (4 user) RAM Limit 1.5	
	Sim	Testbed
SIX	C	C
BR733	A	A
GIGA	B	B

Table 23. Optimal Deployment Pattern S4

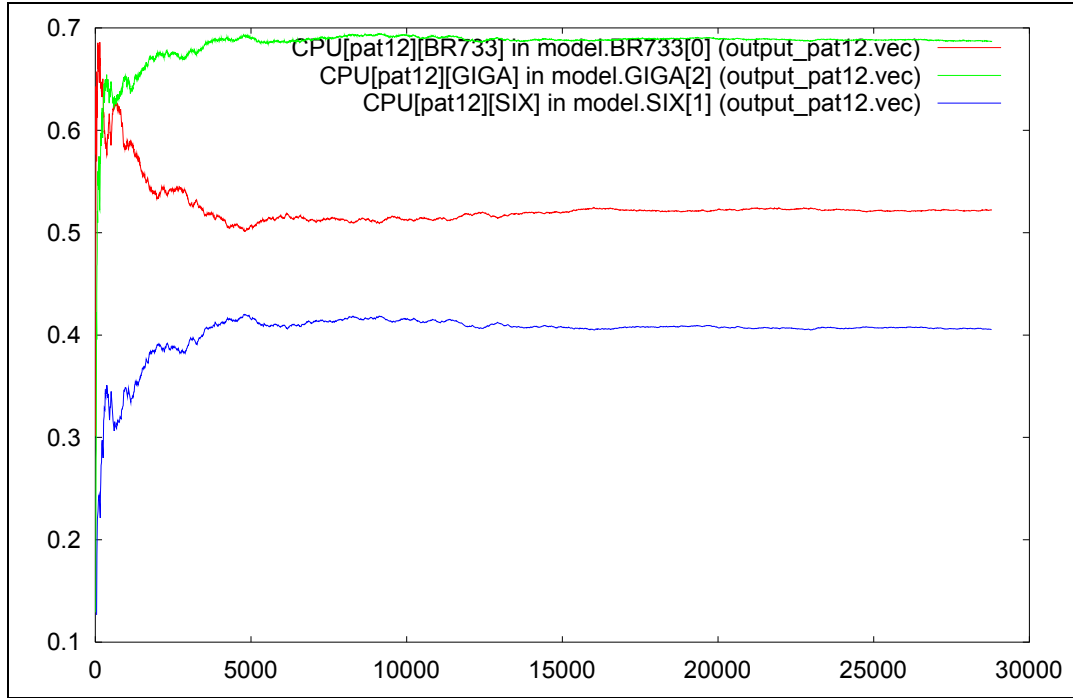


Figure 20. CPU Utilization for S4

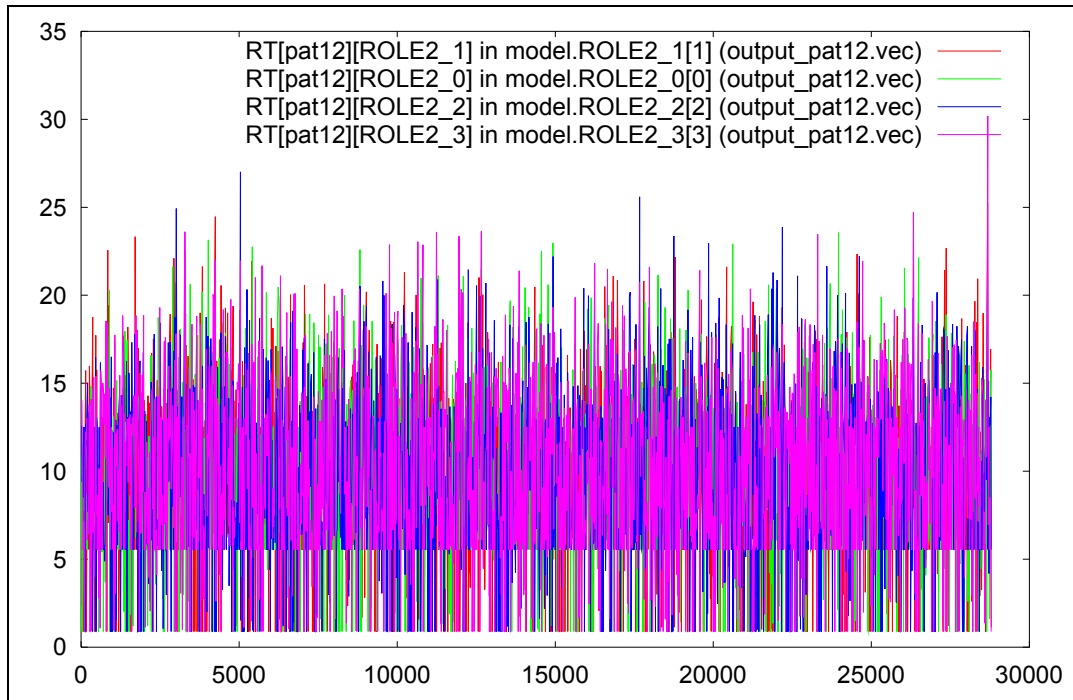


Figure 21. Response Time for S4

The average CPU utilization is GIGA 69%, BR733 52%, SIX 40%. The CPU utilization diagram shows that the CPU utilization of GIGA is high although it has the fastest microprocessor. We can derive that for Role 2 deployment the

object server B is consuming more CPU cycle compared to other object server. Therefore, to support Role 2, the object server B should run on the fastest CPU. The deployment pattern might be able support a small increase in the number roles. However, an increase may result in GIGA being the bottleneck of the deployment server and thus further delaying the response time. The response time range is (30.0, 0.5) sec (max, min).

5. Result for Test Case S5

The graph shows that simulation model modeled the test bed accurately. The highest difference between any two patterns is 2 sec.

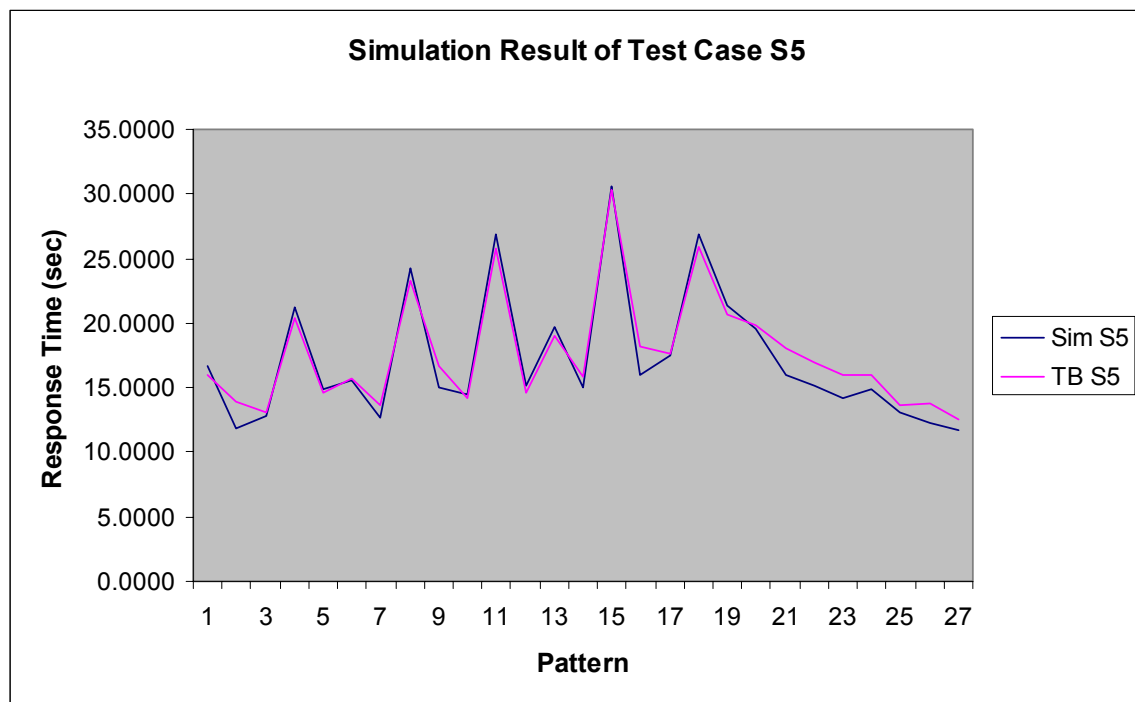


Figure 22. Graphical result of S5

For test case 5, the simulation model chose the same optimal deployment pattern as the test bed.

Roles	Role 3 (3 user) RAM Limit 1.5	
	Sim	Testbed
SIX	A	A
BR733	B	B
GIGA	C	C

Table 24. Optimal Deployment Pattern S5

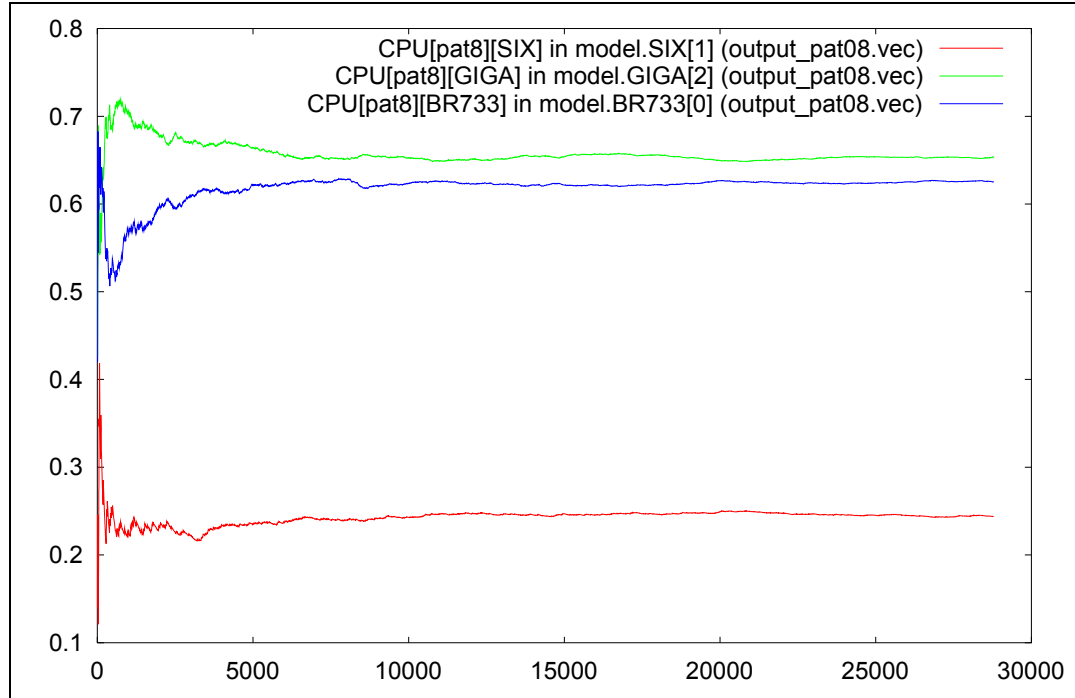


Figure 23. CPU Utilization for S5

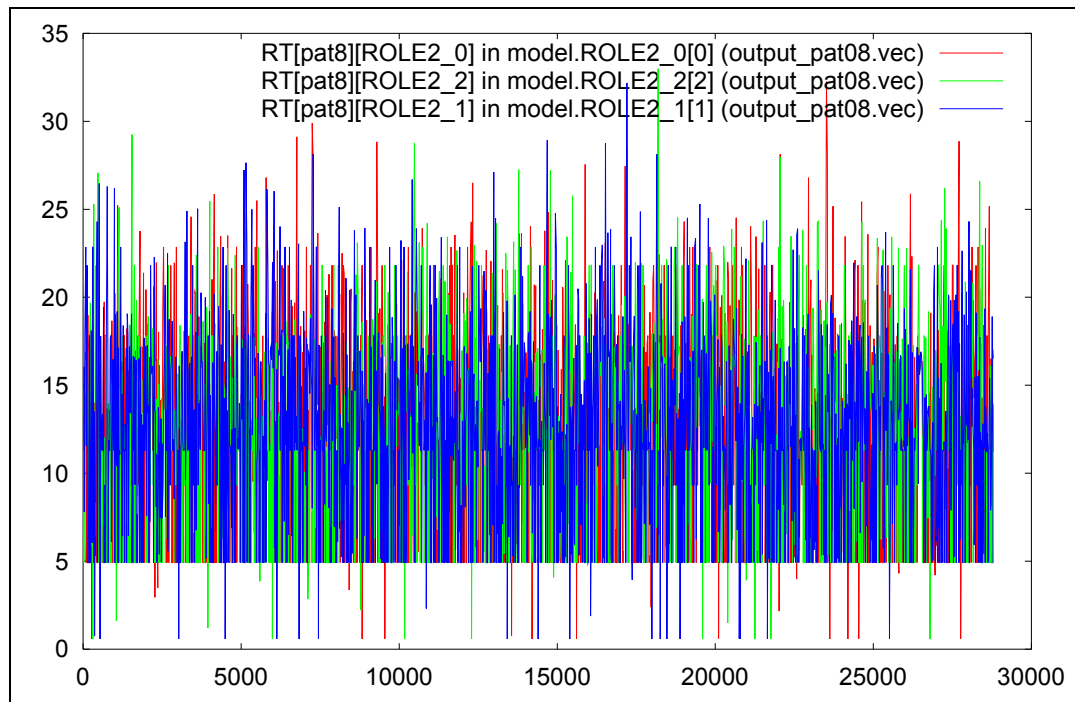


Figure 24. Response Time for S5

The average CPU utilization is GIGA 65%, BR733 62%, SIX 25%. The CPU utilization diagram shows that the CPU utilization of GIGA is high although it has the fastest microprocessor. We can derive that for Role 3 deployment the object server C is consuming more CPU cycle compared to other object server. Therefore, to support Role 3, the object server C should run on the fastest CPU. The deployment pattern might be able support a small increase in the number roles. However, an increase in the number of Role 3 may result in GIGA being the bottleneck of the deployment server and thus further delaying the response time. The response time range is (32.0, 0.5) sec (max, min).

6. Result for Test Case S6

The graph shows that simulation model modeled the test bed accurately. The highest difference between any two patterns is 3 sec. Deployment patterns that fail the RAM limit will be given a response time of 0. Although the total RAM is sufficient to serve the object server, the testbed registered more errors than the simulation model.

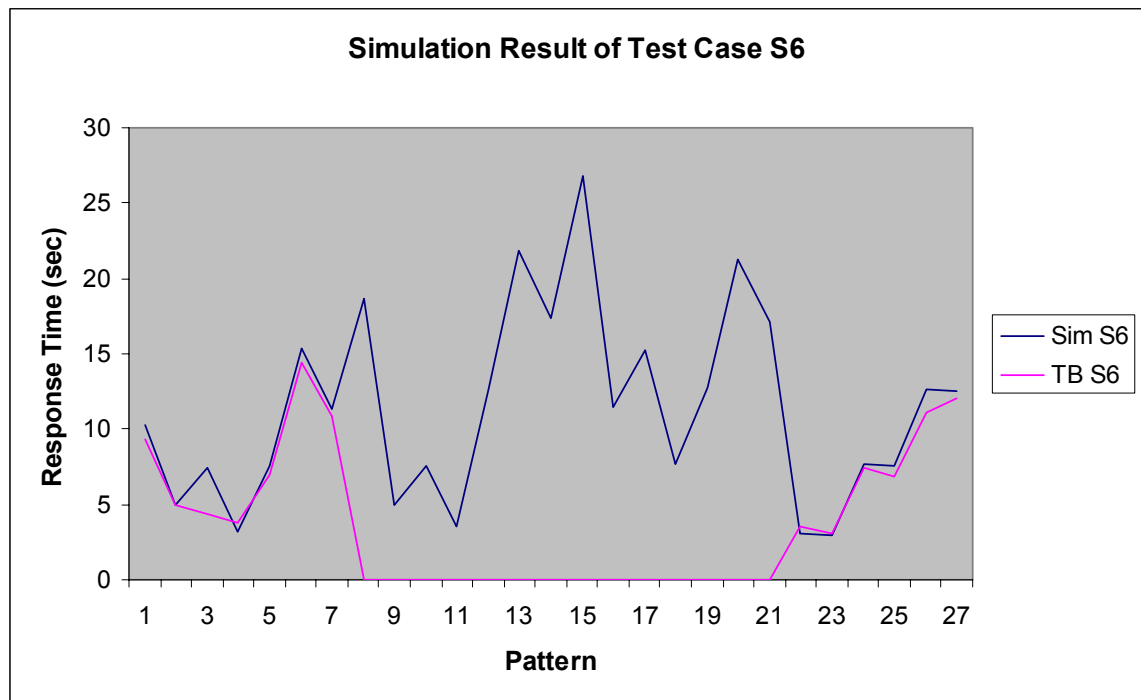


Figure 25. Graphical result of S6

For test case 6, the simulation model chose the same optimal deployment pattern as the test bed.

Roles	Role 1 (28 user) RAM Limit 1.5	
	Sim	Testbed
SIX	C	C
BR733	A	A
GIGA	B	B

Table 25. Optimal Deployment Pattern S6

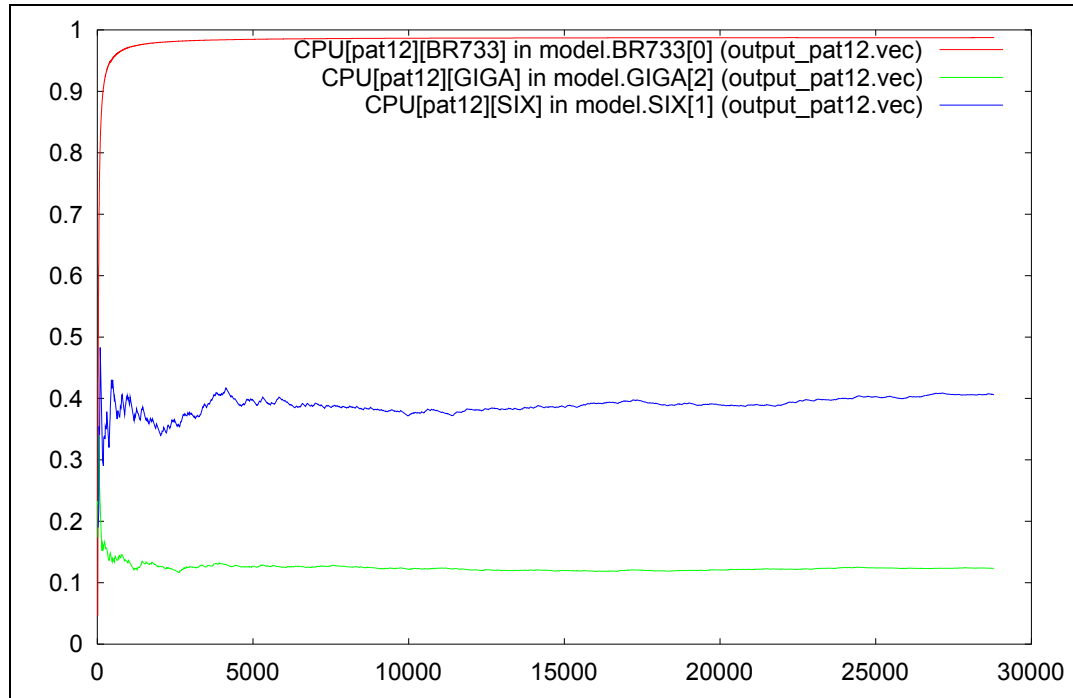


Figure 26. CPU Utilization for S6

The average CPU utilization is GIGA 10%, BR733 99%, SIX 40%. BR733 has reached optimal CPU utilization and therefore an increase in the number of role 1 will result in a significant increase in the response time.

7. Result for Test Case S7

The graph shows that simulation model modeled the test bed accurately. Deployment patterns that fail the RAM limit will be given a response time of 0. The highest difference between any two patterns is 0.1 sec.

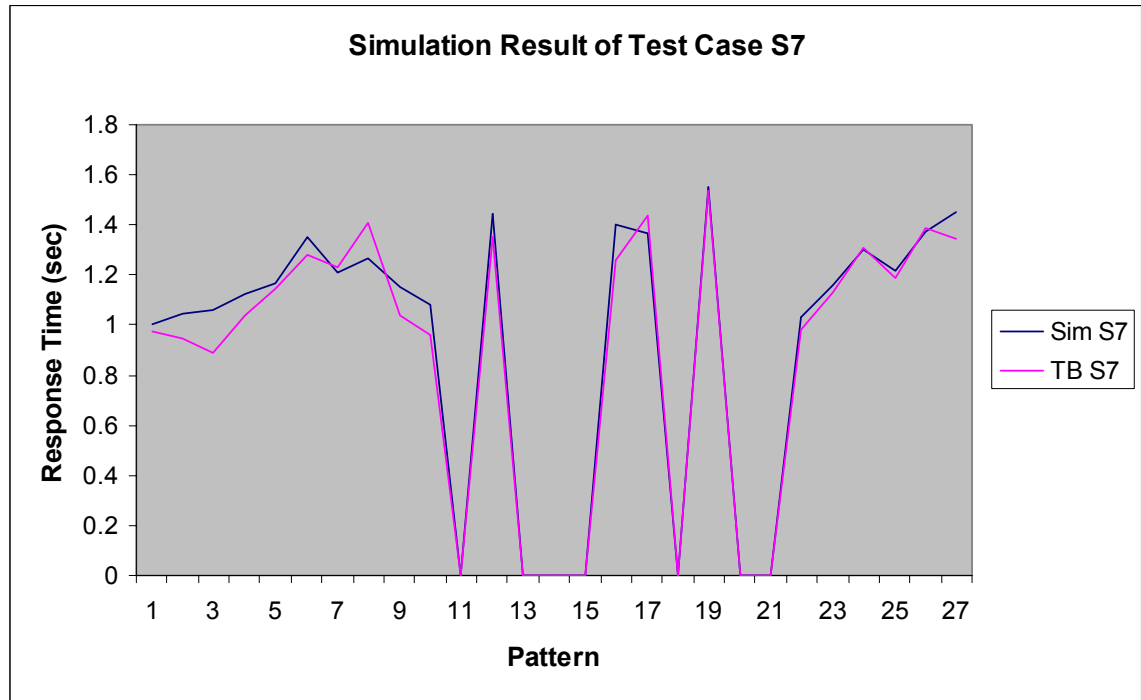


Figure 27. Graphical result of S7

The simulation model chose the 4th optimal deployment pattern as the test bed. The difference in the timing selected is 0.09sec.

Roles	Role 1 (1 user) RAM Limit 1.0	
	Sim	Testbed
SIX	None	None
BR733	None	B
GIGA	A, B, C	A, C

Table 26. Optimal Deployment Pattern S7

The difference in the result could be attributed to the randomness of the test bed and simulation.

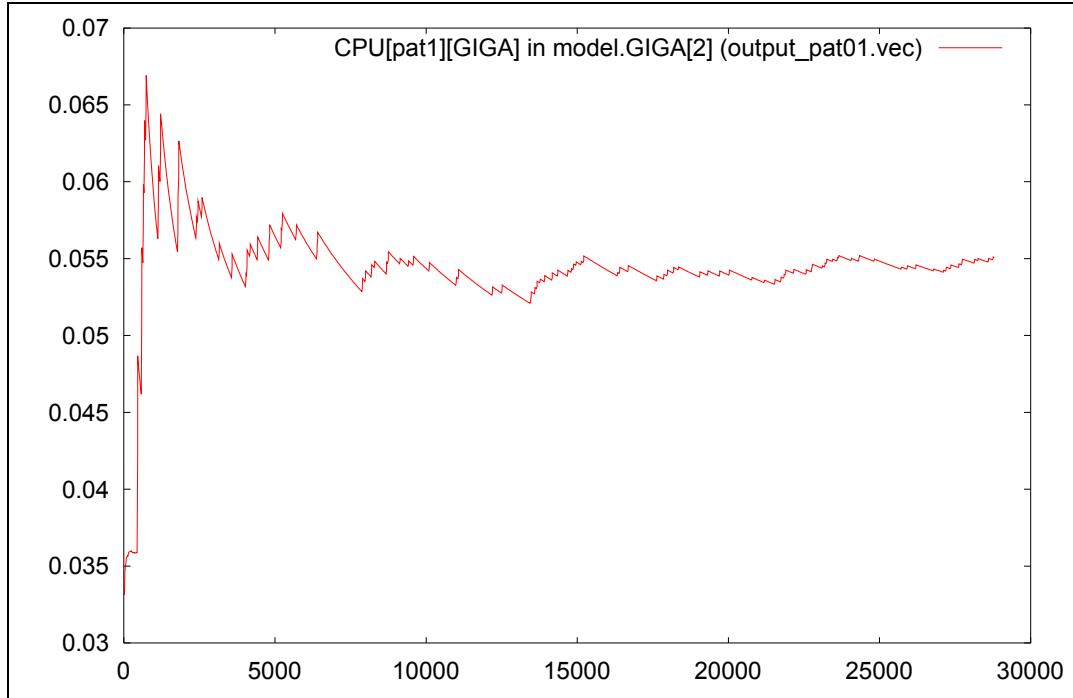


Figure 28. CPU Utilization for S7

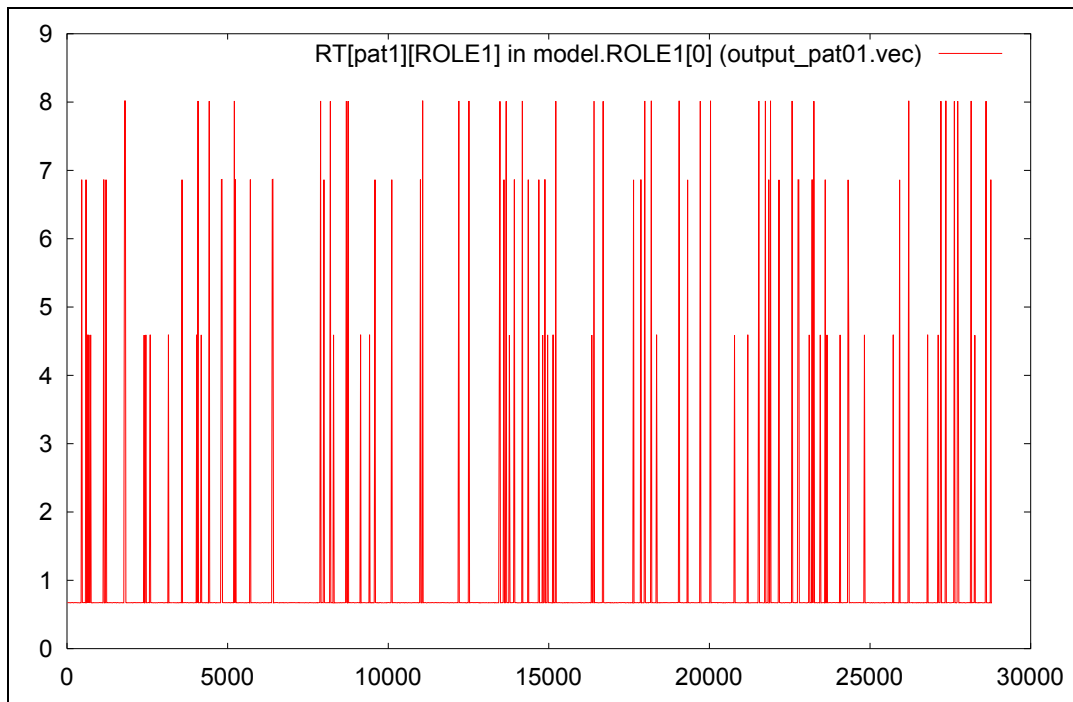


Figure 29. Response Time for S7

The CPU utilization diagram shows that the CPU utilization of GIGA is fairly low at (0.55%) and therefore the deployment pattern is able to support a significant growth in the number of roles. The response time range is (8, 0.8) sec (max, min).

8. Result for Test Case S8

The graph shows that simulation model modeled the test bed accurately. Deployment patterns that fail the RAM limit will be given a response time of 0. The highest difference between any two patterns is 0.17 sec.

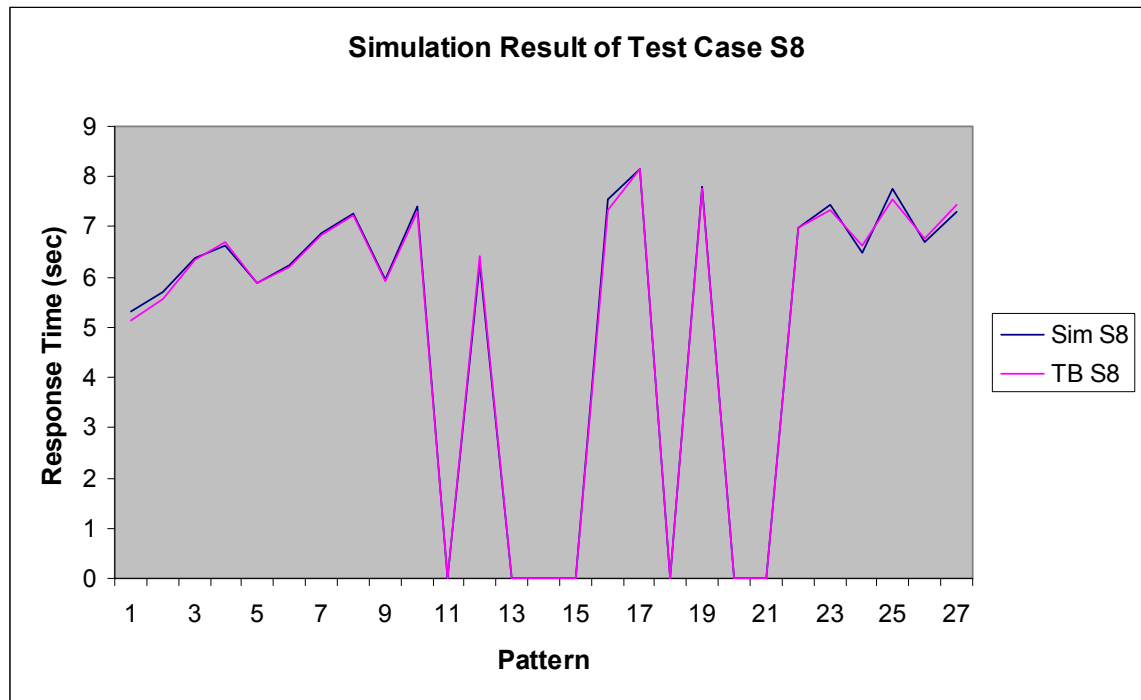


Figure 30. Graphical result of S8

For test case 8, the simulation model chose the same optimal deployment pattern as the test bed.

Roles	Role 2 (1 user) RAM Limit 1.0	
	Sim	Testbed
SIX	None	None
BR733	None	None
GIGA	A, B, C	A, B, C

Table 27. Optimal Deployment Pattern S8

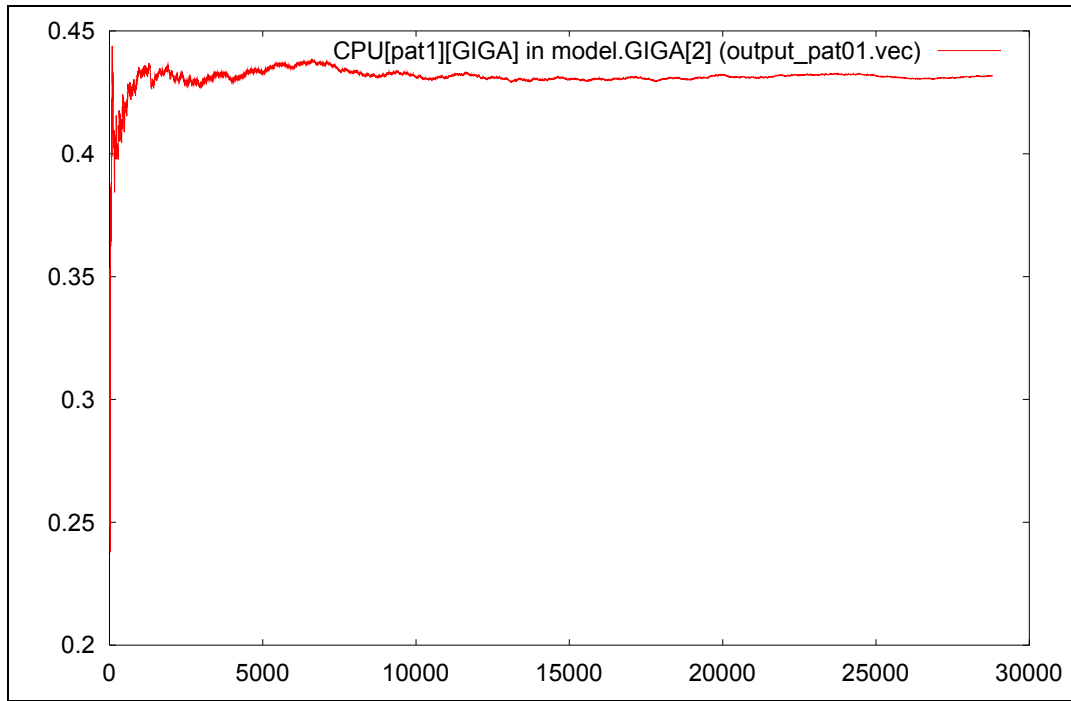


Figure 31. CPU Utilization for S8

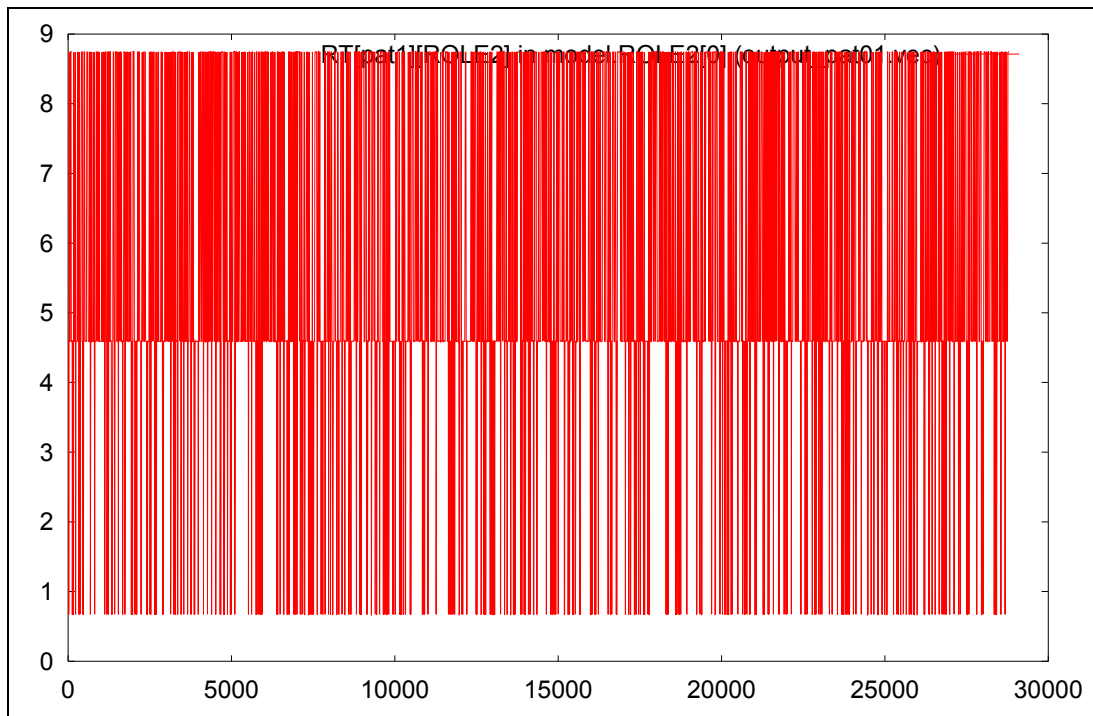


Figure 32. Response Time for S8

The CPU utilization diagram shows that the CPU utilization of GIGA is low at (43%) and therefore the deployment pattern is able to support a higher number of roles. The response time range is (8.8, 0.8) sec (max, min).

9. Result for Test Case S9

The graph shows that simulation model modeled the test bed accurately. The highest difference between any two patterns is 0.3 sec.

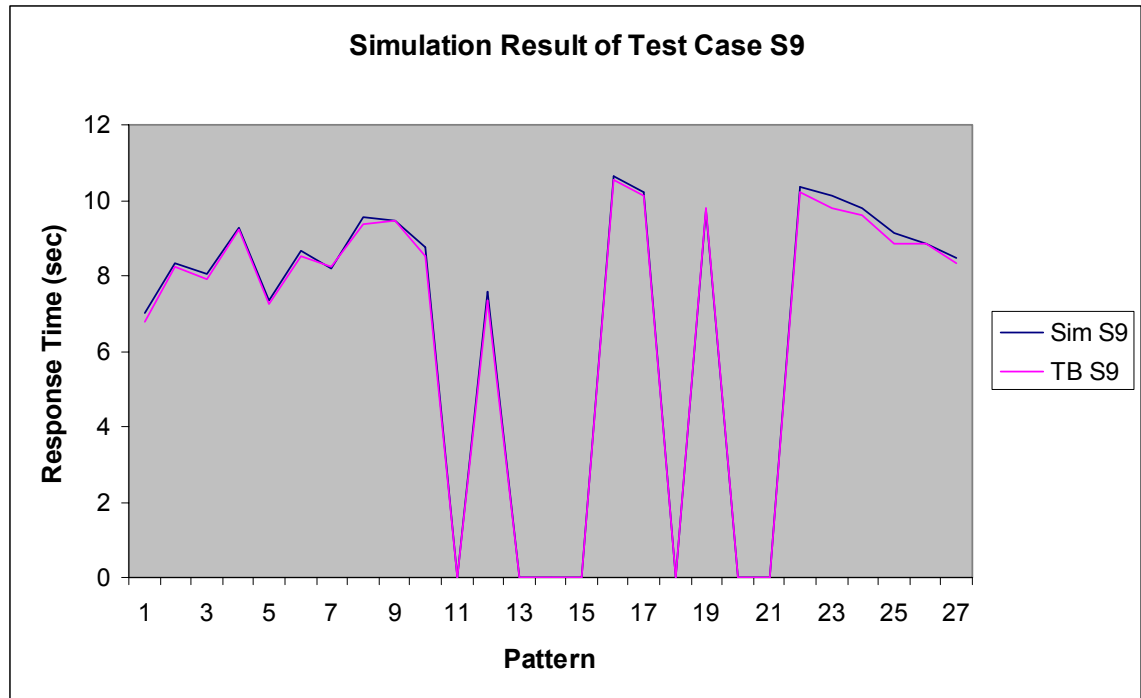


Figure 33. Graphical result of S9

For test case 9, the simulation model chose the same optimal deployment pattern as the test bed.

Roles	Role 3 (1 user) RAM Limit 1.0	
	Sim	Testbed
SIX	None	None
BR733	None	None
GIGA	A, B, C	A, B, C

Figure 34. Optimal Deployment Pattern S9

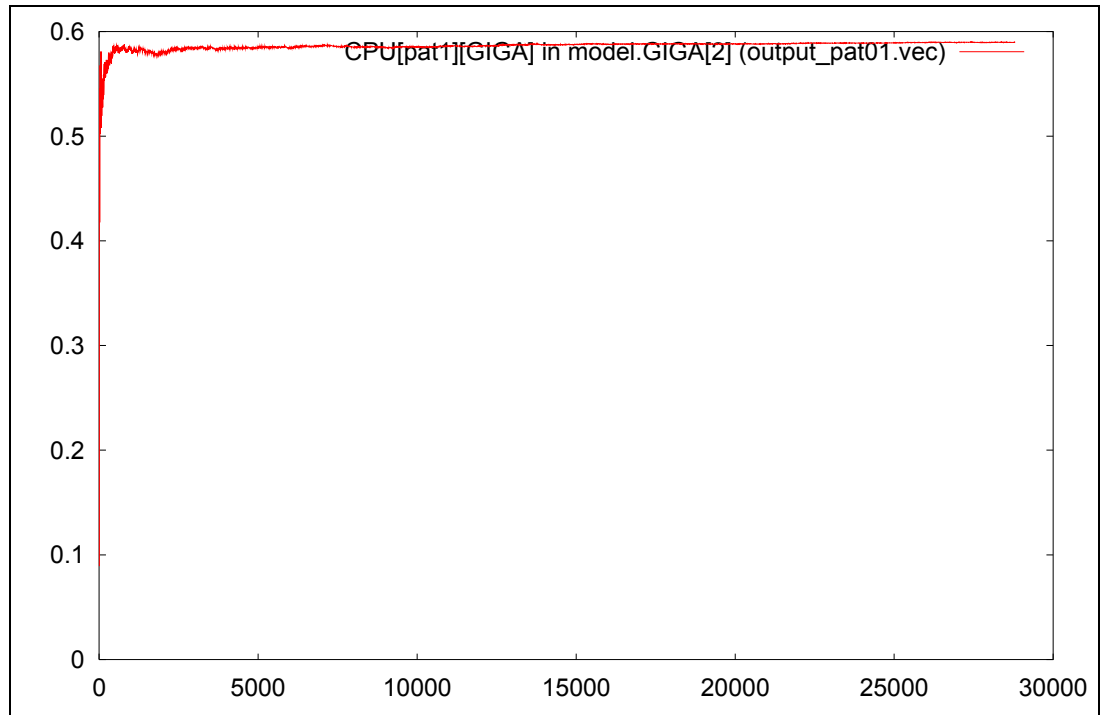


Figure 35. CPU Utilization for S9

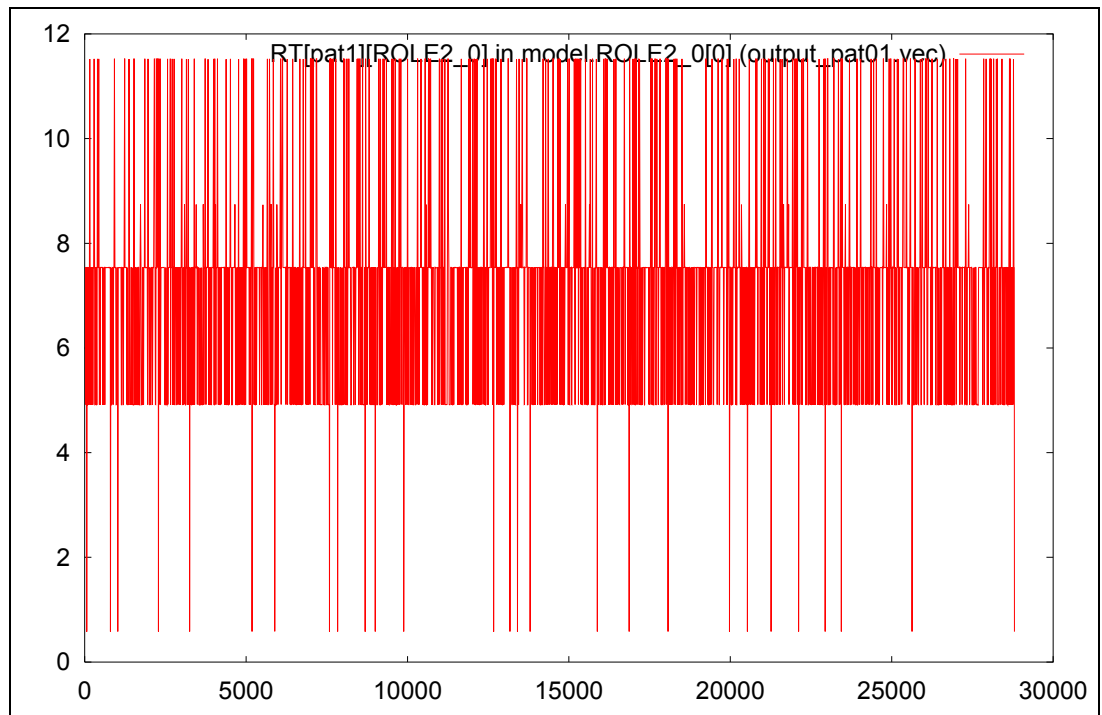


Figure 36. Response Time for S9

The CPU utilization diagram shows that the CPU utilization of GIGA is low at (59%) and therefore the deployment pattern is able to support a small increase in the number roles. The response time range is (11.8, 0.5) sec (max, min).

D. COMPARING SIMULATION RESULT WITH TIME SLICE

The simulation model is executed for the same simulation duration with the time slice scheduling turn on. The time slice execution time is set to 0.3sec, 0.5sec, 1.0sec and 2.0sec.

The result of the simulation run is shown in the following table and graph:-

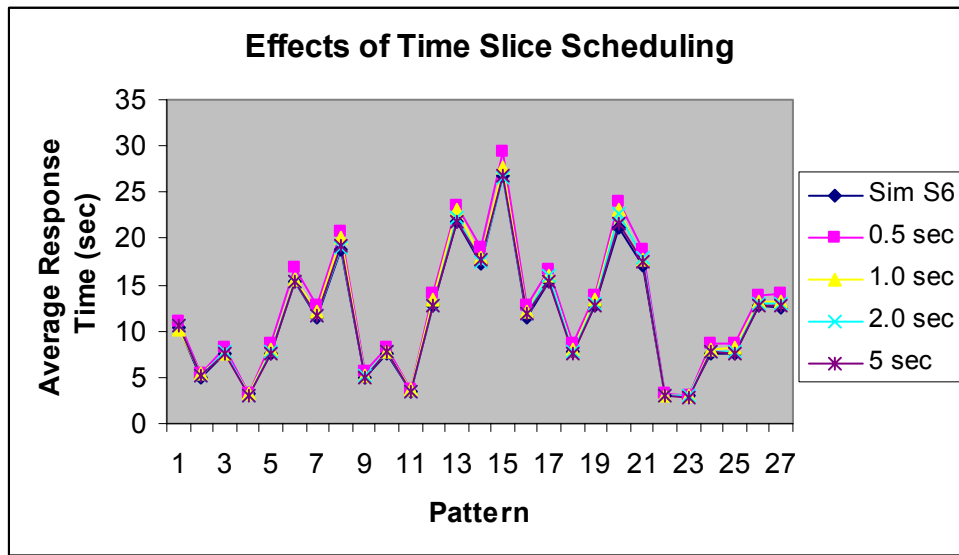


Figure 37. Graphical result of Time Slice

Pat	A	B	C	No Time Slice	Time Slice 0.5s	Time Slice 1.0s	Time Slice 2.0s	Time Slice 5.0s
01	GIGA	GIGA	GIGA	10.2673	10.9358	10.0966	10.6519	10.6164
02	GIGA	GIGA	BR733	5.0118	5.3189	5.3019	5.1548	5.1669
03	GIGA	BR733	GIGA	7.4687	8.1836	7.6092	7.8479	7.4783
04	GIGA	BR733	BR733	3.1608	3.2072	3.2172	3.0915	3.1004
05	BR733	GIGA	GIGA	7.5642	8.7225	8.1007	7.8081	7.6405
06	BR733	GIGA	BR733	15.3563	16.7883	15.5095	15.4238	15.2421
07	BR733	BR733	GIGA	11.3845	12.7411	12.1498	11.6686	11.6405
08	GIGA	GIGA	SIX	18.7025	20.6597	20.0523	18.9089	19.1652
09	GIGA	SIX	GIGA	5.0028	5.6524	5.1628	5.2054	5.0293
10	GIGA	SIX	SIX	7.6163	8.149	7.6878	7.7994	7.7651
11	SIX	GIGA	GIGA	3.5508	3.7054	3.6127	3.5222	3.4656
12	SIX	GIGA	SIX	12.6736	13.9895	13.3366	12.7324	12.8287
13	SIX	SIX	GIGA	21.8104	23.6205	23.0301	22.2968	21.7443
14	SIX	SIX	SIX	17.3469	19.0373	17.9977	17.5824	17.7654

15	BR733	BR733	SIX	26.8308	29.3954	27.6991	26.661	26.788
16	BR733	SIX	BR733	11.4163	12.6777	12.1825	11.8247	11.816
17	BR733	SIX	SIX	15.2354	16.6897	15.9339	15.9211	15.4134
18	SIX	BR733	BR733	7.6992	8.5918	8.0474	7.7426	7.665
19	SIX	BR733	SIX	12.7286	13.9053	13.3378	12.9587	12.8274
20	SIX	SIX	BR733	21.2661	24.0776	23.1732	22.7373	21.6597
21	GIGA	BR733	SIX	17.1591	18.8565	17.7471	17.8394	17.5215
22	SIX	SIX	BR733	3.0755	3.1661	3.0991	3.0019	3.0072
23	BR733	GIGA	SIX	2.9624	2.9561	2.9226	2.9383	2.8604
24	BR733	SIX	GIGA	7.646	8.6442	8.0717	7.7522	7.6896
25	BR733	GIGA	SIX	7.5482	8.6368	8.1036	7.6793	7.6413
26	SIX	GIGA	BR733	12.6422	13.8606	13.228	12.8806	12.6538
27	SIX	BR733	GIGA	12.4794	14.1494	13.1931	12.8856	12.725

Table 28. Test Result with Time Slice

The time slicing scheduling has no effect on the prediction of the optimal deployment. The time slicing interrupt time has effect on the response time. If the time slice interrupt time is much smaller than the execution time of the method call, then the overall response time will increase. This is due to the additional overhead in switching the process. If the time slice interrupt time is close to the execution time, then the overall response time is similar to the result of non-time slicing scheduling.

E. STUDY OF GROWTH POTENTIAL

The simulation model is also useful to study the effect of growth in the number of roles. The simulation model is run with 1, 2, 4 and 8 Role 2 and the following result of the simulation is shown in the following diagram:-

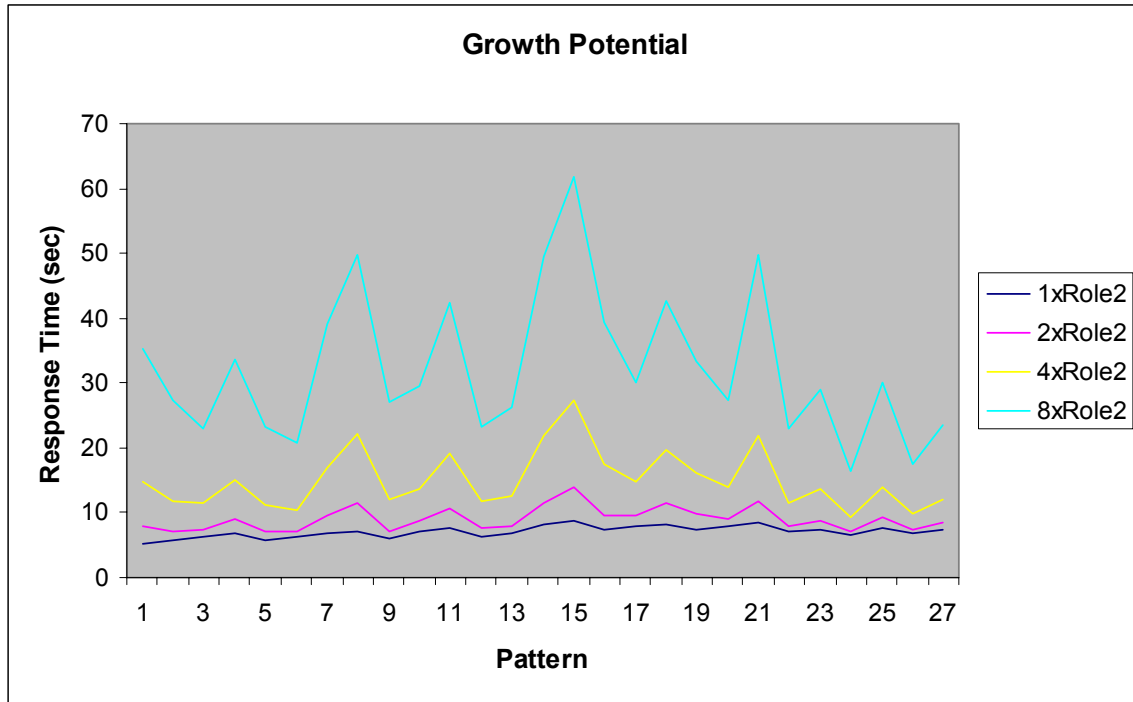


Figure 38. Effect of growth on response time

Roles	Role 2, RAM Limit 0.0			
	1	2	4	8
SIX	None	None	B	B
BR733	None	A, C	A	A
GIGA	A, B, C	B	C	C

Table 29. Effect of growth on deployment pattern

The simulation run reveals that the increase in the number of roles does change the optimal deployment pattern. This implies a change in the number of roles will require the system designer to redetermine the optimal deployment pattern.

F. COMPARING SIMULATION RESULT WITH LINGO MODEL

The result of deployment pattern by the Lingo model is shown in the following tables:-

Roles	Role 1 (1 user) S1		Role 2 (1 user) S2		Role 3 (1 user) S3	
	Sim	Math	Sim	Math	Sim	Math
SIX	None	None	None	None	None	None
BR733	None	None	None	None	None	None
GIGA	A, B, C	A, B, C	A, B, C	A, B, C	A, B, C	A, B, C

Table 30. Result for Test Case S1-S3

Roles	Role 1 (28 user) S4		Role 2 (4 user) S5		Role 3 (3 user) S6	
	Sim	Math	Sim	Math	Sim	Math
SIX	B	None	A	A	A	A
BR733	A	B, C	B	C	B	B
GIGA	C	A	C	B	C	C

Table 31. Result for Test Case S4-S6

Roles	Role 1 (1 user) S7		Role 2 (1 user) S8		Role 3 (1 user) S9	
	Sim	Math	Sim	Math	Sim	Math
SIX	C	None	None	None	None	None
BR733	A	B	None	C	None	A
GIGA	B	A, C	A, B, C	A, B	A, B, C	B, C

Table 32. Result for Test Case S7-S9

5 out of the 9 test case has predicted similar deployment pattern in both methods. For test case which the result differs, the mathematical prediction falls within the top 3 optimal deployment pattern.

G. CONCLUSION

The result of deployment pattern by the Lingo model is shown in the following table. Yes implies a correct matching in the recommended deployment pattern and No implies a mismatch in the recommended deployment pattern.

Test Case	Testbed/Sim	Math/Sim	Testbed/Math
S1	No (4 th)	Yes	No
S2	Yes	Yes	Yes
S3	Yes	Yes	Yes
S4	Yes	No	No
S5	Yes	Yes	No
S6	Yes	Yes	Yes
S7	No (4 th)	No	Yes
S8	Yes	No	No
S9	Yes	No	No

Table 33. Deployment Strategy Result

From the result, the simulation model is able to model the real life test bed accurately. In fact, the accuracy of the simulation model is better than the mathematical model. The simulation model is able to model 7 out of 9 test cases whereas the mathematical model is only able to model the testbed 4 out of 9 test cases.

The simulation model also provides other useful features compare to the mathematical model:-

- The simulation model will be able to rank the deployment pattern according to the response time. The mathematical model will only provide a single deployment pattern without making a reference to the rest of the deployment pattern. The ranking of deployment pattern will allow the system designer to make informed decision if other deployment pattern is chosen beside the optimal deployment pattern.
- The rank of the deployment pattern will also allow the system designer to make weighed consideration to the best deployment pattern. The weighed method in selecting best deployment pattern will be explored in the verification experiment 2.
- The minimum and maximum response time is useful information for system designer to know the upper and lower bound of the expected response time.
- The ability to execute the simulation model without considering RAM limit provide the system designer the ability to determine whether a RAM upgrade will result in better deployment pattern.
- The simulation model also provides the system designer the ability to model the effect of growth in the number of roles and the effect of microprocessor upgrade.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. VERIFICATION EXPERIMENT 2

A. INTRODUCTION

The second verification experiment is a new scenario with 3 machine and 10 object servers. The number of deployment patterns is 59,049. The objective of this verification exercise is to verify the robustness of the simulation model with high number of deployment patterns. The same environment profile is also used on on the mathematical model and the result of the two models is compared.

If a testbed methodology is used to find the optimal deployment for this scenario, the time needed to test all 59049 deployment pattern is $59049 \times 8\text{hrs} = 472392\text{ hrs}$ (i.e. 53 years). In the simulation model, a 8hrs simulation time can be simulated within 20 secs, therefore for 59049 deployment pattern the time needed is $59049 \times 20\text{sec} = 328\text{hrs}$ (i.e. 13 days). The simulation model has reduced the time needed by 1490%.

To further reduce the number of deployment patterns to simulate, the model has bound by setting the ram limit to 1.0. With the bounded ram limit, any deployment pattern that uses more RAM than a machine possessed is rejected. This is a reasonable assumption since the memory swapped time will contribute significantly to the response time and therefore will not result in the optimal response time. For this experiment, the number of deployment pattern is reduced to 1121.

B. SCENARIO PROFILE

The various profiles used in defining the environment is shown in the following table:-

Machine	RAM (MB)	CPU Speed (MHz)
Mozart	256	2000
Handel	512	2400
Beethoven	1000	3000

Table 34. Machine Profile

OBJECT SERVER	METHOD	Number of instruction cycle	Average Size of Message (bits)	RAM Size (MB)
A	1	1579000	216000	139
A	2	1140000	2296000	
A	3	1599000	1736000	
A	4	1394000	528000	
B	1	705000	2752000	122
B	2	1740000	736000	
C	1	243000	1968000	145
C	2	702000	2824000	
C	3	1892000	3728000	
D	1	490000	3752000	153
D	2	1445000	1912000	
E	1	1315000	3640000	231
E	2	1437000	2984000	
E	3	1108000	2616000	
F	1	1286000	2800000	130
F	2	1528000	2064000	
F	3	367000	3328000	
F	4	1750000	3632000	
F	5	1802000	3800000	
G	1	845000	2968000	142
G	2	1437000	104000	
G	3	950000	3848000	
H	1	1395000	2552000	200
H	2	1352000	3384000	
I	1	2215000	1336000	189
J	1	1201000	392000	80
J	2	1606000	2624000	
J	3	557000	856000	
J	4	1101000	248000	

Table 35. Object Server Performance Profile

Primary Method	Secondary Method
D.2	A.2
J.2	B.1
J.3	I.1
E.1	F.5
H.1	G.3

Table 36. Complex Object Server Call Profile

Roles	Call Pattern
Role 0	50 F1.B1 + 1 F1.B2 + 1 F2.B1 + 1 F2.B4
Role 1	10 F1.B1 + 40 F1.B2 + 24 F5.B2
Role 2	50 F4.B1 + 10 F5.B1 + 2 F5.B3 + 1 F3.B1 + 30 F2.B2
Role 3	30 F2.B3

Table 37. Role Call Pattern

Buttons	Method Call
F1.B1	G.2 + C.2 + A.1
F1.B2	F.3
F1.B3	H.2
F2.B1	E.2 + G.2
F2.B2	J.4 + B.2
F2.B3	D.2
F2.B4	B.2 + E.1
F3.B1	J.3
F3.B2	F.2 + B.1 + E.2 + F.3
F4.B1	J.2 + A.2
F5.B1	H.1 + F.3 + J.3
F5.B2	B.1 + E.2
F5.B3	I.1 + J.2
F5.B4	D.1 + I.1
F5.B5	I.1

Table 38. User Interface Call Chart

C. MODEL SETUP

1. Simulation Model Setup

The configuration file in the simulation model is shown in Appendix D.A. A memory leak was encountered with the OMNet++ while executing the 59049 simulation model. The problem was highlighted to the author of OMNet++. To circumvent the problem, the simulation run was divided down into 6 run with 10000 runs each. A batch program was used to automate the running of the 6 runs.

2. Lingo Model Setup

A new Lingo model is created and is shown in Appendix D.B. The Lingo model created supports the computation based on instruction cycle instead of time.

D. WEIGHED MODEL

For an environment with only one role type, the average response time is used to determine the optimal deployment pattern. However, when there is more than one role type, using the average response time of a role may penalize other roles. There are a few methods to determine the optimal deployment pattern:-

1. Ranking

The optimal deployment pattern can be determine by ranking the average response time for each role and comparing the top n deployment pattern and determine the pattern that appears most frequent in all the roles.

2. Highest Priority

The optimal deployment pattern can be determine by selecting the highest priority role and select the lowest average response time. This will ensure that deployment pattern selected will favor the role with the highest priority.

3. Weighted Model

The third method is to assigned weighs to the role response time. Instead of selecting the role highest priority, the weighted model will assign weightage to all roles. The weights can be derived based on the criticality of the machine, and the required response time. For example, the weight of a command and control system (C2) should be higher than a manpower system simply because the time critical nature of C2 system. The weight is used to compute a model value based on the following forumla:-

$$Model = \sum_{i=0}^{n-1} w_i R_i \quad \text{s.t.} \quad \sum_{i=0}^{n-1} w_i = 1$$

where w_i – weigh for object sever i ;

R_i – average response time for object sever i

The deployment pattern with the smallest model value will be selected as optimal. Some examples of weight used are:-

S/No	Weight [Role1, Role2, Role3, Role4]	Remarks
01	[0.25, 0.25, 0.25, 0.25]	The optimal deployment pattern selected will have placed equal weight to each role.
02	[0.0, 0.0, 1.0, 0.0]	The emphasis is placed on Role3. The deployment pattern selected will have the lowest response time for Role3. The result is similar to the highest priority method.
03	[0.1, 0.2, 0.2, 0.5]	An unequal weighted scheme with emphasis placed on Role4.

Table 39. Example of Weighted Model

E. COMPARING SIMULATION RESULT WITH LINGO MODEL

1. Equal Weigh Model

The simulation and Lingo model was run and the equal weighted model is used to determine the optimal deployment pattern for the simulation result. The detail result of the simulation model is shown in Appendix D. The optimal deployment pattern is shown on the following table:-

Model	Deployment Pattern [A, B, C, D, E, F, G, H, I, J]
Simulation	[Bee] [Moz] [Bee] [Bee] [Han] [Han] [Han] [Bee] [Bee] [Moz]
Lingo	[Bee] [Han] [Han] [Bee] [Bee] [Bee] [Moz] [Bee] [Han] [Moz]

Table 40. Deployment Pattern

Both results have spread the object server on all machines. Based on the response time generated by the simulation model, the Lingo model has picked a deployment pattern that is rank 149 positions. Although, the ranking differs significantly, the difference in the response time between the two results is small. The difference in response time in between the two patterns in shown below:

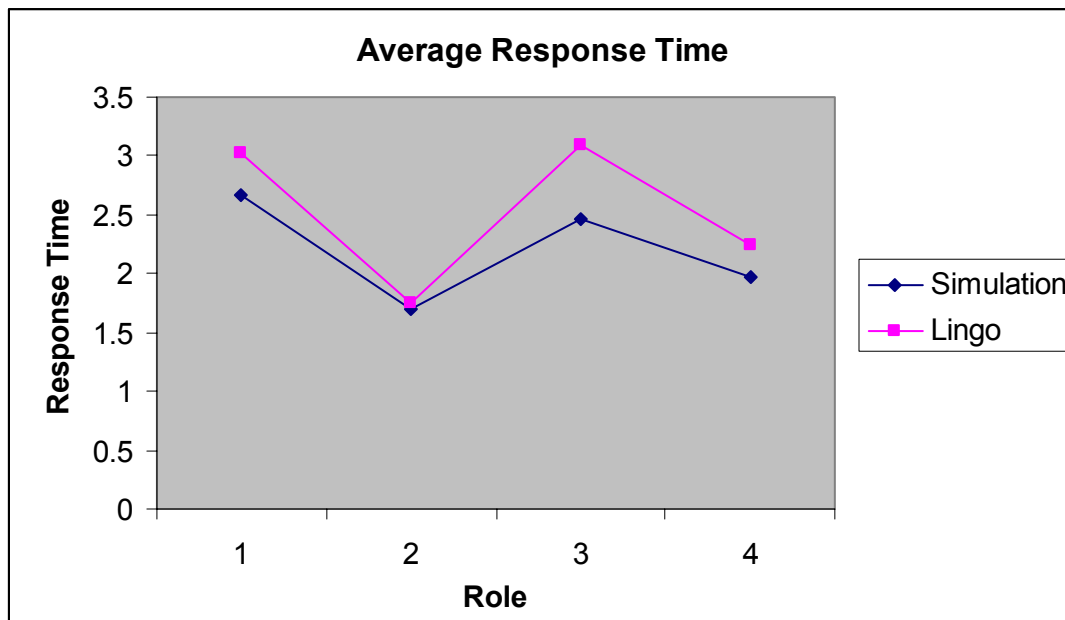


Figure 39. Role Response Time

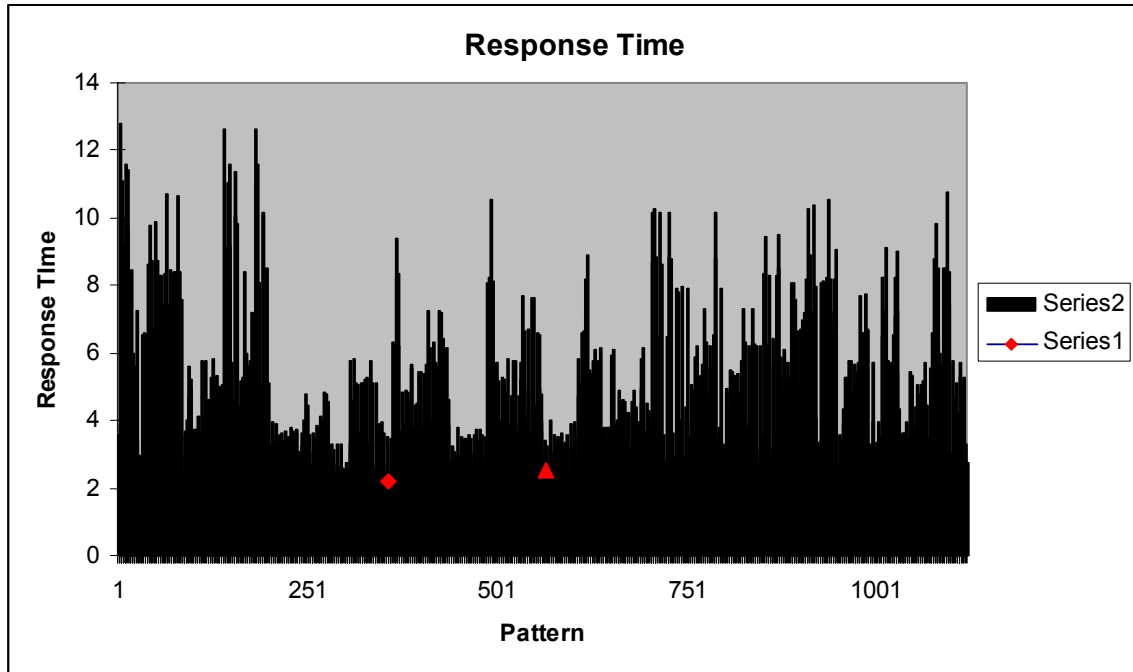


Figure 40. Result wrt to overall response time

(◆,▲) - represents the simulation and Lingo model result respectively.

2. Effect of Weigh Model

Different scenario with different weigh model is used to compute the effect on the deployment pattern. The computation is done using an Excel spreadsheet. The result is shown in the following table:-

Weight [R1, R2, R3, R4]	Deployment Pattern/Response Time [A, B, C, D, E, F, G, H, I, J]
[0.25, 0.25, 0.25, 0.25]	[Bee] [Moz] [Bee] [Bee] [Han] [Han] [Han] [Bee] [Bee] [Moz] [2.6609, 1.7026, 2.4717, 1.9685]
[0.0, 1.0, 0.0, 0.0]	[Han] [Bee] [Bee] [Han] [Bee] [Bee] [Moz] [Han] [Bee] [Moz] [2.9501, 1.2959, 3.1182, 2.5366]
[0.1, 0.2, 0.2, 0.5]	[Bee] [Han] [Bee] [Bee] [Han] [Han] [Moz] [Bee] [Bee] [Moz] [2.9601, 1.4188, 2.9301, 1.8381]
[0.0, 0.2, 0.3, 0.5]	[Bee] [Han] [Moz] [Bee] [Han] [Bee] [Han] [Bee] [Bee] [Moz] [3.0716, 1.9974, 2.3948, 1.8735]

Table 41. Weighted model

The scenario with different weigh model has shown that there is an effect on the deployment pattern in response time of the system. The weighted method is effective to determine the optimal deployment pattern based on difference priority.

F. CONCLUSION

The second verification exercise has shown that the simulation model is robust in a profile with high number of deployment patterns. The Lingo and simulation model has predicted a deployment pattern that has average response time that is close to each other. The verification exercise has also introduced and verified the use of a weighted model to determine the best optimal deployment pattern in an environment with more than one role.

VII. RECOMMENDATION AND FUTURE WORKS

Although the result from two experiments has verified that the simulation model has modeled closely the distributed object server environment, there are multiple ways that the model could be improved.

A. MODELING OF SYSTEM USING UML

The current configuration files are in textual format, and are generated manually by the system designer. This is a tedious task especially with complex environment. It will be more efficient to model the environment profiles as part of the system design process. As UML is becoming the de-facto standard for object oriented design methodology, future work in integrating the simulation model into the UML modeling will be useful. The UML deployment and class diagram can be translated into information needed in the various profiles. Tagged values can be added to the deployment diagram to represents machine profile such as CPU speed, RAM limit, network bandwidth etc. Tagged values can also be added to the class diagram to represents class profile such as complex method calls, computation time and the return message size.

There are existing tools that can convert the UML diagrams to XML schemas or DTD (Document Type Definition). By changing the simulation model to support XML format, integration the simulation model with UML is possible.

B. IMPROVED NETWORK MODELING

The current simulation model only implements a simple network model. Future work can be done to improve the network model by implementing the different layering of network deployment. The model can be improved by modeling network protocol such as ATM, 802.11 and TCP/IP. The model can also be enhanced by allowing system designer to specify the network architecture of the system. For example, the system designer can specify the network domain, router and firewall deployment.

The current network model also assumes a symmetric and error free network between the role and the machine. However, the real-world network configuration could be asymmetric in nature. For example, DSL and broadband network has asymmetric communication link. Machine to machine communication within an enclave will potentially have a higher bandwidth than communication with host outside an enclave. Building support for asymmetric communication will improve the accuracy of the model.

Network error rate will have significant impact in the overall performance of a distributed system. A network segment with a high error rate will result in more loss packet, higher retransmission and thus longer response time. Therefore, the optimal deployment strategy may avoid deploying object server on machine that rides on a network segment with higher network error rate. To model the effect of network error, the following work must be done:-

- Model the error handling mechanism by the network protocol e.g. TCP/IP error handling protocol.
- For every message send or received, model the probability of error using the network error rate.
- For message with error, retransmit or request for the message again.

C. REFINING THE RESULT TO ROLE/METHOD CALL LEVEL

The simulation model determines the optimal deployment pattern based on average response time. However, each method call may not have equal priority. A method call for real-time data update has a higher priority than for static data update. Real-time data update will have a more stringent demand for the response time. For roles with many low priority calls, using the average response time to select a deployment pattern may not be realistic. Ignoring low priority call altogether will also present an unrealistic load on the machine CPU. Therefore, there is a requirement to refine the simulation model to model timing

priority for method call and determine the optimal deployment based on high priority method call.

D. ENTERPRISE RESOURCE PLANNING (ERP)

Enterprise resource planning (ERP) capabilities can be added to the simulation model to enable the system designer to study the effect and expenses of adding memory, upgrading the CPU or adding additional machine. ERP also allows the system designer to study the effect of reducing the number of machines due to machine failure. The simulation model could advise system designer the most cost effective upgrade that would give the most performance improvement for the least amount of money.

E. CONSTRAINING THE DEPLOYMENT PATTERNS

Further works to explore other constraints to the deployment patterns. The aim of introducing more constraint is to reduce the time needed to execute the simulation. Besides limiting the deployment patterns based on RAM limit, the following constraints can be further explored:-

- Constraint based on the CPU limit. Deployment patterns with CPU resource utilization exceeding the preset CPU limit will be rejected.
- Constraint based on probable inferior deployment. For example, deployment pattern which does not involve the most powerful CPU will likely not be the optimal provable deployment.
- Constraint based on network bandwidth. For example, with two identical CPUs, the CPU with the higher network bandwidth with the role should be favored.
- Constraint based on user defined deployment pattern preference. Specific machines are used to run legacy applications due to system and interface constraints. For example, applications written for

Windows 95 environment will not be portable to a machine running WindowsXP.

VII. CONCLUSION

The response time predicted by the simulation model is quite close to the scenario tested. The simulation model is also able to consistently select the top few deployment patterns. Verification experiment one has shown that the simulation model prediction matches the test bed better than the mathematical model.

There are other strengths in the simulation model. First, the simulation model provides a good overview of the response time of the system. The simulation model provides information such as maximum, minimum and average response time of a role. This information is a good benchmark for system designer to set a reasonable expectation of the response time. The model also provides the response time of other deployment pattern other than the optimal deployment, thus allowing the system designer to study the tradeoff of selecting other deployment pattern. For example, if the second optimal deployment meets the operational requirement, it may be favor over the optimal pattern because of deployment preference.

The simulation model is also a valuable tool for the system designer to determine the effect of growth in the number of roles. Since the simulation run is easily repeatable, the system engineer can generate the response time based on many scenario of increasing role count. The maximum number of roles is derived when the response time is greater than the operational requirement. In a deployment environment which the number of roles is constantly changing, the optimal deployment pattern of the highest number of roles should be used to avoid the need to constantly change the deployment pattern.

System designer can use the simulation model to determine the effect of system upgrade i.e. upgrading CPU speed or RAM. For example, the result of the simulation run with no RAM limit can be used to study the effect of upgrading the RAM of the system.

Lastly the simulation model can also be used for “what-if” scenario in the event of system failure. For example, the simulation model allows the system designer to determine what the next best deployment pattern is if one of the CPU fails.

The limitation of the simulation model compared to the mathematical model is the time needed to run the simulation. For verification exercise 2, the simulation run took about 1 day to complete but the mathematical model took about 30 secs to complete. The additional time needed is justifiable with the functionalities mentioned above.

APPENDIX A - SOURCE CODE OF THE SIMULATION MODEL

A. ADDRESS.CPP

```
// Address.cpp: implementation of the CAddress class.
//
////////////////////////////////////////////////////////////////
#include "Address.h"
#include <string.h>

////////////////////////////////////////////////////////////////
// Construction/Destruction
////////////////////////////////////////////////////////////////

CAddress::CAddress()
{
    in_gate = -1;
    out_gate = -1;
}

CAddress::~CAddress()
{
}

void CAddress::SetAddress(int ingate, int outgate, char *svr, char *objapp)
{
    strcpy(svr_name, svr);
    strcpy(obj_name, objapp);
    in_gate = ingate;
    out_gate = outgate;
}

int CAddress::GetInGateId()
{
    return in_gate;
}

int CAddress::GetOutGateId()
{
    return out_gate;
}

char *CAddress::GetName()
{
    return obj_name;
}
```

B. BTNLIST.CPP

```
////////////////////////////////////////////////////////////////
// file: btnlist.cpp
```

```
////////////////////////////////////
```

```
#include <stdio.h>
#include <string.h>
#include "omnetpp.h"
#include "global.h"
#include "simevent.h"
#include "btnlist.h"
```

```
static CBtnList *m_ButtonList = NULL;
CBtnList *CBtnList::GetInstance()
{
    if (m_ButtonList==NULL)
    {
        m_ButtonList = new CBtnList();
        m_ButtonList->initialize();
    }
    return m_ButtonList;
}
```

```
CBtnList::CBtnList()
{
}
```

```
CBtnList::~CBtnList()
{
}
```

```
bool CBtnList::initialize()
{
    buttonCount = 0;
    loadCfg();                //Loading configuration file
    return true;
}
```

```
bool CBtnList::getButton(char *name, sBUTTON &button)
{
    char tmpstr[MAX_NAME_SZ];
    bool found = false;
    int i = 0;

    while ((!found) && (i<buttonCount))
    {
        strcpy(tmpstr, m_Button[i].cName);
        if (strcmp(name, tmpstr) == 0)
        {
            button = m_Button[i];
            found = true;
        }
        i = i+1;
    }

    if (found)
        return true;
    else
```

```

        return false;
    }

//Method to load the configuration file
bool CBtnList::loadCfg()
{
    char seps[] = "\n";
    char *token;

    FILE *stream;
    char line[100];
    int pos, len;
    bool endDef;

    sBUTTON tmpButton;

    /* Open for read (will fail if file "data" does not exist) */
    if( (stream = fopen( "button.def", "r" )) == NULL )
    {
        ev << "[CBtnList] >> The file 'data' was not opened\n";
        return false;
    }
    else
    {
        ev << "[CBtnList] >> The file 'data' was opened\n";

        while ( fgets( line, 100, stream ) != NULL )
        {
            //checking the content of the string
#ifdef DEBUG_FLAG_L1
            ev << "[CBtnList] >> Content " << line << '\n';
#endif

            endDef = false;
            if (strstr( line, "[NEW_DEF]") != NULL)
            {
                //New definition of button, next line is name
                if ( fgets( line, 100, stream ) != NULL )
                {
                    strcpy(tmpButton.cName, line);
                    len = strlen(tmpButton.cName);
                    tmpButton.cName[len-1] = '\0';
                    tmpButton.iNumCall = 0;
                }

                while (!endDef)
                {
                    if (fgets( line, 100, stream ) == NULL)
                        return false;

                    if (strstr( line, "[END_DEF]") != NULL)
                    {
                        m_Button[buttonCount] = tmpButton;
                        buttonCount = buttonCount + 1;
                        endDef = true;
                    }
                }
            }
        }
    }
}

```

```

else
{
    /* Establish string and get the first token: */
    pos = 0;
    token = strtok( line, seps );

    while( token != NULL )
    {
        switch (pos)
        {
            case 0: //Getting the object server name

#ifdef DEBUG_FLAG_L1
            ev << "[CBtnList] >> ObjSvr "<< token << '\n';
#endif

            strcpy(tmpButton.svcCall[tmpButton.iNumCall].cObjsvr, token);
            pos++;
            break;

            case 1: //Getting the method name

#ifdef DEBUG_FLAG_L1
            ev << "[CBtnList] >> cMethod "<< token << '\n';
#endif

            strcpy(tmpButton.svcCall[tmpButton.iNumCall].cMethod, token);
            pos++;
            break;
        }
        token = strtok( NULL, seps );
    }
    tmpButton.iNumCall = tmpButton.iNumCall + 1;
}
}
}

/* Close stream */
if( fclose( stream ) )
    ev << "[CBtnList] >> The file 'data' was not closed\n";

return true;
}

return false;
}

```

C. DNSSVC.CPP

```

// DnsSvc.cpp: implementation of the CDnsSvc class.
//
////////////////////////////////////

#include <string.h>
#include "DnsSvc.h"

```

```

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////

CDnsSvc::CDnsSvc()
{
    counter = 0;
}

CDnsSvc::~CDnsSvc()
{
}

bool CDnsSvc::MapSvcToPort(int ingate, int outgate, char *svr, char *name)
{
    char tmpstr[MAX_NAME_SZ];

    for (int i=0; i<counter; i++)
    {
        strcpy(tmpstr, m_AddMap[i].GetName());
        if (strcmp(name, tmpstr) == 0)
        {
            //name found, there might be a shift in the ingate and outgate
            m_AddMap[i].SetAddress(ingate, outgate, svr, name);
            return true;
        }
    }

    m_AddMap[counter].SetAddress(ingate, outgate, svr, name);
    counter++;

    return true;
}

bool CDnsSvc::FindName(char *name)
{
    char tmpstr[MAX_NAME_SZ];

    for (int i=0; i<counter; i++)
    {
        strcpy(tmpstr, m_AddMap[i].GetName());
        if (strcmp(name, tmpstr) == 0)
            return true;
    }

    return false;
}

int CDnsSvc::FindInGateByName(char *objsvr)
{
    char tmpstr[MAX_NAME_SZ];

    for (int i=0; i<counter; i++)
    {
        strcpy(tmpstr, m_AddMap[i].GetName());

```

```

        if (strcmp(objsvr, tmpstr) == 0)
            return m_AddMap[i].GetInGateId();
    }

    return -1;
}

int CDnsSvc::FindOutGateByName(char *objsvr)
{
    char tmpstr[MAX_NAME_SZ];

    for (int i=0; i<counter; i++)
    {
        strcpy(tmpstr, m_AddMap[i].GetName());
        if (strcmp(objsvr, tmpstr) == 0)
            return m_AddMap[i].GetOutGateId();
    }

    return -1;
}

```

D. INTERACTLIST.CPP

```

// InteractList.cpp: implementation of the CInteractList class.
//
//

```

```

#include <stdio.h>
#include <string.h>
#include "limits.h"

```

```

#include "omnetpp.h"
#include "InteractList.h"

```

```

//
// Construction/Destruction
//
static CInteractList *m_Interact = NULL;
CInteractList *CInteractList::GetInstance()
{
    if (m_Interact==NULL)
    {
        m_Interact = new CInteractList();
        m_Interact->initialize();
    }
    return m_Interact;
}

```

```

CInteractList::CInteractList()
{
}

```

```

CInteractList::~CInteractList()

```

```

{
}

//To initialize the InteractList.
bool CInteractList::initialize()
{
    m_InteractCount = 0;
    loadCfg();                //Loading configuration file
    return true;
}

//To get an interact from the object server name provided.
bool CInteractList::getInteract(char *objsvr, char *method, sInteract &interact)
{
    int index = 0;
    bool found = false;

    while ((index < m_InteractCount) && (!found))
    {
        if ((strcmp(m_InteractList[index].initCall.cObjsvr, objsvr)==0) &&
            (strcmp(m_InteractList[index].initCall.cMethod, method)==0))
        {
            found = true;
            interact = m_InteractList[index];
            return true;
        }
        index = index + 1;
    }

    return false;
}

//Method to load the configuration file
bool CInteractList::loadCfg()
{
    char seps[] = "\n";
    char *token;

    FILE *file;
    char line[100];
    int pos = 0;
    bool endDef = false;

    sInteract tmpInteract;
    int tmpInt = 0;
    char tmpStr[100];
    char buf[1024 + 1];

    if( (file = fopen( "interact.def", "r+t" )) == NULL )
    {
        printf( "The file 'interact.def' was not opened\n" );
        return false;
    }

#ifdef DEBUG_FLAG_L2

```

```

        ev << "[CInteractList] >> The file 'data' was opened\n";
#endif

    tmpInteract.iNumInteract = 0;
    while (fgets(buf, 1024, file))
    {
        if (strstr (buf, "[NEW_INTERACT]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[NEW_INTERACT]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';

            token = strtok( tmpStr, seps );
            while( token != NULL )
            {
                switch (pos)
                {
                    case 0: //Getting the object server name of the method
#ifdef DEBUG_FLAG_L1
                        ev << "[CInteractList] >> ObjSvr "<< token << '\n';
#endif
                        strcpy(tmpInteract.initCall.cObjsvr, token);
                        pos++;
                        break;
                    case 1: //Getting the method name
#ifdef DEBUG_FLAG_L1
                        ev << "[CInteractList] >> cMethod "<< token << '\n';
#endif
                        strcpy(tmpInteract.initCall.cMethod, token);
                        pos++;
                        break;
                }
                token = strtok( NULL, seps );
            }

            tmpInt = 0;
            while (!endDef)
            {
                if (fgets(line, 100, file) == NULL)
                    return false;

                if (strstr( line, "[END_INTERACT]") != NULL)
                {
                    tmpInteract.iNumInteract = tmpInt;
                    m_InteractList[m_InteractCount] = tmpInteract;
                    m_InteractCount = m_InteractCount + 1;
                    endDef = true;
                }
                else if (strstr( line, "[CALL]") != NULL)
                {
                    strcpy(tmpStr, line+strlen("[CALL]")+1);
                    tmpStr[strlen(tmpStr)-1] = '\0';

                    /* Establish string and get the first token: */
                    pos = 0;
                    token = strtok( tmpStr, seps );

```

```

                                while( token != NULL )
                                {
                                    switch (pos)
                                    {
                                        case 0: //Getting the object server name
#ifdef DEBUG_FLAG_L1
ev << "[CInteractList] >> ObjSvr "<< token << '\n';
#endifif

                                strcpy(tmpInt.interactCall[tmpInt].cObjsvr, token);
                                    pos++;
                                    break;
                                        case 1: //Getting the method name
#ifdef DEBUG_FLAG_L1
ev << "[CInteractList] >> cMethod "<< token << '\n';
#endifif

                                strcpy(tmpInt.interactCall[tmpInt].cMethod, token);
                                    pos++;
                                    break;
                                    }
                                token = strtok( NULL, seps );
                                }
                                tmpInt = tmpInt + 1;
                            }
                        }
                    }

/* Close stream */
if( fclose( file ) )
    ev << "[CInteractList] >> The file 'data' was not closed\n";

    return true;
}

return false;
}

```

E. METHOD.CPP

```

// method.cpp: implementation of the Cmethod class.
//
////////////////////////////////////

#include <string.h>
#include "method.h"

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////

CMethod::CMethod()
{

```

```

}

CMethod::~CMethod()
{

}

void CMethod::setMethod(char *str, long cpu, long msg)
{
    strcpy(name, str);
    cpuUtil = cpu;
    msgSz = msg;
}

```

F. OBJ.CPP

```

// obj.cpp: implementation of the obj class.
//
////////////////////////////////////////////////////////////////
#include <string.h>
#include "obj.h"

////////////////////////////////////////////////////////////////
// Construction/Destruction
////////////////////////////////////////////////////////////////

CObj::CObj()
{

}

CObj::~CObj()
{

}

//To retrieve the CPU utilization requirement given the object server name
long CObj::getCPU(char *name)
{
    char tmpstr[MAX_NAME_SZ];

    for (int i=0; i<num_method; i++)
    {
        strcpy(tmpstr, method[i].name);

        //if both string is the same
        if (strcmp(name, tmpstr) == 0)
            return method[i].cpuUtil;
    }

    return -1;
}

```

```

//To retrieve the message size
long CObj::getMsgSize(char *name)
{
    char tmpstr[MAX_NAME_SZ];

    for (int i=0; i<num_method; i++)
    {
        strcpy(tmpstr, method[i].name);

        //if both string is the same
        if (strcmp(name, tmpstr) == 0)
            return method[i].msgSz;
    }

    return -1;
}

```

G. OBJLIST.CPP

```

// ObjList.cpp: implementation of the CObjList class.
//
////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <string.h>
#include "limits.h"

#include "omnetpp.h"
#include "ObjList.h"

////////////////////////////////////////////////////////////////
// Construction/Destruction
////////////////////////////////////////////////////////////////
static CObjList *m_ObjList = NULL;
CObjList *CObjList::GetInstance()
{
    if (m_ObjList==NULL)
    {
        m_ObjList = new CObjList();
        m_ObjList->initialize();
    }
    return m_ObjList;
}

CObjList::CObjList()
{
}

CObjList::~CObjList()
{
}

```

```

//To initialize the ObjList.
bool CObjList::initialize()
{
    m_ObjCount = 0;
    loadCfg();           //Loading configuration file
    return true;
}

//To get an object using the object name
bool CObjList::getObj(char *name, CObj &obj)
{
    char tmpstr[MAX_NAME_SZ];
    bool found = false;
    int i = 0;

    while ((i < m_ObjCount) && (!found))
    {
        strcpy(tmpstr, m_Obj[i].name);
        if (strcmp(name, tmpstr) == 0)
        {
            obj = m_Obj[i];
            found = true;
        }
        i = i + 1;
    }

    if (found)
        return true;
    else
        return false;

    return true;
}

//Method to load the configuration file
bool CObjList::loadCfg()
{
    char seps[] = "\n";
    char *token;

    FILE *stream;
    char line[100];
    int pos;
    bool endDef;

    CObj tmpObj;
    long tmpLong1, tmpLong2;
    char tmpName[MAX_NAME_SZ];

    /* Open for read (will fail if file "data" does not exist) */
    if( (stream = fopen( "obj.def", "r" )) == NULL )
    {
        ev << "[CObjList] >> The file 'data' was not opened\n";
        return false;
    }
    else

```

```

{
    ev << "[CObjList] >> The file 'data' was opened\n";

    while ( fgets( line, 100, stream ) != NULL )
    {
        //checking the content of the string
#ifdef DEBUG_FLAG_L1
        ev << "[CObjList] >> Content "<< line << '\n';
#endif

        endDef = false;
        if (strstr( line, "[NEW_OBJ_DEF]") != NULL)
        {
            //New definition of button, next line is name
            if ( fgets( line, 100, stream ) != NULL )
            {
                pos = 0;
                tmpObj.num_method = 0;
                token = strtok( line, seps );

                while( token != NULL )
                {
                    switch (pos)
                    {
                        case 0: //Getting the object server name
#ifdef DEBUG_FLAG_L1
                            ev << "[CObjList] >> ObjSvr "<< token << '\n';
#endif
                            strcpy(tmpObj.name, line);
                            pos++;
                            break;

                        case 1: //Getting the method name
#ifdef DEBUG_FLAG_L1
                            ev << "[CObjList] >> cMethod "<< token << '\n';
#endif
                            tmpObj.ram_sz = atoi(token);
                            pos++;
                            break;
                    }
                    token = strtok( NULL, seps );
                }
            }

            while (!endDef)
            {
                if (fgets( line, 100, stream) == NULL)
                    return false;

                if (strstr( line, "[END_OBJ_DEF]") != NULL)
                {
                    m_Obj[m_ObjCount] = tmpObj;
                    m_ObjCount = m_ObjCount + 1;
                    endDef = true;
                }
                else
                {

```

```

/* Establish string and get the first token: */
pos = 0;
token = strtok( line, seps );

while( token != NULL )
{
    switch (pos)
    {
        case 0: //Getting the object server name

#ifdef DEBUG_FLAG_L1
        ev << "[CObjList] >> ObjSvr "<< token << '\n';
#endif

        strcpy(tmpName, token);
        pos++;
        break;

        case 1: //Getting the method name

#ifdef DEBUG_FLAG_L1
        ev << "[CObjList] >> cMethod "<< token << '\n';
#endif

        if (atol(token) <= LONG_MAX)
            tmpLong1 = atol(token);
        else
        {
            tmpLong1 =
LONG_MAX;
            ev << "[CObjList] >>
larger than expected value received! \n";
        }
        pos++;
        break;

        case 2: //Getting the method name

#ifdef DEBUG_FLAG_L1
        ev << "[CObjList] >> cMethod "<< token << '\n';
#endif

        tmpLong2 = atol(token);
        break;
    }
    token = strtok( NULL, seps );
}

tmpObj.method[tmpObj.num_method].setMethod(tmpName, tmpLong1, tmpLong2);
tmpObj.num_method = tmpObj.num_method +
1;
    }
}
}

/* Close stream */
if( fclose( stream ) )
    ev << "[CObjList] >> The file 'data' was not closed\n";

delete(token);

return true;
}

```

```

        return false;
    }

```

H. SIMEVENT.CPP

```

////////////////////////////////////
// file: SimEvent.cpp
////////////////////////////////////
#include "omnetpp.h"
#include "global.h"
#include "simevent.h"

CSimEvent::CSimEvent()
{
    numEvent = 0;
    maxPro = 0;
}

CSimEvent::~CSimEvent()
{
}

//Each added event will be indexed and the probability is stored.
//The maxPro is computed upon each additional event. This maxPro
//is used for the random uniform distribution number picked in the
//getNextEvent()
void CSimEvent::addEvent(int index, int pro)
{
    prob[index] = pro;
    maxPro = pro + maxPro;
    numEvent = numEvent + 1;
}

//Is to get the next event. The index return will be used in order
//to determine which button is pressed.
int CSimEvent::getNextEvent()
{
    int state = 0;

    //Generate a random discrete number
    int num = intuniform(0,maxPro);

    for (int i=0; i<numEvent; i++)
    {
        state = state + prob[i];

#ifdef DEBUG_FLAG_L2
        ev << "[CSimEvent] state >> " << state << '\n';
        ev << "[CSimEvent] num >> " << num << '\n';
#endif

        if (num <= state)
        {

```

```

        return i;
    }
}

return -1;
}

```

I. SMACHINE.CPP

```

////////////////////////////////////
// file: sMachine.cpp
////////////////////////////////////

#include <stdio.h>
#include <string.h>

#include "omnetpp.h"
#include "interactlist.h"
#include "global.h"
#include "statlog.h"

class sMachine : public cSimpleModule
{
    Module_Class_Members(sMachine,cSimpleModule,16384)
    virtual void finish();
    virtual void activity();
    virtual void initialize();

protected:
    bool isSwitchGateId(int id);
    bool isObjSvrGateId(int id);
    void InitObjSvrName();
    void AddObjSvrName(char *objsvr, int index);
    int FindObjSvrName(char *objsvr);

    bool initProcess();
    bool addNewProcess(cMessage *msg);
    int findProcessByRequestId(int id);
    bool execNextProcess();
    bool deleteProcess(int id);

    bool initRemoteRequestList();
    int addNewRemoteRequest(int instIndex);
    int findRemoteRequest(int index);
    bool recRemoteResponse(int index);

    double getWaitTime(double cpu, int osid);
    double getWaitTime();
    bool writeData(char *str);
    bool loadDef();

    bool isTimeSlice;
    long cpuPow, ramSz, dataRate2Sw;
    int prevObjSvrId;

```

```

double ramLimit;
double totalRamLimit;
double ramUsed;

double errorRate2Sw;
double processSwapTime;
double diskSwapTime;
double processCallTime;
double execTime;
double ranExec;

int m2swin_sz;
int m2swin_id[MAX_GATE_SZ];
cGate *m2swin_gt;
cGate *m2swout_gt;

int m2osin_sz;
int m2osin_id[MAX_GATE_SZ];
cGate *m2osin_gt;

char defFile[MAX_NAME_SZ];
char machineName[MAX_NAME_SZ];
char objSvrName[MAX_OBJSVR_PER_MACHINE][MAX_NAME_SZ];

slInstruction instControlBlock[MAX_SIM_INSTRUCTION];
int curProcessExecuted;
int curProcessAddPosition;

int remoteRequestIdFlag[MAX_REMOTE_REQUEST_PER_MACHINE];

char runName[MAX_NAME_SZ];
};

Define_Module( sMachine );

void sMachine::initialize()
{
    //initialize the list gateid
    m2osin_gt = gate("m2os_in");
    m2swin_gt = gate("m2sw_in");
    m2swout_gt = gate("m2sw_out");

    m2osin_sz = m2osin_gt->size();
    if (m2osin_sz > MAX_GATE_SZ)
    {
        ev << "[!! Server] >> Maximum number of gate defined (m2osin_sz) " <<
m2osin_sz << "\n";
        m2osin_sz = MAX_GATE_SZ;
    }

    for (int i=0; i<m2osin_sz; i++)
    {
        cGate *tmp = gate("m2os_in", i);
        m2osin_id[i] = tmp->id();
    }
}

#ifdef DEBUG_FLAG_L1

```

```

        ev << "[sMachine] >> ObjSvr (i) " << i << " (gateid) " << tmp->id() << '\n';
    #endif
    }

    //initializing the list of server to switch gate id
    m2swin_sz = m2swin_gt->size();
    if (m2swin_sz > MAX_GATE_SZ)
    {
        ev << "[!! sMachine] >> Maximum number of gate defined (m2swin_sz) " <<
m2swin_sz << '\n';
        m2swin_sz = MAX_GATE_SZ;
    }

    for (i=0; i<m2swin_sz; i++)
    {
        cGate *tmp = gate("m2sw_in", i);
        m2swin_id[i] = tmp->id();
    }

#ifdef DEBUG_FLAG_L1
    ev << "[sMachine] >> Switch (i) " << i << " (gateid) " << tmp->id() << '\n';
#endif
}

    cpuPow = 0;
    ramSz = 0;
    ramUsed = 0;
    processSwapTime = 0.0;
    diskSwapTime = 0.0;
    prevObjSvrId = -1;

    InitObjSvrName();

    //Getting the configuration file name and loading the configuration
    cPar &tmpPar = par("def_file");
    strcpy(defFile, tmpPar.stringValue ());
    isTimeSlice = false;

    tmpPar = parentModule()->par("run_name");
    strcpy(runName, tmpPar.stringValue ());
    ev << "[sRole] >> Initializing " << runName << '\n';

    loadDef();

    //Getting the configuration file name
    setName(machineName);

    //Initialize the process handling list.
    initProcess();

    //To set the data rate.
    tmpPar.setLongValue (dataRate2Sw);
    m2swin_gt->setDataRate(&tmpPar);
    m2swout_gt->setDataRate(&tmpPar);

    //Initialize the interact list
    CInteractList::GetInstance();

```

```

        //To initialize the request list.
        initRemoteRequestList();
    }

    //To handle the finish event
    void sMachine::finish()
    {
        delete(m2osin_gt);
        delete(m2swin_gt);
        delete(m2swout_gt);
    }

    void sMachine::activity()
    {
        double clock_speed = 1.0;
        double process_time = 0.0;
        double avg_utilization = 0.0;

        int type, objsvr_id, msgSz, requestId, index, tmpRam;
        bool newMsg = false;

        cPar tmpPar;
        cPar cpObjSvrName, cpMethodName, cpRequestorName;
        cMessage *svr_resp, *m_resp, *obj_msg;
        cGate *rcv_gate;
        cOutVector resp_v("CPU");

        //waiting code
        process_time = uniform(1.0,3.0);
        wait(process_time);

        char tmpStr[MAX_NAME_SZ];

        for(;;)
        {
            //Listen for incoming message
            newMsg = false;
            cMessage *rcv_msg = receive(0.01);

            if (rcv_msg != NULL)
            {
                newMsg = true;
                type = rcv_msg->kind(); //Message kind
                rcv_gate = rcv_msg->arrivalGate();
            }

#ifdef DEBUG_FLAG_L2
            ev << "[sMachine] >> receiving msg type " << type << "\n";
#endif

            //Message is from the m2sw_in port.
            if (isObjSvrGateId(rcv_gate->id()))
            {
                switch (type)
                {
                    case OBJSVR_REGISTER:

```

```

server                                     //OBJSVR register message to be sent over to the DNS

// Forward the information to the server
NAME_REGISTER );
svr_resp = new cMessage( "NAME_REGISTER",

>par("add_name");
svr_resp->addPar("add_machine") = machineName;
svr_resp->addPar("add_name")      =      rcv_msg-

>par("add_obj");
svr_resp->addPar("add_obj")      =      rcv_msg-

tmpRam = (int)rcv_msg->par("ram_total");

//if the ramLimit is set to 0.0, we will compute the
//best pattern without considering the ram size.
if (ramLimit != 0.0)
{
    ramUsed = ramUsed + tmpRam;
    totalRamLimit = ramLimit*ramSz;

    //checking for the amount of ram used.
    if (ramUsed > totalRamLimit)
    {
        //the amount of ram used is higher than
        //available. Flag an error.
        CStatLog::GetInstance()-
        >writeLogWithRamError(runName, totalRamLimit, ramUsed);
        endSimulation();
    }
}

send( svr_resp, "m2sw_out");

#ifdef DEBUG_FLAG_L2
    ev << "[sMachine] >> Sending NAME_REGISTER response" << '\n';
#endif

//Registering which port is the object server from.
tmpPar = rcv_msg->par("add_name");
strcpy(tmpStr, tmpPar.stringValue ());
AddObjSvrName(tmpStr, rcv_gate->index());

#ifdef DEBUG_FLAG_L2
    ev << "[sMachine] >> Received object server dns request " << tmpPar << '\n';
#endif

//deleting the received message
delete rcv_msg;

break;

//OBJSVR request to execute on CPU
case OBJSVR_EXECUTE_ON_CPU:
#ifdef DEBUG_FLAG_L2
    ev << "[sMachine] >> Received object server request to execute. \n";
#endif

```

```

//Try adding new process. There might be an error if
there is in
//the instControlBlock is full
if (addNewProcess(rcv_msg))
{
    //delete the rcv_msg
    delete rcv_msg;

#ifdef DEBUG_FLAG_L2
    ev << "[sMachine] >> request saved on instControlBlock! \n";
#endif
}
else
{
    //we might want to inform the sender.
    ev << "[sMachine] >> instControlBlock full,
message discarded! \n";
}

break;

//OBJSVR send the response back to the role, or remote object
server
case OBJSVR_RESPONSE:
#ifdef DEBUG_FLAG_L2
    ev << "[sMachine] >> Received object server response. \n";
#endif

//We have to check whether the address is the to an object
server running on
//on the current machine.
tmpPar = rcv_msg->par("add_request");
strcpy(tmpStr, tmpPar.stringValue ());
objsvr_id = FindObjSvrName(tmpStr);
ev << "[sMachine] >> Received object server response to requestor " << tmpPar << " "
<< objsvr_id << "\n";

if (objsvr_id == -1)
{
    //The object server is not running on the current
machine
    msgSz = rcv_msg->par("msgSz");
    m_resp = new cMessage(
"OBJSVR_RESPONSE", OBJSVR_RESPONSE );
    m_resp->addPar("add_request") = rcv_msg-
>par("add_request");
    m_resp->addPar("request_id") = rcv_msg-
>par("request_id");

    m_resp->addPar("msgSz") = msgSz;
    m_resp->setLength(msgSz);
    send( m_resp, "m2sw_out");
    delete(rcv_msg);
}
else
{
    //The object server is running on the current
machine.

    //Process the request at the machine

```

```

        requestId = rcv_msg->par("request_id");
        index = findProcessByRequestId(requestId);
        if (index == -1)
            break;
        else
        {
            recRemoteResponse(index);
        }
    }
    ev << "[sMachine] >> Receiving a remote response id " << requestId << "\n";

        delete(rcv_msg);
    }
    break;
}
}
else if (isSwitchGateId(rcv_gate->id()))
{
    switch( type )
    {
        case OBJSVR_RESPONSE:
#ifdef DEBUG_FLAG_L2
        ev << "[sMachine] >> Received object server response. \n";
#endif

        //receiving a response from the remote object server.
        requestId = rcv_msg->par("request_id");
        index = findProcessByRequestId(requestId);
        if (index == -1)
            break;
        else
        {
            recRemoteResponse(index);
        }
    }
    ev << "[sMachine] >> Receiving a remote response id " << requestId << "\n";

        delete(rcv_msg);
        break;

    case INVOKE_OS2OS_CALL:
        cpObjSvrName = rcv_msg->par("add_os"); //Object
server name

        cpMethodName = rcv_msg->par("add_method");
//Method name

        cpRequestorName = rcv_msg->par("add_request");
//Method name

        requestId = rcv_msg->par("request_id"); //Method name
        delete rcv_msg;

        //Invoking object server call
        obj_msg = new cMessage( "INVOKE_OS2OS_CALL",
INVOKE_OS2OS_CALL );

        obj_msg->addPar("add_os") = cpObjSvrName;
        obj_msg->addPar("add_method") = cpMethodName;
        obj_msg->addPar("add_request") = cpRequestorName;
        obj_msg->addPar("request_id") = requestId;

        strcpy(tmpStr, cpObjSvrName.stringValue ());

```

```

objsvr_id = FindObjSvrName(tmpStr);

//Need to check with port to send out to.
send( obj_msg, "m2os_out", objsvr_id);

#ifdef DEBUG_FLAG_L2
    ev << "[Server] >> Invoking object server call on objsvr " << objsvr_id
<< "\n";
#endif
    break;
case INVOKE_OBJECT_SVR_CALL:
    //Receive message from the switch
    cpObjSvrName = rcv_msg->par("add_os"); //Object
server name
    cpMethodName = rcv_msg->par("add_method");
//Method name
    cpRequestorName = rcv_msg->par("add_request");
//Method name

    requestId = rcv_msg->par("request_id");
    delete rcv_msg;

    //Invoking object server call
    obj_msg = new cMessage(
"INVOKE_OBJECT_SVR_CALL", INVOKE_OBJECT_SVR_CALL );
    obj_msg->addPar("add_os") = cpObjSvrName;
    obj_msg->addPar("add_method") = cpMethodName;
    obj_msg->addPar("add_request") = cpRequestorName;
    obj_msg->addPar("request_id") = requestId;

    strcpy(tmpStr, cpObjSvrName.stringValue ());
    objsvr_id = FindObjSvrName(tmpStr);

    //Need to check with port to send out to.
    send( obj_msg, "m2os_out", objsvr_id);
    ev << "[sMachine] >> " << machineName << " receive from " <<
cpRequestorName << "\n";

#ifdef DEBUG_FLAG_L2
    ev << "[Server] >> Invoking object server call on objsvr " << objsvr_id
<< "\n";
#endif
    break;
}
}
if (!newMsg)
    execNextProcess();
}

}

//*****
//* Protected Method
//*****
//Method to initialize the request list. The request list is used to
//store the list of id that will be used to identify the remote request that was
//forward other object server. This is needed because the machine may process a

```

//few message from the same object server and the completion of the message may
//not be in order.

bool sMachine::initRemoteRequestList()

```
{
    for (int i=0; i<MAX_REMOTE_REQUEST_PER_MACHINE; i++)
    {
        remoteRequestIdFlag[i] = -1;
    }

    return true;
}
```

//Method to add a new remote request in the request list.

//The method will find a remoteRequestIdFlag that is -1 and

//use that position to store the corresponding instIndex.

int sMachine::addNewRemoteRequest(int instIndex)

```
{
    bool found = false;
    int tmpInt = 0;

    //Search the requestIdFlag to find an id that was not in use.
    while ((tmpInt<MAX_REMOTE_REQUEST_PER_MACHINE) && (!found))
    {
        if (remoteRequestIdFlag[tmpInt] == -1)
        {
            remoteRequestIdFlag[tmpInt] = instIndex;
            found = true;
            return tmpInt;
        }
        tmpInt++;
    }

    return -1;
}
```

//Method to return the instruction index of the remote request response.

int sMachine::findRemoteRequest(int index)

```
{
    return remoteRequestIdFlag[index];
}
```

//Method to add a new process in queue

bool sMachine::initProcess()

```
{
    for (int i=0; i<MAX_PROCESS_PER_MACHINE; i++)
    {
        instControlBlock[i].used = false;
    }

    curProcessExecuted = 0;
    curProcessAddPosition = 0;

    return true;
}
```

```

//Method to add a new process in queue
int sMachine::findProcessByRequestId(int id)
{
    bool found = false;
    int tmpInt = 0;

    while ((tmpInt < MAX_PROCESS_PER_MACHINE) && (!found))
    {
        if ((instControlBlock[tmpInt].used == true) &&
            (instControlBlock[tmpInt].remoteRequestId == id))
        {
            return tmpInt;
        }
        else
        {
            tmpInt++;
        }
    }

    return -1;
}

//Method to add a new process in queue
bool sMachine::recRemoteResponse(int index)
{
    int remoteCallIndex = 0;
    int request_id, objsvr_id;
    cMessage *m_resp;

    //reset the remote request id to indicate that the response was received
    remoteRequestIdFlag[instControlBlock[index].remoteRequestId] = -1;

    //There is some remote instruction. Format message to send to the remote objsvr.
    if (instControlBlock[index].curRemoteInstruction !=
instControlBlock[index].maxRemoteInstruction)
    {
        //if there is still some other remote call, process the next one.
        instControlBlock[index].remoteRequestId = addNewRemoteRequest(index);

        remoteCallIndex = instControlBlock[index].curRemoteInstruction;
        instControlBlock[index].curRemoteInstruction++;

        cMessage *call_msg = new cMessage( "INVOKE_OS2OS_CALL",
INVOKE_OS2OS_CALL );
        call_msg->addPar("add_os") =
instControlBlock[index].interactList.interactCall[remoteCallIndex].cObjsvr;
        call_msg->addPar("add_method") =
instControlBlock[index].interactList.interactCall[remoteCallIndex].cMethod;
        call_msg->addPar("add_request") = instControlBlock[index].cObjSvrName;
        call_msg->addPar("request_id") = instControlBlock[index].remoteRequestId;

        int port =
FindObjSvrName(instControlBlock[index].interactList.interactCall[remoteCallIndex].cObjsvr);
        if ( port == -1)
        {

```

```

switch
    //Object server is not running on the current machine, forward to the
    send( call_msg, "m2sw_out");
    }
    else
    {
        //Object server is not running on the current machine, forward to the
        correct object server
        int result = send( call_msg, "m2os_out", port);
    }

    instControlBlock[index].wait = true;
}
else
{
    instControlBlock[index].used = false;
    instControlBlock[index].wait = false;

    request_id = instControlBlock[index].requestId;
    objsvr_id = FindObjSvrName(instControlBlock[index].cObjSvrName);

    // Forward the information to the server
    m_resp = new cMessage( "MACHINE_COMPLETE_EXEC_WITH_REMOTE",
MACHINE_COMPLETE_EXEC_WITH_REMOTE );
    m_resp->addPar("request_id") = request_id;

    //set the remote request id to -1.
    remoteRequestIdFlag[instControlBlock[index].remoteRequestId] = -1;

#ifdef DEBUG_FLAG_L2
    ev << "[sMachine] >> Sending MACHINE_COMPLETE_EXEC response " <<
    instControlBlock[index].cObjSvrName << "\n";
#endif
    send( m_resp, "m2os_out", objsvr_id);
}

return true;
}

//Method to add a new process in queue
bool sMachine::addNewProcess(cMessage *msg)
{
    bool found = false;
    int index;
    int tmpInt = 0;
    cPar cpObjSvrName, cpMethodName, cpOrigRequestName;

    //search for the first empty slot.
    while ((tmpInt<MAX_PROCESS_PER_MACHINE) && (!found))
    {
        if (instControlBlock[tmpInt].used == false)
        {
            //found new used position
            found = true;
            index = tmpInt;
        }
    }
}

```

```

        else
        {
            tmpltnt++;
        }
    }

    if (!found)
        return false;

    int cpu = msg->par("cpu");

    //both cpu and cpuPow is defined with unit 1000Hz
    double process_time = (double)(cpu)*((double)1/cpuPow);
    ev << "[sMachine] (init) >> index " << index << '\n';
    ev << "[sMachine] (init) >> process time " << process_time << '\n';
    ev << "[sMachine] (init) >> cpu " << cpu << '\n';
    ev << "[sMachine] (init) >> cpuPow " << cpuPow << '\n';

    //initializing the value of the instruction control block
    cpObjSvrName = msg->par("add_os"); //Object server name
    strcpy(instControlBlock[index].cObjSvrName, cpObjSvrName.stringValue ());
    cpMethodName = msg->par("add_method"); //Method name
    strcpy(instControlBlock[index].cMethodName, cpMethodName.stringValue ());
    cpOrigRequestName = msg->par("orig_request"); //Method name
    strcpy(instControlBlock[index].cOrigRequest, cpOrigRequestName.stringValue ());

    //wait is used to determine whether the process is waiting for response from
    //a remote object server.
    instControlBlock[index].wait = false;

    //setting the current indexed instControlBlock as used.
    instControlBlock[index].used = true;

    //Setting the request id
    instControlBlock[index].requestId = msg->par("request_id");

    //Setting the process time.
    instControlBlock[index].processTime = process_time;

    //Setting the information for remote instruction.
    sInteract interact;
    found = false;
    found = CInteractList::GetInstance()->getInteract(instControlBlock[index].cObjSvrName,
    instControlBlock[index].cMethodName, interact);

    //if there is an interaction list found for this method implies that remote objsvr call is
    //required.
    if (found)
    {
        instControlBlock[index].curRemoteInstruction = -1;
        instControlBlock[index].maxRemoteInstruction = interact.iNumInteract;
        instControlBlock[index].remoteRequestId = -1;
        instControlBlock[index].interactList = interact;
    }
    else
    {

```

```

        instControlBlock[index].curRemoteInstruction = -1;
        instControlBlock[index].maxRemoteInstruction = 0;
        instControlBlock[index].remoteRequestId = -1;
    }

    return true;
}

//Method to return the next process for processing
bool sMachine::execNextProcess()
{
    bool found = false;
    bool completeExec = false;
    bool execRemote = false;
    int index;

    int objsvr_id, request_id, port;
    int remoteCallIndex;
    cPar cpObjSvrName, cpMethodName;
    cMessage *m_resp;
    int tmpInt = 0;

    //An attempt to make the behaviour of the execution process more random.
    // old code
    // double ranExecTime = uniform(-1*ranExec, ranExec);

    double ranExecTime = uniform(0.0, execTime);
    double ranDelayTime = uniform(0.0, processSwapTime);

    //To find the next instruction to execute
    tmpInt = curProcessExecuted;
    while ((tmpInt < MAX_PROCESS_PER_MACHINE) && (!found))
    {
        if ((instControlBlock[tmpInt].used == true) &&
            (instControlBlock[tmpInt].wait == false))
        {
            //found new used position
            found = true;
            index = tmpInt;
        }
        else
        {
            tmpInt++;
        }
    }

    if (!found)
    {
        tmpInt = 0;
        while ((tmpInt <= (curProcessExecuted-1)) && (!found))
        {
            if ((instControlBlock[tmpInt].used == true) &&
                (instControlBlock[tmpInt].wait == false))
            {
                //found new used position
                found = true;
            }
        }
    }
}

```

```

        index = tmpInt;
    }
    else
    {
        tmpInt++;
    }
}

//No instruction found. CPU remains idle
if (!found)
    return false;

if (instControlBlock[index].processTime == 0.0)
{
    //if processTime is 0.0, this could be the scenario when the machine is
    //waiting for the response from remote system. If wait is true, then it
    //is time to execute the next remote call.
    if (instControlBlock[index].wait == true)
    {
        //instruction still waiting for the remote object server to response.
        //ie do nothing for now.
        return true;
    }
    else
    {
        //instruction not waiting for the response
        //execute next remote instruction call if any.
        if (instControlBlock[index].curRemoteInstruction ==
instControlBlock[index].maxRemoteInstruction)
        {
            //all instruction completed.
            completeExec = true;
            execRemote = false;
        }
    }
}
else
{
    if (isTimeSlice)
    {
        // old
        if (instControlBlock[index].processTime < (execTime+ranExecTime))
        if (instControlBlock[index].processTime < (ranExecTime))
        {
            wait(instControlBlock[index].processTime+ranDelayTime);
            instControlBlock[index].processTime = 0.0;
            execRemote = true;
        }
        else
        {
            // old
            instControlBlock[index].processTime =
instControlBlock[index].processTime-(execTime+ranExecTime);
            // old
            wait(execTime+ranExecTime+ranDelayTime);
            instControlBlock[index].processTime =
instControlBlock[index].processTime-(ranExecTime);
            wait(ranExecTime+ranDelayTime);

```

```

        }
    }
    else
    {
        ev << "[sMachine] >> index " << index << "\n";
        ev << "[sMachine] >> random delay time " << ranDelayTime << "\n";
        ev << "[sMachine] >> process time " << instControlBlock[index].processTime << "\n";
        ev << "[sMachine] >> wait time " << instControlBlock[index].processTime+ranDelayTime << "\n";

        wait(instControlBlock[index].processTime+ranDelayTime);
        instControlBlock[index].processTime = 0.0;
        execRemote = true;
    }
}

if (execRemote)
{
    //The current implementation will requires the machine to execute all the
instruction
    //needed for the local object server before requesting for remote call.

    //After setting the processTime to 0.0, we need to check whether there
    //is any remote obj server call.
    if (instControlBlock[index].maxRemoteInstruction == 0)
    {
        //There is no remote instruction
        completeExec = true;
    }
    else
    {
        completeExec = false;

        //There is some remote instruction. Format message to send to the
remote objsvr.
        instControlBlock[index].remoteRequestId =
addNewRemoteRequest(index);

        if (instControlBlock[index].curRemoteInstruction == -1)
        {
            instControlBlock[index].curRemoteInstruction = 0;
            remoteCallIndex = instControlBlock[index].curRemoteInstruction;
        }
        else
        {
            remoteCallIndex = instControlBlock[index].curRemoteInstruction;
        }

        instControlBlock[index].curRemoteInstruction++;

        cMessage *call_msg = new cMessage( "INVOKE_OS2OS_CALL",
INVOKE_OS2OS_CALL );
        call_msg->addPar("add_os") =
instControlBlock[index].interactList.interactCall[remoteCallIndex].cObjsvr;
        call_msg->addPar("add_method") =
instControlBlock[index].interactList.interactCall[remoteCallIndex].cMethod;

```

```

        call_msg->addPar("add_request") =
instControlBlock[index].cObjSvrName;
        call_msg->addPar("request_id") =
instControlBlock[index].remoteRequestId;
//        call_msg->addPar("request_id") = remoteCallIndex;

        port =
FindObjSvrName(instControlBlock[index].interactList.interactCall[remoteCallIndex].cObjsvr);
        if ( port == -1)
        {
            //Object server is not running on the current machine, forward to
the switch
            send( call_msg, "m2sw_out");
        }
        else
        {
            //Object server is not running on the current machine, forward to
the correct object server
            int result = send( call_msg, "m2os_out", port);
        }

        instControlBlock[index].wait = true;
#ifdef DEBUG_FLAG_L2
ev << "[sMachine] >> Sending remote objsvr call to " << remoteCallIndex << " " <<
instControlBlock[index].interactList.interactCall[remoteCallIndex].cObjsvr << " " <<
instControlBlock[index].interactList.interactCall[remoteCallIndex].cMethod << '\n';
#endif
    }
}

if (completeExec)
{
    instControlBlock[index].used = false;
    request_id = instControlBlock[index].requestId;
    objsvr_id = FindObjSvrName(instControlBlock[index].cObjSvrName);

    // Forward the information to the server
    m_resp = new cMessage( "MACHINE_COMPLETE_EXEC",
MACHINE_COMPLETE_EXEC );
    m_resp->addPar("request_id") = request_id;

#ifdef DEBUG_FLAG_L2
ev << "[sMachine] >> Sending response" << '\n';
#endif
    send( m_resp, "m2os_out", objsvr_id);
}

curProcessExecuted = index++;
return true;
}

//Method to delete the process
bool sMachine::deleteProcess(int id)
{
    return true;
}

```

```

//Method to check whether the gate id is a switch gate id.
bool sMachine::isSwitchGateId(int id)
{
    for (int i=0; i<m2swin_sz; i++)
    {
        if (m2swin_id[i] == id)
            return true;
    }

    return false;
}

//To return the index of the gate.
void sMachine::InitObjSvrName()
{
    char tmpStr[MAX_NAME_SZ] = "\0";

    for (int i=0; i<MAX_OBJSVR_PER_MACHINE; i++)
    {
        strcpy(objSvrName[i], tmpStr);
    }
}

//To save the object server name and the index of the gate,
//information will be used for the reply message
void sMachine::AddObjSvrName(char *objsvr, int index)
{
    strcpy(objSvrName[index], objsvr);
}

//To return the index of the gate.
int sMachine::FindObjSvrName(char *objsvr)
{
    char tmpstr[MAX_NAME_SZ];

    for (int i=0; i<MAX_OBJSVR_PER_MACHINE; i++)
    {
        strcpy(tmpstr, objSvrName[i]);
        if (strcmp(objsvr, tmpstr) == 0)
            return i;
    }

    return -1;
}

//Method to check whether the gate id is a server gate id.
bool sMachine::isObjSvrGateId(int id)
{
    for (int i=0; i<m2osin_sz; i++)
    {
        if (m2osin_id[i] == id)
            return true;
    }

    return false;
}

```

```

}

//Method to compute the total amount of time to wait.
double sMachine::getWaitTime(double cpu, int osid)
{
    double totTime = cpu;
    double waitTime = 0.0;

    if (osid != prevObjSvrId)
    {
        totTime = totTime + processSwapTime;
        prevObjSvrId = osid;
    }

    waitTime = uniform(totTime-0.05, totTime+0.05);
    return waitTime;
}

//Method to compute the total amount of time to wait.
double sMachine::getWaitTime()
{
    double waitTime = processCallTime;

    return waitTime;
}

bool sMachine::writeData(char *str)
{
    //The logging function can be turn off at compilation time.
    #ifdef DEBUG_DETAIL_LOG
        FILE *stream;

        if( (stream = fopen( "test.txt", "a" )) == NULL )
            printf( "The file 'test.txt' was not opened\n" );

        //fwrite( str, sizeof( char ), 100, stream );
        fprintf(stream, str);

        /* Close stream */
        if( fclose( stream ) )
            printf( "The file 'data' was not closed\n" );
    #endif

    return true;
}

//Method to load the configuration file
bool sMachine::loadDef()
{
    char seps[] = "\n";
    char tmpStr[50];
    char buff[1024 + 1];

    FILE *file;

    if( (file = fopen( defFile, "r+t" )) == NULL )

```

```

printf( "The file '%s' was not opened\n", defFile );

if (!file)
{
    ev << "[sMachine] >> The file 'data' was not opened\n";
    return false;
}

//some changes were made in the way we read from the file because
//there is some conflict between omnet++ 2.3 with ifstream.
//The using-declaration generate a error during compilation.
//There is a conflict with including "ifstream.h" and "strstrea.h"
//
//After copying from the buf string to tmpStr, we
//need to set the last character to NULL so that any new-line character
//will not affect the string read.
while (fgets(buf, 1024, file))
{
    if (strstr(buf, "[name]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[name]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        strcpy(machineName, tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Object server name "<< machineName << '\n';
#endif
    }
    else if (strstr (buf, "[cpu_pow]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[cpu_pow]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        cpuPow = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> CPU power "<< cpuPow << '\n';
#endif
    }
    else if (strstr (buf, "[exec_time]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[exec_time]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        execTime = strtod(tmpStr, '\0');
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Execution time "<< execTime << '\n';
#endif
    }
    else if (strstr (buf, "[ran_exec]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[ran_exec]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        ranExec = strtod(tmpStr, '\0');
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Random Execution time "<< ranExec << '\n';
#endif
    }
    else if (strstr (buf, "[ram_sz]") != NULL)
    {

```

```

        strcpy(tmpStr, buf+strlen("[ram_sz"])+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        ramSz = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Ram size " << ramSz << '\n';
#endif
    }
    else if (strstr (buf, "[ram_limit]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[ram_limit"])+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        ramLimit = strtod(tmpStr, '\0');
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Ram limit " << ramSz << '\n';
#endif
    }
    else if (strstr (buf, "[dr_sw]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[dr_sw"])+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        dataRate2Sw = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Data rate to the switch " << dataRate2Sw << '\n';
#endif
    }
    else if (strstr (buf, "[er_sw]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[er_sw"])+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        errorRate2Sw = strtod(tmpStr, '\0');
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Error rate to the switch " << errorRate2Sw << '\n';
#endif
    }
    else if (strstr (buf, "[process_call]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[process_call"])+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        processCallTime = strtod(tmpStr, '\0');
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Process call time " << processCallTime << '\n';
#endif
    }
    else if (strstr (buf, "[process_swap]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[process_swap"])+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        processSwapTime = strtod(tmpStr, '\0');
#ifdef DEBUG_FLAG_L1
        ev << "[sMachine] >> Process swap time " << processSwapTime << '\n';
#endif
    }
    else if (strstr (buf, "[disk_swap]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[disk_swap"])+1);
        tmpStr[strlen(tmpStr)-1] = '\0';

```

```

        diskSwapTime = strtod(tmpStr, '\0');
#ifdef DEBUG_FLAG_L1
    ev << "[sMachine] >> Disk swap time " << diskSwapTime << '\n';
#endif
    }
    else if (strstr (buf, "[time_slice]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[time_slice]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        if (strcmp("no", tmpStr)==0)
            isTimeSlice = false;
        else
            isTimeSlice = true;
#ifdef DEBUG_FLAG_L1
    ev << "[sMachine] >> isTimeSlice " << isTimeSlice << '\n';
#endif
    }
}

if( fclose( file ) )
    printf( "The file '%s' was not closed\n", defFile);

return true;

}

```

J. SOBJSVR.CPP

```

////////////////////////////////////
// file: objsvrapp.cpp
////////////////////////////////////

#include <string.h>
#include "omnetpp.h"
#include "global.h"
#include "objlist.h"

char objname[20];

class sObjSvr : public cSimpleModule
{
    Module_Class_Members(sObjSvr,cSimpleModule,16384)
    virtual void finish();
    virtual void activity();
    virtual void initialize();

protected:
    bool initRequestList();
    bool clearRequestList();

    int addNewRequest(cMessage *msg);
    bool completeRequest(int id);

    bool isMachineGateId(int id);

```

```

    bool loadDef();

    cGate *os2min_gt;
    int os2min_id[MAX_GATE_SZ];
    int os2min_sz;

    char defFile[MAX_NAME_SZ];

    int ramUtil;    //ram usage of the object server without the consideration of the object
    int numObj;
    int ramTotal;
    int machineId;

    char objSvrName[MAX_NAME_SZ];
    char objName[MAX_NAME_SZ];
    CObj objSvc;

    cMessage *requestList[MAX_PROCESS_PER_MACHINE];
    bool requestIdFlag[MAX_PROCESS_PER_MACHINE];
};

Define_Module( sObjSvr );

//initialize method called when the object is first initialize.
void sObjSvr::initialize()
{
    os2min_gt = gate("os2m_in");

    //Initializing the value of the os2m_in gate id
    os2min_sz = os2min_gt->size();
    if (os2min_sz > MAX_GATE_SZ)
    {
        ev << "[!! sObjSvr] >> Maximum number of gate defined (os2min_sz) " <<
os2min_sz << "\n";
        os2min_sz = MAX_GATE_SZ;
    }

    for (int i=0; i<os2min_sz; i++)
    {
        cGate *tmp = gate("os2m_in", i);
        os2min_id[i] = tmp->id();

#ifdef DEBUG_FLAG_L1
        ev << "[sObjSvr] >> Server (i) " << i << " (gateid) " << tmp->id() << "\n";
#endif
    }

#ifdef DEBUG_FLAG_L1
    ev << "***** [sObjSvr] ***** " << "\n";
#endif

    //initialize the value
    ramUtil = 0;
    numObj = 1;
    ramTotal = 0;

```

```

        //Loading the configuration file.
cPar& filename = par("cfg_file");
strcpy(defFile, filename.stringValue ());
loadDef();          //Loading definition file

//Setting the object server name
setName(objSvrName);

//Getting the object server machine id
//machineId = (int)par("machine_id");

CObjList::GetInstance()->getObj(objName, objSvc);
ramTotal = ramUtil;// + numObj*objSvc.ram_sz;

//To initialize the request list.
initRequestList();
}

//To handle the finish event
void sObjSvr::finish()
{
    delete(os2min_gt);
    clearRequestList();
}

//method to handle activity event
void sObjSvr::activity()
{
    int type, requestId;
    int cpu, msgSz;
    bool found;

    cGate *rcv_gate;
    cPar cpMethodName, cpMachineName, cpRequestName, cpObjSvr;
    cMessage *objsvr_resp, *m_resp;
    char tmpStr[MAX_NAME_SZ];

    double process_time = uniform(1.0,3.0);
    wait(process_time);

    //Registering the
    cMessage *dns_reg_msg = new cMessage("OBJSVR_REGISTER",
OBJSVR_REGISTER );
    dns_reg_msg->addPar("add_name") = objSvrName;
    dns_reg_msg->addPar("add_obj") = objName;
    dns_reg_msg->addPar("ram_total") = ramTotal;

    send( dns_reg_msg, "os2m_out");

#ifdef DEBUG_FLAG_L2
    ev << "[ObjSvr] >> Registering object service with DNS" << '\n';
#endif

    for(;;)
    {
        // Receive message from the server

```

```

    cMessage *rcv_msg = receive();
    type = rcv_msg->kind();
    rcv_gate = rcv_msg->arrivalGate();

#ifdef DEBUG_FLAG_L2
    ev << "[ObjSvrApp] >> Received message (type) " << type << " from (gate_id) " <<
    rcv_gate->id() << "\n";
#endif

    //Message is from the machine port.
    if (isMachineGateId(rcv_gate->id()))
    {
        switch( type )
        {
            case INVOKE_OS2OS_CALL:
                //A call to invoke the object server.

                cpObjSvr = rcv_msg->par("add_os"); //Object server name
                cpMethodName = rcv_msg->par("add_method"); //Method name
                cpRequestName = rcv_msg->par("add_request");
                strcpy(tmpStr, cpRequestName.stringValue());
                strcpy(tmpStr, cpMethodName.stringValue());
                cpu = objSvc.getCPU(tmpStr);
                requestId = addNewRequest(rcv_msg);

#ifdef DEBUG_FLAG_L2
                ev << "[ObjSvrApp] >> receiving INVOKE_OS2OS_CALL from (requestor) " <<
                cpRequestName << "\n";
#endif

                // Forward the information to the server
                objsvr_resp = new cMessage( "OBJSVR_EXECUTE_ON_CPU",
                OBJSVR_EXECUTE_ON_CPU );
                objsvr_resp->addPar("request_id") = requestId;

                objsvr_resp->addPar("add_os") = rcv_msg->par("add_os");

                objsvr_resp->addPar("add_method") = rcv_msg->
                >par("add_method");
                objsvr_resp->addPar("orig_request") = rcv_msg->
                >par("add_request");
                objsvr_resp->addPar("cpu") = cpu;
                send( objsvr_resp, "os2m_out");
                break;

            case INVOKE_OBJECT_SVR_CALL:
                //A call to invoke the object server.

#ifdef DEBUG_FLAG_L2
                ev << "[ObjSvrApp] >> receiving INVOKE_OBJECT_SVR_CALL (method) " <<
                method_id << "\n";
#endif

                cpObjSvr = rcv_msg->par("add_os"); //Object server name
                cpMethodName = rcv_msg->par("add_method"); //Method name
                strcpy(tmpStr, cpMethodName.stringValue());
                cpu = objSvc.getCPU(tmpStr);

                requestId = addNewRequest(rcv_msg);

```

```

// Forward the information to the server
objsvr_resp = new cMessage( "OBJSVR_EXECUTE_ON_CPU",
OBJSVR_EXECUTE_ON_CPU );
objsvr_resp->addPar("request_id") = requestId;

objsvr_resp->addPar("add_os") = rcv_msg->par("add_os");

objsvr_resp->addPar("add_method") = rcv_msg-
>par("add_method");
objsvr_resp->addPar("orig_request") = rcv_msg-
>par("add_request");

objsvr_resp->addPar("cpu") = cpu;
send( objsvr_resp, "os2m_out");

#ifdef DEBUG_FLAG_L2
    ev << "[ObjSvr] >> Sending response (cpu) " << cpu << " (ram) " << ram << "\n";
#endif

break;

case MACHINE_COMPLETE_EXEC:
    //A return call from the machine to indicate that the execution has
    completed.

    //Retrieve the requestId to verify that the id is one that was sent
    to from

    //the object server.
    requestId = rcv_msg->par("request_id");
    found = completeRequest(requestId);

#ifdef DEBUG_FLAG_L2
    ev << "[ObjSvrApp] >> receiving MACHINE_COMPLETE_EXEC replied to " <<
    requestId[requestId]->par("add_request") << "\n";
#endif

    //If the request id is found.
    if (found == true)
    {
        m_resp = new cMessage( "OBJSVR_RESPONSE",
OBJSVR_RESPONSE );
        cpRequestName = requestList[requestId]-
>par("add_request");
        m_resp->addPar("add_request") =
requestList[requestId]->par("add_request");
        m_resp->addPar("request_id") = requestList[requestId]-
>par("request_id");

        cpObjSvr = requestList[requestId]->par("add_os");
//Object server name
        cpMethodName = requestList[requestId]-
>par("add_method"); //Method name
        strcpy(tmpStr, cpMethodName.stringValue());
        msgSz = objSvc.getMsgSize(tmpStr);
        m_resp->addPar("msgSz") = msgSz;

        send( m_resp, "os2m_out");
    }

```

```

        ev << "[ObjSvrApp] >> sending OBJSVR_RESPONSE from " << objSvrName << " to
(requestor) " << cpRequestName << "\n";

        //Deleting the message
        delete requestList[requestId];
        delete rcv_msg;
    }
    break;

case MACHINE_COMPLETE_EXEC_WITH_REMOTE:
    //A return call from the machine to indicate that the execution has
completed.

    //Retrieve the requestId to verify that the id is one that was sent
to from

    //the object server.
    requestId = rcv_msg->par("request_id");
    found = completeRequest(requestId);
    ev << "[ObjSvrApp] >> receiving MACHINE_COMPLETE_EXEC_WITH_REMOTE
replied to " << requestList[requestId]->par("add_request") << "\n";

    for (int j=0; j<MAX_REQUEST_PER_OBJSVR; j++)
    {
        if (requestIdFlag[j] == true)
        {
            cpRequestName = requestList[j]->par("add_request");
            strcpy(tmpStr, cpRequestName.stringValue());
            ev << "[ObjSvrApp] >> request id " << j << "(request) "
<< cpRequestName << "\n";
        }
    }

    //ifdef DEBUG_FLAG_L2
    //endif

    //If the request id is found.
    if (found == true)
    {
        m_resp = new cMessage( "OBJSVR_RESPONSE",
OBJSVR_RESPONSE );
        cpRequestName = requestList[requestId]-
>par("add_request");
        strcpy(tmpStr, cpRequestName.stringValue());

        m_resp->addPar("add_request") =
requestList[requestId]->par("add_request");
        m_resp->addPar("request_id") = requestList[requestId]-
>par("request_id");

        cpObjSvr = requestList[requestId]->par("add_os");
        cpMethodName = requestList[requestId]-
>par("add_method"); //Method name
        strcpy(tmpStr, cpMethodName.stringValue());
        msgSz = objSvc.getMsgSize(tmpStr);
        m_resp->addPar("msgSz") = msgSz;
    }
}

```

```

        send( m_resp, "os2m_out");
        ev << "[ObjSvrApp] >> sending OBJSVR_RESPONSE from " << objSvrName << " to
(requestor) " << cpRequestName << "\n";

```

```

        //Deleting the message
        delete requestList[requestId];
        delete rcv_msg;

```

```

    }
    break;

```

```

}

```

```

}

```

```

}

```

```

}

```

```

/*****

```

```

/* Protected Method */

```

```

*****/

```

```

//Method to initialize the request list. The request list is used to
//store the list of id that will be used to identify the request that was
//forward to the machine. This is needed because the machine may process a
//few message from the same object server and the completion of the message may
//not be in order.

```

```

bool sObjSvr::initRequestList()

```

```

{
    for (int i=0; i<MAX_REQUEST_PER_OBJSVR; i++)
    {
        requestIdFlag[i] = false;
    }

    return true;
}

```

```

bool sObjSvr::clearRequestList()

```

```

{
    for (int i=0; i<MAX_REQUEST_PER_OBJSVR; i++)
    {
        if (requestIdFlag[i] == true)
            delete requestList[i];
    }

    return true;
}

```

```

//Method to add a new request in the request list. The message is stored
//in case future work require the message.

```

```

int sObjSvr::addNewRequest(cMessage *msg)

```

```

{
    bool found = false;
    int index;
    int tmpInt = 0;

    //Search the requestIdFlag to find an id that was not in use.
    while ((tmpInt<MAX_REQUEST_PER_OBJSVR) && (!found))
    {
        if (requestIdFlag[tmpInt] == false)

```

```

        {
            found = true;
            index = tmpInt;
            requestIdFlag[tmpInt] = true;
        }
        else
        {
            tmpInt++;
        }
    }

    //If an available id is used.
    if (found)
    {
        requestList[index] = msg;
        return index;
    }
    else
    {
        //return -1 if nothing is found.
        return -1;
    }
}

```

```

//Method to handle a completion of request.
bool sObjSvr::completeRequest(int id)
{
    if (id > MAX_REQUEST_PER_OBJSVR)
    {
        return false;
    }

    if (requestIdFlag[id] == true)
    {
        requestIdFlag[id] = false;

        return true;
    }

    return false;
}

```

```

//Method to check whether the gate id is a server gate id.
bool sObjSvr::isMachineGateId(int id)
{
    for (int i=0; i<os2min_sz; i++)
    {
        if (os2min_id[i] == id)
            return true;
    }

    return false;
}

```

```

//Method to load the configuration file
bool sObjSvr::loadDef()

```

```

{
    char seps[] = "\n";

    char tmpStr[50];
    char buf[1024 + 1];

    FILE *file;

    if( (file = fopen( defFile, "r+t" )) == NULL )
    {
        printf( "The file '%s' was not opened\n", defFile );
        ev << "[sObjSvr] >> The file 'data' was not opened\n";

        return false;
    }

#ifdef DEBUG_FLAG_L2
    ev << "[sObjSvr] >> The file 'data' was opened\n";
#endif

    //some changes were made in the way we read from the file because
    //there is some conflict between omnet++ 2.3 with ifstream.
    //The using-declaration generate a error during compilation.
    //There is a conflict with including "ifstream.h" and "strstrea.h"
    //
    //After copying from the buf string to tmpStr, we
    //need to set the last character to NULL so that any new-line character
    //will not affect the string read.
    while (fgets(buf, 1024, file))
    {
        if (strstr(buf, "[name]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[name]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';
            strcpy(objSvrName, tmpStr);
#ifdef DEBUG_FLAG_L1
            ev << "[sObjSvr] >> Object server name "<< objSvrName << '\n';
#endif
        }
        else if (strstr (buf, "[ram_util]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[ram_util]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';
            ramUtil = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
            ev << "[sObjSvr] >> Ram Util "<< ramUtil << '\n';
#endif
        }
        else if (strstr(buf, "[object]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[object]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';
            strcpy(objName, tmpStr);
#ifdef DEBUG_FLAG_L1
            ev << "[sObjSvr] >> Object Supported "<< objName << '\n';
#endif
        }
    }
}

```

```

    }
    else if (strstr(buf, "[num_obj]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[num_obj]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        numObj = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sObjSvr] >> Number of object "<< numObj << "\n";
#endif
    }
}

if( fclose( file ) )
    printf( "The file '%s' was not closed\n", defFile);

return true;
}

```

K. SROLE.CPP

```

////////////////////////////////////
// file: sRole.cc
////////////////////////////////////
#ifndef __ROLE
#define __ROLE

#include <stdio.h>
#include <string.h>

#include "omnetpp.h"
#include "global.h"
#include "simevent.h"

#include "btnlist.h"
#include "statlog.h"

class sRole : public cSimpleModule
{
    Module_Class_Members(sRole,cSimpleModule,16384)
    virtual void finish();
    virtual void activity();
    virtual void initialize();

protected:
    bool isSwitchGateId(int id);
    bool loadDef();
    bool writeLog(char *str);

    double betweenCallTime;

    int r2swin_sz;
    int r2swin_id[MAX_GATE_SZ];
    cGate *r2swin_gt;
    cGate *r2swout_gt;

```

```

char defFile[MAX_NAME_SZ];
char logFile[MAX_NAME_SZ];
char runName[MAX_NAME_SZ];
char roleName[MAX_NAME_SZ];

int callPatternCount;
int minWait, maxWait, roleId, dataRate2Sw, roleType;
sCallPattern callPattern[MAX_BUTTON_PERROLE];

CSimEvent simEvent;
CBtnList btnList;

cStdDev stat;
};

Define_Module( sRole );

void sRole::initialize()
{
    //initialize the list gateid
    r2swin_gt = gate("r2sw_in");
    r2swout_gt = gate("r2sw_out");

    //initializing the list of switch to client gate id
    r2swin_sz = r2swin_gt->size();
    if (r2swin_sz > MAX_GATE_SZ)
    {
        ev << "[!! sRole] >> Maximum number of gate defined (r2swin_sz) " << r2swin_sz
<< "\n";
        r2swin_sz = MAX_GATE_SZ;
    }

    for (int i=0; i<r2swin_sz; i++)
    {
        cGate *tmp = gate("r2sw_in", i);
        r2swin_id[i] = tmp->id();
#ifdef DEBUG_FLAG_L1
        ev << "[sRole] >> sRole (i) " << i << " (gateid) " << tmp->id() << "\n";
#endif
    }

    //Initializing all the parameters
    callPatternCount = 0;

    //Getting the configuration file name and loading the configuration file
    cPar& tmpPar = par("def_file");
    strcpy(defFile, tmpPar.stringValue ());
    loadDef();

    //Setting the role name
    setName(roleName);

    //Setting the run name
    tmpPar = parentModule()->par("run_name");
    strcpy(runName, tmpPar.stringValue ());

```

```

ev << "[sRole] >> Initializing " << runName << '\n';

//The index will return roleId
roleId = index();
sprintf(logFile, "%s_log%d.txt", runName, roleId);

callPatternCount = 0; //Setting the number of button to 0
CBtnList::GetInstance();

CStatLog::GetInstance()->registerRoleType(roleType);

//Detail logging can be turn off at compilation time.
#ifdef DEBUG_DETAIL_LOG
FILE *stream;

if( (stream = fopen( logFile, "w+" )) == NULL )
    printf( "The file '%s' was not opened\n", logFile );
else
    fclose( stream );
#endif

//Setting for the data rate.
tmpPar.setLongValue (dataRate2Sw);
r2swin_gt->setDataRate(&tmpPar);
r2swout_gt->setDataRate(&tmpPar);
}

//To handle the finish event
void sRole::finish()
{
    char logStr[200];

    long num_samples = stat.samples();
    double smallest = stat.min();
    double largest = stat.max();
    double mean = stat.mean();
    double dev = stat.stddev();
    double var = stat.variance();

    //recording the sample size, minimum, maximum and average value
    CStatLog::GetInstance()->writeLog(runName, roleName, roleType, num_samples,
    smallest, largest, mean);

/*
    char tmpStr[MAX_NAME_SZ];
    sprintf(tmpStr, "stat_%s", logFile);
    FILE *stream;
    if( (stream = fopen( tmpStr, "w+" )) == NULL )
        printf( "The file 'data2' was not opened\n" );
    else
    {
        stat.saveToFile(stream);
        fclose( stream );
    }
*/

```

```

        stat.clearResult ();

        //Storing all information.
        sprintf(logStr, "%s_role%d (Min) %.4f (Max) %.4f (Ave) %.4f", runName, roleId, smallest,
largest, mean);
        writeLog(logStr);
    }

//To handle the activity event
void sRole::activity()
{
    int index;

    int current_server = 0;
    cOutVector resp_v("Role RT");

    //Time waited to ensure that DNS service is all register before sending.
    double process_time = 0.0;
    double response_time = 0.0;
    double cpuTime = 0.0;
    double startTime = 0.0;
    double sendTime = 0.0;
    double rcvTime = 0.0;
    double curSimTime = 0.0;
    double rdelay = 0.0;
    double roleTime = 0.0;

    char logStr[200];
    cMessage *dummy_msg = new cMessage();

    //wait for all the module to stablize before sending the register message
    process_time = uniform(1.0,3.0);
    wait(process_time);

    cMessage *dns_reg_msg = new cMessage("NAME_REGISTER", NAME_REGISTER );
    dns_reg_msg->addPar("add_name") = roleName;
    dns_reg_msg->addPar("add_machine") = roleName;
    send( dns_reg_msg, "r2sw_out");
    ev << "[*****] registering name " << roleName << "\n";

    for(;;)
    {
        ev << "[*****] looping name " << minWait << " " << maxWait << roleName << "\n";
        //Listen for incoming message
        process_time = uniform(minWait,maxWait);
        wait(process_time);

        //This will return the next event to execute.
        index = simEvent.getNextEvent();
        if (index == -1) //Do nothing
            continue;

        //To record the startTime of a button call
        startTime = simTime();
        ev << "[*****] name " << roleName << " " << index << "\n";
    }
}

```

```

//For all the call, send a request message to all the object server.
for (int j=0; j<callPattern[index].button.iNumCall; j++)
{
    cMessage *call_msg = new cMessage(
"INVOKE_OBJECT_SVR_CALL", INVOKE_OBJECT_SVR_CALL );
    call_msg->addPar("add_os") =
callPattern[index].button.svcCall[j].cObjsvr;
    call_msg->addPar("add_method") =
callPattern[index].button.svcCall[j].cMethod;
    call_msg->addPar("add_request") = roleName;
    call_msg->addPar("request_id") = -1;
    call_msg->addPar("roleTime") = simTime();

    int result = send( call_msg, "r2sw_out");
    ev << "[*****] name " << roleName << " send result " << result << " dest objsvr " <<
callPattern[index].button.svcCall[j].cObjsvr << " dest method " <<
callPattern[index].button.svcCall[j].cMethod << "\n";

    //The message receive will be in order of the message sent. So there is
no
    //need to store a request id.
    cMessage *done = receiveOn("r2sw_in");
    delete done;
}

//Completion of sending all the message and compute the time spent.
curSimTime = simTime();
response_time = curSimTime - startTime;
resp_v.record(response_time);
stat.collect(response_time);

sprintf(logStr, "Role%d Total Response \t%d\t%.4f\n", roleId, index,
response_time);
writeLog(logStr);
}
}

//*****
/* Protected Method
//*****
//Method to write additional data in files.
bool sRole::writeLog(char *str)
{
//The logging function can be turn off at compilation time.
#ifdef DEBUG_DETAIL_LOG
    FILE *stream;

    if( (stream = fopen( logFile, "a+" )) == NULL )
        printf( "The file '%s' was not opened\n", logFile );

    //fwrite( str, sizeof( char ), 100, stream );
    fprintf(stream, str);

    /* Close stream */
    if( fclose( stream ) )
        printf( "The file 'data' was not closed\n" );

```

```

#endif

    return true;
}

//Method to check whether the gate id is a server gate id.
bool sRole::isSwitchGateId(int id)
{
    for (int i=0; i<r2swin_sz; i++)
    {
        if (r2swin_id[i] == id)
            return true;
    }

    return false;
}

//Method to load the configuration file
bool sRole::loadDef()
{
    char seps[] = "\n";
    char *token;

    char tmpStr[50];
    char buf[1024 + 1];
    int pos;

    sCallPattern tmpPattern;

    FILE *file;

    if( (file = fopen( defFile, "r+t" )) == NULL )
    {
        printf( "The file '%s' was not opened\n", defFile );
        return false;
    }

#ifdef DEBUG_FLAG_L2
    ev << "[sRole] >> The file 'data' was opened\n";
#endif

    //some changes were made in the way we read from the file because
    //there is some conflict between omnet++ 2.3 with ifstream.
    //The using-declaration generate a error during compilation.
    //There is a conflict with including "ifstream.h" and "strstrea.h"
    //
    //After copying from the buf string to tmpStr, we
    //need to set the last character to NULL so that any new-line character
    //will not affect the string read.
    while (fgets(buf, 1024, file))
    {
        if (strstr (buf, "[call]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[call]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';

```

```

        pos = 0;
        token = strtok( tmpStr, seps );
        while( token != NULL )
        {
            switch (pos)
            {
                case 0: //Getting the name of the button

#ifdef DEBUG_FLAG_L1
                ev << "[sRole] >> Button Name " << token << '\n';
#endif

                strcpy(tmpPattern.cBtnName, token);
                pos++;
                break;

                case 1: //Getting the probability of call

#ifdef DEBUG_FLAG_L1
                ev << "[sRole] >> iProbCall " << token << '\n';
#endif

                tmpPattern.iProbCall = atoi(token);
                break;
            }
            token = strtok( NULL, seps );
        }

        if
        {
            callPattern[callPatternCount] = tmpPattern;
            simEvent.addEvent(callPatternCount, tmpPattern.iProbCall);
            callPatternCount = callPatternCount + 1;
        }
    }
    else if (strstr (buf, "[type]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[type]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        roleType = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sRole] >> Minimum wait " << minWait << '\n';
#endif
    }
    else if (strstr (buf, "[min_wait]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[min_wait]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        minWait = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sRole] >> Minimum wait " << minWait << '\n';
#endif
    }
    else if (strstr (buf, "[max_wait]") != NULL)
    {
        strcpy(tmpStr, buf+strlen("[max_wait]")+1);
        tmpStr[strlen(tmpStr)-1] = '\0';
        maxWait = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
        ev << "[sRole] >> Maximum wait " << maxWait << '\n';
#endif
    }
}

```

```

#endif
        }
        else if (strstr (buf, "[dr_sw]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[dr_sw]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';
            dataRate2Sw = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
            ev << "[sRole] >> Data rate to switch "<< dataRate2Sw << '\n';
#endif
        }
        else if (strstr (buf, "[btw_call]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[btw_call]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';
            betweenCallTime = atoi(tmpStr);
#ifdef DEBUG_FLAG_L1
            ev << "[sRole] >> Time between call "<< betweenCallTime << '\n';
#endif
        }
        else if (strstr (buf, "[name]") != NULL)
        {
            strcpy(tmpStr, buf+strlen("[name]")+1);
            tmpStr[strlen(tmpStr)-1] = '\0';
            strcpy(roleName, tmpStr);
#ifdef DEBUG_FLAG_L1
            ev << "[sRole] >> Role name "<< roleName << '\n';
#endif
        }
    }

    if( fclose( file ) )
        printf( "The file '%s' was not closed\n", defFile);

    return false;
}

#endif

```

L. SSWITCH.CPP

```

////////////////////////////////////
// file: switch.cpp
////////////////////////////////////

#include <string.h>
#include "omnetpp.h"
#include "global.h"
#include "dnssvc.h"

class sSwitch : public cSimpleModule
{
    Module_Class_Members(sSwitch,cSimpleModule,16384)
    virtual void finish();

```

```

virtual void activity();
virtual void initialize();

protected:
    bool isMachineGateId(int id);
    bool isRoleGateId(int id);

    int sw2min_sz;
    int sw2min_id[MAX_GATE_SZ];
    cGate *sw2min_gt;

    int sw2rin_id[MAX_GATE_SZ];
    int sw2rin_sz;
    cGate *sw2rin_gt;

    cGate *sw2mout_gt;

    CDnsSvc dnsObjSvrSvc;
    CDnsSvc dnsRoleSvc;
};

Define_Module( sSwitch );

//initialize method called when the object is first initialize.
void sSwitch::initialize()
{
    //initialize the list gateid
    sw2min_gt = gate("sw2m_in");
    sw2rin_gt = gate("sw2r_in");
    sw2mout_gt = gate("sw2m_out");

    int sw2mout_sz;

    //initializing the list of switch to server gate id
    sw2min_sz = sw2min_gt->size();
    if (sw2min_sz > MAX_GATE_SZ)
    {
        ev << "[!! sSwitch] >> Maximum number of gate defined (sw2min_sz) " <<
sw2min_sz << '\n';
        sw2min_sz = MAX_GATE_SZ;
    }

    for (int i=0; i<sw2min_sz; i++)
    {
        cGate *tmp = gate("sw2m_in", i);
        sw2min_id[i] = tmp->id();
#ifdef DEBUG_FLAG_L1
        ev << "[sSwitch] >> Server (i) " << i << " (gateid) " << tmp->id() << '\n';
#endif
    }

    //initializing the list of switch to client gate id
    sw2rin_sz = sw2rin_gt->size();
    if (sw2rin_sz > MAX_GATE_SZ)
    {

```

```

        ev << "[!! sSwitch] >> Maximum number of gate defined (sz_sw2r) " <<
sw2rin_sz << '\n';
        sw2rin_sz = MAX_GATE_SZ;
    }

    for (i=0; i<sw2rin_sz; i++)
    {
        cGate *tmp = gate("sw2r_in", i);
        sw2rin_id[i] = tmp->id();
#ifdef DEBUG_FLAG_L1
        ev << "[sSwitch] >> Client (i) " << i << " (gateid) " << tmp->id() << '\n';
#endif
    }

    //initializing the list of switch to client out gate id
    sw2mout_sz = sw2mout_gt->size();
    if (sw2mout_sz > MAX_GATE_SZ)
    {
        ev << "[!! sSwitch] >> Maximum number of gate defined (sw2mout_sz) " <<
sw2mout_sz << '\n';
        sw2mout_sz = MAX_GATE_SZ;
    }

    for (i=0; i<sw2mout_sz; i++)
    {
        cGate *tmp = gate("sw2m_out", i);
#ifdef DEBUG_FLAG_L1
        ev << "[sSwitch] >> Server out (i) " << i << " (gateid) " << tmp->id() << '\n';
#endif
    }

#ifdef DEBUG_FLAG_L1
    ev << "***** [sSwitch] *****" << '\n';
#endif
}

//To handle the finish event
void sSwitch::finish()
{
    delete(sw2min_gt);
    delete(sw2rin_gt);
    delete(sw2mout_gt);
}

//Method to handle activity event call
void sSwitch::activity()
{
    int type;
    cGate *rcv_gate;

    cPar cpObjSvrName, cpMethodName, cpMachineName, cpRequestorName, cpName;
    cMessage *sw_msg;
    int ingate, outgate, index, requestId;
    int inport, outport, msgSz;
    double roleTime, tmpDouble;

```

```

        char tmpStr1[MAX_NAME_SZ], tmpStr2[MAX_NAME_SZ];

    for(;;)
    {
        //Listen for incoming message
        cMessage *rcv_msg = receive();

        type = rcv_msg->kind(); //Determine the type of the message.
        rcv_gate = rcv_msg->arrivalGate(); //Determine the gate type

        //If the message is from a machine gate.
        if (isMachineGateId(rcv_gate->id()))
        {
            //if the message is from a server_in gate id.
#ifdef DEBUG_FLAG_L2
            ev << "[sSwitch] >> Received server message (type) " << type << " from sw2s_in" << "\n";
#endif
            switch( type )
            {
                case NAME_REGISTER:
                    //DNS Register message to be sent over to the DNS
server
                    cpMachineName = rcv_msg->par("add_machine");
//Server machine name
                    cpName = rcv_msg->par("add_name"); //Object server
machine name
                    delete rcv_msg;

                    //Extract the string varlu of the machine and object
server name
                    strcpy(tmpStr1, cpMachineName.stringValue ());
                    strcpy(tmpStr2, cpName.stringValue ());

                    //Extract the gate id
                    ingate = rcv_gate->id();
                    index = rcv_gate->index();

                    outgate = gate("sw2m_out", index)->index();

                    //Mapping the port to the machine and object server
name.
                    dnsObjSvrSvc.MapSvcToPort(ingate, outgate, tmpStr1,
tmpStr2);

                    inport = dnsObjSvrSvc.FindInGateByName(tmpStr2);
                    outport = dnsObjSvrSvc.FindOutGateByName(tmpStr2);
break;

                    //Routing the server response
                    case OBJSVR_RESPONSE:
//The requestor address
                        cpRequestorName = rcv_msg->par("add_request");

                        requestId = rcv_msg->par("request_id");
                        msgSz = rcv_msg->par("msgSz"); //Msg Size
                        delete rcv_msg;
                        ev << "[sSwitch] SERVER_RESPONSE to " << cpRequestorName << "\n";

```

```

#ifdef DEBUG_FLAG_L2
    ev << "[sSwitch] SERVER_RESPONSE (client_id) " << client_id << "\n";
#endif

    strcpy(tmpStr1, cpRequestorName.stringValue ());

    OBJSVR_RESPONSE );

    sw_msg = new cMessage( "OBJSVR_RESPONSE",

    sw_msg->addPar("add_request") = cpRequestorName;
    sw_msg->addPar("request_id") = requestId;
    sw_msg->setLength(msgSz);

    //check whether the destination is in the role id
    outport = dnsRoleSvc.FindOutGateByName(tmpStr1);
    if (outport != -1)
    {
        //destination belongs to the role
        send( sw_msg, "sw2r_out", outport);
    }

    outport = dnsObjSvrSvc.FindOutGateByName(tmpStr1);
    if (outport != -1)
    {
        //destination belongs to the role
        send( sw_msg, "sw2m_out", outport);
    }
    break;

    case INVOKE_OS2OS_CALL:
        cpObjSvrName = rcv_msg->par("add_os");
        cpMethodName = rcv_msg->par("add_method");

#ifdef DEBUG_FLAG_L2
    ev << "[sSwitch] INVOKE_OS2OS_CALL (obj) " << cpObjSvrName << " (method) " <<
    cpMethodName << "\n";
#endif

    //Extract the string varlu of the machine and object
    server name

    strcpy(tmpStr1, cpObjSvrName.stringValue ());
    strcpy(tmpStr2, cpMethodName.stringValue ());

    outport = dnsObjSvrSvc.FindOutGateByName(tmpStr1);

    //Send the port to the relevant out port.
    if (outport != -1)
        send( rcv_msg, "sw2m_out", outport);
    break;
}

//If the message is from a role gate.
else if (isRoleGateId(rcv_gate->id()))
{
#ifdef DEBUG_FLAG_L2
    ev << "[sSwitch] >> Received client message (type) " << type << " from sw2s_in" << "\n";
#endif
}

```

```

switch( type )
{
    //Invoke object server call message received.
    case INVOKE_OBJECT_SVR_CALL:
        cpObjSvrName = rcv_msg->par("add_os");
        cpMethodName = rcv_msg->par("add_method");
        cpRequestorName = rcv_msg->par("add_request");
        roleTime = rcv_msg->par("roleTime"); //Msg Size

        tmpDouble = simTime()-roleTime;
        ev << "[sSwitch] receive from " << cpRequestorName << "\n";

#ifdef DEBUG_FLAG_L2
        ev << "[sSwitch] (obj) " << objSvrName << " (method) " << methodName << "\n";
#endif

server name                                //Extract the string varlu of the machine and object

        strcpy(tmpStr1, cpObjSvrName.stringValue ());
        strcpy(tmpStr2, cpMethodName.stringValue ());

        outport = dnsObjSvrSvc.FindOutGateByName(tmpStr1);

        //Send the port to the relevant out port.
        if (outport != -1)
            send( rcv_msg, "sw2m_out", outport);
        break;

    case NAME_REGISTER:
        //DNS Register message to be sent over to the DNS
server
        cpMachineName = rcv_msg->par("add_machine");
//Server machine name
        cpName = rcv_msg->par("add_name"); //Object server
machine name
        delete rcv_msg;

        //Extract the string varlu of the machine and object
server name

        strcpy(tmpStr1, cpMachineName.stringValue ());
        strcpy(tmpStr2, cpName.stringValue ());

        //Extract the gate id
        ingate = rcv_gate->id();
        index = rcv_gate->index();
        outgate = gate("sw2r_out", index)->index();

        //Mapping the port to the machine and object server
name.
        dnsRoleSvc.MapSvcToPort(ingate, outgate, tmpStr1,
tmpStr2);

        inport = dnsRoleSvc.FindInGateByName(tmpStr2);
        outport = dnsRoleSvc.FindOutGateByName(tmpStr2);
        break;

```

```

    }
}

}

}

//*****
/* Protected Method
//*****
//Method to check whether the gate id is a server gate id.
bool sSwitch::isMachineGateId(int id)
{
    for (int i=0; i<sw2min_sz; i++)
    {
        if (sw2min_id[i] == id)
            return true;
    }

    return false;
}

//Method to check whether the gate id is a client gate id.
bool sSwitch::isRoleGateId(int id)
{
    for (int i=0; i<sw2rin_sz; i++)
    {
        if (sw2rin_id[i] == id)
            return true;
    }

    return false;
}

```

M. STATLOG.CPP

```

// StatLog.cpp: implementation of the CStatLog class.
//
//

```

```

#include <stdio.h>
#include <string.h>

```

```

#include "omnetpp.h"
#include "StatLog.h"

```

```

//
// Construction/Destruction
//
static CStatLog *m_StatLog = NULL;
CStatLog *CStatLog::GetInstance()
{
    if (m_StatLog==NULL)
    {
        m_StatLog = new CStatLog();
        m_StatLog->initialize();
    }
}

```

```

    }
    return m_StatLog;
}

CStatLog::CStatLog()
{
}

CStatLog::~CStatLog()
{
}

//To initialize the StatLog.
bool CStatLog::initialize()
{
    for (int i=0; i<MAX_ROLETYPE; i++)
    {
        meanRoleType[i] = 0.0;
        countRoleType[i] = 0;
        maxRoleType[i] = 0;
    }

    return true;
}

//To get an object using the object name
bool CStatLog::registerRoleType(int type)
{
    maxRoleType[type] = maxRoleType[type] + 1;
    return true;
}

//To get an object using the object name
bool CStatLog::writeLog(char *name, char *role, int type, double sz, double smallest, double
largest, double mean)
{
    char str[500];

    FILE *stream;

    if( (stream = fopen( "stat.log", "a+" )) == NULL )
        printf( "The file 'stat.log' was not opened\n" );

    //    sprintf(str, "[Sz\\Min\\Max\\Ave]");
    //    fprintf(stream, str);

    sprintf(str, "%s %s \t%.4f \t%.4f \t%.4f \t%.4f\n", name, role, sz, smallest, largest, mean);
    fprintf(stream, str);

    /* Close stream */
    if( fclose( stream ) )
        printf( "The file 'data' was not closed\n" );

    countRoleType[type] = countRoleType[type] + 1;

```

```

        meanRoleType[type] = meanRoleType[type] + mean;

        if (countRoleType[type] == maxRoleType[type])
        {
            if( (stream = fopen( "computed.log", "a+" )) == NULL )
                printf( "The file 'computed.log' was not opened\n" );

            double computedMean = (double)(meanRoleType[type]/countRoleType[type]);

            sprintf(str, "[Type\\Sz\\Total Mean\\Mean] %s %s \\t%d \\t%.4f \\t%.4f\\n", name,
role, countRoleType[type], meanRoleType[type], computedMean);
            fprintf(stream, str);

            /* Close stream */
            if( fclose( stream ) )
                printf( "The file 'computed.log' was not closed\n" );

            meanRoleType[type] = 0.0;
            countRoleType[type] = 0;
            maxRoleType[type] = 0;
        }
        return true;
    }

bool CStatLog::writeLog(char *str)
{
    FILE *stream;

    if( (stream = fopen( "stat.log", "a+" )) == NULL )
        printf( "The file 'stat.log' was not opened\n" );

    fprintf(stream, str);

    /* Close stream */
    if( fclose( stream ) )
        printf( "The file 'stat.log' was not closed\n" );

    return true;
}

bool CStatLog::writeLogWithRamError(char *name, double sz, double cur)
{
    FILE *stream;
    char str[500];

    if( (stream = fopen( "computed.log", "a+" )) == NULL )
        printf( "The file 'computed.log' was not opened\n" );

    sprintf(str, "[Type\\Sz\\Total Mean\\Mean] %s -- RAM ERROR (Sz)%.2f (Cur)%.2f \\n",
name, sz, cur);
    fprintf(stream, str);

    /* Close stream */
    if( fclose( stream ) )
        printf( "The file 'computed.log' was not closed\n" );
}

```

```

        return true;
    }

```

N. DISOBJSVRSIM.NED NETWORK DESCRIPTION FILE

```

//-----
// file: DisObjSvrSim.ned
//-----

//*****
// Hardware Modelled
//*****

// Role --
//
// A client computer which periodically connects to the
// server for data exchange. In future, role based client should
// be implemented for the thesis.
//
simple sRole
    parameters:
        def_file : string;
    gates:
        //switch connection port
        out: r2sw_out;
        in: r2sw_in;
endsimple

// Switch --
//
// A very simple module which models the network between
// the servers and clients
//
simple sSwitch
    gates:
        out: sw2r_out[];
        in: sw2r_in[];
        out: sw2m_out[];
        in: sw2m_in[];
        in: sw2d_in;
endsimple

// Server --
//
// Models a server which accepts connections from the
// client computers. It serves multiple connections of
// object servers.
//
simple sMachine
    parameters:
        num_objsvr : numeric,
        objsvridx : numeric,
        def_file : string;

    gates:

```

```

        //switch connection port to the s
        out: m2sw_out;
        in: m2sw_in;

        //object server connection port to the object server app
        out: m2os_out[];
        in: m2os_in[];
endsimple

// ObjSvrApp --
//
// Models a client application
//
simple sObjSvr
    parameters:
        cfg_file : string;

    gates:
        //object server application connection port to the server
        out: os2m_out;
        in: os2m_in;
endsimple

// DisObjSvrSim --
//
//
//
module DisObjSvrSim
    //*****
    // Parameters
    //*****
    parameters:
        run_name : string,           //The name of the run

        num_machine : numeric,       //Number of server
        num_role : numeric,          //Total number of roles in the system
        num_objsvr : numeric,        //Total number of object server in the system
        max_objsvr : numeric;        //Max number of object server per server

    //*****
    // Submodules
    //*****
    submodules:
        smachine: sMachine[num_machine];
        gatesizes:
            m2os_out[max_objsvr],
            m2os_in[max_objsvr];
            display: "p=95,250,row,50;i=proc1;b=36,32";

        role: sRole[num_role];
            display: "p=95,450,row,50;i=proc1;b=36,32";

        switch: sSwitch;
        gatesizes:
            sw2m_out[num_machine],
            sw2m_in[num_machine],

```

```

        sw2r_out[num_role],
        sw2r_in[num_role],
        sw2d_in;
        display: "p=155,350,row,50;i=cloud;b=32,32";

    objsvr: sObjSvr[num_objsvr];
        display: "p=51,150,row,50;i=proc2;b=32,32";

//*****
// Connections
//*****
connections nocheck:
    //Connections between server and switch
    for i=0..num_role-1 do
        role[i].r2sw_out --> delay 10ms --> switch.sw2r_in[i];
        role[i].r2sw_in <-- delay 10ms <-- switch.sw2r_out[i];
    endfor;

    for i=0..num_machine-1 do
        smachine[i].m2sw_out --> delay 10ms --> switch.sw2m_in[i];
        smachine[i].m2sw_in <-- delay 10ms <-- switch.sw2m_out[i];
    endfor;

    for i=0..num_machine-1,
    j=smachine[i].objsvridx+0..smachine[i].objsvridx+smachine[i].num_objsvr-1 do
        smachine[i].m2os_out[j-smachine[i].objsvridx] --> delay 10ms --> objsvr[j].os2m_in;
        smachine[i].m2os_in[j-smachine[i].objsvridx] <-- delay 10ms <-- objsvr[j].os2m_out;
    endfor;

endmodule

// theDisObjSvrSim --
//
// Instantiates the DisObjSvrSim network
//
//network theDisObjSvrSim : DisObjSvrSim
network model : DisObjSvrSim
    parameters:
        // ... (parameter assignments)
endnetwork

```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B - SOURCE CODE OF PATTERN GENERATOR

A. PATTERNGENERATOR.CPP

```
// PatternGenerator.cpp : Defines the entry point for the console application.  
//
```

```
#include "stdafx.h"  
#include <stdlib.h>  
#include <stdio.h>  
#include <string.h>  
#include <math.h>  
  
const int MAX_MACHINEINFO = 20;  
const int MAX_ROLE = 20;  
const int MAX_OBJSVR = 20;  
const int MAX_PATTERN = 1000;  
const int MAX_STRING_SZ = 100;  
  
char newBaseFile[MAX_STRING_SZ];  
char baseFile[MAX_STRING_SZ];  
char machineName[MAX_MACHINEINFO][MAX_STRING_SZ];  
char objsvrName[MAX_OBJSVR][MAX_STRING_SZ];  
  
int numRoleType = 0;  
char roleName[MAX_ROLE][MAX_STRING_SZ];  
char roleBaseFile[MAX_ROLE][MAX_STRING_SZ];  
int roleCount[MAX_ROLE];  
  
int patternCount = 0;  
int fileCount = 0;  
  
struct sMACHINEINFO  
{  
    char name[MAX_STRING_SZ];  
    int numOfObj;  
    int objIndex[MAX_OBJSVR];  
    int indexOfObj;  
} machinfo;  
  
sMACHINEINFO machineInfo[MAX_MACHINEINFO];  
int machineInfoCount;  
int objSvrIndex;  
  
//To initialize the machine info array for future storage.  
//The maximum number of machine info is set to 20.  
void initMachineInfo()  
{  
    for (int i=0; i<MAX_MACHINEINFO; i++)  
    {  
        strcpy(machineInfo[i].name, "");  
        machineInfo[i].numOfObj = 0;  
        machineInfo[i].indexOfObj = 0;
```

```

        for (int j=0; j<MAX_OBJSVR; j++)
            machineInfo[i].objIndex[j] = -1;
    }

    machineInfoCount = 0;
}

//To get the number of bits needed to represent the machine.
//For example, if there is 5 machines, we will need 101 i.e. 3 bits
//to represent 5 machines.
int getNumOfBits(int machine)
{
    unsigned int tmpNum1 = 0;
    unsigned int tmpNum2 = 0;
    int counter = 0;

    tmpNum1 = (unsigned int)machine;

    //To shift the bit pattern until the number
    //becomes zero. The counter will represent the number of bits
    //needed.
    while (tmpNum1 != 0)
    {
        tmpNum1 = tmpNum1 >> 1;
        counter = counter + 1;
    }

    return counter;
}

//To get the filter needed for the bits pattern.
//For example, if for 5 machines, base=3 and the filter pattern
//should be 111.
unsigned int getFilter(int base)
{
    unsigned int tmpNum1 = 0;

    //For the number of bit, compute the filter.
    for (int i=0; i<base; i++)
    {
        tmpNum1 = tmpNum1 + pow(2, i);
    }
    return tmpNum1;
}

//To get the bits string to represent the machine and objsvr
unsigned int getBitsString(int base, int machine, int objsvr)
{
    unsigned int tmpNum1 = 0;
    unsigned int tmpNum2 = 0;
    int counter = 0;

    tmpNum1 = (unsigned int)machine;

    while (counter != objsvr)

```

```

    {
        //shift the bit to the left by base length.
        //For example if base=2, tmpNum2=0, the result will be 11
        tmpNum2 = tmpNum2 << base;

        //Forcing an OR sting will map the value to the appropriate value
        //For example, if the machine is 5(101), ie base=3. Having the initial
        //value as 111 will introduce additional processing time that is not necessary
        //Forcing it to 101 will help to reduce unnecessary processing.
        tmpNum2 = tmpNum2 | tmpNum1;
        counter = counter + 1;
    }

    return tmpNum2;
}

```

//To get the bits string to represent the machine and objsvr
 unsigned int getTerminateString(int base, int machine, int objsvr)

```

{
    unsigned int tmpNum1 = 0;
    unsigned int tmpNum2 = 0;
    int counter = 0;

    tmpNum1 = (unsigned int)machine;

    while (counter != objsvr)
    {
        //shift the bit to the left by base length.
        //For example if base=2, tmpNum2=0, the result will be 11
        tmpNum2 = tmpNum2 << base;
        counter = counter + 1;
    }

    return tmpNum2;
}

```

//To check whether the terminate string is reached.
 bool checkFinalString(unsigned int terminateString, unsigned int string, unsigned int filter)

```

{
    if (terminateString > string)
        return true;

    return false;
}

```

//To check whether the string is valid. This method will break up the
 //string pattern into blocks of bit in the same size as the base.
 //For example, 11111111 is broken in 4 blocks of 2 bits each.
 bool checkString(unsigned int string, int base, int machine, int objsvr, unsigned int filter)

```

{
    unsigned int tmpNum1 = 0;
    unsigned int tmpNum2 = 0;
    unsigned int tmpNum3 = 0;
    int counter = 0;

    tmpNum1 = string;

```

```

tmpNum3 = (unsigned int)machine;

//check the length first.

while (counter != objsvr)
{
    //Perform an AND operation.
    //For example 11101010 AND 11 will return 10.
    tmpNum2 = tmpNum1 & filter;

    //if the extracted block is 0 or greater than the number of machine
    //then it is a infeasible pattern. Ignore the pattern.
    if ((tmpNum2 == 0) || (tmpNum2 > machine))
    {
        return false;
    }

    //shift right by base bit.
    tmpNum1 = tmpNum1 >> base;
    counter = counter + 1;
}

return true;
}

//To print the role file content.
// Example : simR1(2).ini
// [Run 1] #0
// model.run_name = "pat1" #1
// model.num_machine = 3 #2
// model.num_objsvr = 3 #3
// model.max_objsvr = 3 #4
// model.smachine[2].def_file = "SIX.def" #5
// model.smachine[2].num_objsvr = 3 #6
// model.smachine[2].objsvridx = 0 #7
// model.objsvr[0].cfg_file = "A.def" #8
// model.objsvr[1].cfg_file = "B.def" #9
// model.objsvr[2].cfg_file = "C.def" #10
bool printPatternString(unsigned int string, int count, int base, int machine, int objsvr, unsigned int
filter)
{
    FILE *in_stream;
    FILE *out_stream;

    unsigned int tmpNum1 = 0;
    unsigned int tmpNum2 = 0;
    unsigned int tmpNum3 = 0;
    int counter = 0;

    tmpNum1 = string;
    tmpNum3 = (unsigned int)machine;
    char tmpStr2[200] = "";
    char tmpStr1[20] = "";
    char tmpStr3[200] = "";

    bool found = false;

```

```

bool breakFile = false;
int maxNumObjSvr = 0;

//The purpose of these code is to break the initiation file
//into block of 1000 call pattern each. These will allow
//smaller file for easy management. To enable the features
//set the flag breakFile to true.
if ((patternCount > MAX_PATTERN) && (breakFile))
{
    fileCount = fileCount + 1;
    sprintf(newBaseFile, "%s_%d.cfg", baseFile, fileCount);
}
else
{
    sprintf(newBaseFile, "%s.cfg", baseFile);
}

//open the pattern file for writing.
if( (out_stream = fopen( newBaseFile, "a+" )) == NULL )
    return 0;

//initialize the machine info block
initMachineInfo();

//Initialize the machine information with the machine name.
for (int j=0; j<machine; j++)
{
    strcpy(machineInfo[j].name, &machineName[j][0]);
    machineInfo[j].numOfObj = 0;
    machineInfo[j].indexOfObj = 0;
}

//Write the information for each object server.
while (counter != objsvr)
{
    found = false;
    tmpNum2 = tmpNum1 & filter;

    //getting the name of the machine
    sprintf(tmpStr1, "%s", &machineName[tmpNum2-1][0]);

    for (int i=0; i<machine; i++)
    {
        //compare the machine name is it is equal than
        //increase the numOfObj by one.
        if (strcmp(machineInfo[i].name, tmpStr1) == 0)
        {
            //found matching string that is already in the machineInfo
            //store the index which is the counter.
            machineInfo[i].objIndex[machineInfo[i].numOfObj] = counter;
            machineInfo[i].numOfObj = machineInfo[i].numOfObj + 1;

            //to determine the max number of object server allocated to any
            machine.

            if (maxNumObjSvr < machineInfo[i].numOfObj)
                maxNumObjSvr = machineInfo[i].numOfObj;
        }
    }
}

```

```

        found = true;
    }
}

sprintf(tmpStr3, "[%s] ", tmpStr1);
strcat(tmpStr2, tmpStr3);
tmpNum1 = tmpNum1 >> base;
counter = counter + 1;
}

objSvrIndex = 0;

//write comment line
sprintf(tmpStr3, "## pat%d %s ##\n", count+1, tmpStr2);
fprintf(out_stream, tmpStr3);
printf(tmpStr3);

//write line #0
sprintf(tmpStr3, "[Run %d]\n", count+1);
fprintf(out_stream, tmpStr3);

//write line #1
sprintf(tmpStr2, "model.run_name = \"%pat%d\"\n", count+1);
fprintf(out_stream, tmpStr2);

//write line #2
sprintf(tmpStr2, "model.num_machine = %d\n", machine);
fprintf(out_stream, tmpStr2);

//write line #3
sprintf(tmpStr2, "model.num_objsvr = %d\n", objsvr);
fprintf(out_stream, tmpStr2);

//write line #4
sprintf(tmpStr2, "model.max_objsvr = %d\n", maxNumObjSvr);
fprintf(out_stream, tmpStr2);

//write line #5-10
for(int k=0; k<machine; k++)
{
    //write line #5
    sprintf(tmpStr2, "model.smachine[%d].def_file = \"%s.def\"\n", k,
machineInfo[k].name);
    fprintf(out_stream, tmpStr2);

    //write line #6
    sprintf(tmpStr2, "model.smachine[%d].num_objsvr = %d\n", k,
machineInfo[k].numOfObj);
    fprintf(out_stream, tmpStr2);

    //write line #7
    if (machineInfo[k].numOfObj != 0)
    {
        sprintf(tmpStr2, "model.smachine[%d].objsvridx = %d\n", k, objSvrIndex);
        fprintf(out_stream, tmpStr2);
    }
}

```

```

    }
    else
    {
        sprintf(tmpStr2, "model.smachine[%d].objsvridx = %d\n", k, 0);
        fprintf(out_stream, tmpStr2);
    }

    //write line #8-10
    for (int m=0; m<machineInfo[k].numOfObj; m++)
    {
        sprintf(tmpStr2, "model.objsvr[%d].cfg_file = \"%s.def\"\n",
m+objSvrIndex, objsvrName[machineInfo[k].objIndex[m]]);
        fprintf(out_stream, tmpStr2);
    }

    objSvrIndex = objSvrIndex + machineInfo[k].numOfObj;
    fprintf(out_stream, "\n");
}

fclose(out_stream);
patternCount = patternCount + 1;
return true;
}

```

```

//To print the role file content.
// Example : role.ini
// model.role[0].def_file = "role0_0.def"           #1
// model.role[1].def_file = "role0_1.def"
// model.role[2].def_file = "role0_2.def"
// model.role[3].def_file = "role0_3.def"
// model.num_role = 4                               #2
//
// Example : RoleA_0.def
// [name] ROLEA_0                                   #1
// [min_wait] 10                                    #2
// [max_wait] 11                                    #3
// [call] F1.B1/50                                  #4
// [call] F1.B2/1                                    #5
// [call] F2.B1/1                                    #6
// [call] F2.B4/5                                    #7
// [dr_sw] 200000000                                #8
// [btw_call] 0.05                                   #9

```

```

bool printRoleFile()
{
    FILE *in_stream;
    FILE *out_stream, *out_stream2;

    int roleIndex = 0;
    int totalRole = 0;

    char tmpStr[200] = "";
    char curFile[200] = "";
    char roleUName[200] = "";

    //To open the baseFile.

```

```

if( (out_stream = fopen( baseFile, "a+" )) == NULL )
    return 0;

//printing the [Parameters] header, this is the format
//needed by the omnet.ini file.
sprintf(tmpStr, "[Parameters]\n");
fprintf(out_stream, tmpStr);

//Generate the file for each role type
while (roleIndex != numRoleType)
{
    //opening the role base file for duplicating
    if( (in_stream = fopen( roleBaseFile[roleIndex], "r+" )) == NULL )
        return 0;

    //generate a role file for each
    for (int i=0; i<roleCount[roleIndex]; i++)
    {
        //creating the role configuration file.
        sprintf(tmpStr, "model.role[%d].def_file = \"%s_%d.def\"\n", totalRole,
roleName[roleIndex], i);
        fprintf(out_stream, tmpStr);
        totalRole = totalRole + 1;

        //creating a configuration file for each role defined.
        fseek(in_stream, 0, SEEK_SET);
        fgets(tmpStr, 100, in_stream);

        //printing a unique role name for each of the role defined.
        //the role name is of the format <RoleName>_<Index of this role>
        strcpy(roleUName, roleName[roleIndex]);

        //make the role name into upper case.
        _strupr(roleUName);

        //creating a filename roleA_1.def
        sprintf(curFile, "%s_%d.def", roleName[roleIndex], i);
        if( (out_stream2 = fopen( curFile, "w+" )) == NULL )
            return 0;
        printf("Writing %s...\n", curFile);

        //print line #1
        sprintf(tmpStr, "[name] %s_%d\n", roleUName, i);
        fprintf(out_stream2, tmpStr);

        //copying from the base file to the new role file name.
        while (fgets(tmpStr, 100, in_stream) != NULL)
        {
            //copying line #2-9 from the role base file into the new file
            fprintf(out_stream2, tmpStr);
        }

        //close the new file
        fclose(out_stream2);
    }
    //close the file only the number of files are generated.
}

```

```

        fclose(in_stream);
        roleIndex = roleIndex + 1;
    }

    //printing the line #2 for role.ini
    sprintf(tmpStr, "model.num_role = %d\n", totalRole);
    fprintf(out_stream, tmpStr);

    //close the role file.
    fclose(out_stream);

    return true;
}

//a. To prompt the user for input that is needed to generate the pattern file
//b. Generate the pattern file
void generatePatternFile()
{
    char tmpStr[100];
    char curFile[100];
    char roleUName[100];

    int numMachine = 0;
    int numObjSvr = 0;
    int numBaseBit = 0;
    int numValidString = 0;

    unsigned int bitsString = 0;
    unsigned int oriString = 0;
    unsigned int filter = 0;
    unsigned int terminateString = 0;

    bool valid = false;
    bool final = false;

    //Enter the new patten file name
    printf("Please enter the base file > ");
    gets( baseFile );
    sprintf(newBaseFile, "%s_%d.ini", baseFile, fileCount);

    //Enter the number of machine
    printf("Please enter the number of machine > ");
    gets( tmpStr );
    numMachine = atoi(tmpStr);

    //For each of the machine, prompt for the machine name.
    for (int i=0; i<numMachine; i++)
    {
        printf("\tPlease enter the name of machine > ");
        gets( tmpStr );
        strcpy(&machineName[i][0], tmpStr);
    }

    //Enter the number of object server
    printf("Please enter the number of object server > ");
    gets( tmpStr );

```

```

numObjSvr = atoi(tmpStr);

//For each of the object server, prompt for the object server name
for (int j=0; j<numObjSvr; j++)
{
    printf("\tPlease enter the name of object server > ");
    gets( tmpStr );
    strcpy(&objsvrName[j][0], tmpStr);
}

//For example the scenario of 3 machines[A,B,C] and 4 objsvrs.
//Each machine can be represented as two bits i.e. 11. A bit string is created such as
11111111
//So 11111111 will means that the 4 objsvrs are run in the 3rd machine.
//
//For each iteration, the bitstring is subtract by 1. And the deployment pattern is
//computed again.
//1st pass : 11111111 > {C}{C}{C}{C}
//2nd pass : 11111110 > {C}{C}{C}{B}
//3rd pass : 11111101 > {C}{C}{C}{A}
//4th pass : 11111100 > No feasible, ignore.
//.....
//nth pass : 00111111 > Terminating string, since any decrease will have no effect.

//To find the number of bits needed to represent the machine
numBaseBit = getNumOfBits(numMachine);

//To get the filter pattern with the bits pattern.
filter = getFilter(numBaseBit);

//To find the bits string to represent all the machine.
bitsString = getBitsString(numBaseBit, numMachine, numObjSvr);

//To find the determining the terminating string. For example, if there
//is 4 objsvr, the terminate String is 00111111 = 63.
terminateString = getTerminateString(numBaseBit, numMachine, numObjSvr-1);

//Copy the string to oristring.
oriString = bitsString;

while ((bitsString != 0) && (!final))
{
    //To check whether the final string has occurred.
    final = checkFinalString(terminateString, bitsString, filter);

    //if the value has not reach the terminating string.
    if (!final)
    {
        //check whether the string is a valid string, in the previous example
        //0 is a non feasible solution and it will be ignore.
        valid = checkString(bitsString, numBaseBit, numMachine, numObjSvr,
filter);

        if (valid)
        {

```

```

        //try printing out the value
        printPatternString(bitsString, numValidString, numBaseBit,
numMachine, numObjSvr, filter);
        numValidString = numValidString + 1;
    }
}

//subtract the value by one.
bitsString = bitsString - 1;
}

printf("Number of Valid String >> %d\n", numValidString);
}

//a. To prompt the user for input that is needed to generate the role file
//b. Generate the role file
void generateRoleFile()
{
    char tmpStr[100];

    //Enter the role configuration file name
    printf("Please enter the new role configuration filename > ");
    gets( baseFile );

    //Enter the number of role type. E.g. if there is
    //RoleA, RoleB, RoleC, the number of role type is 3
    printf("Please enter the number of role type > ");
    gets( tmpStr );
    numRoleType = atoi(tmpStr);

    //For each role type.
    for (int k=0; k<numRoleType; k++)
    {
        //Enter the name of the role.
        printf("\tPlease enter the name of role %d > ", k);
        gets( tmpStr );
        strcpy(&roleName[k][0], tmpStr);

        //Enter the base filename. The base filename will be
        //used to generate the role configuration file.
        //For example, if there is 10 RoleA, then there will
        //be 10 files generated from roleA_0.def to roleA_9.def
        printf("\tPlease enter the base file name > ");
        gets( tmpStr );
        strcpy(&roleBaseFile[k][0], tmpStr);

        //Enter the number of this role type.
        printf("\tPlease enter the number of role >");
        gets( tmpStr );
        roleCount[k] = atoi(tmpStr);
    }

    //To print the role file.
    printRoleFile();
}

```

```

int main(int argc, char* argv[])
{
    bool finish = false;
    char tmpStr[10];
    int choice = 9;

    //Prompt for a choice and if choice is 9, exit the program.
    while (!finish)
    {
        printf("Please select file to generate > \n");
        printf("1. Generate call pattern file. \n");
        printf("2. Generate role configuration file. \n");
        printf("3. Reserved. \n");
        printf("9. Exit \n");
        gets(tmpStr);

        choice = atoi(tmpStr);
        switch (choice)
        {
            //To generate the pattern file.
            case 1: generatePatternFile(); break;

            //To generate the role file.
            case 2: generateRoleFile(); break;

            //To exit the program.
            case 9: finish = true; break;
        }
    }

    return 0;
}

```

APPENDIX C –VERIFICATION EXPERIMENT 1

A. CONFIGURATION FILES

Object server A – *A.def*

```
[name] A  
[ram_util] 44000  
[object] A  
[num_obj] 1
```

Object server B – *B.def*

```
[name] B  
[ram_util] 60000  
[object] B  
[num_obj] 1
```

Object server C – *C.def*

```
[name] C  
[ram_util] 66000  
[object] C  
[num_obj] 1
```

Role0 Configuration File – *role0.def*

```
[name] ROLE0  
[type] 0  
[min_wait] 15  
[max_wait] 16  
[call] C1.B1/50  
[call] C1.B2/1  
[call] C2.B1/1  
[call] C2.B6/1  
[dr_sw] 200000000  
[btw_call] 0.05
```

Role1 Configuration File – *role1.def*

```
[name] ROLE1  
[type] 1  
[min_wait] 6  
[max_wait] 7  
[call] C1.B1/10  
[call] C1.B2/40  
[call] C3.B2/24  
[dr_sw] 200000000  
[btw_call] 0.05
```

Role2 Configuration File – *role2.def*

```
[name] ROLE2
```

[type] 2
[min_wait] 4
[max_wait] 5
[call] C2.B5/50
[call] C2.B9/10
[call] C2.B3/30
[call] C2.B2/1
[call] C3.B2/1
[dr_sw] 200000000
[btw_call] 0.05

MACHINE SIX – *SIX.def*

[name] SIX
[cpu_pow] 600000
[ram_sz] 64000
[ram_limit] 0.0
[dr_sw] 200000000
[er_sw] 0.000000001
[process_call] 0.005
[process_swap] 0.005
[disk_swap] 0.005
[exec_time] 5.0
[ran_exec] 0.05
[time_slice] yes

MACHINE BR733 – *BR733.def*

[name] BR733
[cpu_pow] 733000
[ram_sz] 128000
[ram_limit] 0.0
[dr_sw] 200000000
[er_sw] 0.000000001
[process_call] 0.005
[process_swap] 0.005
[disk_swap] 0.005
[exec_time] 5.0
[ran_exec] 0.05
[time_slice] yes

MACHINE GIGA – *GIGA.def*

[name] GIGA
[cpu_pow] 1000000
[ram_sz] 128000
[ram_limit] 0.0
[dr_sw] 200000000
[er_sw] 0.000000001
[process_call] 0.005
[process_swap] 0.005
[disk_swap] 0.005
[exec_time] 5.0
[ran_exec] 0.05
[time_slice] yes

Button Configuration File – *button.def*

```
[NEW_DEF]
C1.B1
A/1
[END_DEF]
[NEW_DEF]
C1.B2
A/2
B/1
[END_DEF]
[NEW_DEF]
C2.B1
C/1
C/2
[END_DEF]
[NEW_DEF]
C2.B2
C/3
[END_DEF]
[NEW_DEF]
C2.B3
C/2
[END_DEF]
[NEW_DEF]
C2.B4
C/3
[END_DEF]
[NEW_DEF]
C2.B5
A/1
B/2
```

Interaction Configuration File – *interact.def*

```
[NEW_INTERACT] B/2
[CALL] C/1
[END_INTERACT]
```

Object Configuration File – *obj.def*

```
[NEW_OBJ_DEF]
A/10000
1/579600/112000
2/2620300/18400
3/1181750/44800
4/2026400/176000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
B/10000
1/1766550/4000000
2/3700850/2720000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
C/10000
```

```
1/3004300/320000
2/4804000/4000000
3/488150/400000
[END_OBJ_DEF]
```

Role Configuration File for Simulation1

```
[Parameters]
model.role[0].def_file = "role0.def"
model.num_role = 1
```

Role Configuration File for Simulation2

```
[Parameters]
model.role[0].def_file = "role1.def"
model.num_role = 1
```

Role Configuration File for Simulation3

```
[Parameters]
model.role[0].def_file = "role2.def"
model.num_role = 1
```

Role Configuration File for Simulation4

```
[Parameters]
model.role[0].def_file = "ROLE1_0.def"
model.role[1].def_file = "ROLE1_1.def"
model.role[2].def_file = "ROLE1_2.def"
model.role[3].def_file = "ROLE1_3.def"
model.num_role = 4
```

Role Configuration File for Simulation5

```
[Parameters]
model.role[0].def_file = "ROLE2_0.def"
model.role[1].def_file = "ROLE2_1.def"
model.role[2].def_file = "ROLE2_2.def"
model.num_role = 3
```

Role Configuration File for Simulation6

```
[Parameters]
theDisObjSvrSim.role[0].def_file = "role0_0.def"
theDisObjSvrSim.role[1].def_file = "role0_1.def"
theDisObjSvrSim.role[2].def_file = "role0_2.def"
theDisObjSvrSim.role[3].def_file = "role0_3.def"
theDisObjSvrSim.role[4].def_file = "role0_4.def"
theDisObjSvrSim.role[5].def_file = "role0_5.def"
theDisObjSvrSim.role[6].def_file = "role0_6.def"
theDisObjSvrSim.role[7].def_file = "role0_7.def"
theDisObjSvrSim.role[8].def_file = "role0_8.def"
theDisObjSvrSim.role[9].def_file = "role0_9.def"
theDisObjSvrSim.role[10].def_file = "role0_10.def"
```

```

theDisObjSvrSim.role[11].def_file = "role0_11.def"
theDisObjSvrSim.role[12].def_file = "role0_12.def"
theDisObjSvrSim.role[13].def_file = "role0_13.def"
theDisObjSvrSim.role[14].def_file = "role0_14.def"
theDisObjSvrSim.role[15].def_file = "role0_15.def"
theDisObjSvrSim.role[16].def_file = "role0_16.def"
theDisObjSvrSim.role[17].def_file = "role0_17.def"
theDisObjSvrSim.role[18].def_file = "role0_18.def"
theDisObjSvrSim.role[19].def_file = "role0_19.def"
theDisObjSvrSim.role[20].def_file = "role0_20.def"
theDisObjSvrSim.role[21].def_file = "role0_21.def"
theDisObjSvrSim.role[22].def_file = "role0_22.def"
theDisObjSvrSim.role[23].def_file = "role0_23.def"
theDisObjSvrSim.role[24].def_file = "role0_24.def"
theDisObjSvrSim.role[25].def_file = "role0_25.def"
theDisObjSvrSim.role[26].def_file = "role0_26.def"
theDisObjSvrSim.role[27].def_file = "role0_27.def"
model.num_role = 28

```

Pattern Configuration File for

```

## pat1 [GIGA] [GIGA] [GIGA] ##
[Run 1]
model.run_name = "pat1"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 3
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 3
model.smachine[2].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "B.def"
model.objsvr[2].cfg_file = "C.def"

## pat2 [SIX] [GIGA] [GIGA] ##
[Run 2]
model.run_name = "pat2"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"

```

```

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 2
model.smachine[2].objsvridx = 1
model.objsvr[1].cfg_file = "B.def"
model.objsvr[2].cfg_file = "C.def"

## pat3 [BR733] [GIGA] [GIGA] ##
[Run 3]
model.run_name = "pat3"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 2
model.smachine[2].objsvridx = 1
model.objsvr[1].cfg_file = "B.def"
model.objsvr[2].cfg_file = "C.def"

## pat4 [GIGA] [SIX] [GIGA] ##
[Run 4]
model.run_name = "pat4"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 2
model.smachine[2].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"
model.objsvr[2].cfg_file = "C.def"

## pat5 [SIX] [SIX] [GIGA] ##
[Run 5]
model.run_name = "pat5"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"

```

```

model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 2
model.smachine[1].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "B.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "C.def"

## pat6 [BR733] [SIX] [GIGA] ##
[Run 6]
model.run_name = "pat6"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 1
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "B.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "C.def"

## pat7 [GIGA] [BR733] [GIGA] ##
[Run 7]
model.run_name = "pat7"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 2
model.smachine[2].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"
model.objsvr[2].cfg_file = "C.def"

```

```

## pat8 [SIX] [BR733] [GIGA] ##
[Run 8]
model.run_name = "pat8"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 1
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "C.def"

## pat9 [BR733] [BR733] [GIGA] ##
[Run 9]
model.run_name = "pat9"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 2
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "B.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "C.def"

## pat10 [GIGA] [GIGA] [SIX] ##
[Run 10]
model.run_name = "pat10"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 0
model.objsvr[0].cfg_file = "C.def"

```

```

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 2
model.smachine[2].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"
model.objsvr[2].cfg_file = "B.def"

## pat11 [SIX] [GIGA] [SIX] ##
[Run 11]
model.run_name = "pat11"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 2
model.smachine[1].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "B.def"

## pat12 [BR733] [GIGA] [SIX] ##
[Run 12]
model.run_name = "pat12"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 1
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "B.def"

## pat13 [GIGA] [SIX] [SIX] ##
[Run 13]
model.run_name = "pat13"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"

```

```

model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 2
model.smachine[1].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"
model.objsvr[1].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "A.def"

## pat14 [SIX] [SIX] [SIX] ##
[Run 14]
model.run_name = "pat14"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 3
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 0
model.smachine[0].objsvridx = 0

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 3
model.smachine[1].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "B.def"
model.objsvr[2].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

## pat15 [BR733] [SIX] [SIX] ##
[Run 15]
model.run_name = "pat15"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 2
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "B.def"
model.objsvr[2].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

```

```

## pat16 [GIGA] [BR733] [SIX] ##
[Run 16]
model.run_name = "pat16"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 1
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "A.def"

## pat17 [SIX] [BR733] [SIX] ##
[Run 17]
model.run_name = "pat17"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 2
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"
model.objsvr[2].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

## pat18 [BR733] [BR733] [SIX] ##
[Run 18]
model.run_name = "pat18"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 2
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "B.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1

```

```

model.smachine[1].objsvridx = 2
model.objsvr[2].cfg_file = "C.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

## pat19 [GIGA] [GIGA] [BR733] ##
[Run 19]
model.run_name = "pat19"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 2
model.smachine[2].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"
model.objsvr[2].cfg_file = "B.def"

## pat20 [SIX] [GIGA] [BR733] ##
[Run 20]
model.run_name = "pat20"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 1
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "B.def"

## pat21 [BR733] [GIGA] [BR733] ##
[Run 21]
model.run_name = "pat21"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"

```

```

model.smachine[0].num_objsvr = 2
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "B.def"

## pat22 [GIGA] [SIX] [BR733] ##
[Run 22]
model.run_name = "pat22"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 1
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "B.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "A.def"

## pat23 [SIX] [SIX] [BR733] ##
[Run 23]
model.run_name = "pat23"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 1
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 2
model.smachine[1].objsvridx = 1
model.objsvr[1].cfg_file = "A.def"
model.objsvr[2].cfg_file = "B.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

```

```

## pat24 [BR733] [SIX] [BR733] ##
[Run 24]
model.run_name = "pat24"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 2
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1
model.smachine[1].objsvridx = 2
model.objsvr[2].cfg_file = "B.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

## pat25 [GIGA] [BR733] [BR733] ##
[Run 25]
model.run_name = "pat25"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 2
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"
model.objsvr[1].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 1
model.smachine[2].objsvridx = 2
model.objsvr[2].cfg_file = "A.def"

## pat26 [SIX] [BR733] [BR733] ##
[Run 26]
model.run_name = "pat26"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 2
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 2
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "B.def"
model.objsvr[1].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 1

```

```

model.smachine[1].objsvridx = 2
model.objsvr[2].cfg_file = "A.def"

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

## pat27 [BR733] [BR733] [BR733] ##
[Run 27]
model.run_name = "pat27"
model.num_machine = 3
model.num_objsvr = 3
model.max_objsvr = 3
model.smachine[0].def_file = "BR733.def"
model.smachine[0].num_objsvr = 3
model.smachine[0].objsvridx = 0
model.objsvr[0].cfg_file = "A.def"
model.objsvr[1].cfg_file = "B.def"
model.objsvr[2].cfg_file = "C.def"

model.smachine[1].def_file = "SIX.def"
model.smachine[1].num_objsvr = 0
model.smachine[1].objsvridx = 0

model.smachine[2].def_file = "GIGA.def"
model.smachine[2].num_objsvr = 0
model.smachine[2].objsvridx = 0

```

B. DETAILED RESULT

1. Simulation Run 1 (1 role 1, Ram Limit 0.0)

Pattern	Server A	Server B	Server C	Number of Call	Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1750	0.9558	0.6612	8.0006
2	GIGA	GIGA	BR733	1734	1.0962	0.6612	10.8448
3	GIGA	BR733	GIGA	1738	1.0558	0.6612	8.2388
4	GIGA	BR733	BR733	1733	1.1033	0.6612	10.8456
5	BR733	GIGA	GIGA	1718	1.2533	0.8723	8.0006
6	BR733	GIGA	BR733	1720	1.2313	0.8723	10.8447
7	BR733	BR733	GIGA	1727	1.1837	0.8723	8.2386
8	BR733	BR733	BR733	1716	1.2751	0.8723	10.8453
9	GIGA	GIGA	SIX	1723	1.1935	0.6612	13.207
10	GIGA	SIX	GIGA	1736	1.0857	0.6612	9.3585
11	GIGA	SIX	SIX	1721	1.2209	0.6612	13.2068
12	SIX	GIGA	GIGA	1711	1.3226	1.0476	7.9991
13	SIX	GIGA	SIX	1696	1.4807	1.0476	13.2058
14	SIX	SIX	GIGA	1706	1.3782	1.0476	9.3578
15	SIX	SIX	SIX	1693	1.4893	1.0476	13.2071
16	BR733	BR733	SIX	1709	1.3452	0.8723	13.2065
17	BR733	SIX	BR733	1715	1.2877	0.8723	10.8446
18	BR733	SIX	SIX	1697	1.4664	0.8723	13.2069
19	SIX	BR733	BR733	1689	1.5399	1.0476	10.8452
20	SIX	BR733	SIX	1690	1.5391	1.0476	13.2054
21	SIX	SIX	BR733	1693	1.5078	1.0476	10.8455
22	GIGA	BR733	SIX	1724	1.2014	0.6612	13.2065
23	GIGA	SIX	BR733	1739	1.058	0.6612	10.845
24	BR733	GIGA	SIX	1714	1.2905	0.8723	13.2072
25	BR733	SIX	GIGA	1717	1.2798	0.8723	9.3576
26	SIX	GIGA	BR733	1692	1.519	1.0476	10.8453
27	SIX	BR733	GIGA	1701	1.4169	1.0476	8.2397

2. Simulation Run 2 (1 role 2, Ram Limit 0.0)

Pattern	Server A	Server B	Server C	Number of Call	Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	2449	5.2635	0.6612	8.7226
2	GIGA	GIGA	BR733	2356	5.7279	0.6612	9.8576
3	GIGA	BR733	GIGA	2238	6.371	0.6612	10.7552
4	GIGA	BR733	BR733	2179	6.7169	0.6612	11.8079
5	BR733	GIGA	GIGA	2328	5.8716	0.8723	8.7224
6	BR733	GIGA	BR733	2246	6.3193	0.8723	9.8584
7	BR733	BR733	GIGA	2166	6.7915	0.8723	10.7554
8	BR733	BR733	BR733	2118	7.107	0.8723	11.8085
9	GIGA	GIGA	SIX	2321	5.9121	0.6612	10.767
10	GIGA	SIX	GIGA	2119	7.0883	0.6612	12.4087
11	GIGA	SIX	SIX	2029	7.6891	0.6612	14.3698
12	SIX	GIGA	GIGA	2243	6.3354	1.0476	8.7222
13	SIX	GIGA	SIX	2156	6.8569	1.0476	10.767
14	SIX	SIX	GIGA	1967	8.1308	1.0476	12.4089
15	SIX	SIX	SIX	1890	8.7288	1.0476	14.3694
16	BR733	BR733	SIX	2063	7.4566	0.8723	12.7582
17	BR733	SIX	BR733	1990	7.955	0.8723	13.5029
18	BR733	SIX	SIX	1966	8.141	0.8723	14.3706
19	SIX	BR733	BR733	2062	7.4566	1.0476	11.8076
20	SIX	BR733	SIX	1990	7.9663	1.0476	12.7581
21	SIX	SIX	BR733	1929	8.4218	1.0476	13.5031
22	GIGA	BR733	SIX	2131	7.0065	0.6612	12.7588
23	GIGA	SIX	BR733	2074	7.4031	0.6612	13.5038
24	BR733	GIGA	SIX	2216	6.4926	0.8723	10.7665
25	BR733	SIX	GIGA	2029	7.681	0.8723	12.4089
26	SIX	GIGA	BR733	2179	6.7215	1.0476	9.8585
27	SIX	BR733	GIGA	2083	7.318	1.0476	10.7552

3. Simulation Run 3 (1 role 3, Ram Limit 0.0)

Pattern	Server A	Server B	Server C	Number of Call	Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	260	9.284	0.5749	14.5538
2	GIGA	GIGA	BR733	227	11.2949	0.7511	16.8059
3	GIGA	BR733	GIGA	245	10.1469	0.5758	15.9637
4	GIGA	BR733	BR733	212	12.4098	0.7522	18.0903
5	BR733	GIGA	GIGA	256	9.4766	0.5717	16.1524
6	BR733	GIGA	BR733	229	11.1748	0.749	18.4014
7	BR733	BR733	GIGA	241	10.3449	0.576	17.5621
8	BR733	BR733	BR733	208	12.7509	0.7503	19.6866
9	GIGA	GIGA	SIX	212	12.5131	0.898	18.6202
10	GIGA	SIX	GIGA	235	10.7928	0.5716	17.0846
11	GIGA	SIX	SIX	196	13.8507	0.8975	21.0257
12	SIX	GIGA	GIGA	256	9.5256	0.5713	17.4756
13	SIX	GIGA	SIX	203	13.2308	0.8994	21.5432
14	SIX	SIX	GIGA	229	11.1858	0.5713	20.0041
15	SIX	SIX	SIX	183	15.0968	0.8995	23.9454
16	BR733	BR733	SIX	194	13.9055	8.1077	21.5649
17	BR733	SIX	BR733	205	12.9896	0.749	20.8684
18	BR733	SIX	SIX	190	14.4321	0.8967	22.6202
19	SIX	BR733	BR733	211	12.4835	0.7492	21.0107
20	SIX	BR733	SIX	191	14.2588	0.8982	22.8914
21	SIX	SIX	BR733	200	13.3633	0.7496	22.1958
22	GIGA	BR733	SIX	197	13.7414	0.8995	19.9709
23	GIGA	SIX	BR733	205	12.9471	0.7528	19.2751
24	BR733	GIGA	SIX	200	13.4877	0.8973	20.2182
25	BR733	SIX	GIGA	239	10.5546	0.5712	18.6806
26	SIX	GIGA	BR733	229	11.1704	0.7517	19.7244
27	SIX	BR733	GIGA	236	10.703	0.5716	18.8837

4. Simulation Run 4 (4 role 2, Ram Limit 0.0)

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1336	15.0493	0.661525	36.60915
2	GIGA	GIGA	BR733	1584	11.67535	0.661275	25.8519
3	GIGA	BR733	GIGA	1605	11.43463	0.6612	31.5801
4	GIGA	BR733	BR733	1307	15.53168	0.66125	48.36535
5	BR733	GIGA	GIGA	1605	11.43985	0.872325	33.33738
6	BR733	GIGA	BR733	1702	10.41733	0.8723	30.30368
7	BR733	BR733	GIGA	1216	17.17995	0.872475	33.82055
8	BR733	BR733	BR733	995	22.42225	0.8726	50.05655
9	GIGA	GIGA	SIX	1584	11.67603	0.661225	28.12353
10	GIGA	SIX	GIGA	1431	13.61095	0.661225	40.11288
11	GIGA	SIX	SIX	1120	19.2105	0.6612	58.8928
12	SIX	GIGA	GIGA	1548	12.10203	1.0476	34.68138
13	SIX	GIGA	SIX	1540	12.1953	1.047625	32.2754
14	SIX	SIX	GIGA	1010	22.01543	1.04835	39.9369
15	SIX	SIX	SIX	836	27.93548	1.0483	62.90595
16	BR733	BR733	SIX	1207	17.35225	0.8727	34.48555
17	BR733	SIX	BR733	1327	15.1899	0.8723	41.12468
18	BR733	SIX	SIX	1108	19.4705	0.872325	61.46145
19	SIX	BR733	BR733	1277	16.03185	1.047625	48.69093
20	SIX	BR733	SIX	1413	13.8823	1.047675	38.25873
21	SIX	SIX	BR733	1010	21.99245	1.04785	41.59348
22	GIGA	BR733	SIX	1633	11.12345	0.6612	34.45988
23	GIGA	SIX	BR733	1437	13.5477	0.6612	39.30033
24	BR733	GIGA	SIX	1845	9.10485	0.872325	25.15523
25	BR733	SIX	GIGA	1399	14.08285	0.872325	39.62575
26	SIX	GIGA	BR733	1750	9.950025	1.047625	24.67665
27	SIX	BR733	GIGA	1544	12.15075	1.0476	32.21468

5. Simulation Run 5 (3 role 3, Ram Limit 0.0)

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1361	16.6589	1.338	39.90197
2	GIGA	GIGA	BR733	1755	11.9086	0.749467	30.71187
3	GIGA	BR733	GIGA	1662	12.8099	0.572333	37.36043
4	GIGA	BR733	BR733	1121	21.1609	0.7512	42.6214
5	BR733	GIGA	GIGA	1483	14.9085	0.5736	34.62337
6	BR733	GIGA	BR733	1430	15.6307	0.896133	51.17463
7	BR733	BR733	GIGA	1678	12.6598	0.571267	35.87533
8	BR733	BR733	BR733	1000	24.2953	4.2022	61.77937
9	GIGA	GIGA	SIX	1480	14.9543	0.9315	35.4827
10	GIGA	SIX	GIGA	1520	14.4386	0.571467	37.71887
11	GIGA	SIX	SIX	917	26.8708	1.3631	55.16787
12	SIX	GIGA	GIGA	1461	15.2147	0.583933	37.92313
13	SIX	GIGA	SIX	1187	19.7351	0.898733	63.8
14	SIX	SIX	GIGA	1475	15.0132	0.5718	43.90837
15	SIX	SIX	SIX	819	30.6386	3.815067	79.47967
16	BR733	BR733	SIX	1405	15.9958	0.899	41.37097
17	BR733	SIX	BR733	1309	17.4819	0.750667	51.57777
18	BR733	SIX	SIX	918	26.8316333	0.8976	55.31653
19	SIX	BR733	BR733	1111	21.4191667	0.7519	46.0873
20	SIX	BR733	SIX	1193	19.6194	0.897167	62.65613
21	SIX	SIX	BR733	1405	15.9954333	0.7493	45.41987
22	GIGA	BR733	SIX	1468	15.1153667	0.947167	34.0967
23	GIGA	SIX	BR733	1540	14.1918333	0.750333	33.39353
24	BR733	GIGA	SIX	1487	14.8577	0.8994	35.05627
25	BR733	SIX	GIGA	1636	13.0968	0.5717	31.954
26	SIX	GIGA	BR733	1711	12.3277333	0.7494	36.1378
27	SIX	BR733	GIGA	1774	11.7289	0.571333	31.1135

6. Simulation Run 6 (28 role 1, Ram Limit 1.5)

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1117	10.2673	0.670529	60.31194
2	GIGA	GIGA	BR733	1404	5.011779	0.670243	34.84174
3	GIGA	BR733	GIGA	1253	7.468704	0.6703	47.26838
4	GIGA	BR733	BR733	1543	3.160807	0.670204	62.23152
5	BR733	GIGA	GIGA	1248	7.5642	1.937279	35.30255
6	BR733	GIGA	BR733	933	15.3563	2.392	77.13771
7	BR733	BR733	GIGA	1071	11.38449	2.353843	53.42089
8	BR733	BR733	BR733	842	18.70253	2.386021	88.78831
9	GIGA	GIGA	SIX	1404	5.002846	0.670225	56.65871
10	GIGA	SIX	GIGA	1245	7.616296	0.670311	46.1542
11	GIGA	SIX	SIX	error	error	error	error
12	SIX	GIGA	GIGA	1022	12.67358	2.90455	53.462
13	SIX	GIGA	SIX	error	error	error	error
14	SIX	SIX	GIGA	error	error	error	error
15	SIX	SIX	SIX	error	error	error	error
16	BR733	BR733	SIX	1069	11.41627	2.068632	54.62706
17	BR733	SIX	BR733	936	15.23539	2.387989	77.05709
18	BR733	SIX	SIX	error	error	error	error
19	SIX	BR733	BR733	1020	12.72855	2.9049	55.41852
20	SIX	BR733	SIX	error	error	error	error
21	SIX	SIX	BR733	error	error	error	error
22	GIGA	BR733	SIX	1550	3.075521	0.670211	56.89704
23	GIGA	SIX	BR733	1559	2.962407	0.670204	41.20151
24	BR733	GIGA	SIX	1244	7.646021	1.186104	45.96066
25	BR733	SIX	GIGA	1249	7.548154	1.382861	33.51087
26	SIX	GIGA	BR733	1023	12.64221	2.892257	50.43671
27	SIX	BR733	GIGA	1029	12.47938	2.89965	53.38349

7. Simulation Run 7 (1 role 1, Ram Limit 1.5)

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1744	0.6702	8.0191	1.0055
2	GIGA	GIGA	BR733	1739	0.6702	10.8606	1.0488
3	GIGA	BR733	GIGA	1737	0.6702	8.2572	1.0636
4	GIGA	BR733	BR733	1731	0.6702	10.8631	1.1247
5	BR733	GIGA	GIGA	1727	0.8813	8.018	1.1665
6	BR733	GIGA	BR733	1709	0.8813	10.8639	1.3483
7	BR733	BR733	GIGA	1723	0.8813	8.2568	1.2064
8	BR733	BR733	BR733	1717	0.8813	10.8629	1.2656
9	GIGA	GIGA	SIX	1729	0.6702	13.2239	1.1559
10	GIGA	SIX	GIGA	1735	0.6702	9.3768	1.0827
11	GIGA	SIX	SIX	error	error	error	error
12	SIX	GIGA	GIGA	1699	1.0566	8.0187	1.4408
13	SIX	GIGA	SIX	error	error	error	error
14	SIX	SIX	GIGA	error	error	error	error
15	SIX	SIX	SIX	error	error	error	error
16	BR733	BR733	SIX	1703	0.8813	13.2238	1.4044
17	BR733	SIX	BR733	1708	0.8813	10.8639	1.3635
18	BR733	SIX	SIX	error	error	error	error
19	SIX	BR733	BR733	1689	1.0566	10.8631	1.5526
20	SIX	BR733	SIX	error	error	error	error
21	SIX	SIX	BR733	error	error	error	error
22	GIGA	BR733	SIX	1742	0.6702	13.224	1.0286
23	GIGA	SIX	BR733	1729	0.6702	10.8633	1.1604
24	BR733	GIGA	SIX	1712	0.8813	13.224	1.3054
25	BR733	SIX	GIGA	1722	0.8813	9.3763	1.2145
26	SIX	GIGA	BR733	1707	1.0566	10.8638	1.3706
27	SIX	BR733	GIGA	1698	1.0566	8.2572	1.453

8. Simulation Run 8 (1 role 2, Ram Limit 1.5)

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	2440	0.6702	8.7493	5.3012
2	GIGA	GIGA	BR733	2362	0.6702	9.8857	5.6899
3	GIGA	BR733	GIGA	2235	0.6702	10.7829	6.3843
4	GIGA	BR733	BR733	2197	0.6702	11.8352	6.611
5	BR733	GIGA	GIGA	2325	0.8813	8.7498	5.8862
6	BR733	GIGA	BR733	2264	0.8813	9.8855	6.2286
7	BR733	BR733	GIGA	2151	0.8813	10.7822	6.8876
8	BR733	BR733	BR733	2091	0.8813	11.8346	7.2717
9	GIGA	GIGA	SIX	2312	0.6702	10.7934	5.9421
10	GIGA	SIX	GIGA	2073	0.6702	12.4356	7.392
11	GIGA	SIX	SIX	error	error	error	error
12	SIX	GIGA	GIGA	2258	1.0566	8.7493	6.2603
13	SIX	GIGA	SIX	error	error	error	error
14	SIX	SIX	GIGA	error	error	error	error
15	SIX	SIX	SIX	error	error	error	error
16	BR733	BR733	SIX	2050	0.8813	12.7852	7.5513
17	BR733	SIX	BR733	1965	0.8813	13.5297	8.145
18	BR733	SIX	SIX	error	error	error	error
19	SIX	BR733	BR733	2014	1.0566	11.8351	7.7897
20	SIX	BR733	SIX	error	error	error	error
21	SIX	SIX	BR733	error	error	error	error
22	GIGA	BR733	SIX	2138	0.6702	12.7848	6.968
23	GIGA	SIX	BR733	2069	0.6702	13.5306	7.4252
24	BR733	GIGA	SIX	2220	0.8813	10.7936	6.4696
25	BR733	SIX	GIGA	2019	0.8813	12.4357	7.7743
26	SIX	GIGA	BR733	2182	1.0566	9.8856	6.6922
27	SIX	BR733	GIGA	2084	1.0566	10.7824	7.3095

9. Simulation Run 9 (1 role 3, Ram Limit 1.5)

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	2507	0.5802	11.5328	6.9886
2	GIGA	GIGA	BR733	2245	0.758	12.6687	8.3268
3	GIGA	BR733	GIGA	2293	0.5806	12.9238	8.0593
4	GIGA	BR733	BR733	2089	0.7589	13.9755	9.2771
5	BR733	GIGA	GIGA	2431	0.5806	13.1292	7.3435
6	BR733	GIGA	BR733	2185	0.7582	14.2636	8.6711
7	BR733	BR733	GIGA	2271	0.5803	14.5174	8.1804
8	BR733	BR733	BR733	2051	0.758	15.5702	9.5299
9	GIGA	GIGA	SIX	2059	0.9057	13.5774	9.4775
10	GIGA	SIX	GIGA	2170	0.5802	14.0409	8.776
11	GIGA	SIX	SIX	error	error	error	error
12	SIX	GIGA	GIGA	2381	0.5802	14.4546	7.5815
13	SIX	GIGA	SIX	error	error	error	error
14	SIX	SIX	GIGA	error	error	error	error
15	SIX	SIX	SIX	error	error	error	error
16	BR733	BR733	SIX	1903	0.9058	16.5215	10.6311
17	BR733	SIX	BR733	1959	0.7583	16.7318	10.1949
18	BR733	SIX	SIX	error	error	error	error
19	SIX	BR733	BR733	2020	0.758	16.8961	9.7489
20	SIX	BR733	SIX	error	error	error	error
21	SIX	SIX	BR733	error	error	error	error
22	GIGA	BR733	SIX	1941	0.9059	14.9248	10.3324
23	GIGA	SIX	BR733	1967	0.7583	15.1363	10.1292
24	BR733	GIGA	SIX	2017	0.906	15.1736	9.7671
25	BR733	SIX	GIGA	2115	0.5805	15.6375	9.1113
26	SIX	GIGA	BR733	2160	0.7583	15.5885	8.8251
27	SIX	BR733	GIGA	2218	0.5803	15.8443	8.4703

10. Simulation Run 5 with Time Slice 0.3 sec

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1296	0.7865	49.6805	17.72243
2	GIGA	GIGA	BR733	1694	0.812967	29.86783	12.48773
3	GIGA	BR733	GIGA	1676	0.599133	34.1284	12.67423
4	GIGA	BR733	BR733	1068	2.7078	49.7649	22.4408
5	BR733	GIGA	GIGA	1443	1.2133	35.91423	15.44957
6	BR733	GIGA	BR733	1402	0.873833	51.23577	16.0328
7	BR733	BR733	GIGA	1665	0.6028	39.82163	12.79057
8	BR733	BR733	BR733	950	2.3349	72.1737	25.81193
9	GIGA	GIGA	SIX	1459	2.354367	33.978	15.2405
10	GIGA	SIX	GIGA	1546	0.817167	35.08873	14.12207
11	GIGA	SIX	SIX	875	1.941167	67.93237	28.38093
12	SIX	GIGA	GIGA	1437	0.6167	34.92737	15.53837
13	SIX	GIGA	SIX	1174	1.4006	59.8541	20.02303
14	SIX	SIX	GIGA	1449	0.602533	46.08833	15.38293
15	SIX	SIX	SIX	771	1.909433	82.328	32.84013
16	BR733	BR733	SIX	1375	1.1941	40.74587	16.42997
17	BR733	SIX	BR733	1301	0.845133	52.16217	17.6197
18	BR733	SIX	SIX	879	2.5394	63.9537	28.25097
19	SIX	BR733	BR733	1062	2.161067	48.9474	22.61197
20	SIX	BR733	SIX	1172	1.3686	54.9753	20.0478
21	SIX	SIX	BR733	1372	0.8039	50.16107	16.4762
22	GIGA	BR733	SIX	1437	1.352033	34.10403	15.53427
23	GIGA	SIX	BR733	1519	0.815633	33.92017	14.45383
24	BR733	GIGA	SIX	1432	1.3278	34.17067	15.59767
25	BR733	SIX	GIGA	1657	0.602733	35.8253	12.8748
26	SIX	GIGA	BR733	1661	0.7996	31.27937	12.82443
27	SIX	BR733	GIGA	1786	0.620633	31.59183	11.61023

11. Simulation Run 5 with Time Slice 0.5 sec

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1318	2.0484	44.17643	17.33753
2	GIGA	GIGA	BR733	1739	0.7884	28.29127	12.0512
3	GIGA	BR733	GIGA	1719	0.5932	32.17673	12.2479
4	GIGA	BR733	BR733	1102	2.284333	45.7475	21.61923
5	BR733	GIGA	GIGA	1489	1.881233	33.63357	14.834
6	BR733	GIGA	BR733	1427	0.837733	48.7212	15.67527
7	BR733	BR733	GIGA	1702	0.593333	38.26807	12.4122
8	BR733	BR733	BR733	977	1.992567	64.39967	24.9665
9	GIGA	GIGA	SIX	1507	1.257933	32.03333	14.59767
10	GIGA	SIX	GIGA	1587	0.585567	35.26557	13.64613
11	GIGA	SIX	SIX	909	2.675267	60.22457	27.1556
12	SIX	GIGA	GIGA	1468	1.200933	33.7923	15.1225
13	SIX	GIGA	SIX	1203	1.574133	60.17807	19.43113
14	SIX	SIX	GIGA	1515	0.592933	44.8829	14.50497
15	SIX	SIX	SIX	801	1.937767	84.2609	31.42537
16	BR733	BR733	SIX	1403	1.058833	40.1938	16.02557
17	BR733	SIX	BR733	1350	0.785733	46.17747	16.82987
18	BR733	SIX	SIX	904	3.8977	60.2902	27.32653
19	SIX	BR733	BR733	1098	3.553667	46.7826	21.7159
20	SIX	BR733	SIX	1197	1.749233	56.48117	19.56097
21	SIX	SIX	BR733	1395	0.781233	50.98	16.13057
22	GIGA	BR733	SIX	1471	1.246533	33.96027	15.0774
23	GIGA	SIX	BR733	1549	0.787467	34.27813	14.07673
24	BR733	GIGA	SIX	1483	1.208467	33.32643	14.90627
25	BR733	SIX	GIGA	1700	0.590367	30.50503	12.43387
26	SIX	GIGA	BR733	1705	0.775633	31.50473	12.38327
27	SIX	BR733	GIGA	1823	0.589767	29.79467	11.29057

12. Simulation Run 5 with Time Slice 1.0 sec

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1357	1.505133	44.43457	16.71443
2	GIGA	GIGA	BR733	1759	0.763	34.32413	11.86653
3	GIGA	BR733	GIGA	1727	0.580767	33.13533	12.17627
4	GIGA	BR733	BR733	1132	1.316267	47.1201	20.92767
5	BR733	GIGA	GIGA	1515	0.590167	31.0115	14.51573
6	BR733	GIGA	BR733	1455	0.763533	45.88713	15.29217
7	BR733	BR733	GIGA	1724	0.582167	36.49637	12.20677
8	BR733	BR733	BR733	1002	2.175533	67.91843	24.2237
9	GIGA	GIGA	SIX	1521	1.179	32.51987	14.41913
10	GIGA	SIX	GIGA	1615	0.580767	32.8008	13.3197
11	GIGA	SIX	SIX	929	4.623	61.53427	26.47317
12	SIX	GIGA	GIGA	1498	1.309533	38.47523	14.72073
13	SIX	GIGA	SIX	1217	1.237833	60.4929	19.1515
14	SIX	SIX	GIGA	1528	0.581633	43.31977	14.35533
15	SIX	SIX	SIX	824	1.798433	77.29057	30.4625
16	BR733	BR733	SIX	1427	1.2559	42.1291	15.67663
17	BR733	SIX	BR733	1366	0.788967	43.77027	16.5706
18	BR733	SIX	SIX	925	5.0801	57.8997	26.6062
19	SIX	BR733	BR733	1121	2.239667	45.59663	21.1879
20	SIX	BR733	SIX	1218	1.3768	59.71223	19.13137
21	SIX	SIX	BR733	1411	0.7601	46.13793	15.90373
22	GIGA	BR733	SIX	1496	1.0366	34.61547	14.74663
23	GIGA	SIX	BR733	1564	0.770967	35.37123	13.91293
24	BR733	GIGA	SIX	1511	0.9428	32.52393	14.55863
25	BR733	SIX	GIGA	1709	0.5815	29.36543	12.3478
26	SIX	GIGA	BR733	1718	0.760133	31.4568	12.249
27	SIX	BR733	GIGA	1817	0.582533	29.9311	11.34637

13. Simulation Run 5 with Time Slice 2.0 sec

Pattern	Server A	Server B	Server C	Number of Call	Average Response Time		
					Ave	Min	Max
1	GIGA	GIGA	GIGA	1381	1.4063	40.20587	16.34963
2	GIGA	GIGA	BR733	1752	0.759233	39.85037	11.93463
3	GIGA	BR733	GIGA	1704	0.5805	33.02707	12.39853
4	GIGA	BR733	BR733	1146	1.198833	43.55523	20.61667
5	BR733	GIGA	GIGA	1527	1.643933	32.5286	14.36297
6	BR733	GIGA	BR733	1460	0.759467	46.77287	15.21773
7	BR733	BR733	GIGA	1707	0.581233	35.58317	12.3638
8	BR733	BR733	BR733	1012	3.217367	61.37557	23.9637
9	GIGA	GIGA	SIX	1514	1.980367	34.20853	14.51863
10	GIGA	SIX	GIGA	1612	0.580833	33.20963	13.35987
11	GIGA	SIX	SIX	936	4.023167	56.5411	26.261
12	SIX	GIGA	GIGA	1504	0.5862	34.12003	14.64633
13	SIX	GIGA	SIX	1229	2.014333	55.57123	18.9297
14	SIX	SIX	GIGA	1506	0.580667	43.11107	14.62017
15	SIX	SIX	SIX	836	1.8724	75.9428	29.8945
16	BR733	BR733	SIX	1433	0.911367	37.33553	15.59267
17	BR733	SIX	BR733	1362	0.758833	44.3069	16.6259
18	BR733	SIX	SIX	935	4.141867	57.2442	26.29647
19	SIX	BR733	BR733	1131	2.122667	45.03013	20.9292
20	SIX	BR733	SIX	1221	0.909767	56.69247	19.0871
21	SIX	SIX	BR733	1400	0.758867	45.0065	16.06243
22	GIGA	BR733	SIX	1491	1.128633	41.00177	14.8088
23	GIGA	SIX	BR733	1562	0.759833	39.01757	13.92577
24	BR733	GIGA	SIX	1505	1.323267	34.0198	14.62723
25	BR733	SIX	GIGA	1692	0.581067	31.5452	12.51057
26	SIX	GIGA	BR733	1716	0.7604	33.4506	12.283
27	SIX	BR733	GIGA	1795	0.580667	34.9834	11.53953

APPENDIX D – VERIFICATION EXPERIMENT 2

A. CONFIGURATION FILES

Object server A – *A.def*

[name] A
[ram_util] 139000
[object] A
[num_obj] 1

Object server B – *B.def*

[name] B
[ram_util] 122000
[object] B
[num_obj] 1

Object server C – *C.def*

[name] C
[ram_util] 145000
[object] C
[num_obj] 1

Object server D – *D.def*

[name] D
[ram_util] 153000
[object] D
[num_obj] 1

Object server E – *E.def*

[name] E
[ram_util] 231000
[object] E
[num_obj] 1

Object server F – *F.def*

[name] F
[ram_util] 130000
[object] F
[num_obj] 1

Object server G – *G.def*

[name] G
[ram_util] 142000
[object] G
[num_obj] 1

Object server H – *H.def*

[name] H
[ram_util] 200000
[object] H
[num_obj] 1

Object server I – *I.def*

[name] I
[ram_util] 189000
[object] I
[num_obj] 1

Object server J – *J.def*

[name] J
[ram_util] 80000
[object] J
[num_obj] 1

Role0 Configuration File – *role0.def*

[name] ROLE0
[type] 0
[min_wait] 15
[max_wait] 16
[call] F1.B1/50
[call] F1.B2/10
[call] F2.B1/10
[call] F2.B4/5
[dr_sw] 200000000
[btw_call] 0.05

Role1 Configuration File – *role1.def*

[name] ROLE1_0
[type] 1
[min_wait] 15
[max_wait] 16
[call] F1.B1/10
[call] F1.B2/40
[call] F5.B2/24
[dr_sw] 200000000
[btw_call] 0.05

Role2 Configuration File – *role2.def*

[name] ROLE2_0
[type] 2
[min_wait] 12
[max_wait] 13
[call] F4.B1/50

```
[call] F5.B1/10
[call] F5.B3/2
[call] F3.B1/1
[call] F2.B2/30
[dr_sw] 200000000
[btw_call] 0.05
```

Role3 Configuration File – *role3.def*

```
[name] ROLE3_0
[type] 3
[min_wait] 39
[max_wait] 40
[call] F2.B3/30
[dr_sw] 200000000
[btw_call] 0.05
```

MACHINE Handel – *Handel.def*

```
[name] HANDEL
[cpu_pow] 2400000
[ram_sz] 512000
[ram_limit] 1.0
[dr_sw] 200000000
[er_sw] 0.000000001
[process_call] 0.005
[process_swap] 0.005
[disk_swap] 0.005
[exec_time] 0.5
[ran_exec] 0.05
[time_slice] no
```

MACHINE MOZART – *Mozart.def*

```
[name] MOZART
[cpu_pow] 2000000
[ram_sz] 256000
[ram_limit] 1.0
[dr_sw] 200000000
[er_sw] 0.000000001
[process_call] 0.005
[process_swap] 0.005
[disk_swap] 0.005
[exec_time] 0.5
[ran_exec] 0.05
[time_slice] no
```

MACHINE BEETHOVAN – *BEE.def*

```
[name] BEETHOVEN
[cpu_pow] 3000000
[ram_sz] 1000000
[ram_limit] 1.0
[dr_sw] 200000000
[er_sw] 0.000000001
```

[process_call] 0.005
[process_swap] 0.005
[disk_swap] 0.005
[exec_time] 0.5
[ran_exec] 0.05
[time_slice] no

Button Configuration File – *button.def*

[NEW_DEF]
F1.B1
G/2
C/2
A/1
[END_DEF]
[NEW_DEF]
F1.B2
F/3
[END_DEF]
[NEW_DEF]
F1.B3
H/2
[END_DEF]

[NEW_DEF]
F2.B1
E/2
G/2
[END_DEF]
[NEW_DEF]
F2.B2
J/4
B/2
[END_DEF]
[NEW_DEF]
F2.B3
D/2
[END_DEF]
[NEW_DEF]
F2.B4
B/2
E/1
[END_DEF]

[NEW_DEF]
F3.B1
J/3
[END_DEF]
[NEW_DEF]
F3.B2
F/2
B/1
E/2
F/3
[END_DEF]

```
[NEW_DEF]
F4.B1
J/2
A/2
[END_DEF]
```

```
[NEW_DEF]
F5.B1
H/1
F/3
J/3
[END_DEF]
[NEW_DEF]
F5.B2
B/1
E/2
[END_DEF]
[NEW_DEF]
F5.B3
I/1
J/2
[END_DEF]
[NEW_DEF]
F5.B4
D/1
I/1
[END_DEF]
[NEW_DEF]
F5.B5
I/1
[END_DEF]
```

Interaction Configuration File – *interact.def*

```
[NEW_INTERACT] D/2
[CALL] A/2
[END_INTERACT]
```

```
[NEW_INTERACT] J/2
[CALL] B/1
[END_INTERACT]
```

```
[NEW_INTERACT] J/3
[CALL] I/1
[END_INTERACT]
```

```
[NEW_INTERACT] E/1
[CALL] F/5
[END_INTERACT]
```

```
[NEW_INTERACT] H/1
[CALL] G/3
[END_INTERACT]
```

Object Configuration File – *obj.def*

```
[NEW_OBJ_DEF]
A/10000
1/2345600/216000
2/1318100/2296000
3/1963200/1736000
4/1718500/528000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
B/10000
1/2587600/2752000
2/686200/736000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
C/10000
1/1482700/1968000
2/1944700/2824000
3/1087000/3728000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
D/10000
1/975000/3752000
2/2354300/1912000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
E/10000
1/1678500/3640000
2/1002900/2984000
3/2106100/2616000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
F/10000
1/2398600/2800000
2/2402200/2064000
3/2434300/3328000
4/562300/3632000
5/2028300/3800000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
G/10000
1/2065800/2968000
2/1724300/104000
3/1138200/3848000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
H/10000
1/580500/2552000
2/1111500/3384000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
I/10000
1/1131400/1336000
[END_OBJ_DEF]
[NEW_OBJ_DEF]
J/10000
```

1/1467500/392000
2/1309300/2624000
3/2050600/856000
4/2392600/248000
[END_OBJ_DEF]

Role Configuration File

```
[Parameters]
model.role[0].def_file = "role0_0.def"
model.role[1].def_file = "role0_1.def"
model.role[2].def_file = "role0_2.def"
model.role[3].def_file = "role0_3.def"
model.role[4].def_file = "role0_4.def"
model.role[5].def_file = "role0_5.def"
model.role[6].def_file = "role0_6.def"
model.role[7].def_file = "role0_7.def"
model.role[8].def_file = "role0_8.def"
model.role[9].def_file = "role0_9.def"
model.role[10].def_file = "role0_10.def"
model.role[11].def_file = "role0_11.def"
model.role[12].def_file = "role0_12.def"
model.role[13].def_file = "role0_13.def"
model.role[14].def_file = "role0_14.def"
model.role[15].def_file = "role0_15.def"
model.role[16].def_file = "role0_16.def"
model.role[17].def_file = "role0_17.def"
model.role[18].def_file = "role0_18.def"
model.role[19].def_file = "role0_19.def"
model.role[20].def_file = "role1_0.def"
model.role[21].def_file = "role1_1.def"
model.role[22].def_file = "role1_2.def"
model.role[23].def_file = "role1_3.def"
model.role[24].def_file = "role1_4.def"
model.role[25].def_file = "role2_0.def"
model.role[26].def_file = "role2_1.def"
model.role[27].def_file = "role2_2.def"
model.role[28].def_file = "role2_3.def"
model.role[29].def_file = "role2_4.def"
model.role[30].def_file = "role2_5.def"
model.role[31].def_file = "role2_6.def"
model.role[32].def_file = "role2_7.def"
model.role[33].def_file = "role3_0.def"
model.role[34].def_file = "role3_1.def"
model.role[35].def_file = "role3_2.def"
model.role[36].def_file = "role3_3.def"
model.role[37].def_file = "role3_4.def"
model.role[38].def_file = "role3_5.def"
model.role[39].def_file = "role3_6.def"
model.role[40].def_file = "role3_7.def"
model.role[41].def_file = "role3_8.def"
model.role[42].def_file = "role3_9.def"
model.num_role = 43
```

B. LINGO MODEL

MODEL:

SETS:

```
MACHINE / Mozart Handel Beethoven /:
    MEMORY, SPEED;
SERVER / A B C D E F G H I J /:
    MULTIPLIER, MEMORYUSE;
NET_SPD (SERVER, SERVER): U;
DEPLOYMENT (MACHINE, SERVER): V;
MEM_USED (MACHINE): T;
CPU_USED (MACHINE): Q;
CYCLE_AVAIL (MACHINE) : CPU_AVAIL;
```

ENDSETS

DATA:

```
MEMORY SPEED =
256 2000000
512 2400000
1000 3000000;
MULTIPLIER MEMORYUSE =
1733150000 139
901800000 122
456300000 145
433500000 153
591340000 231
356360000 130
1297450000 142
111600000 200
230360000 189
981352000 80;
```

```
MEM_LIMIT = 1;
NET_BW = 100000000;
CPU_TIME = 1200;
```

ENDDATA

```
MIN = PROC_SPEED + NET_SPEED;
```

```
! PROC_SPEED = @SUM( DEPLOYMENT( I, J ):
! V ( I, J ) * MULTIPLIER ( J ) * NORM_SPEED / SPEED( I
! ));
```

```
PROC_SPEED = @SUM( DEPLOYMENT( I, J ):
V ( I, J ) * MULTIPLIER ( J ));
```

```
!
;
! Inter-Server communications function. Ignore Client/Server Comms
;
! because they always exist and we are letting the Client location
;
! be the free variable. NOTE: ASSUME LOCAL TWICE AS FAST AS REMOTE
;
!
```

```

!   NET_SPEED = 1653120000/(U(@INDEX(D),@INDEX(A))*NET_BW) +
!               3434496000/(U(@INDEX(J),@INDEX(B))*NET_BW) +
!               352704000/(U(@INDEX(J),@INDEX(I))*NET_BW) +
!               456000000/(U(@INDEX(E),@INDEX(F))*NET_BW) +
!               923520000/(U(@INDEX(H),@INDEX(G))*NET_BW);

NET_SPEED = 68880000/(U(@INDEX(D),@INDEX(A))*NET_BW) +
143104000/(U(@INDEX(J),@INDEX(B))*NET_BW) +
14696000/(U(@INDEX(J),@INDEX(I))*NET_BW) +
19000000/(U(@INDEX(E),@INDEX(F))*NET_BW) +
38480000/(U(@INDEX(H),@INDEX(G))*NET_BW);

!
!   Figure out if two servers are running on the same machine.
!
@FOR (SERVER(K):
    @FOR (SERVER(L):
        U(K,L) = @SUM ( MACHINE(R): V(R,K)*V(R,L) ) + 1;
    );
);
!
!   A server cannot be split over multiple machines
!
@FOR (DEPLOYMENT: @BIN(V));

!
!   Each server can only run on one machine.
!
@FOR (SERVER(K):
    @SUM ( MACHINE(R): V(R, K) ) = 1;
);
!
!   Constraint for limiting the RAM load on a single machine.
!
@FOR (MACHINE(R):
    T(R) = @SUM ( SERVER(K): V(R, K)*MEMORYUSE(K) );
    T(R) < MEMORY(R)*MEM_LIMIT;
);
!
!   Computing the number of instruction cycle available.
!
@FOR (MACHINE(R):
    CPU_AVAIL(R) = CPU_TIME * SPEED(R);
);
!
!   Constraint for limiting the CPU load on a single machine.
!
@FOR (MACHINE(R):
    Q(R) = @SUM (SERVER(K): V(R, K)*MULTIPLIER(K));
    (Q(R)/1000) < (CPU_AVAIL(R)/1000);
);
END

```

C. DETAILED RESULT

Pattern	Role1	Role2	Role3	Role4	Model Value (Equal Weight)
pat108	3.5795	2.1258	3.1042	3.2595	3.01725
pat10944	4.3416	2.8207	3.5431	3.5956	3.57525
pat10954	13.5533	10.75	14.3082	12.5766	12.797025
pat10956	4.1629	3.2762	4.1032	3.8242	3.841625
pat10960	10.3711	6.9627	9.7473	9.8669	9.237
pat10990	14.8603	10.2881	12.2111	7.1023	11.11545
pat10992	4.7059	3.2482	4.0163	2.5143	3.621175
pat10996	11.1938	6.5015	8.1772	5.8929	7.94135
pat11008	10.5391	8.364	11.1136	6.6134	9.157525
pat11098	14.0368	8.6305	12.6567	11.0602	11.59605
pat11424	5.7975	3.4991	4.9229	4.422	4.660375
pat11428	13.5752	5.2758	9.996	11.2167	10.015925
pat11440	12.3339	7.0866	13.615	12.6974	11.433225
pat11476	13.6738	6.6487	11.3387	7.0902	9.68785
pat1224	2.8819	2.0639	2.5412	2.6198	2.5267
pat1236	2.8283	1.7879	2.4133	2.5902	2.404925
pat12394	8.6625	7.2423	9.6251	8.2768	8.451675
pat12396	5.0022	2.4136	3.5997	4.2395	3.81375
pat1240	3.2182	1.8142	4.1397	3.757	3.232275
pat12400	5.688	4.0441	5.6607	5.101	5.12345
pat12412	5.2784	5.2051	7.3662	6.0677	5.97935
pat12448	6.1746	5.011	6.5349	4.578	5.574625
pat1272	2.8749	1.8383	2.4398	2.6132	2.44155
pat1276	3.7648	1.9074	3.7915	3.0979	3.1404
pat1288	3.4078	2.3473	4.7078	3.2267	3.4224
pat12880	7.5732	4.7586	8.7326	7.897	7.24035
pat13140	3.0679	2.2002	3.2347	3.3456	2.9621
pat13146	3.1361	2.0079	2.9086	3.0134	2.7665
pat13148	3.032	1.9967	2.9524	3.2541	2.8088
pat13158	3.6358	2.3152	3.148	2.4812	2.89505
pat13170	3.4413	2.4605	3.3098	2.1045	2.829025
pat13174	7.8464	5.1445	7.8744	5.3383	6.5509
pat13182	3.5516	2.1038	2.9658	2.5099	2.782775
pat13184	3.5653	2.0948	2.9171	2.3795	2.739175
pat13188	3.3345	2.475	3.4901	2.483	2.94565
pat13192	7.8764	5.2529	7.9102	5.2186	6.564525
pat13196	3.2878	2.4284	3.4394	1.98	2.7839
pat13198	7.7695	5.2079	7.8671	5.2744	6.529725
pat13206	4.2644	2.9201	3.9232	3.4144	3.630525
pat13210	10.7688	5.5876	9.2245	8.924	8.626225
pat13212	3.3381	2.0834	2.9937	3.0151	2.857575
pat13222	9.7177	7.1188	12.1816	10.147	9.791275
pat13224	2.9887	2.1052	3.0723	3.0139	2.795025
pat13228	6.8521	4.0454	7.604	7.0227	6.38105
pat13258	11.1808	7.0986	10.4839	6.11	8.718325
pat13260	3.4998	2.1963	3.1493	2.5141	2.839875
pat13264	8.0042	4.1647	6.8043	4.8121	5.946325
pat13276	7.2493	5.2505	8.9064	5.2937	6.674975
pat13286	4.3458	2.868	3.9823	3.6757	3.71795
pat13288	10.8844	5.7067	9.3505	8.9438	8.72135
pat13294	9.8104	7.2201	12.3638	10.1651	9.88985
pat13312	11.1185	6.9168	10.6	6.2161	8.71285

pat13371	3.9224	1.9031	3.1671	3.1904	3.04575
pat13374	4.0699	2.0324	3.4009	3.5704	3.2684
pat13386	3.7786	2.2597	3.9557	3.744	3.4345
pat13387	9.476	4.0773	9.884	9.7963	8.3084
pat13389	2.9411	1.5639	2.7241	2.9576	2.546675
pat13390	9.4743	3.9757	9.8508	9.8163	8.279275
pat13422	4.4283	2.3985	3.9178	2.4421	3.296675
pat13423	10.8526	4.4688	8.5637	5.8321	7.4293
pat13425	3.2926	1.5845	2.7699	2.5443	2.547825
pat13426	10.7027	4.3629	8.4299	6.0954	7.397725
pat13438	9.8699	5.635	11.187	6.7319	8.35595
pat13441	6.8727	3.106	7.1324	5.0353	5.5366
pat13528	13.0035	5.4377	12.8541	11.546	10.710325
pat13531	9.6654	2.9758	8.4663	8.6409	7.4371
pat13614	3.99	2.0747	3.1872	3.322	3.143475
pat13616	4.0145	1.8651	3.2231	3.4803	3.14575
pat13620	3.5676	2.4582	4.0351	3.7714	3.458075
pat13624	9.6723	4.2041	9.9011	10.0964	8.468475
pat13628	3.639	2.2701	4.0952	4.0082	3.503125
pat13630	9.6008	4.0527	9.9166	9.8948	8.366225
pat13638	4.3352	2.571	3.8156	1.9789	3.175175
pat13642	10.5105	4.1575	8.2166	6.0561	7.235175
pat13654	9.9396	5.7567	11.2542	6.7201	8.41765
pat13664	4.4949	2.4397	3.8947	2.1908	3.255025
pat13666	10.7243	4.2903	8.4026	5.894	7.3278
pat13672	9.9028	5.7276	11.3366	6.6091	8.394025
pat13690	12.8876	5.4194	12.7288	11.4747	10.627625
pat13692	3.8737	2.1691	3.5267	3.3098	3.219825
pat13696	9.4261	2.8875	8.3214	8.4901	7.281275
pat13708	8.4951	4.0538	11.1896	9.7961	8.38365
pat13744	10.2151	4.5178	9.7463	5.9337	7.603225
pat1380	2.895	1.9861	2.3461	2.5888	2.454
pat1384	3.4386	1.5707	3.6481	3.3058	2.9908
pat13860	2.7584	1.7042	2.4495	2.577	2.372275
pat13872	2.7687	1.7123	2.4298	2.5715	2.370575
pat13876	3.5536	2.7636	4.4413	3.8938	3.663075
pat13908	2.7865	1.6437	2.381	2.5439	2.338775
pat13912	4.207	2.8181	4.1255	3.1873	3.584475
pat13924	3.8802	3.4845	5.1973	3.4061	3.992025
pat1396	2.9047	1.7777	4.343	3.6777	3.175775
pat14014	5.6467	4.2153	6.9418	5.6563	5.615025
pat14340	2.7893	1.7842	2.482	2.4706	2.381525
pat14344	5.3071	2.5937	5.4657	5.2158	4.645575
pat14356	4.5016	3.2952	7.0793	5.9846	5.215175
pat14392	5.6981	3.4945	6.404	4.2016	4.94955
pat144	3.3682	2.1748	3.3162	3.8459	3.176275
pat14586	4.4204	1.8924	2.9661	3.7064	3.246325
pat14588	2.7148	1.5782	2.3195	2.4164	2.257225
pat14592	4.4171	2.1129	3.1842	4.0388	3.43825
pat14596	3.5626	2.8762	4.5955	3.9041	3.7346
pat14600	2.8517	1.7216	2.5498	2.8669	2.4975
pat14602	3.7035	2.8313	4.5282	3.8943	3.739325
pat14610	4.6162	2.0794	3.1079	3.6964	3.374975
pat14614	4.1713	2.85	4.203	3.1424	3.591675
pat14626	4.0397	3.589	5.1995	3.5506	4.0947
pat14636	2.7851	1.6579	2.4118	2.7922	2.41175

pat14638	4.1459	2.7905	4.1197	3.4289	3.62125
pat14644	3.8575	3.4842	5.1961	3.8367	4.093625
pat14662	5.697	4.4219	7.1305	5.855	5.7761
pat14664	4.5	1.9565	3.0077	3.7962	3.3151
pat14668	3.8351	2.4603	4.0912	3.5295	3.479025
pat14680	3.4756	2.9271	4.8757	3.9364	3.8037
pat14716	4.0355	2.9509	4.4853	3.4887	3.7401
pat1476	7.7857	3.9016	5.3856	5.9908	5.765925
pat1482	7.321	2.4913	3.4371	4.8747	4.531025
pat14826	4.6476	1.9679	3.3178	4.0419	3.4938
pat14827	5.1274	2.5937	5.5471	5.2055	4.618425
pat14829	4.4903	1.8129	2.9341	3.8047	3.2605
pat14830	5.24	2.5646	5.5353	5.1902	4.632525
pat1484	3.2733	1.772	2.3737	2.4658	2.4712
pat14842	4.6184	3.3412	7.0699	6.0626	5.273025
pat14845	3.3082	1.869	4.0488	3.643	3.21725
pat14878	5.6795	3.4485	6.4335	4.6191	5.04515
pat14881	3.6353	1.9002	3.6448	3.2431	3.10585
pat1494	7.8185	3.9373	5.4363	6.15	5.835525
pat1506	7.5275	2.6841	3.4782	4.8754	4.6413
pat15068	2.8165	1.6504	2.6726	2.7714	2.477725
pat15070	5.1679	2.6423	5.5037	5.1488	4.615675
pat15076	4.4466	3.5246	7.2301	6.0746	5.318975
pat15094	5.6303	3.5805	6.4917	4.3614	5.015975
pat1510	3.5563	2.439	3.5041	2.6548	3.03855
pat15148	4.7662	2.6916	6.1155	5.2366	4.702475
pat1518	6.5779	2.2539	3.6932	7.4933	5.004575
pat1520	2.8094	1.5625	2.2916	3.2481	2.4779
pat1524	6.6221	2.3717	3.731	7.5531	5.069475
pat1528	3.1348	2.3442	3.5094	3.4467	3.108775
pat15318	4.3152	2.8171	3.4336	3.5379	3.52595
pat1532	3.0335	1.5829	2.2393	2.8383	2.4235
pat15328	13.4379	10.6045	14.1879	12.3795	12.65245
pat15330	4.1117	3.3071	4.0497	3.7546	3.805775
pat15334	10.3143	6.7437	9.7046	9.6855	9.112025
pat1534	3.3829	2.2044	3.305	3.0736	2.991475
pat15364	14.8447	10.118	12.0764	7.08	11.029775
pat15366	4.7182	3.2928	3.9784	2.4591	3.612125
pat15370	11.2386	6.484	8.2083	5.8608	7.947925
pat15382	10.6562	8.3974	11.1358	6.5852	9.19365
pat15472	13.8221	8.6782	12.5881	11.2462	11.58365
pat1548	7.7495	3.894	5.333	5.9286	5.726275
pat156	3.3451	1.9258	2.8641	2.9661	2.775275
pat1560	7.4005	2.5922	3.4223	4.9527	4.591925
pat1564	3.4057	2.1539	3.3805	2.7468	2.921725
pat15798	5.7397	3.4449	4.8305	4.3351	4.58755
pat15802	13.586	5.4637	10.005	11.0663	10.03025
pat15814	12.2319	7.1733	13.5804	12.5588	11.3861
pat15850	13.7729	7.0554	11.4195	7.1069	9.838675
pat1596	6.6825	2.3856	3.7351	7.6333	5.109125
pat160	5.0558	3.0601	5.4449	4.0029	4.390925
pat1600	3.294	2.0922	3.1119	3.2926	2.947675
pat1612	3.3136	2.2865	3.4058	3.1196	3.031375
pat1626	7.7213	3.5355	3.7838	5.6136	5.16355
pat1628	3.174	1.9262	2.2535	2.1597	2.37835
pat1632	7.6988	3.5521	3.7957	5.5824	5.15725

pat1636	3.2771	2.3086	3.477	2.7524	2.953775
pat1650	8.0164	3.5702	3.8432	5.7376	5.29185
pat1654	3.6276	2.3724	3.1958	2.4612	2.91425
pat16768	8.5875	7.2245	9.5565	8.272	8.410125
pat16770	4.6742	2.327	3.2805	3.9651	3.5617
pat16774	5.7704	4.0571	5.6398	5.1906	5.164475
pat16786	5.2585	5.0983	7.2933	6.2203	5.9676
pat16822	6.1913	5.0226	6.5003	4.608	5.58055
pat1707	4.3434	1.7685	2.8668	3.6072	3.146475
pat1710	7.7248	3.8234	5.4584	6.1043	5.777725
pat1722	7.1719	2.4855	3.5616	5.0241	4.560775
pat1723	3.3585	1.878	4.2009	3.7987	3.309025
pat1725	7.3785	2.4021	3.3928	4.8423	4.503925
pat17254	7.5823	4.7771	8.7033	7.7167	7.19485
pat1726	3.7788	2.0577	4.2406	3.6301	3.4268
pat17504	4.2592	2.5776	3.1894	3.437	3.3658
pat17506	13.3803	10.4925	14.2692	12.3878	12.63245
pat17512	10.5493	6.9861	9.8975	9.8522	9.321275
pat17524	14.7181	10.0056	11.984	7.1173	10.95625
pat17530	11.2874	6.6633	8.368	6.0156	8.083575
pat17578	13.9013	8.5135	12.6175	11.2769	11.5773
pat1758	6.7627	2.3142	3.8511	7.4952	5.1058
pat17584	10.2806	5.1706	8.3044	8.4595	8.053775
pat1759	3.7124	1.9081	3.7548	3.2999	3.1688
pat1761	6.5825	2.1666	3.6467	7.664	5.01495
pat1762	3.9354	2.0855	3.8675	3.7236	3.403
pat1774	3.9252	2.4592	4.6923	3.8492	3.731475
pat17743	13.8006	5.5444	10.0936	11.0438	10.1206
pat17746	13.9178	5.383	10.069	11.2166	10.1466
pat1777	3.2106	1.6474	3.0364	3.0407	2.733775
pat180	3.617	2.7418	3.5357	3.8488	3.435825
pat18226	8.8087	7.3286	9.6398	8.2347	8.50295
pat18232	5.8855	4.0806	5.6406	5.3084	5.228775
pat1866	7.6481	3.3689	3.8244	5.6034	5.1112
pat1867	3.449	1.5951	3.6053	3.257	2.9766
pat1869	7.781	3.467	3.7533	5.4846	5.121475
pat1870	3.9911	1.9487	3.8256	3.306	3.26785
pat188	2.6922	1.6393	2.5854	2.849	2.441475
pat1950	5.1283	3.0897	3.297	3.9658	3.8702
pat1952	2.7798	1.6913	2.3417	2.3873	2.300025
pat1956	5.0525	3.1236	3.441	4.1941	3.9528
pat1960	3.2705	2.1749	4.2822	3.7336	3.3653
pat1964	2.8736	1.6439	2.478	2.748	2.435875
pat1966	3.4003	1.998	4.142	3.7032	3.310875
pat19737	3.4404	1.9702	3.068	3.3447	2.955825
pat1974	5.189	3.124	3.3606	3.8971	3.892675
pat19755	3.014	1.9352	3.1299	3.9569	3.009
pat19761	3.1596	1.5916	2.8045	3.1685	2.68105
pat19763	3.2908	1.5734	5.2952	4.0533	3.553175
pat1978	3.8286	2.2148	3.8444	2.9674	3.2138
pat198	3.9455	2.8758	3.4996	3.4924	3.453325
pat19851	2.7343	1.7562	2.6074	2.5003	2.39955
pat19853	3.5025	1.6941	5.2807	4.125	3.650575
pat19854	3.7547	2.7352	3.3421	3.6411	3.368275
pat19857	2.7368	1.7895	2.4003	2.7511	2.419425
pat19861	4.1532	2.4608	3.1396	3.9063	3.414975

pat19863	3.514	2.5236	3.3337	3.5783	3.2374
pat19865	3.2764	1.6996	5.2991	4.3601	3.6588
pat19867	4.078	2.5801	3.4868	3.9586	3.525875
pat19870	4.2538	2.5806	3.1328	3.914	3.4703
pat19871	3.2569	1.593	5.1875	4.0356	3.51825
pat19875	2.8533	1.7975	2.5454	2.8699	2.516525
pat19879	4.51	2.5213	3.0632	3.4737	3.39205
pat19881	3.486	2.5672	3.4963	3.8442	3.348425
pat19891	4.3553	3.1226	3.8494	3.8606	3.796975
pat19897	3.1485	1.9646	2.7267	2.8878	2.6819
pat1990	3.7002	2.6012	4.7942	3.42	3.6289
pat19901	3.533	1.7534	5.3432	4.0339	3.665875
pat19903	4.4338	2.5659	3.2486	3.0197	3.317
pat19906	4.629	2.6413	2.9642	3.4483	3.4207
pat19907	3.3334	1.6324	5.3102	4.3328	3.6522
pat19909	4.5287	3.1761	3.7831	3.4587	3.73665
pat19915	3.2223	2.0171	2.6366	2.891	2.69175
pat19953	3.5127	2.068	2.9033	3.2422	2.93155
pat19989	3.1359	1.9791	3.1578	3.9645	3.059325
pat2000	2.7494	1.5809	2.3485	2.7551	2.358475
pat20001	3.3027	1.6325	2.4877	3.1275	2.6376
pat20005	3.9582	2.0564	2.806	3.2325	3.013275
pat2002	3.7566	2.0681	3.809	3.2775	3.2278
pat2008	3.4798	2.58	4.8008	3.7349	3.648875
pat20091	3.0745	1.7693	2.4023	2.8532	2.524825
pat20095	5.7881	2.0884	3.3094	4.8971	4.02075
pat20097	3.6756	2.6157	3.299	3.5675	3.28945
pat20107	5.6346	2.8365	4.6028	5.971	4.761225
pat20113	3.6505	1.6807	2.7581	3.468	2.889325
pat20143	6.421	3.0189	4.1575	4.145	4.4356
pat20149	4.0624	1.8457	2.7192	3.2596	2.971725
pat20187	2.87	2.1091	2.6386	2.851	2.617175
pat20193	2.7219	1.7742	2.5678	2.5034	2.391825
pat20195	3.288	1.5084	5.2101	4.2937	3.57505
pat20205	2.8484	2.1592	2.7808	2.9538	2.68555
pat20217	2.8526	1.7946	2.4778	2.6599	2.446225
pat20221	3.8834	1.9666	2.9525	3.5562	3.089675
pat20229	2.802	1.9633	2.7425	2.7744	2.57055
pat20231	3.5075	1.6372	5.3038	4.1146	3.640775
pat20235	2.8149	1.8327	2.4469	2.6109	2.42635
pat20239	3.9882	1.9916	2.8418	3.2106	3.00805
pat20243	3.2982	1.5683	5.2024	3.769	3.459475
pat20245	3.9306	2.0432	3.1085	3.0729	3.0388
pat2028	5.0585	3.0412	3.2501	4.0259	3.843925
pat2032	3.5285	1.9209	3.7468	3.265	3.1153
pat20333	3.7056	1.6251	5.243	4.3401	3.72845
pat20334	3.277	2.2324	2.5236	2.9254	2.7396
pat20335	5.7396	2.0843	3.6039	4.9369	4.091175
pat20337	2.7986	2.0297	2.7038	2.5219	2.5135
pat20338	5.6971	2.1159	3.2748	4.8519	3.984925
pat20339	3.4444	1.5987	5.2058	4.0845	3.58335
pat20341	5.6814	2.9089	4.7275	6.009	4.8317
pat20343	2.8236	1.9957	2.4378	2.7844	2.510375
pat20347	3.5858	1.5682	2.752	3.4687	2.843675
pat20350	5.5958	2.8684	4.5765	5.9933	4.7585
pat20351	3.1862	1.4773	5.2091	4.2713	3.535975

pat20353	3.5066	1.5638	3.0551	3.5707	2.92405
pat20359	6.3868	2.9166	4.3174	4.5213	4.535525
pat20361	2.8795	2.0763	2.5934	2.9368	2.6215
pat20365	4.0097	1.7501	2.8399	3.3782	2.994475
pat20377	3.8037	1.975	3.4254	3.651	3.213775
pat20439	3.7374	1.8176	3.51	2.6603	2.931325
pat2044	3.181	2.0186	4.4129	3.6392	3.312925
pat20475	3.262	1.7966	3.7426	3.6859	3.121775
pat20487	3.5221	1.4425	3.1198	2.7828	2.7168
pat20491	2.8507	1.535	2.9443	1.9373	2.316825
pat20577	3.0183	1.5157	2.975	2.0917	2.400175
pat20581	3.0359	1.5904	3.0081	1.9476	2.3955
pat20583	3.8348	2.3953	3.8922	3.0619	3.29605
pat20593	2.9684	1.8656	3.2813	2.3749	2.62255
pat20599	2.919	1.5655	2.9527	1.796	2.3083
pat20629	3.0745	1.7654	3.0874	2.1374	2.516175
pat20635	2.7757	1.4834	2.8617	2.1156	2.3091
pat2080	3.6514	2.104	4.0861	3.4001	3.3104
pat20907	3.0676	1.982	3.2952	2.2838	2.65715
pat20919	3.109	1.6255	2.9538	2.1165	2.4512
pat20923	2.9173	1.4368	2.9906	2.2123	2.38925
pat20955	3.0361	1.6637	3.0487	2.4777	2.55655
pat20959	2.9098	1.4441	2.8663	2.0476	2.31695
pat20971	3.0053	1.6465	3.1456	2.099	2.4741
pat21061	3.4035	1.6582	3.3319	2.652	2.7614
pat21063	3.1854	1.9178	3.0701	2.1701	2.58585
pat21067	2.9601	1.4188	2.9301	1.8381	2.286775
pat21079	2.8109	1.4617	3.1116	2.2344	2.40465
pat2108	2.838	1.6787	2.2614	2.4506	2.307175
pat2110	3.5863	1.7928	3.6921	3.2914	3.09065
pat21159	7.7339	3.8258	5.6988	5.8391	5.7744
pat2116	3.0998	2.1521	4.4376	3.7021	3.3479
pat21165	7.2402	2.5407	3.8379	4.6266	4.56135
pat21167	3.8731	1.6914	5.2717	3.7965	3.658175
pat21177	7.5469	3.7367	5.62	6.3888	5.8231
pat21189	7.2909	2.33	3.5229	5.2038	4.5869
pat21193	3.03	1.7989	2.4504	2.405	2.421075
pat21201	6.4623	2.1353	4.1147	7.6029	5.0788
pat21203	3.4664	1.6025	5.3473	4.8297	3.811475
pat21207	6.5718	2.1532	3.8591	7.5235	5.0269
pat21211	2.7546	1.7269	2.3485	2.8945	2.431125
pat21215	3.7012	1.5328	4.9912	4.3483	3.643375
pat21217	2.845	1.6922	2.3986	2.4949	2.357675
pat21305	3.3798	1.5949	5.1055	3.8766	3.4892
pat21306	7.6557	3.4065	3.8294	5.4682	5.08995
pat21307	2.7148	1.7641	2.5422	2.1156	2.284175
pat21309	7.6234	3.3853	4.1933	5.2732	5.1188
pat21310	3.3078	2.1037	2.4685	2.0816	2.4904
pat21311	3.8148	1.8843	5.3189	3.6814	3.67485
pat21313	2.796	1.9624	2.5091	2.4651	2.43315
pat21315	7.8944	3.4811	3.9795	5.485	5.21
pat21319	3.0716	1.9974	2.3948	1.8735	2.334325
pat21331	2.8027	1.8589	2.4855	2.5976	2.436175
pat21333	7.6513	3.324	3.9966	6.0353	5.2518
pat21337	2.9048	1.8851	2.3916	2.2096	2.347775
pat2134	3.7398	2.2198	4.0662	3.0659	3.272925

pat21393	7.7337	3.8189	5.4904	5.9196	5.74065
pat214	5.0513	3.0394	5.4181	3.8662	4.34375
pat21405	7.1472	2.4773	3.4148	4.8221	4.46535
pat21409	3.2236	1.8104	2.4061	2.3051	2.4363
pat21441	6.6544	2.2425	3.8096	7.6135	5.08
pat21445	2.7608	1.6547	2.2893	2.8874	2.39805
pat21457	3.0985	1.782	2.3357	2.8378	2.5135
pat21547	3.2485	1.8608	2.6459	2.897	2.66305
pat21549	7.6652	3.4727	3.8223	5.5118	5.118
pat21553	3.1333	1.9204	2.3376	1.9653	2.33915
pat21633	4.9638	3.1567	3.6415	3.8385	3.900125
pat21635	3.3581	1.7539	5.2599	3.7421	3.5285
pat21639	5.1926	3.1425	3.365	3.9773	3.91935
pat21643	2.7367	1.7689	2.395	2.2316	2.28305
pat21647	3.5885	1.6808	5.2479	4.0688	3.6465
pat21649	2.7632	1.7214	2.5754	2.3231	2.345775
pat21657	5.0204	3.0095	3.4096	4.3394	3.944725
pat21661	2.6035	1.6982	2.3898	2.451	2.285625
pat21673	2.8457	1.8245	2.5647	2.7127	2.4869
pat21683	3.3458	1.6027	5.1786	4.3895	3.62915
pat21685	2.5122	1.6097	2.4298	2.4386	2.247575
pat21691	2.7613	1.8198	2.4205	2.7409	2.435625
pat21790	3.2893	1.9774	2.646	2.8115	2.68105
pat21791	3.4169	1.7158	5.1223	3.7778	3.5082
pat21793	2.6609	1.7026	2.4717	1.9685	2.200925
pat21799	2.7443	1.7577	2.3734	2.389	2.3161
pat21817	2.6792	1.6953	2.3458	2.4682	2.297125
pat2214	3.9309	2.4728	3.3504	3.5694	3.330875
pat224	2.9329	1.7023	2.5299	2.654	2.454775
pat2250	3.9354	2.5882	3.541	3.7126	3.4443
pat2262	3.9229	2.5474	3.3632	2.8731	3.17665
pat2266	7.657	5.1012	7.2575	5.2275	6.3108
pat232	4.9909	3.0288	5.5693	4.1856	4.44365
pat2352	4.3573	2.9682	3.7405	3.5701	3.659025
pat2356	10.4314	5.2709	8.6274	8.5455	8.2188
pat2358	4.1721	2.9181	3.5681	3.8305	3.6222
pat2368	9.4569	6.8042	11.5238	9.7983	9.3958
pat2374	6.7734	4.0124	7.1662	6.9159	6.216975
pat2404	10.7472	6.8897	9.8275	6.0135	8.369475
pat24063	2.8676	1.8374	2.4528	2.4496	2.40185
pat24065	3.7622	1.8195	5.0075	4.2264	3.7039
pat24066	3.1549	2.0212	2.5755	3.0142	2.69145
pat24069	2.8893	1.9774	2.3268	2.7917	2.4963
pat24073	6.2125	4.0764	3.6119	5.5136	4.8536
pat24075	2.7437	1.7595	2.582	2.8479	2.483275
pat24077	3.6017	2.0701	5.2494	4.5761	3.874325
pat24078	3.1198	2.0785	2.4116	2.9232	2.633275
pat24079	6.0741	4.155	3.8654	5.5051	4.8999
pat24081	2.6709	1.5908	2.5683	2.5133	2.335825
pat24082	6.1965	4.1213	3.5823	5.5312	4.857825
pat24083	3.3053	1.6241	5.2061	4.2509	3.5966
pat24087	3.0847	1.997	2.4908	2.4576	2.507525
pat24091	6.6414	3.9684	3.434	4.3282	4.593
pat24093	2.7288	1.757	2.7736	2.9076	2.54175
pat2410	7.5367	4.0765	6.2259	4.8597	5.6747
pat24103	6.9399	5.417	4.8708	5.0106	5.559575

pat24105	2.776	1.6476	2.4661	2.6222	2.377975
pat24109	4.0962	2.7259	2.9732	3.5599	3.3388
pat24113	3.9624	2.1101	5.2505	3.7783	3.775325
pat24114	3.2971	2.0518	2.5577	2.7067	2.653325
pat24115	6.5323	3.9642	3.6316	3.7664	4.473625
pat24117	2.6967	1.5919	2.7155	2.674	2.419525
pat24118	6.7029	4.0389	3.3069	4.0489	4.5244
pat24119	3.4641	1.6199	5.2234	4.128	3.60885
pat24121	6.9719	5.4565	4.7846	4.521	5.4335
pat24123	2.7778	1.6417	2.451	2.5996	2.367525
pat24127	4.2092	2.7024	2.8797	3.208	3.249825
pat24130	7.0081	5.4453	4.6774	4.6802	5.45275
pat24131	3.4349	1.7675	5.2786	3.8491	3.582525
pat24133	4.1169	2.8743	3.1228	3.0704	3.2961
pat24221	3.7567	1.9194	5.2285	4.2885	3.798275
pat24223	6.0268	3.3675	3.6392	4.817	4.462625
pat24229	6.2615	4.4463	4.738	6.0449	5.372675
pat24247	6.8311	4.3032	4.3875	4.5492	5.01775
pat24303	3.5684	2.0373	2.3781	3.0201	2.750975
pat24307	8.652	3.5465	3.589	6.8238	5.652825
pat24309	2.9957	1.7932	2.5654	2.9469	2.5753
pat24319	8.9699	5.2888	5.7278	8.8939	7.2201
pat24321	2.9616	1.6859	2.3262	2.8918	2.466375
pat24325	5.271	2.5427	3.1057	5.0868	4.00155
pat24355	9.4942	4.9669	4.9394	5.2559	6.1641
pat24357	3.0883	1.6836	2.4952	2.7142	2.495325
pat24361	5.7643	2.6292	2.9432	3.8173	3.7885
pat24373	5.9928	3.7686	4.1391	4.3967	4.5743
pat24463	8.87	3.9658	5.0217	7.4142	6.317925
pat24545	4.3108	2.1332	5.2149	4.6632	4.080525
pat24546	3.593	2.2564	2.4281	3.0441	2.8304
pat24547	8.5347	3.5412	3.9512	6.7296	5.689175
pat24549	2.7672	1.7374	2.5116	2.428	2.36105
pat24550	8.5416	3.5164	3.622	6.7776	5.6144
pat24551	3.6264	1.5556	5.0623	4.2661	3.6276
pat24553	9.079	5.2641	5.8468	8.8492	7.259775
pat24555	2.807	1.7634	2.3401	2.7703	2.4202
pat24559	5.3748	2.5075	3.166	5.1118	4.040025
pat24562	9.0103	5.2308	5.6983	8.8198	7.1898
pat24563	3.4827	1.6609	5.2812	4.5401	3.741225
pat24565	5.2386	2.495	3.4689	5.083	4.071375
pat24571	9.694	5.1284	5.1637	5.7236	6.427425
pat24573	2.9007	1.808	2.4646	2.4923	2.4164
pat24577	5.5462	2.4091	3.0144	4.03	3.749925
pat24589	5.9488	3.4479	4.2809	4.7005	4.594525
pat24598	9.4974	4.9122	4.9366	5.2195	6.141425
pat24599	3.7611	1.7823	5.3019	3.8774	3.680675
pat24601	5.8041	2.6976	3.2832	3.6039	3.8472
pat24607	6.1599	3.6266	4.3025	4.2965	4.596375
pat24789	2.8744	1.448	2.899	1.9223	2.285925
pat24793	3.5612	2.062	3.0871	2.4918	2.800525
pat24795	2.8997	1.5068	3.3149	2.2264	2.48695
pat24805	3.6031	2.6876	3.6313	3.137	3.26475
pat24807	3.0448	1.3793	2.9614	2.1107	2.37405
pat24811	2.9899	1.7307	2.9814	2.1915	2.473375
pat24841	3.8472	2.5231	3.4225	2.4728	3.0664

pat24843	2.9501	1.2959	3.1182	2.5366	2.4752
pat24847	2.9681	1.6239	2.9043	2.1073	2.4009
pat24859	3.1044	1.9932	3.1434	2.0814	2.5806
pat24949	3.6187	2.214	3.3905	2.6476	2.9677
pat25273	4.7546	2.6368	3.9309	3.8871	3.80235
pat25275	2.9104	1.5452	2.9688	1.994	2.3546
pat25279	3.2631	1.5104	2.9428	2.3099	2.50655
pat25291	3.3483	1.9765	3.4674	3.034	2.95655
pat25327	3.5521	1.8799	3.2602	2.3607	2.763225
pat25517	3.3129	1.575	5.1829	4.0087	3.519875
pat25518	4.7263	1.9174	3.2285	3.9363	3.452125
pat25519	3.163	2.2178	2.7561	2.6832	2.705025
pat25521	4.5436	1.8259	3.6008	3.7187	3.42225
pat25522	3.3038	2.213	2.5341	2.633	2.670975
pat25523	3.2671	1.5306	5.2429	3.7318	3.4431
pat25525	3.3061	2.7724	3.047	3.349	3.118625
pat25527	4.6592	1.7996	3.2963	3.8756	3.407675
pat25531	2.7785	1.8604	2.4196	2.2953	2.33845
pat25534	3.5516	2.8909	3.0358	3.4463	3.23115
pat25535	3.5067	1.5333	5.1875	4.0006	3.557025
pat25537	2.8068	1.8723	2.5931	2.3093	2.395375
pat25543	3.4632	2.5732	2.9472	3.0999	3.020875
pat25545	4.5434	1.7271	3.3855	4.287	3.48575
pat25549	2.6191	1.7201	2.3984	2.4373	2.293725
pat25561	2.8761	2.0554	2.601	2.7925	2.58125
pat25570	3.6297	2.6162	2.8004	3.0971	3.03585
pat25571	3.3022	1.4115	5.0947	4.3819	3.547575
pat25573	2.5556	1.6772	2.47	2.463	2.29145
pat25579	2.8133	1.9965	2.4352	2.7528	2.49945
pat25759	4.4765	2.7968	3.3582	4.2541	3.7214
pat25761	4.6043	1.7856	3.2402	3.9079	3.3845
pat25765	3.0443	1.7446	2.3785	2.4894	2.4142
pat25777	3.2033	2.0566	2.7657	3.1887	2.803575
pat25813	3.2135	1.9683	2.551	2.945	2.66945
pat26002	4.5136	2.8436	3.3956	4.2614	3.75355
pat26003	3.3262	1.5651	5.2757	4.0061	3.543275
pat26005	2.8864	1.7138	2.612	2.5176	2.43245
pat26011	3.0158	2.044	2.7972	3.158	2.75375
pat26029	3.1179	1.9895	2.751	2.9861	2.711125
pat267	2.9456	1.5609	2.7547	2.9459	2.551775
pat2682	4.1725	2.5907	3.364	3.5919	3.429775
pat2694	3.8861	2.6087	3.8285	3.7577	3.52025
pat2698	9.3726	4.0178	9.1722	9.6814	8.061
pat270	3.8754	2.0803	3.42	3.6779	3.2634
pat2730	4.4115	2.7385	3.7481	2.5316	3.357425
pat2734	10.454	4.2505	7.787	5.8862	7.094425
pat2746	9.7407	5.7785	10.6118	6.7558	8.2217
pat2836	13.0596	5.5004	12.3716	11.3236	10.5638
pat2838	4.1073	2.518	3.5058	3.563	3.423525
pat2842	9.2639	2.9206	7.7793	8.2314	7.0488
pat2854	8.4289	3.9174	10.6646	9.5271	8.1345
pat303	3.4	1.6676	2.8382	2.4953	2.600275
pat306	3.7497	2.0865	3.5659	3.7864	3.297125
pat318	3.7465	1.9293	3.3142	2.9722	2.99055
pat319	6.9112	3.1443	7.2025	5.1338	5.59795
pat321	3.3059	1.5453	2.6659	3.0269	2.636

pat322	7.0496	3.1208	7.3038	5.2886	5.6907
pat32817	2.948	2.0686	2.4005	2.7804	2.549375
pat32821	6.539	4.2769	3.986	5.8586	5.165125
pat32823	2.6956	1.7273	2.5082	2.8237	2.4387
pat32825	3.7147	2.1875	5.4308	4.6886	4.0054
pat32826	3.1669	2.1724	2.4673	2.9716	2.69455
pat32827	6.504	4.3744	4.36	5.8605	5.274725
pat32829	2.6396	1.6058	2.5344	2.4693	2.312275
pat32830	6.5802	4.3848	3.9837	5.912	5.215175
pat32831	3.3901	1.646	5.3037	4.3704	3.67755
pat32835	3.1585	2.0846	2.5087	2.3969	2.537175
pat32839	7.0125	4.2015	3.7321	4.4767	4.8557
pat32841	2.7548	1.7719	2.7267	2.926	2.54485
pat32851	7.1844	5.6379	5.3276	5.0998	5.812425
pat32853	2.7543	1.6677	2.442	2.5733	2.359325
pat32857	4.4093	2.9127	3.2175	3.7163	3.56395
pat32861	4.0306	2.2433	5.33	3.7373	3.8353
pat32862	3.3631	2.0886	2.5525	2.6813	2.671375
pat32863	6.8589	4.2358	3.9357	3.928	4.7396
pat32865	2.7166	1.6092	2.6826	2.6459	2.413575
pat32866	6.9382	4.1662	3.6105	4.1981	4.72825
pat32867	3.4861	1.6676	5.2295	4.0553	3.609625
pat32869	7.3349	5.6996	5.2341	4.7176	5.74655
pat32871	2.7744	1.6865	2.4432	2.5562	2.365075
pat32875	4.4979	2.9599	3.1487	3.3294	3.483975
pat32878	7.3162	5.7342	5.148	4.8367	5.758775
pat32879	3.4124	1.8192	5.3168	3.8187	3.591775
pat32881	4.2981	3.0114	3.3808	3.1595	3.46245
pat32969	3.8232	1.9478	5.2872	4.349	3.8518
pat32971	6.355	3.4979	3.9022	5.0606	4.703925
pat32977	6.5279	4.7683	5.2477	6.3134	5.714325
pat32995	7.1343	4.4646	4.7413	4.6086	5.2372
pat33051	3.6576	2.0583	2.4444	3.0537	2.8035
pat33057	3.0404	1.7723	2.5127	2.9472	2.56815
pat33067	9.4468	5.6524	6.3302	9.2297	7.664775
pat33069	2.9606	1.678	2.3203	2.8669	2.45645
pat33073	5.6352	2.6655	3.4436	5.5006	4.311225
pat33103	10.2275	5.3559	5.5175	5.3725	6.61835
pat33105	3.1374	1.7107	2.4631	2.6008	2.478
pat33109	6.0617	2.7412	3.1945	3.8687	3.966525
pat33121	6.4065	3.8497	4.6038	4.6721	4.883025
pat33211	9.2682	4.1712	5.5366	7.7953	6.692825
pat33293	4.3872	2.0801	5.2419	4.6886	4.09945
pat33294	3.6592	2.2286	2.4428	3.0963	2.856725
pat33297	2.7948	1.7213	2.43	2.4207	2.3417
pat33299	3.7066	1.5756	5.1531	4.3852	3.705125
pat33301	9.3488	5.4427	6.421	9.2135	7.6065
pat33303	2.8342	1.8128	2.3342	2.771	2.43805
pat33307	5.8486	2.7207	3.5699	5.6791	4.454575
pat33310	9.2698	5.567	6.3093	9.3019	7.612
pat33311	3.5473	1.6658	5.3465	4.6335	3.798275
pat33313	5.5837	2.5845	3.7915	5.4237	4.34585
pat33319	9.8971	5.0789	5.6012	5.7859	6.590775
pat33321	2.95	1.8493	2.4642	2.4781	2.4354
pat33325	6.1234	2.716	3.3524	4.2871	4.119725
pat33337	6.37	3.7183	4.7505	4.9479	4.946675

pat33346	10.1253	5.1865	5.4133	5.4125	6.5344
pat33347	3.791	1.777	5.2405	3.831	3.659875
pat33349	5.9405	2.6902	3.5143	3.7487	3.973425
pat33355	6.3449	3.7313	4.6605	4.4394	4.794025
pat33537	2.8908	1.469	2.8958	1.9345	2.297525
pat33541	3.7818	2.1646	3.2852	2.6175	2.962275
pat33543	2.8865	1.4911	3.1915	2.1397	2.4272
pat33553	3.7019	2.8425	3.902	3.2589	3.426325
pat33555	3.0061	1.4287	2.9253	2.1022	2.365575
pat33559	3.0279	1.758	3.0883	2.2351	2.527325
pat33589	4.0671	2.7053	3.6632	2.5674	3.25075
pat33591	2.9248	1.348	3.0336	2.4163	2.430675
pat33595	3.0495	1.6868	2.9942	2.1052	2.458925
pat33607	3.1513	2.0388	3.2889	2.1553	2.658575
pat33697	3.7024	2.3216	3.5785	2.7461	3.08715
pat34021	4.9255	2.8206	4.1909	4.0608	3.99945
pat34023	2.8803	1.5176	2.8847	1.9613	2.310975
pat34027	3.4358	1.5921	3.0593	2.4455	2.633175
pat34039	3.3668	2.0053	3.5964	3.1019	3.0176
pat34075	3.7108	1.9489	3.3895	2.4448	2.8735
pat34265	3.3648	1.6017	5.3223	4.0674	3.58905
pat34266	4.4395	1.9368	2.9866	3.7343	3.2743
pat34267	3.2644	2.287	2.9047	2.7962	2.813075
pat34269	4.3031	1.7614	3.3201	3.5479	3.233125
pat34270	3.4256	2.3166	2.617	2.7846	2.78595
pat34271	3.2947	1.5358	5.3042	3.9055	3.51005
pat34273	3.3937	2.9104	3.2629	3.5172	3.27105
pat34275	4.4705	1.7422	3.0164	3.7057	3.2337
pat34279	2.7634	1.8817	2.468	2.3628	2.368975
pat34282	3.5905	2.979	3.2204	3.5319	3.33045
pat34283	3.5299	1.5405	5.2304	4.1125	3.603325
pat34285	2.7964	1.9532	2.6814	2.3529	2.445975
pat34291	3.629	2.7869	3.143	3.1833	3.18555
pat34293	4.4817	1.7356	3.1176	4.1671	3.3755
pat34297	2.6809	1.7912	2.4625	2.5481	2.370675
pat34309	2.9082	2.122	2.7065	2.8222	2.639725
pat34318	3.7687	2.8144	2.9997	3.1385	3.180325
pat34319	3.2821	1.4451	5.1405	4.4025	3.56755
pat34321	2.6235	1.7214	2.5121	2.444	2.32525
pat34327	2.8635	2.0458	2.5519	2.7169	2.544525
pat34507	4.6445	2.9535	3.6474	4.4527	3.924525
pat34509	4.397	1.785	2.9104	3.6792	3.1929
pat34513	3.1086	1.785	2.4575	2.6083	2.48985
pat34525	3.229	2.1304	2.9195	3.3285	2.90185
pat34561	3.3208	1.9982	2.651	2.9661	2.734025
pat34750	4.6993	2.974	3.6499	4.5374	3.96515
pat34751	3.2875	1.5378	5.208	3.9972	3.507625
pat34753	2.9386	1.7466	2.687	2.6338	2.5015
pat34759	3.0361	2.1113	2.9223	3.289	2.839675
pat34777	3.2214	2.0207	2.8204	2.9957	2.76455
pat3654	7.7559	3.9189	5.4889	6.0254	5.797275
pat3666	7.4489	2.7469	3.5913	5.0741	4.7153
pat3670	4.094	3.0174	4.3594	3.7269	3.799425
pat3702	6.7189	2.4559	3.8677	7.4025	5.11125
pat3706	4.2053	2.8774	3.8928	3.7256	3.675275
pat3718	4.2062	3.4797	4.7293	3.7972	4.0531

pat37181	4.5384	2.6707	5.2082	4.6311	4.2621
pat37183	9.4403	5.6434	4.322	6.952	6.589425
pat37186	9.675	5.7027	4.0166	7.134	6.632075
pat37187	3.7636	1.884	5.0238	4.268	3.73485
pat37189	9.819	7.8034	6.2625	8.8864	8.192825
pat37195	6.1566	4.0233	3.5459	5.4321	4.789475
pat37207	10.3148	7.0599	5.4838	5.7224	7.145225
pat37213	6.4325	3.8135	3.3828	4.2662	4.47375
pat37423	12.9329	6.5952	6.135	9.9419	8.90125
pat37429	8.3758	3.5959	3.4815	6.6076	5.5152
pat37909	5.3779	3.9113	4.2561	4.1583	4.4259
pat37915	3.578	2.0476	3.0945	2.4402	2.790075
pat3808	5.5446	4.2009	6.4271	5.4319	5.401125
pat3810	7.7573	3.6308	3.8604	5.6764	5.231225
pat3814	4.1884	2.7035	3.8706	3.2452	3.501925
pat39420	6.161	3.3343	8.147	5.4106	5.763225
pat39438	4.4008	3.0232	8.9026	7.9416	6.06705
pat39446	2.7496	1.5628	2.7609	2.9006	2.493475
pat39456	3.4878	2.29	5.4787	3.975	3.807875
pat39468	3.9818	1.9475	4.406	3.8009	3.53405
pat39472	4.2711	2.6178	3.2911	3.8459	3.506475
pat39474	6.188	3.35	8.1599	5.3853	5.7708
pat39504	3.3754	1.7013	4.8088	4.5338	3.604825
pat39508	4.5936	2.6657	3.2065	3.5512	3.50425
pat39510	4.4745	3.1129	8.9822	8.0577	6.156825
pat39520	4.5549	3.2029	3.742	3.6629	3.790675
pat39526	3.3328	2.0382	3.0287	3.0768	2.869125
pat39534	3.7791	2.2614	4.9511	3.917	3.72715
pat39536	2.7691	1.6988	2.6805	2.4791	2.406875
pat39540	3.6862	2.3387	5.0831	4.1404	3.8121
pat39544	4.119	2.6711	3.4303	3.8631	3.520875
pat39548	2.6934	1.6219	2.3867	2.7469	2.362225
pat39550	4.2491	2.5334	3.179	3.9245	3.4715
pat39558	3.9313	2.3705	5.0739	3.859	3.808675
pat39562	4.5951	2.7009	3.2589	3.0562	3.402775
pat39574	4.5934	3.2374	3.7631	3.4918	3.771425
pat39584	2.8226	1.6691	2.5649	2.824	2.47015
pat39586	4.5783	2.5679	3.0851	3.4711	3.4256
pat39592	4.4912	3.2519	3.8595	3.7642	3.8417
pat39633	3.9981	1.6866	4.3717	3.75	3.4516
pat39636	6.461	3.4516	8.2667	5.6014	5.945175
pat39669	3.2299	1.5191	4.7159	4.5681	3.50825
pat39672	4.6128	3.0144	8.8956	7.8925	6.103825
pat39685	3.966	1.9419	2.8112	3.2883	3.00185
pat39688	4.1414	2.1161	3.2833	3.6883	3.307275
pat39774	4.117	2.3067	5.0418	4.1397	3.9013
pat39775	5.7156	2.156	3.2526	4.8876	4.00295
pat39777	3.7481	2.166	5.0016	3.9036	3.704825
pat39778	5.6511	2.1676	3.5406	4.7705	4.03245
pat39790	5.8282	2.9109	4.6925	6.0885	4.880025
pat39793	3.6801	1.618	2.8058	3.4662	2.892525
pat39826	6.3085	2.9152	4.2627	4.3518	4.45955
pat39829	3.98	1.7508	2.8142	3.2833	2.957075
pat39870	3.9766	3.7559	6.3361	4.4379	4.626625
pat39876	4.678	2.9647	4.9763	3.965	4.146
pat39878	2.7212	1.6554	2.4415	2.6299	2.362

pat39888	4.0514	3.7216	6.3156	4.1867	4.568825
pat39900	4.6958	3.0676	4.9904	3.7988	4.13815
pat39904	4.1066	2.2543	2.9839	3.0482	3.09825
pat39912	3.7431	2.8125	5.3983	4.9722	4.231525
pat39914	2.6516	1.7524	2.6173	2.8834	2.476175
pat39918	3.7022	2.806	5.4048	5.02	4.23325
pat39922	3.9492	2.3106	3.1479	3.5627	3.2426
pat39926	2.7269	1.6206	2.2863	2.3982	2.258
pat39928	4.0823	2.1303	2.8985	3.374	3.121275
pat39936	3.3469	2.2461	3.7466	3.2945	3.158525
pat39940	5.962	2.3656	3.4095	5.0152	4.188075
pat39942	4.0176	3.6992	6.324	4.2378	4.56965
pat39952	5.8367	3.0198	4.5799	6.1809	4.904325
pat39954	4.6355	2.9326	4.9631	4.0069	4.134525
pat39958	3.7373	1.8599	2.9501	3.4649	3.00305
pat39988	6.3806	3.0138	4.0667	4.1696	4.407675
pat39990	3.8018	2.8427	5.3918	5.0481	4.2711
pat39994	3.9729	1.9906	2.9644	3.356	3.070975
pat40006	3.9772	2.1748	3.3055	3.4676	3.231275
pat40016	3.0103	1.6725	2.4118	2.8354	2.4825
pat40018	5.6701	2.1166	3.2399	4.8298	3.9641
pat40024	5.8483	2.972	4.655	6.2031	4.9196
pat40042	6.4961	3.1176	4.114	4.1943	4.4805
pat40122	6.415	3.4496	8.1829	5.196	5.810875
pat40158	4.5257	2.9453	9.0106	8.077	6.13965
pat40174	2.677	1.6752	2.6246	2.5654	2.38555
pat40260	3.7838	2.1712	5.1034	3.7465	3.701225
pat40264	2.7712	1.7991	2.6125	2.0782	2.31525
pat40276	2.8309	1.936	2.5438	2.4586	2.442325
pat40312	2.7573	1.8408	2.4926	2.6149	2.4264
pat40590	3.9869	3.6929	6.3401	4.0913	4.5278
pat40602	4.6934	2.8577	4.8872	3.8327	4.06775
pat40606	2.7294	1.7267	2.4242	2.2524	2.283175
pat40638	3.7327	2.7426	5.3676	5.1816	4.256125
pat40642	2.5345	1.7409	2.4262	2.5495	2.312775
pat40654	2.7596	1.7731	2.3658	2.5048	2.350825
pat40744	3.1794	1.818	2.6511	2.8742	2.630675
pat40842	11.6586	5.4009	14.9932	8.6782	10.182725
pat40848	12.7206	3.7472	10.5555	7.6055	8.6572
pat40850	4.7072	2.3108	4.2948	2.0566	3.34235
pat40860	11.6011	5.4846	15.1287	8.9431	10.289375
pat40872	12.9315	3.6814	10.7221	7.9919	8.831725
pat40876	4.6328	2.508	4.2534	1.9108	3.32625
pat40886	3.4955	1.9792	4.5357	4.4434	3.61345
pat40894	3.4628	2.1268	4.4333	4.0987	3.5304
pat40908	7.9626	2.4622	8.9146	6.5306	6.4675
pat40912	3.1398	1.9563	3.3185	1.9899	2.601125
pat40914	11.4766	5.4867	14.9135	8.6899	10.141675
pat40924	3.4947	2.0498	3.0753	2.4054	2.7563
pat40926	12.8402	3.7686	10.3258	7.5698	8.6261
pat40930	4.6387	2.5204	4.2377	1.7981	3.298725
pat40960	2.949	1.7561	2.948	3.1168	2.692475
pat40966	3.5341	2.1965	4.4447	4.0275	3.5507
pat40988	3.1408	1.657	2.9705	2.0853	2.4634
pat40990	3.3259	1.93	3.031	1.9933	2.57005
pat40996	3.2464	2.1922	3.3277	2.4201	2.7966

pat41014	3.3975	2.1132	3.1108	2.125	2.686625
pat41073	7.8856	2.4959	8.9416	6.5131	6.45905
pat41076	11.5459	5.3846	14.9779	8.654	10.1406
pat41088	13.0374	3.8494	10.6674	7.8271	8.845325
pat41089	3.4246	1.6181	2.8408	2.2261	2.5274
pat41091	13.0271	3.7763	10.6207	7.7478	8.792975
pat41092	4.6909	2.3715	4.2641	2.0834	3.352475
pat411	3.2222	1.6309	2.6909	2.9304	2.6186
pat41125	2.7715	1.3986	2.7382	3.0227	2.48275
pat41128	3.6295	2.0851	4.5029	4.2091	3.60665
pat41230	3.7006	1.9635	3.3491	2.6186	2.90795
pat41233	3.1782	1.6585	2.9477	1.8681	2.413125
pat41316	9.3482	5.354	9.9933	6.8731	7.89215
pat41318	3.1791	2.3491	3.5435	1.9541	2.75645
pat41322	9.1421	5.3354	9.8824	6.7643	7.78105
pat41326	3.2159	2.5051	3.5376	2.1654	2.856
pat41330	3.6946	2.0226	2.9565	2.3003	2.7435
pat41332	3.7536	2.1911	3.0659	2.2367	2.811825
pat4134	5.2229	3.2217	3.5134	4.146	4.026
pat41340	9.3787	5.463	10.0407	7.0469	7.982325
pat41344	3.2338	2.4527	3.4424	1.951	2.769975
pat41356	3.8117	2.309	3.1332	2.2622	2.879025
pat41366	2.964	1.8819	2.9675	3.4382	2.8129
pat41368	2.9932	1.9646	2.9358	3.1542	2.76195
pat41374	3.0625	2.1817	3.1867	3.4211	2.963
pat4138	5.0653	2.6596	5.1154	4.7822	4.405625
pat41392	3.2439	1.9794	2.9634	2.7021	2.7222
pat41394	9.4038	5.4968	9.9462	6.8201	7.916725
pat41398	3.2677	2.4297	3.4704	1.8168	2.74615
pat414	4.0633	2.5222	3.6063	3.858	3.51245
pat41410	3.6568	2.1941	3.1256	2.2366	2.803275
pat41446	2.979	1.9959	2.9498	3.1371	2.76545
pat4150	4.6396	3.3975	6.5853	5.6527	5.068775
pat41562	3.8823	2.5734	5.3866	4.1167	3.98975
pat41574	4.2854	2.2453	4.3444	3.9642	3.709825
pat41578	6.2465	4.1724	3.7635	5.4124	4.8987
pat41580	6.3928	3.5036	8.146	5.5012	5.8859
pat41610	3.8222	2.11	4.7023	4.3951	3.7574
pat41614	6.5485	3.9984	3.5638	4.2976	4.602075
pat41616	4.6731	3.1787	8.8931	7.9659	6.1777
pat41626	6.906	5.3998	4.6909	4.8135	5.45255
pat41632	4.33	2.9081	3.2899	3.6693	3.549325
pat41718	4.2046	2.6665	5.0839	4.2087	4.040925
pat41722	6.1345	3.4511	3.5983	4.7365	4.4801
pat41734	6.2166	4.5002	4.6524	5.8875	5.314175
pat41770	6.7206	4.3115	4.294	4.3619	4.922
pat4186	5.5441	3.5934	5.991	4.4504	4.894725
pat42042	4.002	2.6669	3.7345	3.6676	3.51775
pat42046	8.5812	3.73	3.6698	6.7028	5.67095
pat42048	4.2616	3.804	6.1468	4.3287	4.635275
pat42058	8.9777	5.4544	5.7478	8.993	7.293225
pat42060	4.8449	3.0763	4.9151	4.1894	4.256425
pat42064	5.4113	2.7574	3.344	4.9176	4.107575
pat42094	9.7151	5.1909	4.9399	5.4574	6.325825
pat42096	4.1186	3.0649	5.4544	5.0211	4.41475
pat42100	5.932	3.0433	3.3047	4.1722	4.11305

pat42112	6.1795	3.9132	4.2462	4.6781	4.75425
pat42202	8.7352	3.9544	4.9214	7.2428	6.21345
pat427	6.0923	2.1102	7.172	6.9118	5.571575
pat4294	4.8874	2.7787	5.6612	4.9673	4.57365
pat430	6.13	2.088	7.0508	6.8085	5.519325
pat43014	8.0224	2.5739	9.013	6.5941	6.55085
pat43018	3.6725	2.433	3.4006	2.4933	2.99985
pat43020	11.5592	5.5035	14.8823	8.782	10.18175
pat43030	4.1376	2.8906	3.4723	3.2321	3.43315
pat43032	13.0115	3.928	10.5232	7.7298	8.798125
pat43036	4.7624	2.7189	4.2322	2.0934	3.451725
pat43066	3.7157	2.4529	3.2684	3.4897	3.231675
pat43072	3.6682	2.3003	4.4738	4.2048	3.661775
pat43174	3.8776	2.5449	3.3939	2.67	3.1216
pat43498	4.5993	2.9709	3.6046	4.0129	3.796925
pat43500	9.3325	5.4761	9.9609	6.9171	7.92165
pat43504	3.6552	2.6031	3.5512	2.2588	3.017075
pat43516	4.1652	2.6748	3.4476	2.9405	3.307025
pat43552	3.5768	2.4279	3.2585	3.5378	3.20025
pat43746	3.2418	1.7787	3.5712	3.1548	2.936625
pat43748	3.04	1.7932	2.3322	2.7774	2.4857
pat43752	3.0739	2.0103	3.7457	3.516	3.086475
pat43756	6.2441	4.226	3.6779	5.6444	4.9481
pat43760	2.9711	2.0097	2.2874	3.141	2.6023
pat43762	6.1693	4.0984	3.535	5.5216	4.831075
pat43770	3.4725	2.0749	3.7635	2.812	3.030725
pat43774	6.5951	3.9669	3.3476	3.8943	4.450975
pat43786	7.0643	5.5337	4.683	4.7361	5.504275
pat43796	3.2525	2.0568	2.446	2.538	2.573325
pat43798	6.653	4.0875	3.3525	4.0747	4.541925
pat43804	6.9741	5.362	4.7229	4.7606	5.4549
pat43824	3.2372	1.8429	3.6871	3.2593	3.006625
pat43828	6.0542	3.3302	3.3411	4.8557	4.3953
pat43840	6.2392	4.5047	4.5565	6.0456	5.3365
pat43876	6.8306	4.3766	4.1534	4.2561	4.904175
pat43904	3.0844	1.8447	2.4444	2.7944	2.541975
pat43906	6.1803	3.3178	3.3341	4.8757	4.426975
pat43912	6.2231	4.538	4.6494	6.0924	5.375725
pat4392	3.0315	2.2093	3.1966	3.377	2.9536
pat43930	6.7335	4.2795	4.1236	4.1595	4.824025
pat4398	3.1618	1.9872	2.8161	3.0308	2.748975
pat43986	3.8041	1.9315	3.7436	3.6454	3.28115
pat43987	8.8323	3.7007	3.5412	6.9475	5.755425
pat43989	3.0689	1.5	3.4954	3.0951	2.78985
pat43990	8.6614	3.6915	3.6922	6.828	5.718275
pat4400	2.9635	1.9212	2.8056	3.2212	2.727875
pat44002	9.1263	5.2974	5.7152	9.0423	7.2953
pat44005	5.401	2.5428	3.1135	5.0989	4.03905
pat44038	9.6698	5.0119	4.9961	5.5007	6.294625
pat44041	5.7765	2.6209	2.956	3.8242	3.7944
pat4410	3.5878	2.299	3.1133	2.4953	2.87385
pat44146	8.7055	3.8999	4.94	7.3184	6.21595
pat44149	5.1827	2.0432	2.9177	4.4556	3.6498
pat4422	3.4212	2.3968	3.2005	2.1526	2.792775
pat44228	3.6972	2.0187	2.3647	3.323	2.8509
pat44230	8.7066	3.6826	3.5567	6.8731	5.70475

pat44236	9.1665	5.3756	5.6578	9.0525	7.3131
pat44254	9.7477	4.9181	4.8116	5.3726	6.2125
pat4426	7.5894	5.0677	7.1877	5.1661	6.252725
pat44308	8.8169	3.9195	4.8594	7.399	6.2487
pat4434	3.4896	2.0643	2.9275	2.6062	2.7719
pat4436	3.4689	2.06	2.7802	2.3987	2.67695
pat4440	3.2888	2.388	3.3702	2.5951	2.910525
pat4444	7.5262	4.9614	7.2363	5.0886	6.203125
pat44472	2.8346	1.4593	3.5816	2.8544	2.682475
pat44476	3.2488	2.158	2.4683	2.6809	2.639
pat4448	3.2546	2.3598	3.2374	2.0474	2.7248
pat44488	3.3923	2.7772	2.8827	3.3758	3.107
pat4450	7.5456	5.0149	7.2473	5.0507	6.214625
pat44524	3.6583	2.6243	2.7541	2.8682	2.976225
pat4458	4.1805	2.8382	3.6999	3.3469	3.516375
pat4462	10.5771	5.4989	8.6098	8.777	8.3657
pat44632	3.3977	2.3769	2.7458	2.9561	2.869125
pat4464	3.2974	2.0523	2.9876	3.083	2.855075
pat4474	9.5604	6.9578	11.5438	9.6652	9.4318
pat4476	3.0059	2.073	3.0063	3.0278	2.77825
pat4480	6.7876	3.9773	7.2334	6.988	6.246575
pat44956	4.5979	2.7785	3.2203	4.2701	3.7167
pat4510	10.771	6.7117	9.8387	5.8599	8.295325
pat4512	3.4981	2.1496	3.096	2.6415	2.8463
pat4516	7.6076	4.074	6.2518	4.6898	5.6558
pat45200	2.7121	1.4425	2.6645	2.3022	2.280325
pat45202	3.385	2.1014	2.8116	2.6524	2.7376
pat45208	3.389	2.7302	3.3725	3.2839	3.1939
pat45226	3.6095	2.5688	3.1151	2.6707	2.991025
pat4528	7.0493	5.1503	8.3165	5.1544	6.417625
pat45280	3.3903	2.2378	3.0483	2.7641	2.860125
pat4538	4.3271	2.8273	3.8303	3.6541	3.6597
pat4540	10.4129	5.3675	8.6347	8.7837	8.2997
pat45442	4.5773	2.6247	3.6617	4.0752	3.734725
pat45445	3.0669	1.6133	2.6518	2.4204	2.4381
pat4546	9.572	6.9234	11.5817	10.0185	9.5239
pat4564	11.0118	6.8814	10.0127	6.1789	8.5212
pat45948	4.4015	2.3667	4.4974	4.0419	3.826875
pat45952	6.7272	4.4179	4.1518	5.8124	5.277325
pat45954	6.3065	3.5335	8.1101	5.4668	5.854225
pat45984	3.8831	2.1877	4.9506	4.4262	3.8619
pat45988	6.8922	4.2234	3.8445	4.3965	4.83915
pat45990	4.6492	3.1667	8.7466	7.8044	6.091725
pat46000	7.2733	5.6623	5.1946	5.0704	5.80015
pat46006	4.5282	3.0086	3.5124	3.7447	3.698475
pat46092	4.1376	2.5738	5.0577	4.0942	3.965825
pat46096	6.4513	3.6044	3.8892	5.0569	4.75045
pat46108	6.5893	4.7442	5.1842	6.3073	5.70625
pat46144	7.1807	4.5815	4.6963	4.5945	5.26325
pat4623	3.8061	1.9121	3.0007	3.1651	2.971
pat4626	4.011	2.0249	3.3428	3.6006	3.244825
pat463	6.9677	2.3435	6.2072	4.6137	5.033025
pat4638	3.7963	2.3003	3.8623	3.7796	3.434625
pat4639	9.269	4.0608	9.2109	9.6874	8.057025
pat4641	2.9651	1.5579	2.6897	2.9249	2.5344
pat46416	4.1256	2.7143	3.8468	3.7152	3.600475

pat4642	9.3663	3.9686	9.1565	9.6816	8.04325
pat46422	4.3383	3.8308	6.2238	4.4278	4.705175
pat46432	9.3881	5.4431	6.2966	9.2318	7.5899
pat46434	4.85	3.1529	4.9655	4.2117	4.295025
pat46438	5.7574	2.8659	3.6742	5.3298	4.406825
pat46468	10.0819	5.3633	5.4579	5.6723	6.64385
pat46470	4.1658	3.0336	5.4649	4.8926	4.389225
pat46474	6.0376	3.065	3.5277	4.2616	4.222975
pat46486	6.4776	3.9924	4.6528	4.8788	5.0004
pat46576	9.2522	4.2377	5.4844	7.7368	6.677775
pat466	7.1038	2.4374	6.2997	4.9121	5.18825
pat4674	4.3063	2.2535	3.739	2.5349	3.208425
pat4675	10.29	4.127	7.8169	5.7643	6.99955
pat4677	3.3131	1.6362	2.7939	2.5961	2.584825
pat4678	10.576	4.3052	7.9323	5.9397	7.1883
pat4690	9.6977	5.6989	10.5637	6.6592	8.154875
pat4693	6.5779	2.9846	6.5681	4.9066	5.2593
pat47388	8.0303	2.6807	8.9245	6.5836	6.554775
pat47392	3.7622	2.4856	3.5503	2.6296	3.106925
pat47394	11.7795	5.4742	15.1284	8.781	10.290775
pat47404	4.2086	3.0551	3.73	3.3575	3.5878
pat47406	13.1004	3.9934	10.6908	7.7965	8.895275
pat47410	4.8013	2.754	4.372	2.1333	3.51515
pat47440	3.9106	2.6835	3.5216	3.6409	3.43915
pat47446	3.7313	2.3421	4.5229	4.2036	3.699975
pat47548	3.9972	2.6457	3.5745	2.7618	3.2448
pat4780	12.7741	5.2478	12.1866	11.2613	10.36745
pat4783	9.618	3.0195	8.0245	8.3682	7.25755
pat47872	4.8773	3.1414	3.9367	4.3028	4.06455
pat47874	9.4493	5.4744	10.0337	6.9259	7.970825
pat47878	3.6842	2.6589	3.6301	2.3069	3.070025
pat47890	4.2116	2.6862	3.5547	3.0136	3.366525
pat47926	3.6955	2.474	3.3895	3.5791	3.284525
pat4866	3.8666	2.0603	3.0827	3.2039	3.053375
pat4868	3.9909	1.8646	3.0749	3.4303	3.090175
pat4872	3.5481	2.4817	3.9009	3.7562	3.421725
pat4876	9.3016	4.0323	9.2686	9.6428	8.061325
pat4880	3.5881	2.26	3.9127	3.9662	3.43175
pat4882	9.3793	4.0898	9.3186	9.7539	8.1354
pat4890	4.2002	2.5317	3.6489	2.0393	3.105025
pat4894	10.4551	4.2226	7.7313	5.9337	7.085675
pat4906	9.5944	5.5552	10.599	6.5963	8.086225
pat4916	4.2563	2.2815	3.6207	2.1971	3.0889
pat4918	10.4365	4.2961	7.8795	5.8449	7.11425
pat4924	9.7343	5.886	10.7047	6.5594	8.2211
pat4942	13.0765	5.5842	12.214	11.3352	10.552475
pat4944	3.7799	2.0962	3.4037	3.2958	3.1439
pat4948	9.4441	2.9874	7.8823	8.44	7.18845
pat4960	8.4792	4.0081	10.7225	9.5019	8.177925
pat4996	9.7687	4.3038	9.1211	5.7699	7.240875
pat50304	4.1085	2.6297	3.8013	3.6939	3.55835
pat50308	9.4722	5.539	4.0684	7.0829	6.540625
pat50320	9.9583	7.876	6.145	8.8518	8.207775
pat50356	10.5668	7.2546	5.4236	5.5577	7.200675
pat504	3.0262	2.1525	3.1675	3.344	2.92255
pat50464	9.8373	6.0813	5.4434	7.622	7.246

pat50788	13.2343	6.8233	6.0385	10.1599	9.064
pat510	3.104	1.8349	2.8388	3.005	2.695675
pat5112	2.7701	1.7773	2.5136	2.6047	2.416425
pat512	2.877	1.4885	2.8191	3.2042	2.5972
pat5124	2.8312	1.706	2.4077	2.5724	2.379325
pat5128	3.5389	2.706	4.2219	3.7881	3.563725
pat5160	2.8182	1.6194	2.4792	2.7021	2.404725
pat5164	4.0754	2.7472	3.8362	3.1335	3.448075
pat5176	3.8392	3.4918	4.9292	3.3882	3.9121
pat51760	5.1635	3.945	3.9668	4.276	4.337825
pat522	3.5496	2.421	3.0813	2.572	2.905975
pat52500	3.109	2.0961	3.7056	3.6016	3.128075
pat52504	6.5824	4.4431	4.0627	6.0358	5.281
pat52508	3.067	2.0785	2.4012	3.1682	2.678725
pat52510	6.645	4.4055	3.9343	6.0397	5.256125
pat52518	3.4779	2.091	3.6447	2.7274	2.98525
pat52522	6.8266	4.1513	3.5903	4.0491	4.654325
pat52534	7.3535	5.7561	5.0993	4.8738	5.770675
pat52544	3.3799	2.11	2.5044	2.4701	2.6161
pat52546	7.0071	4.1549	3.6194	4.1946	4.744
pat52552	7.3169	5.6913	5.1195	4.9062	5.758475
pat52572	3.325	1.9344	3.6204	3.252	3.03295
pat52576	6.5445	3.5852	3.7122	5.2148	4.764175
pat52588	6.5056	4.7258	5.125	6.3313	5.671925
pat52624	7.0373	4.4399	4.4672	4.361	5.07635
pat52652	3.1452	1.9322	2.5499	2.9147	2.6355
pat52654	6.4831	3.4191	3.6334	5.1462	4.67045
pat5266	5.5625	4.205	6.5158	5.5517	5.45875
pat52660	6.5096	4.7013	5.117	6.4456	5.693375
pat52678	7.1553	4.4385	4.524	4.3178	5.1089
pat52734	3.9542	2.0652	3.6119	3.7181	3.33735
pat52737	3.1323	1.55	3.4147	3.1315	2.807125
pat52750	9.548	5.5455	6.364	9.3726	7.707525
pat52753	5.6003	2.6103	3.3891	5.4021	4.25045
pat52786	10.1362	5.1945	5.4181	5.6954	6.61105
pat52789	6.1303	2.7506	3.2001	3.9961	4.019275
pat52894	9.1577	3.9331	5.4815	7.7096	6.570475
pat52897	5.589	2.1527	3.1896	4.6809	3.90305
pat52976	3.8372	2.1193	2.4613	3.362	2.94495
pat52984	9.5747	5.5944	6.26	9.4811	7.72755
pat53002	10.221	5.2544	5.3767	5.6603	6.6281
pat53056	9.3584	4.0948	5.3804	7.9214	6.68875
pat53220	2.7812	1.5059	3.3857	2.8124	2.6213
pat53224	3.3684	2.2423	2.5717	2.7838	2.74155
pat53236	3.5009	2.9659	3.1283	3.5428	3.284475
pat53272	3.8269	2.8654	2.9885	2.9811	3.165475
pat53380	3.5083	2.4474	2.9106	3.0641	2.9826
pat534	3.4009	2.1782	3.1936	2.2055	2.74455
pat53704	4.8952	2.9788	3.5408	4.4413	3.964025
pat538	7.056	3.1734	7.3025	5.2182	5.687525
pat53948	2.6872	1.4855	2.6044	2.3633	2.2851
pat53950	3.4205	2.1743	2.8634	2.6927	2.787725
pat53956	3.4533	2.8983	3.5173	3.4392	3.327025
pat53974	3.8192	2.7564	3.2395	2.8168	3.157975
pat54	3.7669	2.2659	3.2129	2.931	3.044175
pat54028	3.52	2.3838	3.1846	2.9347	3.005775

pat54190	4.7371	2.8422	3.8549	4.3222	3.9391
pat54193	3.1456	1.6427	2.6723	2.5278	2.4971
pat546	3.475	2.0088	2.8933	2.5375	2.72865
pat54678	4.0786	2.5956	3.6624	3.6676	3.50105
pat54682	9.4391	5.5225	4.0581	6.912	6.482925
pat54694	9.9323	7.8896	6.1728	8.923	8.229425
pat54730	10.571	7.3419	5.4032	5.578	7.223525
pat548	3.3239	1.6517	2.7586	2.4028	2.53425
pat54838	9.6548	5.9281	5.3358	7.5822	7.125225
pat55162	13.265	6.7672	6.0786	10.3058	9.10415
pat552	3.1924	2.2467	3.3586	2.5677	2.84135
pat556	7.1772	3.2464	7.3522	5.224	5.74995
pat5592	2.8169	1.758	2.5397	2.5004	2.40375
pat5596	5.0395	2.4242	5.0449	4.9444	4.36325
pat560	3.0299	1.7887	3.1836	2.0404	2.51065
pat5608	4.3821	3.1947	6.6102	5.7651	4.988025
pat56134	5.1627	4.0466	3.932	4.3916	4.383225
pat562	7.1009	3.1702	7.2744	5.2217	5.6918
pat5644	5.5735	3.4124	6.0343	4.1393	4.789875
pat56864	4.0368	2.6872	2.4768	3.3998	3.15015
pat56866	9.6353	5.6374	3.8732	7.0958	6.560425
pat56872	9.9616	7.872	6.0447	9.0265	8.2262
pat56890	10.5114	7.093	5.2877	5.4404	7.083125
pat56944	9.8177	6.1764	5.3451	7.6526	7.24795
pat57106	13.0767	6.5914	6.0935	10.2031	8.991175
pat57109	8.2887	3.5879	3.403	6.4902	5.44245
pat57592	5.2293	4.0975	3.5717	4.4776	4.344025
pat576	3.3207	2.1347	2.9484	3.0995	2.875825
pat5838	4.7737	1.9875	3.2514	3.9228	3.48385
pat5840	2.7111	1.5987	2.3068	2.3879	2.251125
pat5844	4.564	2.0886	3.3744	4.0373	3.516075
pat5848	3.5046	2.8322	4.3531	3.745	3.608725
pat5852	2.9014	1.7291	2.5183	2.7893	2.484525
pat5854	3.7094	2.7815	4.2614	3.8075	3.63995
pat5862	4.8192	2.1033	3.3392	3.8627	3.5311
pat5866	4.1041	2.8384	3.9656	3.0402	3.487075
pat5878	3.9946	3.4578	4.902	3.4813	3.958925
pat588	2.987	1.8606	2.954	3.0345	2.709025
pat5888	2.7619	1.6018	2.3695	2.827	2.39005
pat5890	4.0305	2.6612	3.8542	3.3396	3.471375
pat5896	3.7511	3.3662	4.8199	3.7517	3.922225
pat5914	5.4568	4.1964	6.5554	5.5131	5.430425
pat5916	4.6403	1.9804	3.2702	3.937	3.456975
pat592	5.887	1.9668	6.8904	6.6493	5.348375
pat5920	3.7634	2.4253	3.8633	3.4153	3.366825
pat5932	3.4892	2.9073	4.6024	3.8454	3.711075
pat5968	3.9412	2.7847	4.2403	3.47	3.60905
pat6078	4.7571	2.0116	3.5347	4.219	3.6306
pat6079	4.9302	2.4901	5.0903	5.0469	4.389375
pat6081	4.7729	1.8819	3.2608	4.0223	3.484475
pat6082	5.0334	2.5337	5.1178	4.8552	4.385025
pat6094	4.5865	3.2662	6.6566	5.7679	5.0693
pat6097	3.3536	1.893	3.9366	3.6579	3.210275
pat6130	5.4584	3.2504	5.9567	4.4678	4.783325
pat6133	3.546	1.8265	3.4772	3.2201	3.01745
pat624	3.4046	2.0979	3.0529	2.5528	2.77705

pat628	6.9585	2.3175	6.1479	4.6777	5.0254
pat6320	2.8172	1.6481	2.6148	2.7694	2.462375
pat6322	4.9224	2.5224	5.122	4.9164	4.3708
pat6328	4.3949	3.4586	6.8582	5.9065	5.15455
pat6346	5.5282	3.5447	6.1355	4.2204	4.8572
pat640	6.4125	3.1851	8.1894	5.1357	5.730675
pat6400	4.6045	2.5854	5.7079	4.9709	4.467175
pat654	3.4052	2.0617	2.783	3.0077	2.8144
pat656	3.1662	1.5162	2.6974	2.9054	2.5713
pat6588	3.9895	2.615	3.5118	3.688	3.451075
pat660	2.9407	2.1232	3.1128	3.3166	2.873325
pat6624	4.0088	2.6123	3.7038	3.7067	3.5079
pat6636	3.9017	2.5421	3.4858	2.9035	3.208275
pat664	6.0445	2.1788	7.1183	6.8642	5.55145
pat6640	7.8723	5.1891	7.776	5.4397	6.569275
pat668	2.701	1.5019	3.0713	3.3031	2.644325
pat670	5.9882	2.0295	6.994	6.8396	5.462825
pat6726	4.4338	3.0015	3.9775	3.6392	3.763
pat6730	11.0234	5.6589	9.3809	9.0411	8.776075
pat6732	4.2126	2.9564	3.6713	3.8583	3.67465
pat6742	9.8147	7.1062	12.2687	10.0314	9.80525
pat6748	6.8857	4.0973	7.665	7	6.412
pat6778	10.8992	6.678	10.3864	5.9988	8.4906
pat678	3.5417	2.3259	3.0442	2.4351	2.836725
pat6784	7.8549	4.3442	6.7063	5.0026	5.977
pat682	7.0343	2.4227	6.1609	4.6559	5.06845
pat694	6.366	3.1065	8.137	5.1195	5.68225
pat7056	4.2796	2.5438	3.5159	3.6176	3.489225
pat7068	3.9369	2.6693	4.0527	3.8294	3.622075
pat7072	9.7822	4.2459	10.0166	10.0164	8.515275
pat7104	4.579	2.8507	3.9656	2.4935	3.4722
pat7108	10.62	4.2427	8.4322	6.048	7.335725
pat7120	9.9701	5.8716	11.3949	6.7586	8.4988
pat72	3.1754	2.2288	3.4604	3.8354	3.175
pat7210	13.15	5.6521	12.8406	11.4566	10.774825
pat7212	4.1236	2.5321	3.6835	3.561	3.47505
pat7216	9.7017	3.0073	8.4077	8.5949	7.4279
pat7228	8.5784	3.9564	11.2714	9.8225	8.407175
pat756	3.6629	2.0988	2.8708	3.1094	2.935475
pat78	3.4421	1.8432	2.8027	3.0462	2.78355
pat792	3.1187	1.932	2.9855	4.0165	3.013175
pat80	2.7834	1.7213	2.6495	2.2968	2.36275
pat8028	7.6769	3.8743	5.4541	5.9608	5.741525
pat804	3.3347	1.6755	2.4616	3.1263	2.649525
pat8040	7.3833	2.9023	3.6344	5.0713	4.747825
pat8044	4.154	3.0735	4.5638	3.8214	3.903175
pat8076	6.6787	2.5138	3.9109	7.34	5.11085
pat808	2.9929	2.1295	3.3387	2.7	2.790275
pat8080	4.4006	3.0016	4.2706	3.8143	3.871775
pat8092	4.3204	3.579	5.0403	3.89	4.207425
pat8182	5.7264	4.4124	6.9375	5.6941	5.6926
pat8184	7.7851	3.6416	3.9263	5.6279	5.245225
pat8188	4.3601	2.7873	4.1618	3.4464	3.6889
pat8508	5.1957	3.2504	3.5379	4.1631	4.036775
pat8512	5.2784	2.851	5.5432	5.092	4.69115
pat8524	4.6827	3.4891	7.0319	5.9039	5.2769

pat8560	5.6956	3.6628	6.4452	4.5749	5.094625
pat8668	4.9297	2.879	6.0136	5.0284	4.712675
pat900	3.7486	2.6658	3.1981	3.4618	3.268575
pat916	2.8527	1.9411	3.2607	2.7832	2.709425
pat952	3.1052	1.9917	3.0131	2.8312	2.7353

BIBLIOGRAPHY

- [Andras 2003a] András Varga OMNet++ User Manual, Budapest University of Technology and Economics.
- [Andras 2003b] András Varga OMNet++ Online help, Budapest University of Technology and Economics.
- [Bernd 2000] Bernd Bruegge, Allen H Dutoit – Object Oriented Software Engineering. Prentice Hall. 2000.
- [Hassan] Hassan Gomaa, Designing Concurrent, Distributed, and Real Time Applications with UML.
- [William 2001] William J, Ray, Valdis Berzins. Optimization of Distributed Object-Oriented Servers. Naval Postgraduate School. Sept 2001.
- [William 2002] William J, Ray, Luqi, Valdis Berzins. Optimizing Systems by Work Schedule. Naval Postgraduate School.
- [William 2003] William J, Ray SW4555 – Engineering Network Centric Systems, Naval Postgraduate School. 2003/3.
- [William 2004] William J, Ray SW4599 – Automated Hardware/Software Integration, Naval Postgraduate School. 2004/1

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. Tan Lai Poh
Temasek Defense System Institute
Singapore
4. Seah Siew Hwee, Human Resource Department
Defence Science Technology Agency
Singapore