INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Sensitivity Analysis of Performance Predictions for Distributed Applications

by

Fahim A. Sheikh, B.Eng.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements of the degree of

Master of Engineering

Ottawa-Carleton Institute for Electrical Engineering
Faculty of Engineering
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario
CANADA, K1S 5B6

January 27, 1998

©1998, Fahim A. Sheikh

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-27026-2

Canada

The undersigned recommend to the Faculty of Graduate Studies
and Research the acceptance of the thesis

Sensitivity Analysis of Performance Predictions for Distributed Applications

submitted by Fahim A. Sheikh, B. Eng in partial fulfillment of the
requirements for the degree of Master of Engineering

Chair, Department of Systems and Computer Engineering

Thesis Supervisor

Carleton University
January 27, 1998

Abstract

The goal of software performance engineering is to provide solid feedback to designers and developers as early as possible in the system development cycle. This can potentially mean lower costs in the long run. In distributed system designs often parameters are based on estimates. By performing a Sensitivity Analysis on a model of a system one can help determine which parameters are critical to system performance. Designers and developers can then use this information to diagnose problems and improve designs. A rapid and exhaustive approach for obtaining parameter sensitivities/elasticities on analytic models is designed. This approach is integrated to the existing Layered Queueing Modelling framework. The approach is then evaluated using a number of client-server systems architectures including a Call Processing Application. A study on how sensitivities can be used to identify bottlenecks and optimize designs is also carried out.

Acknowledgements

I would like to thank my supervisor, Murray Woodside for his guidance, encouragement and support in my research. Working with Jerry Rolia has also been a very rewarding experience. The knowledge and experience of these two “performance gurus” is inspiring.

I would like to thank my parents and my sisters for all their support and understanding during the course of my studies. I wish to also thank my wonderful fiancé for all her encouragement and comfort during my Masters. I love you all!!!

Many thanks to all my friends and colleagues (a.k.a. RADS Rats). You guys (I.A., C.S., G.F., A.H., J.I., M.C., O.A., G.S., H.E., L.S., and others) have been great!!!

Financial assistance provided by the National Science and Engineering Research Council of Canada and Carleton University was greatly appreciated.

Table of Contents

Abstract	.iii
Acknowledgements	.iv
Table of Contents	.v
List of Figures	.x
List of Tables	.xiii
Glossary Of Terms	.xv
Chapter 1 Introduction	.1
1.1 Motivation	1
1.2 Layered Queueing Modelling	4
1.3 Thesis Goal	5
1.3.1 Thesis Contributions	5
1.4 Thesis Outline	6
Chapter 2 Background	.7
2.1 Sensitivity Analysis	7
2.2 Previous Work on Sensitivity Analysis	8
2.2.1 Related Research	9
2.3 Sensitivity Analysis in Queueing Networks	13
Chapter 3 Layered Queueing Modelling	.18
3.1 Analytic Modelling	18
3.2 Layered Queueing Models	19
3.2.1 Model Components	19
3.2.2 Model Parameters	21
3.2.3 Multi-servers	23
3.2.4 Performance Results or Model Outputs	23
3.3 Layered Queueing Network Solver	24
3.3.1 Class hierarchy of the LQNS Solver	24
3.3.2 Hierarchical Decomposition	26
3.3.3 Layered Queueing Network Solver Operation	27

Chapter 4 Sensitivity Analysis in Layered Queueing Models29

4.1	Sensitivity Analysis in Layered Queueing Models	29
4.1.1	Mean Value Analysis	30
4.2	Bard-Schweitzer Approximate MVA	30
4.2.1	The Bard-Schweitzer Approximate MVA Equations	31
4.2.2	Partial derivatives of the Bard-Schweitzer Approximate MVA Equations	32
4.2.3	Interaction Equations: Coupling between MVA Submodels	34
4.2.4	Partial Derivatives of Interaction Equations	37
4.3	Worked Example of a Simple Client-Server System . . .	37
4.3.1	Submodel Equations for the Simple Client-Server System . .	38
4.3.1.1	Submodel I	38
4.3.1.2	Submodel II	42
4.3.2	Interaction between MVA Submodels for the Simple Client-Server System	44
4.4	Assembly of System Matrix	44
4.4.1	Strategy for Solving for the Sensitivities	46
4.4.1.1	An Artificial Example	48
4.5	Traffic Dependencies in Client-Server Systems	53
4.5.1	Adjustment for Interlocking Effects	54
4.6	Waiting times equations and scheduling disciplines . . .	56
4.6.1	FCFS Scheduling with Multiple Heterogeneous Classes	57
4.7	Sensitivities at Multi-servers	59
4.8	Deliberate Limitations to the Analysis	61

Chapter 5 lqnSENS: Implementation of the Approach . . .62

5.1	lqnSENS	62
5.2	Why Perl?	63
5.3	Model Solution	63
5.3.1	Adding the Sensitivity Option to the Solver	64
5.3.2	Sensitivity header file	64
5.3.3	Derivatives and iterations	66
5.3.4	Instrumenting and Extending the LQNS Solver	67
5.3.5	Instrumenting for the Derivatives	68
5.4	Stage I : Reduction and Matrix Generation	69
5.4.1	The .der file	70

5.4.1.1	Derivative File Reduction	71
5.4.1.2	Reduction Algorithm	72
5.4.2	The .int file	72
5.4.3	The .id file	73
5.5	Stage II: Sparse Matrix Operations	75
5.5.1	Stage III: Translation / Identification	78
5.5.2	Stage IV: Ranking	79
Chapter 6	Results	80
6.1	Simple Client-Server Case with no Interlocking	81
6.2	Second Simple Client-Server Case with Interlocking	87
6.2.1	Example with Processor Sharing	89
6.2.2	Example with First-Come First-Serve (FCFS)	93
6.2.3	Example with FCFS Scheduling with Multiple Heterogeneous Classes	97
6.3	Multiple Client Chains	101
6.3.1	Multiple Client Chains with no Interlocking	101
6.3.2	Multiple Client Chains with Interlocking	103
6.4	Distributed Database Server	105
6.5	Multi-level Case with two levels of Interlocking	109
6.6	Discussion	113
Chapter 7	Call Processing Application	115
7.1	Call Processing System	115
7.1.1	System Architecture	115
7.1.2	Layered Queueing Model of the Call Processing System ...	116
7.1.3	Discussion	123
Chapter 8	Using the Sensitivity Information	125
8.1	Bottleneck Identification in the Distributed Database Server Case	125
8.1.1	Effect of Dispatcher Population on Client Throughput	125
8.1.2	Effect of Average Visits from Client to Dispatcher on Client Throughput	129
8.2	The Multi-level Case Revisited	131
8.2.1	Improving Client Throughput via Server Cloning	136
8.2.2	Using Sensitivity Information to Determine Cloning Strategy	139
8.2.3	Sensitivity based Cloning Strategy	140

8.2.3.1	Elasticity based Cloning Strategy	143
8.2.4	Results	146
8.3	Discussion	146
Chapter 9	Conclusions	148
	Future Work	149
References		151
Appendix A		155

Carleton University
Ottawa, Canada K1S 5B6

Text complete; leaf ~~IX~~ omitted in numbering.

List of Figures

Figure 1-1	Two Approaches for determining parameter sensitivities	3
Figure 2-1	Identifying regions of sensitivity	11
Figure 2-1	Sensitivity Analysis of Product-form and Layered Queueing Networks	14
Figure 3-1	Layered Queueing Network Model: Notation	20
Figure 3-2	Layered Queueing Network Model Components	21
Figure 3-3	Execution slices in Layered Models	22
Figure 3-4	LQNS Class Hierarchy	25
Figure 3-5	Example LQN Model.	26
Figure 3-6	Decomposition of a Layered Model into MVA submodels.	28
Figure 4-1	Layered Model and MVA Submodels for a Client-Server System	35
Figure 4-2	Determining the service time of the Client based on the response of the Client at servers in other layers	35
Figure 4-3	Determining the service time of the Server based on the response of the Server at servers in other layers.	36
Figure 4-4	Layered Model of a Simple Distributed Client-Server System.	38
Figure 4-5	Submodel I of the Simple Client-Server System.	39
Figure 4-6	Simple Client-Server System Submodel I Matrix (A1).	41
Figure 4-7	Submodel II of the Simple Client-Server System	42
Figure 4-8	Simple Client-Server System Submodel II Matrix (A2)	43
Figure 4-9	System Matrix Structure	45
Figure 4-10	Internal and External parameters with three submodels	46
Figure 4-11	(a) Software Interlock (b) Hardware Interlock	54
Figure 5-1	Input and output to the lqnSENS tool	63
Figure 5-2	The stages within lqnSENS	64
Figure 5-3	Inside lqnSENS	65
Figure 5-4	Invoking the LQNS Solver	66
Figure 5-5	Iteration flags	67
Figure 5-6	Highlighted classes were instrumented	68

Figure 5-7	Instrumentation of a utilization equation in mva class of the LQNS Solver	69
Figure 5-8	The Derivative (.der) file	70
Figure 5-9	Determining the interactions between submodels, for “wait” variables	74
Figure 5-10	Derivative and Matrix Generation Stage	76
Figure 5-11	Setup of a Submodel matrix.	77
Figure 5-12	Setup of a System matrix.	77
Figure 5-13	Translation and Identification Stage	78
Figure 6-1	Simple Client-Server Case with no Interlocking.	81
Figure 6-2	Ranking of Elasticities for the Simple Client-Server case with no Interlocking.	88
Figure 6-3	Simple Client-Server Case with Interlocking and PS	89
Figure 6-4	Client throughput as the Client population is increased	91
Figure 6-5	Simple Client-Server Case with Interlocking and FCFS.	94
Figure 6-6	Simple Client-Server Case with Interlocking and HV-FCFS	98
Figure 6-7	Multiple Client Chains with no Interlocking	102
Figure 6-8	Multiple Client Chains with Interlocking	104
Figure 6-9	Distributed Database Server	107
Figure 6-10	Multi-level Case with two levels of Interlocking	111
Figure 7-1	Call Processing System Hardware Infrastructure [ppj91]	116
Figure 7-2	Layered Queueing Model of the Call Processing System	118
Figure 7-3	Call Processing System Submodels.	119
Figure 7-4	Ranking of elasticities for the Call Processing System	121
Figure 8-1	Ranking of Elasticities for the Distributed Server Case	127
Figure 8-2	Sensitivity of Client throughput wrt to Dispatcher population for increasing number of Dispatcher clones	128
Figure 8-3	Elasticity of Client throughput wrt to Dispatcher population for increasing number of Dispatcher clones	128
Figure 8-4	Client throughput for increasing number of Dispatcher clones.	129
Figure 8-5	Client throughput as the avg number of visits from Client	

	to Dispatcher are varied	130
Figure 8-6	Sensitivity of Client throughput to avg. visits from Client to Dispatcher.	131
Figure 8-7	Elasticity of Client throughput to avg. visits from Client to Dispatcher.	131
Figure 8-9	Sensitivity of Client throughput to the average visits from Client to server S2.	132
Figure 8-8	Ranking of Elasticities for the Multi-level Case	133
Figure 8-10	Elasticity of Client throughput to the average number of visits from Client to server S2.	134
Figure 8-11	Client throughput as the average visits from Client to server S2 are increased	134
Figure 8-12	Sensitivity of S2 throughput to the average number of visits from Client to S2	135
Figure 8-13	Elasticity of S2 to the average visits from Client to S2.	135
Figure 8-14	S2 throughput as the average number of visits from Client to S2 are increased	136
Figure 8-15	Ranking of elasticities for the Multi-level Case II.	138
Figure 8-16	Multi-level Case II.	139
Figure 8-17	Sensitivity based Cloning: Sensitivity of Client throughput to the multiplicity of the servers in the model	141
Figure 8-18	Sensitivity based Cloning: Elasticity of Client throughput to the multiplicity of the servers in the model	141
Figure 8-19	Sensitivity based Cloning Decision Stages	142
Figure 8-20	Elasticity based Cloning Decision Stages	144
Figure 8-21	Elasticity based Cloning: Sensitivity of Client throughput to the multiplicity of the servers in the model	145
Figure 8-22	Elasticity based Cloning: Elasticity of Client throughput to the multiplicity of the servers in the model	145
Figure 8-23	Client throughput	146

List of Tables

Table 3-1 :	Class Description Table	25
Table 6-1 :	Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)	84
Table 6-2 :	Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0).....	84
Table 6-3 :	Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)	85
Table 6-4 :	Sensitivity/Elasticity wrt to Client Think Time (Z=8)	86
Table 6-5 :	Sensitivity/Elasticity wrt to Client Population (N=2).....	86
Table 6-6 :	Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)	91
Table 6-7 :	Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0)	92
Table 6-8 :	Sensitivity/Elasticity wrt to Visits from Client to Server [E1] (V=2)	92
Table 6-9 :	Sensitivity/Elasticity wrt to Client Think Time (Z=8)	93
Table 6-10 :	Sensitivity/Elasticity wrt to Client Population (N=2).....	93
Table 6-11 :	Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)	94
Table 6-12 :	Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0).....	95
Table 6-13 :	Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)	95
Table 6-14 :	Sensitivity/Elasticity wrt to Client Think Time (Z=8)	96
Table 6-15 :	Sensitivity/Elasticity wrt to Client Population (N=2).....	96
Table 6-16 :	Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)	98
Table 6-17 :	Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0).....	99
Table 6-18 :	Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)	99
Table 6-19 :	Sensitivity/Elasticity wrt to Client Think Time (Z=8)	100
Table 6-20 :	Sensitivity/Elasticity wrt to Client Population (N=2).....	100
Table 6-21 :	Sensitivity/Elasticity wrt to Service Time of Client 1 (ST=2.0).....	103
Table 6-22 :	Sensitivity/Elasticity wrt to Service Time of Client 1 (ST=2.0).....	104
Table 6-23 :	Sensitivity/Elasticity wrt to Service Time of Dispatcher (ST=2).....	108
Table 6-24 :	Sensitivity/Elasticity wrt to Visits from Client to Dispatcher [D_ENTRY] (V=2)	109
Table 6-25 :	Sensitivity/Elasticity wrt to Service Time of S3 at P4 (ST=4)	112
Table 7-1 :	Sensitivity/Elasticity wrt to Visits from Term Integrated Device Driver to Term Agent Factory (V=1).....	120

Table 7-2 :	Sensitivity/Elasticity of the utilization of some processes to the population of the Client (TM=182638)	123
Table 8-1 :	Sensitivity/Elasticity of Client Throughput wrt Dispatcher Multiplicity . . .	126
Table 8-2 :	Sensitivity/Elasticity of Client Throughput wrt Avg Visits from Client to Dispatcher [E1]	129
Table 8-3 :	Sensitivity/elasticities of software servers	136
Table 8-4 :	Effect on Client throughput as software server multiplicity is increased	137

Glossary Of Terms

$\frac{\partial y}{\partial x}, \frac{\Delta y}{\Delta x}$	sensitivity of output parameter y to input parameter x
E_{yx}	elasticity of output parameter y to input parameter x
$D_{A,B}$	demand of class A at server B
$S_{A,B}$	service time of class A at server B
$V_{A,B}$	avg. visits from class A to server B
R_A	response time of class A
$R_{A,B}$	response time of class A at server B
Q_B	queue length at server B
$Q_{A,B}$	queue length of class A at server B
X_A	throughput of class A
ρ_B	utilization of server B
N_A	population of class A
Z_A	think time of class A
$R_{A,B}^{[m]}$	response time of class A at server B in submodel m
$S_A^{[m]}$	service time of server A in submodel m
ξ	any arbitrary parameter; a dummy variable
TM	task multiplicity

ST[P1,P2]	service time; P1 phase 1; P2 phase 2
Z	think time
V	visits
FCFS	First-Come First-Serve
PS	Processor Sharing
HV-FCFS	High Variance FCFS
E1	entry 1
Px	processor x

Chapter 1

Introduction

1.1 Motivation

Large distributed computer systems and networks are important to industry and society. Businesses are globalizing and the need to share information quickly and efficiently among many organizational units is paramount. Client-server computing achieves these goals by using a higher level of integration in Information Systems. With client-server computing, corporations can increase efficiency of production or service delivery and stay competitive in a rapidly evolving marketplace. Business banking, telecommunications, manufacturing, government services and natural resources are all industries in which client-server computing technology is being deployed. For example, being able to provide more responsive service in the corporate banking world can mean getting a leg up on the competition. Faster response and higher transaction throughput are aspects of performance.

Software performance engineering aims at understanding, predicting and improving the performance of software systems at design time, during development and in operation. Being able to predict the performance of a proposed system as early as possible and in step with the system development cycle can potentially mean lower costs in the long run. Identifying performance problems such as overloaded resources or bottlenecks at design time so that changes at a later stage (which can be more expensive) are avoided. However, software performance engineering still suffers from weak integration with the design process because it fails to provide solid feedback to designers and developers. In today's very competitive industries, time-to-market pressure is forcing firms to adopt an approach that focuses on development first and performance optimization later. In a number of cases this approach has actually resulted in an increase in the time to market and development cost because system prototypes suffered from very poor performance.

The design of these systems is a severe challenge. System parameters such as the computational work for a user request may be difficult to determine, or may change as users become more sophisticated leading to a severe impact on system performance. In addition, the behaviour of these applications can depend on hundreds of different system parameters. As a result, for the designer of these types of applications knowing the impact of input parameters on performance predictions can be non-intuitive and challenging.

Sensitivity analysis can help determine which parameters are critical to system performance. Designers and developers can then use these results to improve designs or optimize performance. For example, in design time modelling execution times of procedures or processes are often estimates provided by designers. By performing a sensitivity analysis one can determine which of these service times are affecting performance predictions most (response time or throughput). As a result, designers can concentrate on these values. This helps integrate software performance engineering as early as possible in the system development cycle.

Simulation and analytic techniques are two approaches that exist for modelling distributed applications. In simulation one can describe models with greater detail than analytic models however simulation models typically have long executions times and one can also get bogged down debugging and verifying that the simulation is correct. Analytic models on the other hand are based on mathematical expressions. Analytic models have faster solutions times making it easier to explore different design alternatives easily. Disadvantages of analytic models are that often one has to make more modelling assumptions.

Two approaches exist for obtaining parameter sensitivities (see Figure 1-1). The *Perturbation Approach* determines sensitivities by calculating the slope of output parameters to input parameters at some operating point X_i^* . The slope is then determined using the difference in the output parameter and the change in the input parameter, or interval width. Therefore, the sensitivity coefficients, $S_{i,j}$ are determined as such,

$$S_{i,j} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y_i}{\Delta x_j}$$

where $x_1, \dots, x_n$ are model inputs and $y_1, \dots, y_n$ are model outputs.

We name the second approach for determining parameter sensitivities as the *Direct Approach*. The Direct approach applies to analytic models where partial derivatives of the equations used to form the analytic solution are determined. The partial derivative information gives the slope. This approach is further described in Chapter 4 of this thesis. Essentially the sensitivity coefficients, $S_{i,j}$, are

$$S_{i,j} = \frac{\partial y_i}{\partial x_j}$$

Figure 1-1 Two Approaches for determining parameter sensitivities

The disadvantage of calculating the sensitivities using the perturbation approach is that the model has to be solved $2n$ times for n parameters. Depending on solution time this can be computationally very expensive and even infeasible [andres95][banks95]. Furthermore, additional processing is required to calculate the sensitivities from the solution results.

Further motivation for this research was a distributed application that was modelled by [sheikh97] and [garg]. In these studies, both simulation and analytic models were used to model a proposed design of Hewlett-Packard's next generation distributed network

management application. The goal of this study was to understand the performance and scalability of this application and identify possible design problems that might limit this application from supporting its intended target environment. The application modelled was a CORBA¹ based application and it consisted of five layers of service. The design also contained over 100 different parameters.

In our study with Hewlett-Packard the perturbation approach was used with both Simulation and Analytic models. Depending on the configuration of the simulation for 100 parameters this approach was found to be computationally expensive (many hours), at the same time perturbation using the analytic model took almost an hour. Therefore, it was decided to investigate a Direct approach to obtaining sensitivities.

1.2 Layered Queueing Modelling

Layered Queueing modelling is an analytic modelling technique developed for modelling distributed applications. Layered Queueing Networks (LQN) are a special form of models for distributed applications[srvn95][rolia95]. Previous research investigated Sensitivity Analysis (SA) techniques for Queueing Networks (QN) [liu91][opdahl96]. QN's however can not represent logical resources such as a processes, critical sections or locks in a natural way, rather special approximations are required. In LQN models, software and hardware can be represented within one model where no special approximations are required. The framework provides a transparent representation of the software architecture, which makes models easy to develop and understand and represents logical resources such as processes, critical sections or locks. The solution approach is based on Approximate Mean Value Analysis. A number of tools such as LQNS², MOL³ that support the LQN framework for performance analysis exist however none have support for performing a sensitivity analysis.

1. Common Object Request Broker [corba]

2. Layered Queueing Network Solver, written G. Franks, Carleton University

3. Method of Layers Solver, written by J. Rolia, Carleton University

1.3 Thesis Goal

The goal of this thesis was to design, implement, and evaluate a rapid and exhaustive analytic approach to determine sensitivities of performance predictions to model input parameters.

1.3.1 Thesis Contributions

An approach for evaluating parameter sensitivities in Distributed Application models was developed. It is a direct approach making it very efficient where sensitivities of all *output* parameters with respect to all *input* parameters can be obtained using only **one** model solution. Matrix based techniques were used on the partial derivatives of the equations of the systems to determine the sensitivities.

The sensitivity analysis approach was integrated to the existing Layered Queueing Modelling framework, in particular to the Layered Queueing Network Solver. The implementation of the approach has been designed such that future extensions to the solution technique can easily be adapted to be included as part of the sensitivity analysis.

The approach was found to be almost two times faster for small models and ten times faster for larger models than the traditional perturbation based approach.

In addition to the parameter sensitivities, parameters *elasticities* are also computed. Elasticities are percentage changes of output values to input parameters. The output from the sensitivity analysis technique is a ranking of elasticities from most elastic to least. This approach can be easily repeated for different values of input parameters. The model is simply re-solved with the new parameters and sensitivities are based on that solution.

Sensitivities and elasticities were used to detect bottlenecks in client-server systems. These numbers were also used to make decisions on how to improve the throughput of the system.

To summarize, the following are the contributions made by this thesis:

- [1] Design of a rapid and exhaustive direct approach of determining sensitivities

in Layered Queueing Models

- [2] Implementation of the approach to the Layered Queueing Network Solver
- [3] Evaluation of the approach using a number of case studies
- [4] Demonstration of how sensitivities/elasticities can provide insight into the performance of distributed applications

1.4 Thesis Outline

The next chapter, Chapter 2, furthers motivates the need of performing a sensitivity analysis by describing how sensitivity information has been used to gain further insight into other studies. This chapter describes related research, where sensitivity analysis has been performed on product-form type queueing networks.

Chapter 3 outlined the Layered Queueing Modelling framework and the approach that the Layered Queueing Modelling Solver (LQNS) uses to solve layered systems.

Chapter 4 discusses an overview of the sensitivity analysis technique as applied to Layered Queueing Models. It is motivated by a simple client-server example. Issues such as adjustments for interlocking and multi-servers is also touched upon.

The implementation of the approach discussed in Chapter 4 is described in Chapter 5. Issues such as instrumenting the LQNS solver, processing output files, generation of matrices, solution of matrices etc. are also discussed in this chapter.

Chapter 6 describes different client-server systems that were studied to test the direct sensitivity analysis technique. The sensitivities obtained using the approach described in Chapter 3 are compared to the perturbation approach. This comparison is based on accuracy and speed.

In Chapter 7, the sensitivity analysis technique is applied to a Prototype of a Call Processing Application. Results of this analysis are presented as well.

The sensitivity analysis information is used in Chapter 8 to identify and resolve bottlenecks using some of the earlier examples presented in Chapter 6.

Finally, in Chapter 9, conclusions are drawn.

Chapter 2

Background

In this chapter, Sensitivity Analysis will be defined and its need further motivated. Previous work on sensitivity analysis and in particular how sensitivity analysis has been applied to queueing networks will be discussed.

2.1 Sensitivity Analysis

“A procedure to determine the sensitivity of the outcomes of an ALTERNATIVE to changes in its parameters (as opposed to changes in the ENVIRONMENT). If a small change in a parameter results in relatively large changes in the outcomes, the outcomes are said to be sensitive to that parameter. This may mean that the parameter has to be determined very accurately or that the alternative has to be redesigned for low sensitivity”

- The Web Dictionary of Cybernetics and Systems¹

Sensitivity analysis, also referred to as what-if analysis, is applied to analytic or simulation models of very different types of systems found in a broad number of disciplines such as chemistry, physics, economics, and of course engineering. Sensitivity analysis is simply the study of the relationship or correlation between the predicted outputs of a model to its inputs.

A sensitivity analysis can prove very useful in the design or understanding of the behaviour of a system. In a model of a system where there are a number of uncertain parameters, by performing a sensitivity analysis those parameters can be screened and ranked. By doing so, we can identify critical regions in the input parameter space. Sensitivity information can also be used in system optimization. For example if one is interested in minimizing the call processing time of a telephone switch, one can create a model of the telephone switch, perform a sensitivity analysis to determine which parameters affect system response the most. By optimizing those sensitive parameters it is possible that one can reduce the call processing time on the telephone switch.

1. <http://pespmc1.vub.ac.be>

In summary, sensitivity analysis can be useful for the following:

- Determine the most sensitive parameters in the system
- Identify areas where changes can be made to improve system performance and identify hot spots
- Use sensitivity information for system optimization (ie. gradient based optimization)
- Identify areas of risk to guide future design and development

2.2 Previous Work on Sensitivity Analysis

Sensitivity analysis is carried out in wide spectrum of disciplines ranging from Mechanics to Economics. An international symposium (Sensitivity Analysis of Model Output - SAMO) dedicated to sensitivity analysis has been organized to bring these people together to exchange ideas. One goal of this organization is to strengthen the role of sensitivity analysis as a prerequisite for model building in all settings - be it diagnostic or prognostic - and all fields - be it exact or social sciences - where models are used and developed.

The sensitivity analysis methods employed may differ considerably from one discipline to another, depending upon the particular problem being addressed in the analysis. The goal however is to use sensitivities for screening active factors in systems with many uncertain parameters, for identifying crucial regions in the input parameter space, for model calibration in the presence of uncertainties, and as a tool to assist with complex models. Sensitivity analysis may be crucial in model verification and quality assurance.

Traditional approaches for performing a sensitivity analysis and the choice of most analysts today is to observe the output while varying one and only one variable at a time [reilly95]. This approach was referred to as perturbation in Chapter 1. The direct approach as described in Chapter 1 is often used where analytic (as apposed to simulation models) models of systems can be created. By creating a set of balanced equations and

solving the system of partial differential equations sensitivities can be obtained. The partial differential equations are derived for each input parameter and then solved. This is slightly different from the approach presented in Chapter 4 of this thesis. In this thesis a set of so-called *general* partial differential equations are determined so that sensitivities of all output parameters with respect to all input parameters can be determined in one step, rather than setting up different partial differential equations for each input parameter. By getting all the sensitivities at once, one can easily screen parameters by simply ranking the sensitivities. In this thesis elasticities are ranked.

The next section highlights some previous work on how to sensitivities can play an important role in better understanding how systems can be diagnosed and optimized and it presents how sensitivities have been employed by authors in their fields of study. In the next section, x , will be used to indicate a vector input parameter and x_n is defined a some input parameter to a model or system. Therefore, x is said to be equal to $(x_1, \dots, x_n)$. $f(x)$ on the other hand is the output of a model for a particular input parameter set x , and it can be computed from a set of equations. Therefore, the function f is effected by variations of parameters in x .

2.2.1 Related Research

This section describes related work on sensitivity analysis presented by authors from a wide spectrum of disciplines. The goal of this section is to highlight how sensitivities were used by the authors based on their experiences.

Andres

Andres in [andres95] studied the sensitivity of outputs with respect to input parameters of a system with large number of input parameters. He studied a geological waste disposal where there were approximately 3300 parameters in the model. He points out that if your parameter set is greater than 500 then it becomes very difficult to identify a few important parameters. In [andres95] an efficient sensitivity analysis procedure designed to elicit as much information as possible about the parameter dependencies of

some output based on a few specific input points is presented. This procedure involves deciding how to choose parameters for the sensitivity analysis, and then setting up sample sets. Andres recommends that the variability in the input parameters should also be kept in mind. Determining sensitivities over ranges of values for input parameters would help in the screening. Screening refers to identifying a parameter subset that controls the variability in the output of a model.

A good sensitivity analysis according to Andres should be able to:

- Reliably rank parameter effects
- Produce repeatable results using the same procedure but with different data sets
- Minimize cost: the number of function evaluations needed should be much less than the number of parameters when n is large and few effects are important
- Detect sensitivities across the entire domain of each parameter
- Provide useful estimates of f and of parameter influences.

The goal of a modeller is to understand the model, be able to explain an observed value of f by referring to values of a designated parameter subset where $f(x)$, is an output function and $x = (x_1, \dots, x_n)$ is a vector of the input parameters. Andres points out the first step is to screen the data. This means determining which parameters are controlling the variability of f the most. Various options exists for setting up a sample set for screening analysis. These are described in [andres95].

Banks and Lucas

Banks and Lucas use a model of a large military combat system to support their research, however their ideas are germane and can be applied to large models in other analytic applications. Combat models are often used to reason about hypothetical futuristic systems. As a results there is typically uncertainty about parameter settings. A sensitivity analysis is key in identifying the critical parameters.

A motivation for using analytic techniques as the basis for model solution can be drawn from this research. Simulations can provide important insights that support decisions making, however the computational infeasibility of conducting a thorough sensitivity analysis within simulation is an issue. Hence, due to the processing constraints only a few, believed a priori, key variables can be investigated. How does one identify these parameters? Analytic models can help to identify these key variables which can be further studied in detail using high resolution simulations.

Huber

Determining regions where models tend to be more or less sensitive can provide a lot of insight. In this paper [huber95], a new data-based approach for locating sensitive regions is presented. See Figure 2-1.

Figure 2-1 Identifying regions of sensitivity

A rule based learning algorithm is used to extract knowledge about the model behaviour based on data from simulation experiments. The resulting rules describe regions where the model tends to be more or less sensitive. The approach presented consists of three steps:

- [1] Data generation
- [2] Computation of difference quotients

[3] Learning sensitive regions

The process involves planning a number of experiments and performing simulation runs with different factor combinations. Then one interesting output, y , is selected for further sensitivity investigation. The next step is the computation of difference quotients. For each point, P , a set of adjacent points are taken and the difference quotient between the two points is computed:

$$Sensitivity(P) = dq(P) = \frac{\Delta y}{\Delta x} = \left| \frac{y_2 - y_1}{x_2 - x_1} \right|$$

Here Δx represents the Euclidean distance between the points in a normalized parameter space and Δy is the difference between output values. The last step in the process is to apply efficient learning algorithms where the data examples act as inputs to construct sets of hyper-rectangles in the parameter space.

They have found that in regions of low sensitivity the models seems to behave uncritically, while in regions of high sensitivity a more detailed investigation may be useful. A high sensitivity level can be caused by a strong increase, strong decrease and a steep optima.

Kleijnen

In [kleijnen95] a review of five state of the art related types of analysis techniques for simulation models are presented in order:

- [1] validation
- [2] screening
- [3] sensitivity or what-if analysis
- [4] uncertainty and risk analysis
- [5] optimization

The question Kleijnen tries to address is when and which analysis technique should be used. The main conclusion of his research is that the simulation analyst should distinguish the five phases in a simulation study. One should start off with *validation* which might involve use of regression and experimentation. *Screening* follows validation and he states that simple, efficient and effective techniques exist. *Sensitivity analysis* can then be performed on the screen parameter set. As a result of the screening fewer simulation runs are required. First-order effects and interactions effects can also be investigated. Next is *uncertainty analysis* where techniques such as Monte Carlo sampling, variance reduction techniques etc. can be used. Finally, for *optimization* Response Surface Methodology, which builds on regression analysis and experimentation is used.

2.3 Sensitivity Analysis in Queueing Networks

A number of studies exist where sensitivity analysis has been performed on product-form queueing networks. Sensitivity analysis for Queueing Networks (QN) has been reported by several authors. It is generally limited to *product-form*¹ or *separable models*. Lui and Nain in [lui91], studied the sensitivity of performance measures of the network with respect to input parameters such as arrival rates, service times, population etc. for product form queueing networks. A number of studies where partial derivatives of queueing measures in product-form queueing networks have been carried out. Trivedi et al in [tri80] used the partial derivative information to consider issues of optimal selection of CPU speed and device capacities so as to maximize throughput. Another study where partial derivative information was used was in [koba83] where the aim was to maximize system throughput by routing and load balancing. This work is described below.

Opdahl

Andreas Opdahl, in [opdahl96], presents an approach to software performance modelling based on annotating the design specifications with performance parameters and operational analysis of queueing networks representing the hardware. The approach to

1. a special class of queueing networks

Figure 2-1 Sensitivity Analysis of Product-form and Layered Queueing Networks

sensitivity analysis that is presented in the paper is used to point out where model refinement and parameter capture effort should be focused and suggests performance optimizations in the design specifications. Sensitivities are obtained by differentiation of the combined software and hardware performance model, but open queueing networks are only considered at the hardware level. This approach does not consider queueing for logical resources such as processes, critical sections, locks etc. By annotating the software design with the performance parameters, the workload on the underlying hardware can be determined but the notion that queueing can occur at software resources is not captured.

Opdahl points out that contemporary approaches to software performance engineering are time and knowledge intensive, which is why there is weak penetration of these techniques into the software development cycle. The approaches are time and knowledge intensive because they suffer from the following problems:

- addition modelling effort required to establish a software performance model
- the cost of capturing parameters required to establish a performance model
- lack of feedback to developers on possible performance improvements in the design

Sensitivity analysis addresses the latter two points by indicating where model refinement and parameter capture effort should be focussed, and it also suggests performance optimizations in the design specification.

Liu and Nain

In [liu91], Liu and Nain considered the sensitivity of open, closed and mixed product form queueing networks. Their objective was to derive explicit formulas for the derivatives measures in BCMP networks. They showed that the derivative of the expectation of any function of the system with respect to any model parameter can simply be expressed in terms of known functions of state variables such as queue lengths. With this result, the derivatives can be evaluated using computational algorithms described in [liu91]. Using this approach they can determine the quantitative and qualitative effects due to changes in model parameters such as monotonicity/non-monotonicity and concavity/convexity.

Akyildiz and Liu

Closed queueing networks with one multiprocessor stage and $N-1$ server stations with arbitrary connections are investigated in [aky88] for optimization of performance measures. Optimal service rates are determined such that throughput is maximized and the solutions minimize the response time at the multiprocessor. Mean value analysis is used as the basis for the optimization. Expression for the throughput and response time are derived using the Bard-Schweitzer approximate MVA equations. A cost function for the network based on a total budget for purchasing a given number of devices with specific service rates is also included. Based on the relationships of the total number of jobs and performance parameters, Lagrange multiplier technique is applied to determine the optimal utilization of the multiserver station, the optimal service rate, and optimal throughput of the system and the optimal response time of the multiprocessor station.

Cao and Ma

In [cao96], the focus is on determining performance sensitivities for single-class closed queueing networks with exponential service times and load-dependent service rates. In the paper, formulae are derived for the *elasticities* of general performance measures such as mean response time and mean queue length with respect to systems parameters such as mean service requirements, service rates, and visit ratios. These formulae can then be used for new on-line estimation schemes for the elasticities. These schemes are based solely on the averages of the performance measures and therefore require no knowledge of system parameters.

Others work

Williams and Bhandiwad calculated *elasticities* of various performance measures with respect to relative utilization of a server. The elasticity of a performance measure y with respect to an input parameter, x , is defined as follows:

$$E_{yx} = \frac{x}{y} \left(\frac{\partial y}{\partial x} \right)$$

This is interpreted as the relative change in x divided by the relative change in y . The authors also derive the following result on the elasticities in closed, single class networks with load independent exponential FCFS servers.

$$\left| \frac{x}{y} \frac{\partial y}{\partial x} \right| \leq 1$$

Elasticity is a measure of the response of one value to a relative change in another. For example, if throughput is not affected by a given percentage change in service time, then throughput is said to be inelastic to service time. If throughput increases greatly when service times are changed slightly and drops sharply with a small service time increase, throughput is said to be highly elastic to service time.

In [gordon80] the effect of parameter estimation error in closed product form queueing networks is investigated. The authors address questions such as:

- How sensitive are the performance measures to errors in model parameters?
- Are some performance measures more sensitive than others?
- How do the sensitivities of the performance measures vary with the type of the queueing network considered?

Partial derivatives with respect to server utilization of the equations for load independent networks are determined.

Performing a sensitivity analysis on a system where parameters such as service time, think times, etc. are based on estimates is a common practice. Depending at what stage you undergo this performance study, certain parameters will be based on estimates. Due to the complexity of these distributed applications, where the number of system parameters can be in the hundreds, estimating parameters introduce random quantities to the model and can affect the overall quality of your analysis. There exist different ways of regarding such randomness in simulation based analysis but no single viewpoint exists. Cheng and Holland [cheng96] state two sources of variation:

- estimation of parameters (parameter uncertainty)
- inherent variation in the solution technique because of their dependence on certain probability distributions

Chapter 3

Layered Queueing Modelling

This chapter summarizes the Layered Queueing Modelling framework as context to the sensitivity analysis approach discussed in Chapter 4. The components of a Layered Queueing Model are described and the solution approach is briefly described.

3.1 Analytic Modelling

A number of analytic techniques exist for analysis of computer systems. Analytic techniques are derived from mathematical expressions that are used to determine results for a system under study. An advantage of this approach to modelling systems is that solution time is typically small, enabling the designer/modeler to explore the design space more efficiently. However, analytic expressions are typically based on assumptions which may restrict the amount of detail of the actual system the model can represent.

Analytic approaches to performance modelling involve understanding of queueing theory, petri-nets, markov chain analysis. To many users, these approaches to performance modelling seem impractical and require knowledge of a fair amount of background theory to make the right modelling assumptions. Solutions for product-form queueing network models can be obtained simply. Both Exact and Approximate solution techniques exists for these types of queueing networks however a number of assumptions must be made before these techniques can be applied. In addition, the modelling of logical resources such as processes, critical sections, locks, etc. are not part of the framework. Since contention for logical resources can be a source of performance problems in computer systems these must be considered as part of the analysis. Therefore, an approach that is easy to use, easy to understand, models software and hardware components in a natural way, and is fast would be the ideal solution for modelling computer systems. Layered Queueing Modelling combines the natural representation of logical resources with the simple solution techniques found for product-form queueing, making it a powerful approach for modelling distributed computer systems.

3.2 Layered Queueing Models

Layered Queueing Networks (LQN) are a special form of *analytic* models for distributed applications [woodside89][woodside95][rolia95]. LQN modelling provides a transparent representation of the software architecture, which makes models easy to develop and understand and represents logical resources such as processes, critical sections or locks. A number of tools such as LQNS¹, MOL² that support the LQN framework for performance analysis. In this section the Layered Queueing Network (LQN) Modelling framework will be described.

3.2.1 Model Components

This section describes the components of a LQN model using a client-server example (See Figure 3-1). Parallelograms in a LQN model represent operating system processes. Processes are assigned to processors and are divided into three groups: *pure clients* (Client), *active servers* (Server) and *pure servers* (P1). Pure clients, also called reference tasks, only send requests. These tasks cycle continuously, performing computation work, sleep or send requests. The intent of a pure client is to model actual users or types of input sources of a system like cpu's or disk's. *Pure servers* only receive requests and they represent devices or stations found in conventional queueing networks. Finally, active servers can be thought of as a combination of clients and servers. Active servers are software processes that can both receive and generate requests.

A task pool is a group of identical processes that service requests from a common queue. Clients can also be replicated, as shown in Figure 3-1 where the population of Client is 2. Processes have entries. For example, entries on server processes represent services provided by that server. Associated with each entry is a service time, which corresponds to the amount of computational work that entry generates on the processes host processor. Message are serviced on a First-Come-First-Served (FCFS) basis. An active server can either be idle waiting for a request, awaiting a response from lower level

-
1. Layered Queueing Network Solver, written by G. Franks, Carleton University
 2. Method of Layers Solver, written by J. Rolia, Carleton University

servers, or executing on it processor. The service time of an entry is broken down into phases. The time between receiving and replying to a request is called *phase one*, the computational work done after replying to that request is classified as *phase two*.

Two types of interaction can take place between processes. Synchronous or blocking calls which can be thought of as synchronous RPCs and asynchronous messages. The numbers on the arcs between the processes represent the average number of visits from one process to the other.

Figure 3-1 Layered Queueing Network Model: Notation

The LQN model presented in Figure 3-1 consists of four *layers*. The Client is at the top most layer, S1, S2 and P1 form the next layer below and so on. Requests are made from one layer to another, hence the name layered modelling.

Figure 3-2 Layered Queueing Network Model Components

3.2.2 Model Parameters

This section highlights the various parameters to a layered model. The parameters of a particular model can also be seen in Figure 3-1. As mentioned processes have entries. Each entry in a layered model has a *service time* that represents the total execution time of it on its host processor. This does not include the queueing time of the process at the processor. Service times are exponential distributed. A receiving or server task can execute generally in two phases. The first phase is a service phase, which is similar to the service given in a queueing network and the second phase is an autonomous phase, where the server task acts independently. The second phase which is launched after the completion of the first phase is used to model operations such as writing to disk/cache, cleanup, etc. The following notation, $ST = [\text{phase 1 service time}, \text{phase 2 service time}]$, is used to identify the two phases of service at a server task.

The service times entered as part of the LQN model are broken down into what are referred to as slices. A slice is equal to the service time divided by the number of requests

plus one made by the process. Therefore in between requests, the process executes a slice of its execution on its host processor. In Figure 3-3 the execution of a Server process on processor P1 is given. The service time of the Server which happens to be 5 msec is broken down into 5 equal slices each with a 1 msec duration.

Figure 3-3 Execution slices in Layered Models

The numbers on the arcs between the processes in Figure 3-1 are the average number of *visits* from one process to another. Visits are geometrically distributed however it is possible to have deterministic visits as well as service times.

The *population* of a client pool, the *threading level* of a server process, and the *multiplicity* of a processor (represent a symmetric processor) are other input parameters to a layered queueing model. By varying the number of clients in a client pool, the load on

the system can be varied, or by increasing the multiplicity of the number of threads of a server process we can eliminate software bottlenecks and increase system throughput.

Another input parameter to a layered model is the *think time* of a client. Think time represents the delay between a response of client request to the next client request.

Customers at a resource compete for service. Thus the time spent by a customer at a resource depends on the scheduling policy at the resource. This is another input parameter to a layered model however sensitivities with respect to scheduling are not considered.

Therefore, to summarize, inputs to a LQN model that sensitivities considered with respect to are:

- average service times (computational work)
- average number of visits (between processes)
- population (multiplicity, or population)
- client think time (client time between successive requests)

3.2.3 Multi-servers

A set of identical tasks that service requests from a common input queue are referred to as multiservers. Different multiserver techniques exist and are best described in [franks97]. Three techniques exist, however the technique developed by Rolia [rolia92] which is a simple, fast, and accurate technique was used in this research. This will be discussed further in Chapter 4.

3.2.4 Performance Results or Model Outputs

Response time is defined as the interval of time between the submitting of a request and the receiving of a response. The response time is the sum of the residence times at the various resources, hardware or software, in the system.

Throughput is a performance result defined as the number of completed requests in a particular interval of time.

The proportion of time a resource is busy, or equivalently, the average number of customers in service at resource is the *utilization* of a device. This is another model output of a layered model.

The *queue length* at a resource includes all customers at the resource whether they are waiting or receiving service.

The performance metrics or outputs of a the model are:

- mean response times
- throughputs
- utilizations
- mean queue lengths

3.3 Layered Queueing Network Solver

This section gives a brief overview of parts of the LQNS solver which will help in understanding the changes that were required to integrate the sensitivity analysis technique are presented in Chapter 5. For further detail on the solver refer to [franks97]. The technique is outlined in Chapter 4 and the implementation is discussed in Chapter 5.

3.3.1 Class hierarchy of the LQNS Solver

It was decided that the best approach for obtaining parameter sensitivities was to use an approach that integrates the technique to the existing LQNS solver. By doing so duplication in the solution and post processing can be avoided and future changes to the solution technique can also be easily integrated to the current analysis.

In order to extend the LQNS it was important to understand the overall structure of the LQNS class hierarchy. The LQNS solver is written in C++ and is over 30,000 lines of code. It is designed such that different solution algorithms can be easily added to the existing code, making the code easily extensible. The class hierarchy for the LQNS solver is shown below in Figure 3-4. In Figure 3-4, the dot identifies a relationship while the triangle identifies a different subclass.

Figure 3-4 LQNS Class Hierarchy

Table 3-1 : Class Description Table

Class Name	Description
load	<i>load the model</i>
store	<i>store the model</i>
layerize	<i>constructs the MVA submodels and invokes the appropriate MVA solver</i>
call	<i>represents arcs and are used to connect entries in the layered model</i>
entry	<i>class used to describe an entry</i>
entity	<i>class used to describe an entity</i>
MVA	<i>provides the different solution techniques such as exact MVA or Schweitzer and Linearizer approximations</i>
server	<i>class used to describe the different type of servers</i>

3.3.2 Hierarchical Decomposition

A common approach to modelling large scale systems is to partition a model in smaller submodels. Each of the submodels are then evaluated, and the individual solutions are combined to obtain the solution of the original model. An approach called *Batched Layering* is used for hierarchical modelling in the solution of Layered Queueing Network Models.

Figure 3-5 Example LQN Model

The batch layering approach constructs a number of *complementary* models, where each model corresponds to a particular MVA submodel. Clients in the top most level and its servers form a submodel. The CLIENT and its servers S1, S2, and P1 form Submodel I. Servers S1, and S2 also act as clients to servers P2, S3, S4 and P3 and therefore these form the clients and servers in Submodel II. Finally, S3, and S4 are clients in Submodel II and P4 is the server. Software servers can be found in multiple MVA submodels whereas a device such as a processor is found only in one submodel. This decomposition technique does not assume that the LQN model be strictly layered. The decomposition of the LQN model found in Figure 3-5, into MVA submodels can be found in Figure 3-6.

3.3.3 Layered Queueing Network Solver Operation

The steps to solving a LQN model are summarized below. Applications typically consists of many layers of geographically distributed software and hardware components.

- [1] Read input file and construct object database consisting of tasks, processors, and entries and the calls between them.
- [2] Identify the layers and create MVA submodels for each layer.
- [3] Solve individual MVA submodels, where outputs of one submodel form inputs to another (interaction variables). Iterate among the submodels until waiting time results converge for all layers.
- [4] Write results out

Solution Algorithm

The pseudo-code for the solution algorithm is presented below. The repeat-until loop corresponds to the outer iterations. Within each repeat-until loop each submodel is solved. Outputs of a submodel can act as inputs to another submodel solution. The iterations for a submodel solution are referred to as inner iterations.

```

repeat
  foreach submodel  $l$  ( $1..L$ )
    set MVA parameters for submodel  $l$ 
    solve submodel  $l$ 
    set waiting times for submodel  $l$  and think times
      for submodel  $l+1$ 
  endfor
until convergence
calculate and output partial sensitivities

```

The sensitivities are only calculated after convergence. An extra pass through each submodel is performed where quantities called the “partial sensitivities” are calculated. The code where the solution takes place is instrumented for the partial sensitivities and then written to file. The partial sensitivities are then processed to determine parameter sensitivities. This is described in Chapter 5.

Figure 3-6 Decomposition of a Layered Model into MVA submodels

Chapter 4

Sensitivity Analysis in Layered Queueing Models

In this chapter, the general approach of performing a sensitivity analysis in Layered Queueing Models is presented. A simple worked example will be used to demonstrate the approach. Chapter 5, titled Implementation of Approach, will discuss how the approach presented in this chapter was implemented.

4.1 Sensitivity Analysis in Layered Queueing Models

Bard Schweitzer Approximate Mean Value Analysis and Linearizer are two approximate Mean Value Analysis (MVA) algorithms that have been implemented in the Layered Queueing Network Solver. The sensitivity analysis technique presented and implemented is based on the Bard-Schweitzer Approximate Mean Value Analysis algorithms. The sensitivity analysis approach can in principle also be based on the Linearizer Approximate MVA, however this was not carried out. Therefore sensitivities that are obtained are based on the Bard-Schweitzer solution.

In Chapter 3, the decomposition of a Layered Queueing Model into submodels was described. These submodels, which are simply queueing networks, are then solved using Bard-Schweitzer Approximate Mean Value Analysis algorithm. This chapter will derive the Bard-Schweitzer equations for a simple client-server example. There are also equations that describe the interaction between submodels. These equations, which are also critical to the solution, will also be described in this chapter.

The next section briefly describes MVA and the Bard-Schweitzer Approximate MVA solution technique.

4.1.1 Mean Value Analysis

Mean Value Analysis is an approach for solving queueing networks that yields mean performance results. MVA applies to networks with a variety of service disciplines and service time distributions. Both Exact and Approximate MVA techniques exist.

Exact MVA techniques are exact for product form queueing networks, where the computation of the performance with N jobs in the system requires the knowledge of the system performance with $N-1$ jobs. For small values of N , this procedure is not computationally expensive, however as the number of jobs increases to large values of N , the recursive nature of this procedure becomes computationally quite expensive.

Several approximate techniques have been developed that avoid this computational inefficiency. One such technique is known as Bard-Schweitzer Approximate MVA. This technique avoids the recursion by estimating the queue length with N jobs. This is an iterative approach where the response times and throughputs, determined using the initial queue length estimate, are used to calculate the queue length. If the difference between the previous is large then one continues to iterate. The detailed MVA algorithm for the Bard-Schweitzer Approximate MVA technique can be found on page 581 in [jain91] and pages 142-145 in [lazow84].

The Bard-Schweitzer approximations were used because the equations are simpler. One other important point to keep in mind is that layered models do not necessarily satisfy the product-form assumptions.

4.2 Bard-Schweitzer Approximate MVA

The sub-sections to follow will describe the Bard-Schweitzer Approximate MVA equations and their partial derivatives. The equations that corresponds to the interaction between submodels and their partial derivatives will also be described. Section 4.3 will apply these equations to a simple client-server example.

4.2.1 The Bard-Schweitzer Approximate MVA Equations

This section will provide background on the equations used as part of the Bard-Schweitzer MVA analysis and it will also demonstrate how those equations are used to arrive at the sensitivities in a LQN Model.

The response time of class, c , R_c is the sum of the all the responses times of class c , at all devices it visits, number 1 to K:

$$R_c = \sum_{k=1}^K R_{c,k}(\vec{N}) \quad \text{Eqn 4-1}$$

The response time of class c , at device k , is defined as such:

$$R_{c,k}(\vec{N}) = D_{c,k} \left[1 + \left(\frac{N_c - 1}{N_c} \right) Q_{c,k}(\vec{N}) + \sum_{\substack{j=1 \\ j \neq c}}^C Q_{j,k}(\vec{N}) \right] \quad \text{Eqn 4-2}$$

The key to the Bard-Schweitzer approximation is the assumption that the queue length of a particular class at a station with one job removed from that class is equal to one job removed from that class divided by the total number in the class multiplied by the queue length with N jobs.

$$\left(\frac{N_c - 1}{N_c} \right) Q_{c,k}(\vec{N}) \approx Q_{c,k}(\vec{N} - 1) \quad \text{Eqn 4-3}$$

$D_{c,k}$ is the demand of a particular class c at device k .

$$D_{c,k} = V_{c,k} S_{c,k} \quad \text{Eqn 4-4}$$

where $V_{c,k}$ is the average number of visits and $S_{c,k}$ is the average service time of class c at device k .

The average number of jobs of class c at device k is the product of the throughput of that class and the response time of that class at that particular device.

$$Q_{c,k}(\vec{N}) = X_c(\vec{N})R_{c,k}(\vec{N}) \quad \text{Eqn 4-5}$$

The throughput of class c is defined as such:

$$X_c(\vec{N}) = \frac{N_c}{Z_c + \sum_{k=1}^K R_{c,k}(\vec{N})} \quad \text{Eqn 4-6}$$

where N_c is the total number of customers of that class, Z_c is the average think time of that class, and finally $\sum_{k=1}^K R_{c,k}(\vec{N})$ which is the sum of the response times of that class at all the devices that it visits or simply R_c .

4.2.2 Partial derivatives of the Bard-Schweitzer Approximate MVA Equations

The partial derivatives of the Bard-Schweitzer approximate are used to determine the parameter sensitivities. In this section, the partial derivative of the equation outlined in the previous section are presented. Since each MVA submodel is a queueing network, equations for each of submodel will be constructed and then submodel matrices for each submodel will be created using the partial derivatives derived below. The next section will demonstrate this by working through a simple client-server system.

Assuming we are interested in determining the sensitivity of the CLIENT response time to some parameter in our LQN model, which will be referred to as ξ . Therefore, we are interested in finding partial derivative of the response time of the CLIENT class to ξ :

$$\frac{\partial R_c(\vec{N})}{\partial \xi} \quad \text{Eqn 4-7}$$

We also know the following

$$\frac{\partial}{\partial \xi} R_c(\vec{N}) = \sum_{k=1}^K \frac{\partial}{\partial \xi} R_{c,k}(\vec{N}) \quad \text{Eqn 4-8}$$

Similarly, the partial derivative of the response time of a class c at device k is

$$\begin{aligned} \frac{\partial R_{c,k}(\vec{N})}{\partial \xi} = D_{c,k} & \left[\frac{Q_{c,k}(\vec{N})}{N_c^2} \frac{\partial N_c}{\partial \xi} + \left(\frac{N_c - 1}{N_c} \right) \frac{\partial Q_{c,k}(\vec{N})}{\partial \xi} + \sum_{\substack{j=1 \\ j \neq c}}^C \frac{\partial Q_{j,k}(\vec{N})}{\partial \xi} \right] \\ & + \left[1 + \left(\frac{N_c - 1}{N_c} \right) Q_{c,k}(\vec{N}) + \sum_{\substack{j=1 \\ j \neq c}}^C Q_{j,k}(\vec{N}) \right] \frac{\partial D_{c,k}}{\partial \xi} \end{aligned} \quad \text{Eqn 4-9}$$

The partial derivative of the demand, queue length and throughput are also shown below:

Demand:

$$\frac{\partial D_{c,k}}{\partial \xi} = S_{c,k} \frac{\partial V_{c,k}}{\partial \xi} + V_{c,k} \frac{\partial S_{c,k}}{\partial \xi} \quad \text{Eqn 4-10}$$

Queue Length:

$$\frac{\partial Q_{c,k}(\vec{N})}{\partial \xi} = R_{c,k}(\vec{N}) \frac{\partial X_c(\vec{N})}{\partial \xi} + X_c(\vec{N}) \frac{\partial R_{c,k}(\vec{N})}{\partial \xi} \quad \text{Eqn 4-11}$$

Throughput of a particular class c is defined as such:

$$\begin{aligned} \frac{\partial X_c(\vec{N})}{\partial \xi} = N_c & \left(\frac{-1}{\left(Z_c + \sum_{k=1}^K R_{c,k}(\vec{N}) \right)^2} \left(\frac{\partial Z_c}{\partial \xi} + \sum_{k=1}^K \frac{\partial R_{c,k}(\vec{N})}{\partial \xi} \right) \right. \\ & \left. + \left(\frac{1}{Z_c + \sum_{k=1}^K R_{c,k}(\vec{N})} \right) \frac{\partial N_c}{\partial \xi} \right) \end{aligned} \quad \text{Eqn 4-12}$$

4.2.3 Interaction Equations: Coupling between MVA Submodels

One must consider parameter flow from one MVA submodel to another. A client and server in a particular submodel can have both service times as well as think times. These are both determined from solutions of other submodels. The think time of a server in submodel m is the rate at which the server will generate requests to its lower level servers. This rate is partly derived from the server's utilization in some other submodel l . Service times for servers in submodel l are based on the response times found in all submodels. The interaction equations will be further explained using the client-server architecture in Figure 4-1.

When a task is acting as a client in MVA submodel l , its service time is found by taking the sum of the response time to all of its servers in all other submodels except for those in the current submodel. M is defined as all the submodels in the system, and J is defined to be all the servers that the client visits in the submodel.

$$s_i^{[l]} = \sum_{m, m \neq l}^M \sum_j^J R_{i,j}^{[m]} \quad \text{Eqn 4-13}$$

Figure 4-1 Layered Model and MVA Submodels for a Client-Server System

Figure 4-2 Determining the service time of the Client based on the response of the Client at servers in other layers

When a task is acting as a server in a MVA submodel, its service time is found by taking the sum of the response times to all its servers.

$$s_i^{[l]} = \sum_m^M \sum_{j \in J(i, m)} R_{i, j}^{[m]} \quad \text{Eqn 4-14}$$

Figure 4-3 Determining the service time of the Server based on the response of the Server at servers in other layers

The Think time, $Z_i^{[l]}$, for chain i in an MVA submodel l is found from the utilization, $\rho_i^{[l]}$, of task i , when it is acting as a server in submodel m where m can be any other submodel. The equation that describes this relationship is as follows:

$$Z_{i, l} = \frac{N_i^{[l]}(1 - \rho_i^{[m]})}{X_i^{[m]}} = N_i^{[l]} \left(\frac{1}{X_i^{[m]}} - s_i^{[m]} \right) \quad \text{Eqn 4-15}$$

where $N_i^{[l]}$ is the population of chain i and also the multiplicity of the server in submodel m .

4.2.4 Partial Derivatives of Interaction Equations

The partial derivative of client service time in submodel l is:

$$\frac{\partial s_i^{[l]}}{\partial \xi} = \sum_{m, m \neq l}^M \sum_{j \in J(i, m)} \frac{\partial R_{i,j}^{[m]}}{\partial \xi} \quad \text{Eqn 4-16}$$

For task acting as a server in submodel l , the partial derivative of its service time with respect to input parameter ξ .

$$\frac{\partial s_i^{[l]}}{\partial \xi} = \sum_m \sum_{j \in J(i, m)} \frac{\partial R_{i,j}^{[m]}}{\partial \xi} \quad \text{Eqn 4-17}$$

The partial derivative of a think time, Z , is determined as follows. Using Eqn 4-15 we can determine the partial derivative of the interaction think time equation. The identity in Eqn 4-18 for the utilization, ρ , of a device is used in the simplification:

$$\rho_i^{[m]} = X_i^{[m]} D_i^{[m]} \quad \text{Eqn 4-18}$$

$$Z_i^{[l]} = \frac{N_i^{[l]}}{X_i^{[m]}} - N_i^{[l]} D_i^{[m]} \quad \text{Eqn 4-19}$$

$$\frac{\partial Z_i^{[l]}}{\partial \xi} = \left(\frac{-N_i^{[m]}}{(X_i^{[m]})^2} \right) \frac{\partial X_i^{[m]}}{\partial \xi} + \left(\frac{1}{X_i^{[m]}} + D_i^{[m]} \right) \frac{\partial N_i^{[l]}}{\partial \xi} - N_i^{[l]} \frac{\partial D_i^{[m]}}{\partial \xi} \quad \text{Eqn 4-20}$$

4.3 Worked Example of a Simple Client-Server System

In this section, the approach described in the previous sections will be demonstrated using an example of simple client-server system.

The client-server system in Figure 4-4, consists of a pool of clients each with a service time of 5 msec and a think time of 20 msec. The clients which reside on processor P1 make on the average two requests to the server, Server, for service E1. The service time of entry E1 is 2 msec. The server resides on processor P2.

Figure 4-4 Layered Model of a Simple Distributed Client-Server System

This LQN model, when solved is decomposed into two MVA submodels that are solved alternately until convergence. The next section determines the partial derivative equation for each submodel and demonstrates how those equations are used to create the submodel matrices.

4.3.1 Submodel Equations for the Simple Client-Server System

4.3.1.1 Submodel I

The approximate MVA equations that are used to solve this submodel are presented in this section. The decomposed submodel queueing network is shown in Figure 4-5. The submodel consists of one client, called CLIENT and two servers. SERVER is the server station in the queueing network that corresponds to the software SERVER in the layered model. The P1 station represents the processor that the CLIENT processor resides on.

Figure 4-5 Submodel I of the Simple Client-Server System

Equations for Submodel I are determined using the equations derived in Section 4.2.2.

$$\frac{\partial R_{C,S}}{\partial \xi} = \left(1 + Q_{C,S} - \frac{Q_{C,S}}{N_C}\right) \frac{\partial D_{C,S}}{\partial \xi} + \left(D_{C,S} - \frac{D_{C,S}}{N_C}\right) \frac{\partial Q_{C,S}}{\partial \xi} + \left(\frac{Q_{C,S} D_{C,S}}{N_C^2}\right) \frac{\partial N_C}{\partial \xi}$$

$$\frac{\partial D_{C,S}}{\partial \xi} = S_{C,S} \frac{\partial V_{C,S}}{\partial \xi} + V_{C,S} \frac{\partial S_{C,S}}{\partial \xi}$$

$$\frac{\partial Q_{C,S}}{\partial \xi} = R_{C,S} \frac{\partial X_C}{\partial \xi} + X_C \frac{\partial R_{C,S}}{\partial \xi}$$

$$\frac{\partial R_{C,P1}}{\partial \xi} = \left(1 + Q_{C,P1} - \frac{Q_{C,P1}}{N_C}\right) \frac{\partial D_{C,P1}}{\partial \xi} + \left(D_{C,P1} - \frac{D_{C,P1}}{N_C}\right) \frac{\partial Q_{C,P1}}{\partial \xi} + \left(\frac{Q_{C,P1} D_{C,P1}}{N_C^2}\right) \frac{\partial N_C}{\partial \xi}$$

$$\frac{\partial D_{C,P1}}{\partial \xi} = S_{C,P1} \frac{\partial V_{C,P1}}{\partial \xi} + V_{C,P1} \frac{\partial S_{C,P1}}{\partial \xi}$$

$$\frac{\partial Q_{C,P1}}{\partial \xi} = R_{C,P1} \frac{\partial X_C}{\partial \xi} + X_C \frac{\partial R_{C,P1}}{\partial \xi}$$

$$\frac{\partial X_C}{\partial \xi} = N_C \left(\frac{-1}{(Z_C + R_{C,S} + R_{C,P1})^2} \right) \left(\frac{\partial Z_C}{\partial \xi} + \frac{\partial R_{C,S}}{\partial \xi} + \frac{\partial R_{C,P1}}{\partial \xi} \right) + \left(\frac{1}{Z_C + R_{C,S} + R_{C,P1}} \right) \frac{\partial N_C}{\partial \xi}$$

These equations are then used to form the matrix for this submodel. These equations are in vector matrix form $\frac{\partial y_i}{\partial \xi} = \sum_j A_{i,j} \frac{\partial x_j}{\partial \xi}$ but when building the submodel matrix, the equations are re-worked to $\frac{\partial y_i}{\partial \xi} - \sum_j A_{ij} \frac{\partial x_j}{\partial \xi} = 0$. The matrix for this submodel is given in Figure 4-6.

1	0	$\left(1 + Q_{C,S} \frac{W_{C,S}}{N_C}\right)$	0	$\left(D_{C,S} \frac{D_{C,S}}{N_C}\right)$	0	0	0	0	0	$\left(\frac{Q_{C,S} D_{C,S}}{N_C^2}\right)$	0	$\frac{\partial W_{C,S}}{\partial N_C}$
0	1	0	$\left(1 + Q_{C,P1} \frac{Q_{C,S}}{N_C}\right)$	0	$\left(D_{C,P1} \frac{D_{C,S}}{N_C}\right)$	0	0	0	0	$\left(\frac{Q_{C,P1} D_{C,S}}{N_C^2}\right)$	0	$\frac{\partial W_{C,P1}}{\partial N_C}$
0	0	1	0	0	0	$S_{C,S} \frac{S_{C,S}}{N_C}$	0	0	0	0	0	$\frac{\partial W_{C,S}}{\partial N_C}$
0	0	0	1	0	0	0	$S_{C,P1} \frac{S_{C,P1}}{N_C}$	0	0	0	0	$\frac{\partial W_{C,P1}}{\partial N_C}$
x_C	0	0	0	1	0	0	0	0	0	0	0	$\frac{\partial x_C}{\partial N_C}$
0	x_C	0	0	0	1	0	0	0	0	$R_{C,S}$	0	$\frac{\partial x_C}{\partial N_C}$
0	0	0	0	0	0	0	1	0	0	$R_{C,P1}$	0	$\frac{\partial x_C}{\partial N_C}$
N_C	$\frac{(L_C + R_{C,S} + R_{C,P1})^2}{N_C}$	0	0	0	0	0	0	0	0	1	N_C	$\frac{\partial N_C}{\partial N_C}$
0	0	0	0	0	0	0	1	0	0	$(L_C + R_{C,S} + R_{C,P1})$	0	$\frac{\partial W_{C,S}}{\partial N_C}$
0	0	0	0	0	0	0	0	1	0	0	0	$\frac{\partial W_{C,P1}}{\partial N_C}$
0	0	0	0	0	0	0	0	0	1	0	0	$\frac{\partial W_{C,S}}{\partial N_C}$
0	0	0	0	0	0	0	0	0	0	1	0	$\frac{\partial W_{C,P1}}{\partial N_C}$
0	0	0	0	0	0	0	0	0	0	0	1	$\frac{\partial W_{C,S}}{\partial N_C}$

Figure 4-6 Simple Client-Server System Submodel I Matrix (A_I)

4.3.1.2 Submodel II

The equations for submodel II are presented in this section. Submodel II as shown in Figure 4-7 consists of a single client called SERVER, and processor P2 which is the

Figure 4-7 Submodel II of the Simple Client-Server System

processor that the SERVER process resides on.

The following are the partial derivative equations for this queueing network:

$$\frac{\partial R_{S,P2}}{\partial \xi} = \left(1 + Q_{S,P2} - \frac{Q_{S,P2}}{N_S}\right) \frac{\partial D_{S,P2}}{\partial \xi} + \left(D_{S,P2} - \frac{D_{S,P2}}{N_S}\right) \frac{\partial Q_{S,P2}}{\partial \xi} + \left(\frac{Q_{S,P2} D_{S,P2}}{N_S^2}\right) \frac{\partial N_S}{\partial \xi}$$

$$\frac{\partial D_{S,P2}}{\partial \xi} = S_{S,P2} \frac{\partial V_{S,P2}}{\partial \xi} + V_{S,P2} \frac{\partial S_{S,P2}}{\partial \xi}$$

$$\frac{\partial Q_{S,P2}}{\partial \xi} = R_{S,P2} \frac{\partial X_S}{\partial \xi} + X_S \frac{\partial R_{S,P2}}{\partial \xi}$$

$$\frac{\partial X_S}{\partial \xi} = N_S \left(\frac{-1}{(Z_S + R_{S,P2})^2} \left(\frac{\partial Z_S}{\partial \xi} + \frac{\partial R_{S,P2}}{\partial \xi} \right) + \left(\frac{1}{Z_S + R_{S,P2}} \right) \frac{\partial N_S}{\partial \xi} \right)$$

$$\begin{bmatrix}
1 & -\left(1 - Q_{S, P2} - \frac{Q_{S, P2}}{N_S}\right) & \left(D_{S, P2} - \frac{D_{S, P2}}{N_S}\right) & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -S_{S, P2} & -V_{S, P2} & 0 & 0 \\
-X_S & 0 & 1 & 0 & 0 & -R_{S, P2} & 0 \\
N_S & 0 & 0 & 1 & 0 & 0 & \frac{N_S}{(Z_S + R_{S, P2})^2} \\
\frac{(Z_S + R_{S, P2})^2}{(Z_S + R_{S, P2})^2} & 0 & 0 & 0 & 1 & 0 & \frac{-1}{(Z_S + R_{S, P2})} \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
\partial R_{S, P2} \\
\partial D_{S, P2} \\
\partial Q_{S, P2} \\
\partial X_S \\
\partial S_{S, P2} \\
\partial V_{S, P2} \\
\partial N_S \\
\partial Z_S
\end{bmatrix}$$

Figure 4-8 Simple Client-Server System Submodel II Matrix (A_2)

4.3.2 Interaction between MVA Submodels for the Simple Client-Server System

In this example, the interaction or coupling between submodels is limited to the Server. The response time of the SERVER in Submodel II is used to form the service time of the SERVER in Submodel I. The utilization of the SERVER in Submodel I is used to determine the think time of the SERVER in submodel II.

The equations that describe the interaction between submodels are formulated below. Let S represent the SERVER process.

$$S_S^{[1]} = R_{S,P2}$$

$$Z_S^{[2]} = N_S^{[2]} \frac{1 - \rho_S^{[1]}}{X_S^{[2]}}$$

The partial derivatives of the above equations are:

$$\frac{\partial S_S^{[1]}}{\partial \xi} = \frac{\partial R_{S,P2}^{[2]}}{\partial \xi}$$

$$\frac{\partial Z_S^{[2]}}{\partial \xi} = \left(\frac{-N_S^{[2]}}{(X_S^{[2]})^2} \right) \frac{\partial X_{S,I}}{\partial \xi} + \left(\frac{1}{(X_S^{[2]})^2} + D_S^{[1]} \right) \frac{\partial N_S^{[2]}}{\partial \xi} - N_S^{[2]} \frac{\partial D_S^{[1]}}{\partial \xi}$$

4.4 Assembly of System Matrix

Once the submodel equations have been determined and the submodel matrices have been created, these can be assembled along with the interaction equations to produce the global *System Matrix*, *A*. A high level view of the System Matrix, *A*, is given in Figure 4-9.

The System Matrix governs a number of linear equations. To obtain the sensitivities these equations are solved using gaussian elimination. This is a very common approach for solving a sets of *n* equations with *n* unknowns.

Figure 4-9 System Matrix Structure

As explained earlier, the solution technique involves the use of several submodels. As a result, one needs to distinguish between input and outputs of the overall model (layered model) called *external parameters*, and the inputs and outputs of submodels, called *internal parameters*. In some cases, internal parameters can also be external

parameters. External parameters can be further broken down into *external input* or *external output* parameters. As part of our goal for this study, we are interested in the sensitivities of the external output parameters to the external input parameters. However, the approach used will give sensitivities of ALL output parameters (internal or external) with respect to ALL input parameters (internal or external).

Figure 4-10 Internal and External parameters with three submodels

4.4.1 Strategy for Solving for the Sensitivities

In this section, to demonstrate the approach used for setting up the solution for the *sensitivity matrix* the simple-client server example motivated earlier will be used. The

sensitivity matrix contains sensitivities of all output parameters with respect to all input parameters.

Consider these equations taken from Section 4.3.1.1:

$$D_{C,S} = V_{C,S} S_{C,S}$$

$$Q_{C,S} = X_C R_{C,S}$$

$$X_C = \frac{N_C}{Z_C + R_{C,S} + R_{C,P1}}$$

$$R_{C,S} = D_{C,S} \left[1 + \left(\frac{N_C - 1}{N_C} \right) Q_{C,S} \right]$$

The partial derivatives of these equations with respect to some parameter ξ are:

$$\frac{\partial D_{C,S}}{\partial \xi} = S_{C,S} \frac{\partial V_{C,S}}{\partial \xi} + V_{C,S} \frac{\partial S_{C,S}}{\partial \xi}$$

$$\frac{\partial Q_{C,S}}{\partial \xi} = R_{C,S} \frac{\partial X_C}{\partial \xi} + X_C \frac{\partial R_{C,S}}{\partial \xi}$$

$$\frac{\partial R_{C,S}}{\partial \xi} = \left(1 + Q_{C,S} - \frac{Q_{C,S}}{N_C} \right) \frac{\partial D_{C,S}}{\partial \xi} + \left(D_{C,S} - \frac{D_{C,S}}{N_C} \right) \frac{\partial Q_{C,S}}{\partial \xi} + \left(\frac{Q_{C,S} D_{C,S}}{N_C^2} \right) \frac{\partial N_C}{\partial \xi}$$

$$\frac{\partial X_C}{\partial \xi} = N_C \left(\frac{-1}{(Z_C + R_{C,S} + R_{C,P1})^2} \right) \left(\frac{\partial Z_C}{\partial \xi} + \frac{\partial R_{C,S}}{\partial \xi} + \frac{\partial R_{C,P1}}{\partial \xi} \right) + \left(\frac{1}{Z_C + R_{C,S} + R_{C,P1}} \right) \frac{\partial N_C}{\partial \xi}$$

$R_{C,S}$, $Q_{C,S}$, X_C , $R_{C,P1}$ can be thought of a output parameters that are classified as both internal and external because they are dependent on each other. $V_{C,S}$, $S_{C,S}$, N_C , Z_C are simply external input parameters, so these are the parameters we are interested in getting the sensitivities with respect to.

For each submodel there are sets of partial derivative equations. The equations for each submodel are then grouped into one matrix, A, or referred to as the *System Matrix*. The partial derivatives of the interaction equations are also entered into the system matrix.

The system matrix therefore represents all the equations that describe the system as a whole. Using the standard solution approach (matrix operations) where:

$$X = A^{-1}B$$

we can solve for the sensitivity matrix, X. B is what is referred to as the *Selector Matrix*.

In the next section an artificial example is used to demonstrate the complete setup of the system and selector matrices. Unfortunately due to the size of the previous simple client-server example this was not possible.

4.4.1.1 An Artificial Example

An artificial yet simpler example with two submodels will be used to better demonstrate the construction of the System and Selector Matrix. Suppose there is some kind of Submodel I with input parameters x_1, x_2, x_3 , and output parameters y_1, y_2, y_3 . y_4, y_5 are output parameters and x_4 is the input parameter to some Submodel II. ξ stands for any parameter which may be an input or output parameter.

Submodel I

Equations

$$y_1 = 5y_2 + 2x_1$$

$$y_2 = y_3 + 3x_1 + 5x_2$$

$$y_3 = 2y_1 + 3y_2 + x_3$$

Partial Derivatives wrt to any parameter ξ

$$\frac{\partial y_1}{\partial \xi} = 5 \frac{\partial y_2}{\partial \xi} + 2 \frac{\partial x_1}{\partial \xi}$$

$$\frac{\partial y_2}{\partial \xi} = \frac{\partial y_3}{\partial \xi} + 3 \frac{\partial x_1}{\partial \xi} + 5 \frac{\partial x_2}{\partial \xi}$$

$$\frac{\partial y_3}{\partial \xi} = 2 \frac{\partial y_1}{\partial \xi} + 3 \frac{\partial y_2}{\partial \xi} + \frac{\partial x_3}{\partial \xi}$$

Submodel I Matrix

$$\begin{bmatrix} 1 & -5 & 0 & -2 & 0 & 0 \\ 0 & 1 & -1 & -3 & -5 & 0 \\ -2 & -3 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Submodel II

Equations

$$y_4 = 3y_5 + 2x_4$$

$$y_5 = y_4 - x_4$$

Partial Derivatives wrt to ξ

$$\frac{\partial y_4}{\partial \xi} = 3 \frac{\partial y_5}{\partial \xi} + 2 \frac{\partial x_4}{\partial \xi}$$

$$\frac{\partial y_5}{\partial \xi} = \frac{\partial y_4}{\partial \xi} - \frac{\partial x_4}{\partial \xi}$$

Submodel II Matrix

$$\begin{bmatrix} 1 & -3 & -2 \\ -1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Interaction Effects between submodels*Interaction Equation*

$$y_4 = 2y_1 + 3y_3$$

Partial Derivative of Interaction Equation

$$\frac{\partial y_4}{\partial \xi} = 2 \frac{\partial y_1}{\partial \xi} + 3 \frac{\partial y_3}{\partial \xi}$$

System Matrix

$$A = \begin{bmatrix} 1 & -5 & 0 & -2 & 0 & 0 \\ 0 & 1 & -1 & -3 & -5 & 0 \\ -2 & -3 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -2 & 0 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & -3 & -2 \\ -1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

If we are interested in only determining the sensitivity of all output parameters to a single input parameter, x_1 , then the equation $AX = B$ would be set up as such:

$$\begin{bmatrix}
 1 & -5 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & -1 & -3 & -5 & 0 & 0 & 0 & 0 \\
 -2 & -3 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 -2 & 0 & -3 & 0 & 0 & 0 & 1 & -3 & -2 \\
 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
 \end{bmatrix}
 \begin{bmatrix}
 \frac{\partial y_1}{\partial x_1} \\
 \frac{\partial y_2}{\partial x_1} \\
 \frac{\partial y_3}{\partial x_1} \\
 \frac{\partial x_1}{\partial x_1} \\
 \frac{\partial x_2}{\partial x_1} \\
 \frac{\partial x_3}{\partial x_1} \\
 \frac{\partial y_4}{\partial x_1} \\
 \frac{\partial y_5}{\partial x_1} \\
 \frac{\partial x_4}{\partial x_1}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 0 \\
 1 \\
 0 \\
 0 \\
 0 \\
 0 \\
 0
 \end{bmatrix}$$

We can however extend the Sensitivity and Selector matrices so that we can solve for the sensitivities of all output parameters to all input parameters in one matrix operation. The new Sensitivity and Selector matrices look as follows:

Sensitivity Matrix

$$X = \begin{bmatrix}
 \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} & \frac{\partial y_1}{\partial x_4} \\
 \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} & \frac{\partial y_2}{\partial x_4} \\
 \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} & \frac{\partial y_3}{\partial x_4} \\
 \frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} & \frac{\partial x_1}{\partial x_3} & \frac{\partial x_1}{\partial x_4} \\
 \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial x_2} & \frac{\partial x_2}{\partial x_3} & \frac{\partial x_2}{\partial x_4} \\
 \frac{\partial x_3}{\partial x_1} & \frac{\partial x_3}{\partial x_2} & \frac{\partial x_3}{\partial x_3} & \frac{\partial x_3}{\partial x_4} \\
 \frac{\partial y_4}{\partial x_1} & \frac{\partial y_4}{\partial x_2} & \frac{\partial y_4}{\partial x_3} & \frac{\partial y_4}{\partial x_4} \\
 \frac{\partial y_5}{\partial x_1} & \frac{\partial y_5}{\partial x_2} & \frac{\partial y_5}{\partial x_3} & \frac{\partial y_5}{\partial x_4} \\
 \frac{\partial x_4}{\partial x_1} & \frac{\partial x_4}{\partial x_2} & \frac{\partial x_4}{\partial x_3} & \frac{\partial x_4}{\partial x_4}
 \end{bmatrix}$$

Selector Matrix

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -5 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & -3 & -5 & 0 & 0 & 0 & 0 \\ -2 & -3 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -2 & 0 & -3 & 0 & 0 & 0 & 1 & -3 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} & \frac{\partial y_1}{\partial x_4} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} & \frac{\partial y_2}{\partial x_4} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} & \frac{\partial y_3}{\partial x_4} \\ \frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} & \frac{\partial x_1}{\partial x_3} & \frac{\partial x_1}{\partial x_4} \\ \frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial x_2} & \frac{\partial x_2}{\partial x_3} & \frac{\partial x_2}{\partial x_4} \\ \frac{\partial x_3}{\partial x_1} & \frac{\partial x_3}{\partial x_2} & \frac{\partial x_3}{\partial x_3} & \frac{\partial x_3}{\partial x_4} \\ \frac{\partial y_4}{\partial x_1} & \frac{\partial y_4}{\partial x_2} & \frac{\partial y_4}{\partial x_3} & \frac{\partial y_4}{\partial x_4} \\ \frac{\partial y_5}{\partial x_1} & \frac{\partial y_5}{\partial x_2} & \frac{\partial y_5}{\partial x_3} & \frac{\partial y_5}{\partial x_4} \\ \frac{\partial x_4}{\partial x_1} & \frac{\partial x_4}{\partial x_2} & \frac{\partial x_4}{\partial x_3} & \frac{\partial x_4}{\partial x_4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The resultant Sensitivity Matrix can be determined:

$$\begin{bmatrix}
\frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} & \frac{\partial y_1}{\partial x_4} \\
\frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} & \frac{\partial y_2}{\partial x_4} \\
\frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} & \frac{\partial y_3}{\partial x_3} & \frac{\partial y_3}{\partial x_4} \\
\frac{\partial x_1}{\partial x_1} & \frac{\partial x_1}{\partial x_2} & \frac{\partial x_1}{\partial x_3} & \frac{\partial x_1}{\partial x_4} \\
\frac{\partial x_2}{\partial x_1} & \frac{\partial x_2}{\partial x_2} & \frac{\partial x_2}{\partial x_3} & \frac{\partial x_2}{\partial x_4} \\
\frac{\partial x_3}{\partial x_1} & \frac{\partial x_3}{\partial x_2} & \frac{\partial x_3}{\partial x_3} & \frac{\partial x_3}{\partial x_4} \\
\frac{\partial y_4}{\partial x_1} & \frac{\partial y_4}{\partial x_2} & \frac{\partial y_4}{\partial x_3} & \frac{\partial y_4}{\partial x_4} \\
\frac{\partial y_5}{\partial x_1} & \frac{\partial y_5}{\partial x_2} & \frac{\partial y_5}{\partial x_3} & \frac{\partial y_5}{\partial x_4} \\
\frac{\partial x_4}{\partial x_1} & \frac{\partial x_4}{\partial x_2} & \frac{\partial x_4}{\partial x_3} & \frac{\partial x_4}{\partial x_4}
\end{bmatrix} = \begin{bmatrix}
1 & -5 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & -3 & -5 & 0 & 0 & 0 & 0 \\
-2 & -3 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
-2 & 0 & -3 & 0 & 0 & 0 & 1 & -3 & -2 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}^{-1} \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}$$

4.5 Traffic Dependencies in Client-Server Systems

In multi-level client-server systems certain traffic dependencies might require special treatment. We refer to the special cases where these interactions occur as interlocking. In order to obtain accurate performance estimates interlocking should be taken into consideration.

Interlocking occurs when a client and its server share a common resource. The resource can be either a software or hardware resource. In Figure 4-11, the two types of interlocking are presented. For the software interlock case, the client task can only be blocked waiting for either server1 or server2, but not both simultaneously. Similarly, in the hardware interlock case the client can either be executing on the processor or blocked waiting on the server1 but not both. Solutions that ignore the effects of interlock have the tendency to be pessimistic, since they overestimate contention.

[franks95] discusses the issue of interlocking and demonstrates why adjustment for interlock is necessary. The adjustment required to handle interlocking effects are summarized in the next section.

Figure 4-11 (a) Software Interlock (b) Hardware Interlock

4.5.1 Adjustment for Interlocking Effects

In this section, the interlocking equations and their partial derivatives will be described. The assumptions made will also be highlighted in this section.

Adjustment for interlocking involves the removal of flow from common sources that arrive at a common server by different paths. The equation for determining the queue length of a chain k at station m is:

$$Q_{mk} = X_{mk} R_{mk}$$

However, when we introduce the effects of interlocking the queue length equation is adjusted as follows:

$$Q_{mk} = \left[\left(\frac{\eta_S - 1}{\eta_S} \right) X_{mk}^{IL} + X_{mk}^{NOIL} \right] R_{mk}$$

The interlocked flow is reduced in proportion to the total number of sources, η_S , of flow along all chains with interlock. These sources are the tasks at the point where the

interlock paths diverge plus the immediate parents of the tasks that lie along the interlocked paths that source non-interlocked flow.

$$X_{mk} = X_{mk}^{IL} + X_{mk}^{NOIL}$$

$$X_{mk}^{IL} = X_{mk} Pr(IL)$$

$$X_{mk}^{NOIL} = X_{mk} (1 - Pr(IL))$$

$$Q_{mk} = \left(1 - \frac{1}{\eta_S}\right) X_{mk}^{IL} R_{mk} + X_{mk}^{NOIL} R_{mk}$$

The probability of interlocking or $Pr(IL)$, is the probability that an arrival that comes from an interlocked client. The calculation is described in [franks95]. In this thesis it will be taken as a constant and its derivation will not be calculated, ignoring some possible effects. For the sensitivity analysis, the partial derivatives of the above equations are determined.

$$\begin{aligned} \frac{\partial Q_{mk}}{\partial \xi} &= \left(X_{mk}^{IL} - \frac{X_{mk}^{IL}}{\eta_S} + X_{mk}^{NOIL} \right) \frac{\partial R_{mk}}{\partial \xi} + \left(R_{mk} - \frac{1}{\eta_S} R_{mk} \right) \frac{\partial X_{mk}^{IL}}{\partial \xi} + (R_{mk}) \frac{\partial X_{mk}^{NOIL}}{\partial \xi} + \\ &\quad \left(\frac{R_{mk} X_{mk}^{IL}}{\eta_S^2} \right) \frac{\partial \eta_S}{\partial \xi} \end{aligned}$$

$$\frac{\partial X_{mk}}{\partial \xi} = \frac{\partial X_{mk}^{IL}}{\partial \xi} + \frac{\partial X_{mk}^{NOIL}}{\partial \xi}$$

$$\frac{\partial X_{mk}^{IL}}{\partial \xi} = X_{mk} \frac{\partial Pr(IL)}{\partial \xi} + Pr(IL) \frac{\partial X_{mk}}{\partial \xi}$$

$$\frac{\partial X_{mk}^{NOIL}}{\partial \xi} = (1 - Pr(IL)) \frac{\partial X_{mk}}{\partial \xi} + X_{mk} \frac{\partial Pr(IL)}{\partial \xi}$$

For simplicity, let us assume that $\frac{\partial Pr(IL)}{\partial \xi} = 0$ and $\frac{\partial \eta_S}{\partial \xi} = 0$.

Therefore,

$$\frac{\partial X_{mk}^{IL}}{\partial \xi} = Pr(IL) \frac{\partial X_{mk}}{\partial \xi}$$

$$\frac{\partial X_{mk}^{NOIL}}{\partial \xi} = (1 - Pr(IL)) \frac{\partial X_{mk}}{\partial \xi}$$

$$\frac{\partial Q_{mk}}{\partial \xi} = \left(X_{mk}^{IL} - \frac{X_{mk}^{IL}}{\eta_S} + X_{mk}^{NOIL} \right) \frac{\partial R_{mk}}{\partial \xi} + \left(R_{mk} - \frac{1}{\eta_S} R_{mk} \right) \frac{\partial X_{mk}^{IL}}{\partial \xi} + R_{mk} \frac{\partial X_{mk}^{NOIL}}{\partial \xi}$$

The assumption about the probability of interlocking was employed. The probability is determined between solution iterations. For the interlocking calculations at a server the number of parents sources are required. These sources can reside in different submodels. Therefore, in the interlocking calculations the structure of the layered queueing model is used instead of the structure of the individual submodels. The probability is determined between iterations such that during solution the probability for each device is simply taken from a global variable and used. In addition, because the probabilities are determined outside the MVA model level, a translation stage where each station, entry, etc. would have to be translated to their equivalent representation in the MVA model would be required. This makes the integration of the probability equations into the existing analysis quite complex which is why it was not carried out.

4.6 Waiting times equations and scheduling disciplines

Different scheduling disciplines lead to different equations for waiting times. Section 4.2 considered FCFS scheduling, however because other disciplines have been implemented as part of the sensitivity analysis they will be considered.

There are a number of scheduling disciplines that can be represented in multi-class queueing models. The following are scheduling disciplines that are typically found:

- FCFS (First Come First Serve)
- PS (Processor Sharing)
- LCFS (Last Come First Served)
- Delay

The scheduling policies that are supported by the solver and that are part of the sensitivity analysis are FCFS, PS, and Delay. The residence time expressions for each are given below:

FCFS

$$R_{c,k} = V_{c,k} \left[S_{c,k} + \sum_{i=1}^C S_{i,k} Q_{i,k} (I - 1_c) \right]$$

PS

$$R_{c,k} = V_{c,k} \left[S_{c,k} + S_{c,k} \sum_{i=1}^C Q_{i,k} (I - 1_c) \right]$$

Delay

$$R_{c,k} = D_{c,k}$$

4.6.1 FCFS Scheduling with Multiple Heterogeneous Classes

The above response time equation for FCFS scheduling is one solution technique where the average service times per visit differ among customer classes. In this section we discuss another technique that yields better results for FCFS servers with high service time variation. In the LQNS solver this is referred to as HV-FCFS.

The residence time of a job at a service centre consists of service time plus queueing time. For FCFS centres, all the jobs already present at the centre form the queueing time.

Therefore each arriving job must wait on each customer class found in the queue but not yet in service plus the customer currently in service. Therefore the response time expression is:

$$R_{c,k} = V_{c,k} \left[S_{c,k} + \sum_{i=1}^C S_{i,k} [Q_{i,k}(I-1_c) - U_{i,k}(I-1_c)] + \sum_{j=1}^C r_{j,k} U_{j,k}(I-1_c) \right]$$

where $r_{j,k}$ is the average time until completion of a class j customer found to be in service by a class c arrival at centre k . $r_{j,k}$ is referred to as the mean residual service time of class j at centre k . The mean residual service time as described in [lazow84] is:

$$r_{j,k} = \frac{S_{j,k}}{2} + \frac{\sigma_k^2}{2S_{j,k}}$$

σ_k^2 is the service time variation at the server k . This is determined by the different average service time requests $S_{j,k}$ and variances in service time requests made by the arriving customers. The service time variance for server entries is based upon the series-parallel model. This is quite complex and is discussed in [rolia92] pages 123-136.

As part of this analysis, the service time variation at a server has been assumed to be a constant, $\frac{\partial \sigma_k^2}{\partial \xi} = 0$. Therefore, in the equations for the sensitivity analysis the partial derivative of the mean residual service time is:

$$\frac{\partial r_{j,k}}{\partial \xi} = \frac{1}{2} \frac{\partial S_{j,k}}{\partial \xi} - \frac{\sigma_k^2}{2S_{j,k}^2} \frac{\partial S_{j,k}}{\partial \xi}$$

instead of

$$\frac{\partial r_{j,k}}{\partial \xi} = \frac{1}{2} \frac{\partial S_{j,k}}{\partial \xi} - \frac{\sigma_k^2}{2S_{j,k}^2} \frac{\partial S_{j,k}}{\partial \xi} + \frac{\sigma_k}{S_{j,k}} \frac{\partial \sigma_k}{\partial \xi}$$

where subsequent equations for $\frac{\partial \sigma_k}{\partial \xi}$ would be required.

The reason why the partial derivatives for the variance are not included as part of the sensitivity analysis is due to the complexity in obtaining its partial derivatives. The variance calculations that are performed between iterations are separated among a number of different classes (Server, Variance, Probability, Entity, Entry, SeriesParallel). The variance calculations from each of the parts would have to be put together before they could be put in with the partial derivatives obtained in the MVA model. The equations obtained from the classes above would not be in same integer representation found in the MVA model. As a result translation to the MVA model representation would be required so that the variance equations could be used in the overall analysis. As with the probability of interlocking calculation, the variance calculation although possible using some sophisticated scripts was deemed to be beyond the scope of this thesis.

4.7 Sensitivities at Multi-servers

Multi-servers are service centres at which there is a single queue of customers waiting for service from two or more identical servers. Therefore, a multi-server can provide service to multiple requests simultaneously. The ROLIA Multi-server was used in the implementation of the sensitivity analysis. Details of the ROLIA Multi-server can be found in [rolia92].

The response time equation for a customer of class c arriving at a queueing centre k is:

$$R_{c,k} = V_{c,k} \left[S_{c,k} + \frac{\sum_{i=1}^C S_{i,k} Q_{i,k} (I - 1_c)}{M_k} P_B \right]$$

where M_k is the number of servers or multiplicity of device k and P_B is probability that a customer arriving at device k will find all servers busy. For the case where there is

only one server, $M_k = 1$ then $P_B = 1$ because all arriving customers will find the server busy. Therefore the response time equation will reduce simply to the response time equation for the FCFS server.

However, if the $M_k > 1$ then the probability of finding all the servers busy is approximately by $U_k^{M_k}$, where U_k is the utilization per server of the station k . Therefore, the response time equation for the $M_k > 1$ case is:

$$R_{c,k} = V_{c,k} \left[S_{c,k} + \frac{\sum_{i=1}^C S_{i,k} Q_{i,k} (I - 1_c)}{M_k} U_k^{M_k} \right]$$

To determine sensitivities with respect to server multiplicity we need to determine the partial derivatives of the response time equation with respect to server multiplicity. For the single server case, the partial derivative of the response time equation with respect to multiplicity is:

$$\frac{\partial R_{c,k}(\vec{N})}{\partial M_k} = - \left(\sum_{i=1}^C V_{i,k} S_{i,k} Q_{i,k} (I - 1_c) \right)$$

where $M_k = 1$. For the case where $M_k > 1$ the partial derivative of the response time equation to multiplicity is:

$$\frac{\partial R_{c,k}(\vec{N})}{\partial M_k} = \frac{- \left(\sum_{i=1}^C V_{i,k} S_{i,k} Q_{i,k} (I - 1_c) \right)}{M_k^2} + \frac{1}{M_k} (\log U_k) U_k^{M_k}$$

4.8 Deliberate Limitations to the Analysis

To keep the programming and debugging effort manageable, some of the calculations were excluded from the sensitivity analysis. As mentioned earlier, a number of assumptions have been made in the waiting times time equations for heterogeneous classes and the interlocking equations. The interlocking and variance calculations are simply special cases of a general fact that any parameter can be treated as a constant instead of a function (of other parameters). The effect of this depends on its (uncalculated) sensitivity.

The next chapter will make the scale of the effort more clear and Chapter 6 will evaluate the resulting limitations.

Chapter 5

lqnSENS: *Implementation of the Approach*

In this chapter the implementation of the approach outlined in Chapter 4 is described. lqnSENS (layered queueing network sensitivities) is the name of the tool that has been created to determine the sensitivities of all output parameters to all input parameters for a layered model. This chapter will describe the internals of lqnSENS, in particular the additions that were required to the LQNS solver, and the post model solution processing that was crucial in the determining the sensitivities.

5.1 lqnSENS

lqnSENS consist of five post solution stages as shown in Figure 5-3 below. *Stage I* is the Reduction and Matrix Generation stage. The derivative file obtained from the solver is reduced and a submodel matrix is generated. These matrices are then combined to form the global system matrix. *Stage II* takes the system matrix and solves for the sensitivities using gaussian elimination. Using the sensitivities determined in *Stage II*, elasticities are calculated. The sensitivity information calculated in *Stage II* is represented using integer identification for stations, chains, entry, etc. In *Stage III*, these integer identifications are translated into a more readable string format. Finally, in *Stage IV*, the elasticities are ranked. All of these stages will be described in further detail in the sections to follow. In addition Appendix A provides examples of outputs for each of the stages.

Input to this tool is simply a lqns model input file (without the.in extension). lqnSENS will produce a file of sensitivities (.sens) and a file with the ranking (.rank) of the absolute values of the sensitivities.

Figure 5-1 Input and output to the lqnSENS tool

5.2 Why Perl?

Stages I, III and IV shown in Figure 5-2, were written in Perl. Perl is a programming language that makes it easy to manipulate text files, and numbers. One of its advantages is that it is portable. Perl can run on all modern operating systems such as UNIX, MS-DOS, OS/2, Macintosh, etc. It has a rich software development environment complete with debuggers, profilers, compilers, interpreters, libraries syntax-directed editors. Overall, Perl is a very simple language to learn and code and it just executes like a shell script. The Perl scripts for Stages I, III and IV can be found in Appendix A.

5.3 Model Solution

As mentioned earlier it was decided to base sensitivities on Bard-Schweitzer Approximate MVA. The sensitivities obtained will therefore be sensitivities of Bard-Schweitzer solutions with respect the model input parameters. The approach employed can be extended to handle the equations of Linearizer [chandy82], a more accurate yet more complicated algorithm. It is also possible to use a combination of the results of a Linearizer solution for performance and sensitivities based on the Bard-Schweitzer equation. This however was not carried out.

Figure 5-2 The stages within lqnSENS

5.3.1 Adding the Sensitivity Option to the Solver

To perform a sensitivity analysis an new “-s” option has been added to the existing LQNS solver. With the -s option you simply provide the name you want a number of intermediate output files. This is however transparent when using lqnSENS. lqnSENS will automatically use the “-s” when using it invokes the solver. Refer to Figure 5-4 to see how the “-s” option is used.

5.3.2 Sensitivity header file

The so-called *plug-and-play* concept was used when deciding how to incorporate the sensitivity analysis approach into the solver. A header file containing global variables was created for the sensitivity analysis. Code examples of these global variables are file pointers and iteration flags. File pointers are included because it was necessary for different objects of different classes to write their output to a common file. The need for iteration flags is explained in section 5.3.3 . By grouping all sensitivity related variables in

Figure 5-3 Inside lqnSENS

Figure 5-4 Invoking the LQNS Solver

one file it makes it easy to add “instrumentation”. Instrumentation is the term used here to describe the approach where so-called “hooks” were placed inside the solver to output the partial derivatives of the various equations involved in the solution process. The sensitivity header file is included in each module that the partial sensitivities need to be obtained from.

5.3.3 Derivatives and iterations

There are two levels of iteration in the LQNS Solver, the first level being *outer* iterations and the second being *inner* iterations. These are explained in Chapter 3. After the model solution has converged to within some specified convergence criteria (default is successive differences less than 1×10^{-9}) one last outer iteration step is performed for the sensitivity analysis. In an outer iteration each MVA submodel is solved. Inner iterations refer to the iterations required when solving each individual MVA submodel. As part of the sensitivity analysis, an extra inner iteration is also performed. Derivatives are determined and outputted at the end of this extra inner iteration.

Two flags were created to identify when the last (extra) inner and outer iterations were being performed. These flags were then used to determine when to output the derivative information. `sensFlag` indicates the last outer iteration and `coreFlag` is used to indicate the last inner iteration. The use of these flags is demonstrated in Figure 5-

5. Derivative information is computed and written out only when both `sensFlag` and `coreFlag` have been set.

Figure 5-5 Iteration flags

5.3.4 Instrumenting and Extending the LQNS Solver

“Instrumenting” and extracting the necessary information from the solver proved to be one of the most challenging aspects of this project. The LQNS Solver is over 30,000 lines of C++ code with over 30 different class definitions. Understanding the structure of the classes was necessary. As is the case in many software development projects documentation was limited.

One goal when instrumenting was to minimize the number of changes to the existing LQNS code. As a result no changes have been made to the existing class definitions other than what is mentioned below. The highlighted classes in Figure 5-6 indicate the classes to which instrumentation was added.

For the `server` class a public variable `sens_id` has been added to the class definition. Each new instantiation of a server object is given a unique identification which is stored in the variable called `sens_id`. The LQNS solver has been set up such that clients and servers are given integer identifications only within a MVA submodel. Outside of the submodel their string names are used. Submodel interaction effects are coordinated

at a level above the MVA model level, in the submodel class. However, in order to introduce both interaction and submodel partial derivative information within the system matrix, a server appearing in an interaction equation must have the same identification as in the mva model and vice-versa. `sens_id` and the `get_sub_num()` function (described in Section 5.3.5) ensure that the same server has the same unique string identification.

Figure 5-6 Highlighted classes were instrumented

The advantage of this approach is that we can apply the underlying sensitivity analysis technique easily to other distributed system solvers such as MOL.

5.3.5 Instrumenting for the Derivatives

All equations that were part of the solution were instrumented. This involved understanding the structure of the code and then tracing the call sequence for a typical solution. Once the important functions/methods were identified instrumentation was added so that the value of the derivative of the equations used in the solution could be written out.

The instrumentation of the utilization equation below is shown in Figure 5-7 .

$$U = XVS$$

```

U[m][e][k][n] = X[k][n] * Q[m]->V(e,k) * Q[m]->S(e,k);          /* Eq:3.7 */

if ((sensFlag) && (coreFlag)) {
    der_out << "U[" << get_sub_num(m) << "]" << e << "[" << k << "]" << n << "]" << X[" << k << "]" << n << "]" << Q[m]-
>V(e,k)*Q[m]->S(e,k)) << endl;
    der_out << "U[" << get_sub_num(m) << "]" << e << "[" << k << "]" << n << "]" << "V(" << e << "," << k << ")" << Q[m]-
>sens_id << "x" << X[k][n] * Q[m]->S(e,k) << endl;
    der_out << "U[" << get_sub_num(m) << "]" << e << "[" << k << "]" << n << "]" << "S(" << e << "," << k << ")" << Q[m]-
>sens_id << "x" << X[k][n] * Q[m]->V(e,k) << endl;
}

```

U : Utilization
 X : Throughput
 V : Visits
 S : Service Time

m : station
 e : entry
 k : chain
 n : population

Figure 5-7 Instrumentation of a utilization equation in mva class of the LQNS Solver

The function `get_sub_num()` is used to determine the global station number of device `m` (the device the utilization is being calculated for). `m` is the station local identification within this submodel, however the interaction equations use a global station identifier. In order to keep both consistent, the global identifier for this station is determined using its local identification and knowledge of the submodel layer. This will become more clear in Section 5.4.2, where the interaction instrumentation is explained.

5.4 Stage I : Reduction and Matrix Generation

Three files for the sensitivity analysis are created by the solver as shown in Figure 5-3. The three files are:

- Derivatives File (.der)
- Interaction File (.int)
- Identification File (.id)

5.4.1 The .der file

The .der file consists of three columns as shown below in Figure 5-8. Consider the following equation occurring within a submodel. This is the equation shown above in Figure 5-7.

$$U = XVS$$

The partial derivative of U is:

$$\frac{\partial U}{\partial X} = VS \quad \frac{\partial U}{\partial V} = XS \quad \frac{\partial U}{\partial S} = XV$$

Dependent Variable Name (string name)			Independent Variable Name (string name)		
Column 1	Column 2	Column 3			
U	X	VS			
U	V	XS			
...	...	...			

Figure 5-8 The Derivative (.der) file

The first column in the derivative file names the result variable, the second column names the parameter what the derivative is taken with respect to and finally the third column is the value of the partial derivative. The .der file obtained from the solver still

contains intermediate variables. These intermediate variables can be eliminated. It was decided to perform some reduction on the .der file. The advantage of doing so is that final size of the system matrix is reduced. The reduction algorithm employed is described in Section 5.4.1.1 .

See Appendix A for an example of what a .der file looks like prior to reduction.

5.4.1.1 Derivative File Reduction

There are often variables included in the derivative file which can be eliminated right away because they are not external (data or parameters of the entire system) or internal variables. These variables are eliminated by using the chain rule of differentiation. An example of part of a derivative file is given below:

sumOf_SL_m_21	S (1,1)_2	0.327148
sumOf_SL_m_21	L [2] [1] [1] [0]	2.14163
W [1] [1] [0]_2	S (1,1)_2	1.0
W [1] [1] [0]_2	sumOf_SL_m_21	1.0
W [1] [1] [0]_2	M [2]	-0.70063

In the above example `sumOf_SL_m_21` is an example of a variable that can be reduced. `M [2]` is an external variable, and `W [1] [1] [0]_2` is an example of an internal variable. The reduction step will eliminate all non internal and non external variables until only internal and external variables are in column 1 and column 2 of the derivatives file In this case substituting any occurrences of `sumOf_SL_m_21` in the second column with `S (1,1)_2`, and `L [2] [1] [1] [0]`. Therefore, the above example is reduced to:

W [1] [1] [0]_2<1>	L [2] [1] [1] [0] <1>	2.14163
W [1] [1] [0]_2<1>	M [2] <1>	-0.70063
W [1] [1] [0]_2<1>	S (1,1)_2<1>	1.327148

The algorithm to perform the reduction has been optimized by using hash tables to avoid the cost of performing linear searches at the expense of increased memory use.

The reduction stage is optional and can be turned off if desired. It was found that on some very large submodels the overhead of performing the reduction simply outweighs the solution time of larger matrices.

5.4.1.2 Reduction Algorithm

The first step in the reduction is to put aside all entries in the derivative file that cannot be reduced, those which do not have non-reducible parameters in column 1 and 2. Reduction then takes place on the rest of the file. The derivative file is stored in an array (in memory) to minimize the overhead of opening/closing and files. If the variable, X , in the second column of a line is not an internal or external variable then it must be reduced. The derivative file is searched for the line that contains a variable X in its first column. A candidate reduction variable can be reduced to derivatives of internal / external variables or other variables that need to be reduced further. If the substitution or chain rule differentiation leads to internal and external variables in both columns then this line cannot be further reduced. Therefore, this line is put aside with the other lines that can not be reduced. Reduction continues on lines throughout the file until nothing can be further reduced, rather everything has been put aside.

See Appendix A for a sample of a reduced .der file.

5.4.2 The .int file

From the .int file the interaction between submodels is determined. Within the LQNS solver, the transfer of waiting times is coordinated through “wait” array variables. Each entry of a task in the layered model has specific data object that is simply an array of waiting times for that entry, where each element in the array corresponds to the number of submodels. Figure 5-9 demonstrates how waiting times are stored and retrieved from the entry *wait* array. The interaction file captures this storage and retrieval from the wait arrays to determine the transfer of waiting times or interactions between submodels. For example, in Figure 5-9 the service time of the entry of server S2 is determined by summing the waiting times of this server in another submodel. In this case, S2 only waits on processor P2 in submodel 2. After solving submodel 2, this waiting time is stored in the

wait array for this server entry. Submodel 1 is this waiting time is used as the service time for server S2.

The utilization of a software server is used to determine the think time of that server in another submodel. The interaction file contains derivatives for the transfer of these utilizations equations.

Interaction effects and transfer of waiting times and utilization between layers or submodels are accounted for at a level above the mva level. Identification of stations, processes, entries, chains differ at the mva level from the level above the mva level. In order to obtain a consistent set of interaction that can go along with mva equations (required for the construction of the System Matrix), the interaction effects obtained in the .int file have to be translated to their mva level representation. This is done using the identification (.id) file.

See Appendix A for a sample of a .int file and a condensed and translated version of the interaction file. The translated version has the .itx extension.

5.4.3 The .id file

The .id file serves a very important role in the sensitivity analysis process. It is used in stages I and III. In Stage I, the .id file is used to generate the submodel interaction equations so that they are in the format that is consistent with the derivative equations found in the .der file. This conversion involves substituting string representations of stations, entries, etc. with their equivalent integer representation that is found in the derivative file.

As mentioned earlier, in the mva model all devices or stations, entries, and chains are identified using numbers or integer representations. In Stage II, the identification or .id file is used to translate those numbers to their string equivalent such that sensitivity results can be easily interpreted.

Within the mva solution service times of tasks are not classified as external variables. In the layered model definition file you specify the service time at tasks, however tasks are

Figure 5-9 Determining the interactions between submodels, for "wait" variables

mapped on to processors. It is the sensitivity of the output measures to the service times of the tasks at the processors that is required. The .id file also identifies whether a server is a task or a processor. If the server is a processor, then the service times at the processor are the service times we want the sensitivities with respect to.

See Appendix A for a sample of a .id file.

5.5 Stage II: Sparse Matrix Operations

Matlab was used to perform the matrix solution of the system matrix ($AX=B$). Matlab was used because it has built in gaussian elimination algorithms for solution of a set of linear equations, and support for sparse matrices. Matrix manipulation operations have been optimized in Matlab.

When dealing with large matrices, issues of memory storage become very important. It was discovered that the majority of entries in our matrices were zero which is why it was decided to use sparse matrix storage and operations for the matrices.

The creation of submodel matrices occurs after the derivative file has been reduced. The first step in the creation of the matrices involves determining which variables are internal and external.

Once the internal and external variables have been determined, the rows and columns of the submodel matrices can be determined. Each row and column in the submodel matrix corresponds to a particular internal or external variable.

The following figure shows the structure of the submodel matrix. However, because sparse matrices are being used the zero elements are removed.

Figure 5-10 Derivative and Matrix Generation Stage

Figure 5-11 Setup of a Submodel matrix

Figure 5-12 Setup of a System matrix

Once the matlab file has been created and the stage 1 perl script ends, lqnSENS will automatically invoke Matlab to execute the .m file that was created in stage 1. Matlab solves the sets of linear equations and outputs the solution to a .res file.

5.5.1 Stage III: Translation / Identification

Stage III was employed to translate the sensitivity results into a more readable format. In the derivative file names have integer representations such as `w[1][2][1]_2`. This is read as the waiting time of entry 1, chain 2, phase 2 at station 2. This is obviously not very readable. String names for these integers are obtained from the .id file. Stage III is used to convert these integer representations into their equivalent string representation making it more readable. See Figure 5-13.

Figure 5-13 Translation and Identification Stage

5.5.2 Stage IV: Ranking

For the ranking of the elasticities another perl script was used. The sensitivities in the.sens file are read in and stored into an array. The elasticities are then sorted in descending order based on their absolute values.

See Appendix A for an example of .rank file.

Chapter 6

Results

In this chapter, results from the study of a number of client-server examples have been presented. These results demonstrate that the accuracy and speed of the technique. After each example a brief discussion of the results is presented.

A brief discussion of the results are included at the end of each example. Some examples include all features (interlocking etc.) and the results have been found to be very accurate. Other examples are sensitive to the assumptions outlined in Chapter 4 and the accuracy of the results is mixed.

A table containing the executions times¹ of the model solution and for each stage of the post-processing is presented for each example. Executions times were obtained using the UNIX *time* system call. All the models were solved on a Sun SPARC Ultra Workstation.

Sample calculations are only presented for the first case. The table below will help in the understanding of the Layered Queueing Network (LQN) Models and tables presented in this chapter.

TM	Task Multiplicity
ST	Service Time
Z	Think Time
V	Visits
FCFS	First-Come First-Serve
PS	Processor Sharing
HV-FCFS	High Variance FCFS
E1	Entry 1
PX	Processor X

1. all experiments were run on an SUN ULTRA SPARC Workstation running Solaris

S	Sensitivity
E	Elasticity

6.1 Simple Client-Server Case with no Interlocking

The first case studied is a simple Client-Server system with no interlocking. A LQN model of the system is presented in Figure 6-1. In this example, both Client and Server reside on their own processor. The scheduling discipline at both processors is FCFS. There are two Clients ($TM=2$) in this example and each makes on the average two ($V=2$ for Client calls to the Server) requests to entry E1 of the Server. The service time of the Client at processor P1 is 2 msec ($ST=[0,2]$ for Client) and similarly the server time of the Server at processor P2 is also 2 msec ($ST=[2,0]$ for Server). The difference being that a clients service always occurs as a phase 2 execution while the service time of the server can either be phase 1 or phase 2. In this example the server only has phase 1 execution. The think time of the Client is 8 msec ($Z=8$) which means that after receiving the reply of a request it waits on the average 8 msecs before generating the subsequent request.

At solution time this system is decomposed into two separate submodels. The first submodel consists of a client, Client and two servers, Server and P1 as shown in Figure 6-1. The second submodel simply consists of one client, Server, and one server, P2.

Figure 6-1 Simple Client-Server Case with no Interlocking

Execution time results for this example are given below. The model took approximately 0.4 seconds to solve. The post processing stages of the sensitivity analysis technique took a total of 5.13 seconds to yield a sensitivity matrix of size 68 x 13. That is a total of 884 different sensitivities. It was decided to keep only sensitivities of task/processor throughput, task/processor utilization, task response time per entry, and queue length per entry even though sensitivities for such things as throughput of entries for each task, waiting times per entry are available. Of those 884 sensitivities, only approximately 300 are kept and outputted to the .sens and .rank file.

In this example there were 13 input parameters to the model. These can be found in the .etl files for the different submodels of this example and others. The sensitivities using the perturbation approach would require 26 separate model solutions each executing for approximately 0.4 secs for a total of 10 secs. Therefore, the Direct approach in this case is said to be twice as fast as the Perturbation approach.

Solution Time	0.4 sec	
Stage I (Reduction and Matrix Generation)	1.48 sec	Total: 5.13 sec
Stage II (Sparse Matrix Operations)	3.14 sec	
Stage III (Post Processing)	0.41 sec	
Stage IV (Ranking)	0.11 sec	
SENSITIVITY MATRIX SIZE	68 x 13	
Estimated Perturbation Approach Time	~10 sec	Ratio: ~2
Direct Approach Time	~5 sec	

The next tables verify the sensitivities and elasticities obtained using the Direct approach against those obtained using the Perturbation approach. Percentage difference between the two approaches have also been calculated and are included in the tables.

Sample Calculations

Perturbation Sensitivity

When determining sensitivities with respect to service times using the perturbation approach LQNS makes it more convenient to use the slice time of a process rather than its total execution time. The slice time is the actual processor service time when executing an entry, as described in Chapter 3. This is demonstrated in the sample calculations below for Table 6-1 :

Client Throughput at Client service time of 1.9 msec is 0.128619.

Client Throughput at Client service time of 2.1 msec is 0.126738.

Client total service time at P1 is 2.0 msec.

Client slice time at P1 is $\frac{2.0ms}{3} = 0.6667$.

Perturbation slice time interval is $\frac{2.1ms}{3} - \frac{1.9ms}{3} = 0.067$.

Perturbation Sensitivity is $\frac{0.126738 - 0.128619}{0.0667} = -0.0282$

Percentage difference calculation is

$$\frac{Perturbation - Direct}{Perturbation} \times 100 = \frac{-0.0282 - (-0.0282)}{-0.0282} \times 100 = 0.0\%$$

Perturbation Elasticity

Client Throughput at Client service time of 2.0 msec (operating point) is 0.1277.

Client slice time at P1 is $\frac{2.0ms}{3} = 0.6667$.

Perturbation Sensitivity is -0.0282.

Perturbation Elasticity is $\frac{\partial Y}{\partial X} \frac{X}{Y} = (-0.0282) \frac{0.6667}{0.1277} = -0.1473$

Percentage difference calculation is

$$\frac{\text{Perturbation} - \text{Direct}}{\text{Perturbation}} \times 100 = \frac{-0.1473 - (-0.1473)}{-0.1473} \times 100 = 0.0\%$$

In Table 6-1 the sensitivities and elasticities of the output parameter in column 1 of the table with respect the service time of the Client are presented. Below each output parameter is the value of the output parameter at the operating point

Table 6-1 : Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)

Output Parameter (and value)		Direct	Perturbation (1.9, 2.1)	%diff
Client Throughput (0.1277)	<i>S</i>	-0.0282	-0.0282	0.0 %
	<i>E</i>	-0.1473	-0.1473	0.0 %
Server Throughput (0.2554)	<i>S</i>	-0.0562	-0.0561	0.0 %
	<i>E</i>	-0.1473	-0.1472	0.0 %
Server Utilization (0.5107)	<i>S</i>	-0.1128	-0.1128	0.0 %
	<i>E</i>	0.8527	0.8527	0.0 %
P1 Utilization (0.2553)	<i>S</i>	0.3266	0.3266	0.0 %
	<i>E</i>	0.8527	0.8527	0.0 %
P2 Utilization (0.5107)	<i>S</i>	-0.1128	-0.1128	0.0 %
	<i>E</i>	-0.1473	-0.1473	0.0 %

In Table 6-2 the sensitivities and elasticities of the output parameters in column 1 of the table with respect to the service time of the Server are presented.

Table 6-2 : Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0)

Output Parameter (and value)		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1277)	<i>S</i>	-0.02581	-0.0258	0.0 %
	<i>E</i>	-0.4043	-0.4041	0.0 %
Server Throughput (0.2554)	<i>S</i>	-0.05162	-0.0516	0.0 %
	<i>E</i>	-0.4043	-0.4041	0.0 %

Table 6-2 : Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0)

Output Parameter (and value)		Direct	Perturbation (1.9,2.1)	%diff
Server Utilization (0.5107)	<i>S</i>	0.1521	0.1521	0.0 %
	<i>E</i>	0.5957	0.5956	0.0 %
P1 Utilization (0.2553)	<i>S</i>	-0.05162	-0.0516	0.0 %
	<i>E</i>	-0.4043	-0.4041	0.0 %
P2 Utilization (0.5107)	<i>S</i>	0.1521	0.1521	0.0 %
	<i>E</i>	0.5957	0.5956	0.0 %

In Table 6-3 the sensitivities and elasticities of the output parameters in column 1 of the table with respect to the number of visits from Client to Server [E1] are given.

Table 6-3 : Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)

Output Parameter (and value)		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1277)	<i>S</i>	-0.02581	-0.0258	0.0 %
	<i>E</i>	-0.4043	-0.4043	0.0 %
Server Throughput (0.2554)	<i>S</i>	0.0761	0.0761	0.0 %
	<i>E</i>	0.5957	0.5960	0.0 %
Server Utilization (0.5107)	<i>S</i>	0.1521	0.1521	0.0 %
	<i>E</i>	0.5957	0.5956	0.0 %
P1 Utilization (0.2553)	<i>S</i>	-0.0516	-0.0516	0.0 %
	<i>E</i>	-0.4043	-0.4041	0.0 %
P2 Utilization (0.5107)	<i>S</i>	0.1521	0.1521	0.0 %
	<i>E</i>	0.5957	0.5956	0.0 %

In Table 6-4 the sensitivities and elasticities of the output parameters in column 1 of the table with respect to the think time of the Client are presented

Table 6-4 : Sensitivity/Elasticity wrt to Client Think Time (Z=8)

Output Parameter (and value)		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1277)	<i>S</i>	-0.0072	-0.0072	0.0 %
	<i>E</i>	-0.4484	-0.4511	0.6 %
Server Throughput (0.2554)	<i>S</i>	-0.0143	-0.0143	0.0 %
	<i>E</i>	-0.4484	-0.4480	0.0 %
Server Utilization (0.5107)	<i>S</i>	-0.0286	-0.0286	0.0 %
	<i>E</i>	-0.4484	-0.4480	0.0 %
P1 Utilization (0.2553)	<i>S</i>	-0.0143	-0.0143	0.0 %
	<i>E</i>	-0.4484	-0.4480	0.0 %
P2 Utilization (0.5107)	<i>S</i>	-0.0286	-0.0286	0.0 %
	<i>E</i>	-0.4484	-0.4480	0.0 %

In Table 6-5 the sensitivities and elasticities of the output parameters in column 1 of the table with respect to the Client population.

Table 6-5 : Sensitivity/Elasticity wrt to Client Population (N=2)

Output Parameter (and value)		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1277)	<i>S</i>	0.0483	0.0484	0.3 %
	<i>E</i>	0.7559	0.7581	0.3 %
Server Throughput (0.2554)	<i>S</i>	0.0965	0.0968	0.3 %
	<i>E</i>	0.7559	0.7582	0.3 %
Server Utilization (0.5107)	<i>S</i>	0.1930	0.1936	0.3 %
	<i>E</i>	0.7559	0.7582	0.3 %
P1 Utilization (0.2553)	<i>S</i>	0.0965	0.0968	0.3 %
	<i>E</i>	0.7559	0.7582	0.3 %
P2 Utilization (0.5107)	<i>S</i>	0.1930	0.1936	0.3 %
	<i>E</i>	0.7559	0.7582	0.3 %

From the above results it is clear that the sensitivities/elasticities obtained using the Direct approach are accurate when compared to the Perturbation approach for this case. A ranking of the elasticities are given in Figure 6-2. From this figure one can see that both Client and Server throughput are most sensitive to the population of the Client. A description of each column is given in Figure 6-2.

The reason for the error in the sensitivities with respect to population is explained in Section 6.2.1.

6.2 Second Simple Client-Server Case with Interlocking

In the previous chapter, it was noted that the variation in the interlocking parameter is ignored and its derivative is set to zero. The previous example did not have any interlocking so the interlocking parameter did not play a role. The next example investigates this assumption using a simple example.

In the next few examples, the Client and Server have been placed on a common processor, P1. This system is said exhibit the interlocking effect because both Client and Server share a common resource, P1. This example has a very strong interlocking effect.

For this example three different waiting time calculations (PS, FCFS, and HV-FCFS) were investigated. Two very significant assumptions were made in Chapter 4 and the following client-server examples will test these assumptions.

The first assumption assumes that the probability of interlocking is a constant or $\frac{\partial Pr(IL)}{\partial \xi} = 0$ and the number of sources to the interlocked device are also a constant or $\frac{\partial \eta_S}{\partial \xi} = 0$. This was described in Section 4.5.1. The other assumption made is about the variance calculation in the HV-FCFS waiting time calculation. As described in subsection 4.6.1, the variance is treated as a constant, or $\frac{\partial \sigma_k^2}{\partial \xi} = 0$. This assumption is tested for its accuracy and only applies to the third example in this section, or in Section 6.2.3.

ELASTICITY	SENSITIVITY	OUTPUT VALUE <submodel#>	PARAMETER <submodel#>
0.7561555966	+	4.7916824e-02	THROUGHPUT_[CLIENT] <1>
0.7561527555	+	9.5833666e-02	THROUGHPUT_[SERVER] <2>
0.7561526056	+	9.5833647e-02	THROUGHPUT_[SERVER] <1>
0.6006199616	+	7.6121673e-02	THROUGHPUT_[SERVER] <2>
0.6006178470	+	7.6121405e-02	THROUGHPUT_[SERVER] <1>
0.4451460683	-	-7.0521153e-03	THROUGHPUT_[CLIENT] <1>
0.4451443879	-	-1.4104233e-02	THROUGHPUT_[SERVER] <2>
0.4451443247	-	-1.4104231e-02	THROUGHPUT_[SERVER] <1>
0.3993797914	-	-2.5308298e-02	THROUGHPUT_[CLIENT] <1>
0.3993791917	-	-2.5308260e-02	THROUGHPUT_[CLIENT] <1>
0.3993778370	-	-5.0616548e-02	THROUGHPUT_[SERVER] <2>
0.3993776161	-	-5.0616520e-02	THROUGHPUT_[SERVER] <1>
0.1554780208	-	-2.8149962e-02	THROUGHPUT_[CLIENT] <1>
0.1554774378	-	-5.6299935e-02	THROUGHPUT_[SERVER] <2>
0.1554774074	-	-5.6299924e-02	THROUGHPUT_[SERVER] <1>
0.10123334896	+	1.2830130e-02	THROUGHPUT_[CLIENT] <1>
0.10123331060	+	2.5660264e-02	THROUGHPUT_[SERVER] <2>
0.10123330862	+	2.5660259e-02	THROUGHPUT_[SERVER] <1>
			POPULATION_[CLIENT] <1>
			POPULATION_[CLIENT] <1>
			POPULATION_[CLIENT] <1>
			VISITS_[CLIENT] -> [SERVER] [SERVER_E1] <1>
			VISITS_[CLIENT] -> [SERVER] [SERVER_E1] <1>
			THINK_TIME_[CLIENT] <1>
			THINK_TIME_[CLIENT] <1>
			THINK_TIME_[CLIENT] <1>
			VISITS_[CLIENT] -> [SERVER] [SERVER_E1] <1>
			SERVICE_TIME_[SERVER]@[proc2] [SERVER_E1] <2>
			SERVICE_TIME_[SERVER]@[proc2] [SERVER_E1] <2>
			SERVICE_TIME_[SERVER]@[proc2] [SERVER_E1] <2>
			SERVICE_TIME_[CLIENT]@[proc1] [CLIENT] <1>
			SERVICE_TIME_[CLIENT]@[proc1] [CLIENT] <1>
			SERVICE_TIME_[CLIENT]@[proc1] [CLIENT] <1>
			POPULATION_[SERVER] <1>
			POPULATION_[SERVER] <1>
			POPULATION_[SERVER] <1>

Figure 6-2 Ranking of Elasticities for the Simple Client-Server case with no Interlocking

The parameters used in the previous example (no interlocking example) have been maintained for the next three examples, the only difference being that both Client and Server reside on the same processor and the waiting time calculations at the processor have been varied.

6.2.1 Example with Processor Sharing

In this example the scheduling discipline at the processor is Processor Sharing or PS.

Figure 6-3 Simple Client-Server Case with Interlocking and PS

Similar to the previous example solution times as well as sensitivity/elasticity data have been included in the tables below. The Direct Approach is found to be approximately three times as fast as the Perturbation based approach.

Solution Time	0.59 sec
---------------	----------

Stage I (Reduction and Matrix Generation)	2.32 sec	Total: 7.38 sec
Stage II (Sparse Matrix Operations)	3.92 sec	
Stage III (Post Processing)	0.97 sec	
Stage IV (Ranking)	0.17 sec	
SENSITIVITY MATRIX SIZE	125 x 19	
Estimated Perturbation Approach Time	~23 sec	Ratio: 2.9
Direct Approach Time	~ 8 sec	

From Table 6-6 to Table 6-10 sensitivity/elasticity results for the two approaches for this example are presented. In general the results for the Direct approach are close to the Perturbation based approach. However, the results for the sensitivities/elasticities with respect to the population of the client (See Table 6-10) are not very close to the Perturbation approach. The results are different because with populations we are more restricted when selecting the interval width. With populations we are restricted to integer number of clients, the interval selected was 1 and 3. For example, in Figure 6-4 a graph of the throughput of Client vs. Client Population is presented. In this case, one could say that the perturbation results are not really comparable, and the idea of a derivative or elasticity is rather loose.

Figure 6-4 Client throughput as the Client population is increased

Table 6-6 : Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Client Throughput (0.1172)	<i>S</i>	-0.0343	-0.0343	0.0 %
	<i>E</i>	-0.1950	-0.1951	0.0 %
Server Throughput (0.2344)	<i>S</i>	-0.0686	-0.0686	0.0 %
	<i>E</i>	-0.1950	-0.1951	0.0 %
Server Utilization (0.5431)	<i>S</i>	-0.0656	-0.0656	0.0 %
	<i>E</i>	-0.0805	-0.0805	0.0 %
P1 Utilization (0.7033)	<i>S</i>	0.1459	0.1459	0.0 %
	<i>E</i>	0.1383	0.1383	0.0 %

Table 6-7 : Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1172)	<i>S</i>	-0.0257	-0.0257	0.0 %
	<i>E</i>	-0.4388	-0.4390	0.0 %
Server Throughput (0.2344)	<i>S</i>	-0.0514	-0.0514	0.0 %
	<i>E</i>	-0.4388	-0.4390	0.0 %
Server Utilization (0.5431)	<i>S</i>	0.1392	0.1392	0.0 %
	<i>E</i>	0.5124	0.5128	0.0 %
P1 Utilization (0.7033)	<i>S</i>	0.0801	0.0806	0.6 %
	<i>E</i>	0.2278	0.2294	0.7 %

Table 6-8 : Sensitivity/Elasticity wrt to Visits from Client to Server [E1] (V=2)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1172)	<i>S</i>	-0.0257	-0.0257	0.0 %
	<i>E</i>	-0.4388	-0.4390	0.0 %
Server Throughput (0.2344)	<i>S</i>	0.0658	0.0658	0.0 %
	<i>E</i>	0.5612	0.5615	0.0 %
Server Utilization (0.5431)	<i>S</i>	0.1391	0.1392	0.0 %
	<i>E</i>	0.5124	0.5128	0.0 %
P1 Utilization (0.7033)	<i>S</i>	0.0801	0.0801	0.0 %
	<i>E</i>	0.2278	0.2280	0.0 %

Table 6-9 : Sensitivity/Elasticity wrt to Client Think Time (Z=8)

Output Parameter		Direct	Perturbation (7.9,8.1)	%diff
Client Throughput (0.1172)	<i>S</i>	-0.0054	-0.0054	0.0 %
	<i>E</i>	-0.3661	-0.3662	0.0 %
Server Throughput (0.2344)	<i>S</i>	-0.0107	-0.0107	0.0 %
	<i>E</i>	-0.3661	-0.3662	0.0 %
Server Utilization (0.5431)	<i>S</i>	-0.0293	-0.0293	0.0 %
	<i>E</i>	-0.4319	-0.4319	0.0 %
P1 Utilization (0.7033)	<i>S</i>	-0.0322	-0.0321	0.0 %
	<i>E</i>	-0.3661	-0.3661	0.0 %

Table 6-10 : Sensitivity/Elasticity wrt to Client Population (N=2)

Output Parameter		Direct	Perturbation (1,3)	%diff
Client Throughput (0.1172)	<i>S</i>	0.0436	0.0337	-29.4 %
	<i>E</i>	0.7438	0.5745	-29.5 %
Server Throughput (0.2344)	<i>S</i>	0.0872	0.0674	-29.4 %
	<i>E</i>	0.7437	0.5746	-29.4 %
Server Utilization (0.5431)	<i>S</i>	0.1976	0.2218	10.9 %
	<i>E</i>	0.7278	0.8167	10.9 %
P1 Utilization (0.7033)	<i>S</i>	0.1443	0.2020	28.6 %
	<i>E</i>	0.4104	0.5746	28.6 %

6.2.2 Example with First-Come First-Serve (FCFS)

In this example the scheduling discipline at the processor is First-Come First-Serve or FCFS. The only difference between this and the previous example is the waiting time expression at the processor.

Figure 6-5 Simple Client-Server Case with Interlocking and FCFS

The following table presents the time taken to obtain all the sensitivities using the direct approach. Recall that the comparison is also made again to the most optimistic estimate of the perturbation approach.

Solution Time	0.51 sec	
Stage I (Reduction and Matrix Generation)	2.01 sec	Total: 6.8 sec
Stage II (Sparse Matrix Operations)	3.84 sec	
Stage III (Post Processing)	0.77 sec	
Stage IV (Ranking)	0.19 sec	
SENSITIVITY MATRIX SIZE	120 x 20	
Estimated Perturbation Approach Time	~20 sec	Ratio: ~2.8
Direct Approach Time	~7 sec	

Table 6-11 : Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Client Throughput (0.1151)	S	-0.0264	-0.0264	0.0 %
	E	-0.1532	-0.1532	0.0 %

Table 6-11 : Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Server Throughput (0.2302)	<i>S</i>	-0.0528	-0.0529	0.0 %
	<i>E</i>	-0.1531	-0.1531	0.0 %
Server Utilization (0.4930)	<i>S</i>	-0.0424	-0.0424	0.0 %
	<i>E</i>	-0.0573	-0.0573	0.0 %
P1 Utilization (0.6906)	<i>S</i>	0.1866	0.1867	0.0 %
	<i>E</i>	0.1801	0.1801	0.0 %

Table 6-12 : Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1151)	<i>S</i>	-0.0278	-0.0278	0.0 %
	<i>E</i>	-0.4824	-0.4825	0.0 %
Server Throughput (0.2302)	<i>S</i>	-0.0555	-0.0555	0.0 %
	<i>E</i>	-0.4824	-0.4825	0.0 %
Server Utilization (0.4930)	<i>S</i>	0.1130	0.1131	0.0 %
	<i>E</i>	0.4586	0.4589	0.0 %
P1 Utilization (0.6906)	<i>S</i>	0.0636	0.0637	0.0 %
	<i>E</i>	0.1843	0.1844	0.0 %

Table 6-13 : Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1151)	<i>S</i>	-0.0246	-0.0267	7.9 %
	<i>E</i>	-0.4272	-0.4641	8.0 %

Table 6-13 : Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Server Throughput (0.2302)	<i>S</i>	0.0659	0.0617	-6.8 %
	<i>E</i>	0.5728	0.5363	-6.8 %
Server Utilization (0.4930)	<i>S</i>	0.1377	0.1213	-13.5 %
	<i>E</i>	0.5586	0.4923	-13.5 %
P1 Utilization (0.6906)	<i>S</i>	0.0827	0.0700	-18.1 %
	<i>E</i>	0.2395	0.2028	-18.1 %

Table 6-14 : Sensitivity/Elasticity wrt to Client Think Time (Z=8)

Output Parameter		Direct	Perturbation (7.9,8.1)	%diff
Client Throughput (0.1151)	<i>S</i>	-0.0052	-0.0052	0.0 %
	<i>E</i>	-0.3644	-0.3642	0.0 %
Server Throughput (0.2302)	<i>S</i>	-0.0105	-0.0105	0.0 %
	<i>E</i>	-0.3644	-0.3645	0.0 %
Server Utilization (0.4930)	<i>S</i>	-0.0247	-0.0247	0.0 %
	<i>E</i>	-0.4013	-0.4013	0.0 %
P1 Utilization (0.6906)	<i>S</i>	-0.0315	-0.0315	0.0 %
	<i>E</i>	-0.3644	-0.3644	0.0 %

Table 6-15 : Sensitivity/Elasticity wrt to Client Population (N=2)

Output Parameter		Direct	Perturbation (1,3)	%diff
Client Throughput (0.1151)	<i>S</i>	0.0430	0.0339	-26.8 %
	<i>E</i>	0.7470	0.5884	-27.0 %

Table 6-15 : Sensitivity/Elasticity wrt to Client Population (N=2)

Output Parameter		Direct	Perturbation (1,3)	%diff
Server Throughput (0.2302)	<i>S</i>	0.0860	0.0683	-26.0 %
	<i>E</i>	0.7470	0.5936	-25.8 %
Server Utilization (0.4930)	<i>S</i>	0.1850	0.1822	-1.5 %
	<i>E</i>	0.7504	0.7394	-1.5 %
P1 Utilization (0.6906)	<i>S</i>	0.1429	0.2050	30.5 %
	<i>E</i>	0.4137	0.5936	30.3 %

In general the sensitivity/elasticity results obtained using the Direct approach with the FCFS waiting time expressions yield good results when compared to perturbation, typically 0 %. In Table 6-13 however, the error ranges from 8% to -18%. The reason for this error can be attributed to the fact that varying the average visits from one task to the other, we are in turn also varying the service (slice time) of the calling task at its host processor. This was not a problem in the previous case with PS at the processor because the time each process spends on the cpu is divided into equal slices in addition to the process slices, so the waiting times are not effected as much, resulting in significantly less error.

6.2.3 Example with FCFS Scheduling with Multiple Heterogeneous Classes

In this example the scheduling discipline at the processor is First-Come First-Serve, or FCFS, however the residence expressions have been adjusted to account for variability in the service times of the arriving classes. The only difference between this example and the previous example is the waiting time expression at the processor P1.

Figure 6-6 Simple Client-Server Case with Interlocking and HV-FCFS

Solution Time	0.50 sec	
Stage I (Reduction and Matrix Generation)	2.14 sec	Total: 7.28 sec
Stage II (Sparse Matrix Operations)	4.06 sec	
Stage III (Post Processing)	0.92 sec	
Stage IV (Ranking)	0.16 sec	
SENSITIVITY MATRIX SIZE	120 x 20	
Estimated Perturbation Approach Time	~20 sec	Ratio: ~2.8
Direct Approach Time	~7 sec	

Table 6-16 : Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Client Throughput (0.1104)	S	-0.0200	-0.0173	-15.6 %
	E	-0.1208	-0.1042	-15.9 %
Server Throughput (0.2208)	S	-0.0400	-0.0345	-15.9 %
	E	-0.1208	-0.1043	-15.8 %

Table 6-16 : Sensitivity/Elasticity wrt to Service Time of Client (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Server Utilization (0.5396)	<i>S</i>	-0.0457	-0.0729	37.3 %
	<i>E</i>	-0.0565	-0.0900	37.2 %
P1 Utilization (0.6626)	<i>S</i>	0.2112	0.2277	7.2 %
	<i>E</i>	0.2125	0.2291	7.2 %

Table 6-17 : Sensitivity/Elasticity wrt to Service Time of Server (ST=2.0)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1104)	<i>S</i>	-0.0208	-0.0310	32.9 %
	<i>E</i>	-0.3782	-0.5614	32.6 %
Server Throughput (0.2208)	<i>S</i>	-0.0418	-0.0620	32.6 %
	<i>E</i>	-0.3782	-0.5613	32.6 %
Server Utilization (0.5396)	<i>S</i>	0.1110	0.1340	17.2 %
	<i>E</i>	0.4076	0.4966	17.9 %
P1 Utilization (0.6626)	<i>S</i>	0.0956	0.0350	-173.1 %
	<i>E</i>	0.2885	0.1057	-172.9 %

Table 6-18 : Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.1104)	<i>S</i>	-0.0225	-0.0291	22.7 %
	<i>E</i>	-0.4084	-0.5271	22.5 %
Server Throughput (0.2208)	<i>S</i>	0.0653	0.0523	-24.8 %
	<i>E</i>	0.5916	0.4734	-25.0 %

Table 6-18 : Sensitivity/Elasticity wrt to Visits from Client to Server[E1] (V=2)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Server Utilization (0.5396)	<i>S</i>	0.1406	0.1374	-2.3 %
	<i>E</i>	0.5213	0.5094	-2.3 %
P1 Utilization (0.6626)	<i>S</i>	0.0856	0.0463	-84.9 %
	<i>E</i>	0.2583	0.1398	-84.8 %

Table 6-19 : Sensitivity/Elasticity wrt to Client Think Time (Z=8)

Output Parameter		Direct	Perturbation (7.9,8.1)	%diff
Client Throughput (0.1104)	<i>S</i>	-0.0046	-0.0046	0.0 %
	<i>E</i>	-0.3362	-0.3343	-0.6 %
Server Throughput (0.2208)	<i>S</i>	-0.0093	-0.0092	-1.0 %
	<i>E</i>	-0.3362	-0.3345	-0.5 %
Server Utilization (0.5396)	<i>S</i>	-0.0275	-0.0274	-0.4 %
	<i>E</i>	-0.4002	-0.4062	1.5 %
P1 Utilization (0.6626)	<i>S</i>	-0.0278	-0.0277	-0.4 %
	<i>E</i>	-0.3362	-0.3345	-0.5 %

Table 6-20 : Sensitivity/Elasticity wrt to Client Population (N=2)

Output Parameter		Direct	Perturbation (1,3)	%diff
Client Throughput (0.1104)	<i>S</i>	0.0427	0.0303	-40.9 %
	<i>E</i>	0.7735	0.5495	-40.8 %
Server Throughput (0.2208)	<i>S</i>	0.0854	0.0607	-40.7 %
	<i>E</i>	0.7735	0.5495	-40.8 %

Table 6-20 : Sensitivity/Elasticity wrt to Client Population (N=2)

Output Parameter		Direct	Perturbation (1,3)	%diff
Server Utilization (0.5396)	<i>S</i>	0.2002	0.2128	5.9 %
	<i>E</i>	0.7422	0.7890	5.9 %
P1 Utilization (0.6626)	<i>S</i>	0.1458	0.1820	19.8 %
	<i>E</i>	0.4402	0.5495	19.9 %

From the results presented in the tables below it is clear that the sensitivities obtained using the Direct Approach are not accurate. In Chapter 4, for the HV-FCFS server, an assumption about the variance is discussed. Using the perturbation approach the value of the variance was observed. The variance for the service time of the two servers (Server and P1) is given in the brackets beside each input parameter. It was discovered that the variance did not remain constant when the average number of visits (2.28 -> 2.30, 4 -> 4), Client service time (2.28 -> 2.30, 4 -> 4), Server service time (2.08 -> 2.524 , 3.61 -> 4.41) were varied around the operating point. Server service time has a particularly large effect. The variance did remain constant when the Client think time (2.29 -> 2.29, 4 -> 4) and Client population (2.29 -> 2.29, 4 -> 4) was varied. The sensitivity/elasticity with respect to Client think time are accurate and confirm the fact that the variance assumption is the problem in the other sensitivities.

6.3 Multiple Client Chains

The two examples below further explore the interlocking probability assumption. It is expected that by introducing another source at an interlocked resource the probability of interlocking assumption should prove to be inaccurate. This is demonstrated using the examples below.

6.3.1 Multiple Client Chains with no Interlocking

In this example, there are two clients, Client 1 and Client 2 each making requests to separate entries at a common software server, Server. Both clients and server reside on

separate processor therefore eliminating the possibility of interlocking. Service times, avg. visits and think times are given in Figure 6-7.

Figure 6-7 Multiple Client Chains with no Interlocking

Solution Time	0.70 sec	
Stage I (Reduction and Matrix Generation)	3.87 sec	Total: 10.4 sec
Stage II (Sparse Matrix Operations)	4.56 sec	
Stage III (Post Processing)	1.73 sec	
Stage IV (Ranking)	0.19 sec	
SENSITIVITY MATRIX SIZE	180 x 28	
Estimated Perturbation Approach Time	~40 sec	Ratio: ~4
Direct Approach Time	~10 sec	

Table 6-21 : Sensitivity/Elasticity wrt to Service Time of Client 1 (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Client 1 Throughput (0.1083)	<i>S</i>	-0.0186	-0.0185	-0.5 %
	<i>E</i>	-0.1145	-0.1138	-0.6 %
Client 2 Throughput (0.0849)	<i>S</i>	0.0031	0.0031	0.0 %
	<i>E</i>	0.0241	0.0245	1.6 %
SERVER Throughput (0.3016)	<i>S</i>	-0.0342	-0.0340	-0.6 %
	<i>E</i>	-0.0755	-0.0755	0.0 %
SERVER Utilization (0.6881)	<i>S</i>	-0.0652	-0.0649	-0.5 %
	<i>E</i>	-0.0632	-0.0632	0.0 %
P1 Utilization (0.2167)	<i>S</i>	0.2878	0.2864	-0.5 %
	<i>E</i>	0.8854	0.8855	0.0 %
P2 Utilization (0.1698)	<i>S</i>	0.0061	0.0061	0.0 %
	<i>E</i>	0.0241	0.0241	0.0 %
P3 Utilization (0.6881)	<i>S</i>	-0.0692	-0.0649	-6.6 %
	<i>E</i>	-0.0671	0.0632	-6.2 %

For this example, we can conclude the results obtained using the Direct approach are very accurate. In the next example, a similar example that introduces interlocking is investigated.

6.3.2 Multiple Client Chains with Interlocking

This example is similar to the example presented in the previous section. The only difference being Client 1 and Server 1 now reside on a common processor, P1. By doing so we have introduced the interlocking effect at the common processor. Service times, average visits, and think times have been maintained.

Figure 6-8 Multiple Client Chains with Interlocking

Solution Time	0.89 sec	
Stage I (Reduction and Matrix Generation)	6.19 sec	Total: 17.1 sec
Stage II (Sparse Matrix Operations)	6.76 sec	
Stage III (Post Processing)	3.86 sec	
Stage IV (Ranking)	0.26 sec	
SENSITIVITY MATRIX SIZE	243 x 37	
Estimated Perturbation Approach Time	~67 sec	Ratio: ~3.9
Direct Approach Time	~17 sec	

Table 6-22 : Sensitivity/Elasticity wrt to Service Time of Client 1 (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Client 1 Throughput (0.1003)	S	-0.0154	-0.0150	-2.7 %
	E	-0.1026	-0.0997	-2.9 %

Table 6-22 : Sensitivity/Elasticity wrt to Service Time of Client 1 (ST=2.0)

Output Parameter		Direct	Perturbation (1.9, 2.1)	%diff
Client 2 Throughput (0.0713)	<i>S</i>	0.0031	0.0031	0.0 %
	<i>E</i>	0.0286	0.0286	0.0 %
SERVER Throughput (0.2719)	<i>S</i>	-0.0278	-0.0277	-0.3 %
	<i>E</i>	-0.0682	-0.0679	-0.4 %
SERVER Utilization (0.7116)	<i>S</i>	-0.0605	-0.0600	-0.8 %
	<i>E</i>	-0.0567	-0.0562	-0.9 %
P1 Utilization (0.2006)	<i>S</i>	0.2700	0.2686	-0.5 %
	<i>E</i>	0.8974	0.8927	-0.5 %
P2 Utilization (0.7578)	<i>S</i>	-0.0497	-0.0460	-8.0 %
	<i>E</i>	-0.0437	-0.0404	-8.2 %

The results in Table 6-22 indicate that the Direct approach for obtaining sensitivities is also accurate for the case with multiple client chains and interlocking with processor sharing at the processor. The slight error can be attributed to the assumptions made in the interlocking probability calculations.

6.4 Distributed Database Server

In the previous examples the systems studied were simple, typically with one client and one server. In the next two examples larger systems many servers and clients will be studied.

The next example, referred to as the *Distributed Database Server* was studied in [neilson95],[sheikh95]. In this example, there is one group of clients that make requests to a lower level server. The lower level server, called the Dispatcher, does some processing and then forwards on the request to a number of its servers. In this case, as shown in Figure 6-9, the Dispatcher forwards on requests to three lower level servers, Server 1, Server 2, and Server 3. There are four layers in this model resulting in three MVA submodels. Model parameters are given in Figure 6-9.

If the Dispatcher process is limiting the throughput of the client by not being able to service client requests faster than they are generated then it is classified as a software bottleneck. In [neilson95] it was found that cloning (or multi-threading) the dispatcher can result in a significant increase in client throughput. This example has been designed as such that the Dispatcher is a software bottleneck. Later on this example will be used in Chapter 7 to determine whether sensitivity/elasticity information can be used to identify bottlenecks in client-server type systems. Note that this example does not contain any interlocking effects.

As expected, as the model size increased the time difference between the Direct approach and the Perturbation approach also increased. In this example the Direct approach is approximately 7.5 times faster than perturbation.

Solution Time	1.13 sec	
Stage I (Reduction and Matrix Generation)	6.39 sec	Total: 15.18 sec
Stage II (Sparse Matrix Operations)	3.97 sec	
Stage III (Post Processing)	4.38 sec	
Stage IV (Ranking)	0.44 sec	
SENSITIVITY MATRIX SIZE	313 x 51	
Estimated Perturbation Approach Time	~113 sec	Ratio: 7.5
Direct Approach Time	~15 sec	

Sensitivity/elasticity results for a number of model outputs with respect to the service time of the Dispatcher can be found in Table 6-23 below. Sensitivity/elasticity results of model outputs with respect to the average number of visits from Client to Dispatcher [E1] are given in Table 6-24 .

Figure 6-9 Distributed Database Server

Table 6-23 : Sensitivity/Elasticity wrt to Service Time of Dispatcher (ST=2)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.0332)	<i>S</i>	-0.0155	-0.0155	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %
Dispatcher Throughput (0.0664)	<i>S</i>	-0.0309	-0.0309	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %
Dispatcher Utilization (0.9969)	<i>S</i>	0.0015	0.0015	0.0 %
	<i>E</i>	0.0004	0.0004	0.0 %
Server 1 Utilization (0.3988)	<i>S</i>	-0.1855	-0.1855	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %
Server 2 Utilization (0.1994)	<i>S</i>	-0.0928	-0.0928	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %
Server 3 Utilization (0.2658)	<i>S</i>	-0.1237	-0.1237	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %
P2 Utilization (0.1329)	<i>S</i>	0.4034	0.4034	0.0 %
	<i>E</i>	0.8671	0.8671	0.0 %
P3 Utilization (0.3988)	<i>S</i>	-0.1855	-0.1855	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %
P4 Utilization (0.1994)	<i>S</i>	-0.0928	-0.0928	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %
P5 Utilization (0.2658)	<i>S</i>	-0.1237	-0.1237	0.0 %
	<i>E</i>	-0.1329	-0.1329	0.0 %

The percentage difference between the two approaches is 0%. The results in the table above prove that the Direct approach for obtaining sensitivities/elasticities for this case is very accurate. The same conclusion can be drawn from the results in the table below.

**Table 6-24 : Sensitivity/Elasticity wrt to Visits from Client to Dispatcher
[D_ENTRY] (V=2)**

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.0332)	<i>S</i>	-0.0166	-0.0166	0.0 %
	<i>E</i>	-0.9968	-0.9991	0.2 %
Dispatcher Throughput (0.0664)	<i>S</i>	0.0001	0.0001	0.0 %
	<i>E</i>	0.0032	0.0030	-6.7 %
Dispatcher Utilization (0.9969)	<i>S</i>	0.0016	0.0016	0.0 %
	<i>E</i>	0.0032	0.0032	0.0 %
Server 1 Utilization (0.3988)	<i>S</i>	0.0006	0.0006	0.0 %
	<i>E</i>	0.0032	0.0030	-6.7 %
Server 2 Utilization (0.1994)	<i>S</i>	0.0003	0.0003	0.0 %
	<i>E</i>	0.0032	0.0030	-6.7 %
Server 3 Utilization (0.2658)	<i>S</i>	0.0004	0.0004	0.0 %
	<i>E</i>	0.0032	0.0030	-6.7 %
P2 Utilization (0.1329)	<i>S</i>	0.0002	0.0002	0.0 %
	<i>E</i>	0.0032	0.0030	-6.7 %
P3 Utilization (0.3988)	<i>S</i>	0.0006	0.0006	0.0 %
	<i>E</i>	0.0032	0.0030	-6.7 %
P4 Utilization (0.1994)	<i>S</i>	0.0003	0.0003	0.0%
	<i>E</i>	0.0032	0.0030	-6.7 %
P5 Utilization (0.2658)	<i>S</i>	0.0004	0.0004	0.0%
	<i>E</i>	0.0032	0.0030	-6.7 %

6.5 Multi-level Case with two levels of Interlocking

The example presents a client-server system with multiple servers, a number of clients, with interlocking at both hardware and software resources. The layered model of

the system can be found in Figure 6-10. Three separate submodels are solved alternately to arrive at the solution.

The software interlocked resource is server, S4. The hardware interlocked resource is processor, P4. Server S1 is a multi-server with a multiplicity of 2. All other servers have a single copy.

Solution Time	1.99 sec	
Stage I (Reduction and Matrix Generation)	7.42 sec	TOTAL: 19.75 sec
Stage II (Sparse Matrix Operations)	7.12 sec	
Stage III (Post Processing)	4.99 sec	
Stage IV (Ranking)	0.42 sec	
SENSITIVITY MATRIX SIZE	280 x 43	
Estimated Perturbation Approach Time	~170 sec	Ratio: 8.5
Direct Approach Time	~20 sec	

The results in the above table demonstrate that the Direct approach is approximately 8.5 times faster than the Perturbation based approach.

Figure 6-10 Multi-level Case with two levels of Interlocking

In Table 6-25 sensitivity/elasticity results of model output parameters to the service

Table 6-25 : Sensitivity/Elasticity wrt to Service Time of S3 at P4 (ST=4)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput (0.01864)	<i>S</i>	-0.0019	-0.0018	-5.9 %
	<i>E</i>	-0.0667	-0.0644	-3.6 %
S1 Throughput (0.0559)	<i>S</i>	-0.0056	-0.0054	-3.7 %
	<i>E</i>	-0.0667	-0.0641	-4.1 %
S2 Throughput (0.0559)	<i>S</i>	-0.0056	-0.0054	-3.7 %
	<i>E</i>	-0.0667	-0.0641	-4.1 %
S3 Throughput (0.1118)	<i>S</i>	-0.0119	-0.0108	-10.2 %
	<i>E</i>	-0.0667	-0.0642	-3.9 %
S4 Throughput (0.2237)	<i>S</i>	-0.0224	-0.0215	-4.2 %
	<i>E</i>	-0.0667	-0.0640	-4.2 %
S1 Utilization (1.9680)	<i>S</i>	0.0340	0.0360	5.6 %
	<i>E</i>	0.0115	0.0122	5.7 %
S2 Utilization (0.5342)	<i>S</i>	-0.0766	-0.0819	6.5 %
	<i>E</i>	-0.0956	-0.1022	6.5 %
S3 Utilization (0.6639)	<i>S</i>	-0.0962	-0.0940	-2.3 %
	<i>E</i>	-0.0966	-0.0944	-2.3 %
S4 Utilization (0.6454)	<i>S</i>	-0.0861	-0.0901	4.4 %
	<i>E</i>	-0.0889	-0.0930	4.4 %
P2 Utilization (0.2237)	<i>S</i>	0.3031	0.3140	0.3 %
	<i>E</i>	0.9333	0.9360	0.3 %
P3 Utilization (0.2796)	<i>S</i>	-0.0280	-0.0269	-4.1 %
	<i>E</i>	-0.0667	-0.0640	-4.2 %
P4 Utilization (0.8947)	<i>S</i>	-0.0863	-0.0860	-0.3 %
	<i>E</i>	-0.0643	-0.0641	-0.3 %

time of server, S3, are presented. The percentage difference between the Direct and

Perturbation based approach is at most 10%. Even though there are two level of interlocking by changing the service time of S3, the probability of interlocking at the two servers does not change by much which is why the results are good.

6.6 Discussion

Using the technique described in Chapter 4 and the implementation discussed in Chapter 5, sensitivities/elasticities of ALL output parameters with respect to ALL input parameters were obtained.

In general the results for systems without interlocking are very accurate. Error for the cases presented and others studied is almost zero. However, the examples chosen with interlocking substantiate the fact that some of our assumptions in the derivatives of the interlocking equations (See Chapter 4) are weak. The sensitivities with respect to model inputs that don't affect the interlocking probabilities typical yield accurate results. However, cases where the probability of interlocking changes with a parameter change then the percentage difference between the two approaches is large, and in some cases it has been to almost 200% (See Table 6-17). On the other hand, in the example in Figure 6-8, the results for a system with multiple client chains with interlocking and with PS discipline the accuracy is excellent.

Interlocking effects are strongest for small client populations, where a larger fraction of the demand is interlocked. It was found that by increasing the number of clients in the Simple Client-server example with interlocking that error in the results decreased. It was also found that by introducing another client pool the error in the sensitivities also increased.

The other assumption about the variance term in the HV-FCFS waiting time equation being a constant is also poor. Although, the FCFS can be used for the models, HV-FCFS has been proven to yield more accurate results and it would have been preferable if sensitivity results using this was more accurate. However because of the implementation of the LQNS solver the incorporation of the partial derivatives of the variance calculation was not carried out.

The speed of the Direct approach was found to be at minimum twice as fast at the Perturbation based technique. In some cases, the Direct approach was found to be almost 8 times faster.

Chapter 7

Call Processing Application

To evaluate the approach in terms of practicality and feasibility a large industrial application was studied. A sensitivity analysis was performed on a prototype Call Processing System studied by Jogalekar in [ppj91].

7.1 Call Processing System

A layered queueing model of the Call Processing System was created based measurements from a prototype. As described in [ppj91] the system uses the basic two half-call model for call processing. This system is referred to as the two half-call model because it separates tasks for call origination and call termination. The basic functionality of the system software is as follows:

- Detect request for service
- Interpret dialed digits
- Alert called party
- Establish connection
- Disconnect call
- Charge parties

The system can support other features such as conference calls, call waiting, call display, which are all extensions of the basic call processing model. For the purposes of this thesis the details of this system are not important.

7.1.1 System Architecture

The system studied is implemented using message oriented middleware based on a fast ATM network. The systems used fast ATM technology to minimize latency, therefore the communication infrastructure is assumed to be bottleneck free. In Figure 7-1 a high-

level view of the system is presented. It features a number of telephones connected to computers that hook into the ATM backbone.

Figure 7-1 Call Processing System Hardware Infrastructure [ppj91]

The next section will describe the Layered Queueing model of this Call Processing system.

7.1.2 Layered Queueing Model of the Call Processing System

A Layered Queueing Model of the Call Processing System can be found in Figure 7-2. The tasks in the model are the tasks that are involved in the call processing at each of the sites in the system. As mentioned earlier, the system can be divided into two parts: call origination and call termination. The communications infrastructure, ATM network and the Integrated Device Driver (enables applications to directly send messages on the ATM network) have been modelled as delay servers.

All the processes in the system are assumed to have infinite number of threads. There are 182638 clients in the system each residing on a separate cpu. These clients make requests to the call origination servers shown in Figure 7-2. This does contain a fair

amount of interlocking at processors. For more detail about the structure and creation of the Layered Queueing Model refer to [ppj91].

The Call Processing System is broken down in nine separate submodels. The nine submodels are given in Figure 7-3.

The results of the sensitivity analysis will be presented in the same manner as in Chapter 6. The table below contains timing information for the solution of the system, and the times for each stage of the sensitivity analysis.

Solution Time	7.38 sec	
Stage I (Reduction and Matrix Generation)	169.93 sec	Total: 420.45 sec or 7.01 min
Stage II (Sparse Matrix Operations)	87.19 sec	
Stage III (Post Processing)	157.10 sec	
Stage IV (Ranking)	6.23 sec	
SENSITIVITY MATRIX SIZE	2371 x 302	
Estimated Perturbation Approach Time	4228 sec	Ratio: ~10
Direct Approach Time	420 sec	

The system contained over 300 input parameters. It took approximately 7 minutes to get the sensitivities of all output parameters to those 300 input parameters. A conservative estimate on the time it would take using the perturbation approach is over an hour. Therefore, the Direct approach can be said to be at least ten times faster than the perturbation approach. Remember for the perturbation approach, only solution time is considered. The overhead of taking the information from each solution and then using that to determine sensitivities has not been included in its estimates.

A few of the ranked elasticities can be found in Table 7-4 .

Figure 7-2 Layered Queueing Model of the Call Processing System

Figure 7-3 Call Processing System Submodels

Table 7-1 : Sensitivity/Elasticity wrt to Visits from Term Integrated Device Driver to Term Agent Factory (V=1)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Client Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
Orig_Agent_Factory Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
Orig_Term_Agent Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
Orig_User_Agent Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
Orig_Call_Setup Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
Database Manager Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
IM_Orig_Integrated_Dev_Driver Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
IM_ATM_Net Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
IM_Term_Int_Dev_Driver Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
Term_Agent_Factory Throughput	<i>S</i>	0.3044	0.3044	0.0 %
	<i>E</i>	1.0000	1.0000	0.0 %
Term_Term_Agent Throughput	<i>S</i>	0.3044	0.3044	0.0 %
	<i>E</i>	1.0000	1.0000	0.0 %
Term_User_Agent Throughput	<i>S</i>	0.3044	0.3044	0.0 %
	<i>E</i>	1.0000	1.0000	0.0 %

1.2355045063	+	9.8634649e-02	UTILIZATION_ [Term_ User_ Agent] <7>	VISITS_ [IM_ Term_ Int_ Dev_ Driver] -> [Term_ Agent_ Factory] [Term_ Agent_ Factory]
1.2355045063	+	9.8634649e-02	UTILIZATION_ [Term_ Call_ Setup] <7>	VISITS_ [IM_ Term_ Int_ Dev_ Driver] -> [Term_ Agent_ Factory] [Term_ Agent_ Factory]
1.2355042559	+	5.4005535e-07	UTILIZATION_ [Term_ User_ Agent] <7>	POPULATION_ [IM_ client] <1>
1.2355042559	+	5.4005535e-07	UTILIZATION_ [Term_ Call_ Setup] <7>	POPULATION_ [IM_ client] <1>
1.2355014374	+	9.8634404e-02	UTILIZATION_ [Term_ User_ Agent] <7>	VISITS_ [IM_ ATM_ Net] -> [IM_ Term_ Int_ Dev_ Driver] [Term_ Int_ Dev_ Driver] <5>
1.2355014374	+	9.8634404e-02	UTILIZATION_ [Term_ Call_ Setup] <7>	VISITS_ [IM_ ATM_ Net] -> [IM_ Term_ Int_ Dev_ Driver] [Term_ Int_ Dev_ Driver] <5>
1.2354986691	+	9.8634183e-02	UTILIZATION_ [Term_ User_ Agent] <7>	VISITS_ [IM_ Orig_ Int_ Dev_ Driver] -> [IM_ ATM_ Net] [ATM_ Net] <4>
1.2354986691	+	9.8634183e-02	UTILIZATION_ [Term_ Call_ Setup] <7>	VISITS_ [IM_ Orig_ Int_ Dev_ Driver] -> [IM_ ATM_ Net] [ATM_ Net] <4>
1.2354961138	+	9.8633979e-02	UTILIZATION_ [Term_ User_ Agent] <7>	VISITS_ [Orig_ Call_ Setup] -> [IM_ Orig_ Int_ Dev_ Driver] [Orig_ Int_ Dev_ Driver] <1>
1.2354961138	+	9.8633979e-02	UTILIZATION_ [Term_ Call_ Setup] <7>	VISITS_ [Orig_ Call_ Setup] -> [IM_ Orig_ Int_ Dev_ Driver] [Orig_ Int_ Dev_ Driver] <1>
1.2354944478	+	9.8633846e-02	UTILIZATION_ [Term_ User_ Agent] <7>	VISITS_ [IM_ client] -> [Orig_ Agent_ Factory] [Orig_ Agent_ Factory] <1>
1.2354944478	+	9.8633846e-02	UTILIZATION_ [Term_ Call_ Setup] <7>	VISITS_ [IM_ client] -> [Orig_ Agent_ Factory] [Orig_ Agent_ Factory] <1>
1.2354935960	+	9.8633778e-02	UTILIZATION_ [Term_ User_ Agent] <7>	VISITS_ [Orig_ Agent_ Factory] -> [Orig_ Call_ Setup] [Orig_ Call_ Setup] <2>
1.2354935960	+	9.8633778e-02	UTILIZATION_ [Term_ Call_ Setup] <7>	VISITS_ [Orig_ Agent_ Factory] -> [Orig_ Call_ Setup] [Orig_ Call_ Setup] <2>
1.2354926816	+	9.8633705e-02	UTILIZATION_ [Term_ User_ Agent] <7>	VISITS_ [IM_ client] -> [IM_ client] [client] <1>
1.2354926816	+	9.8633705e-02	UTILIZATION_ [Term_ Call_ Setup] <7>	VISITS_ [IM_ client] -> [IM_ client] [client] <1>
1.2318135376	+	5.0348097e-07	UTILIZATION_ [Orig_ Term_ Agent] <2>	POPULATION_ [IM_ client] <1>
1.2318109835	+	5.2997926e-07	UTILIZATION_ [Orig_ User_ Agent] <2>	POPULATION_ [IM_ client] <1>
1.2318037666	+	9.1954028e-02	UTILIZATION_ [Orig_ Term_ Agent] <2>	VISITS_ [IM_ client] -> [Orig_ Agent_ Factory] [Orig_ Agent_ Factory] <1>
1.2318020118	+	9.1953897e-02	UTILIZATION_ [Orig_ Term_ Agent] <2>	VISITS_ [IM_ client] -> [Orig_ Agent_ Factory] [Orig_ Agent_ Factory] <1>
1.2318012087	+	9.6793584e-02	UTILIZATION_ [Orig_ User_ Agent] <2>	VISITS_ [IM_ client] -> [Orig_ Agent_ Factory] [Orig_ Agent_ Factory] <1>
1.2317994525	+	9.6793446e-02	UTILIZATION_ [Orig_ User_ Agent] <2>	VISITS_ [IM_ client] -> [IM_ client] [client] <1>

Figure 7-4 Ranking of elasticities for the Call Processing System

Table 7-1 : Sensitivity/Elasticity wrt to Visits from Term Integrated Device Driver to Term Agent Factory (V=1)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Term_Call_Setup Throughput	<i>S</i>	0.3044	0.3044	0.0 %
	<i>E</i>	1.0000	1.0000	0.0 %
IM_Term_Client Through- put	<i>S</i>	0.3044	0.3044	0.0 %
	<i>E</i>	1.0000	1.0000	0.0 %
Location Server Through- put	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
IM_Trans_Ohead Throughput	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
CPU1 Utilization	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
CPU2 Utilization	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
CPU3 Utilization	<i>S</i>	0.1796	0.2374	24.4 %
	<i>E</i>	0.7564	1.0000	24.4 %
CPU4 Utilization	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
NA_NET_CPU	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
NA_TRANS_CPU	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %
NA_IDD_CPU	<i>S</i>	0.0000	0.0000	0.0 %
	<i>E</i>	0.0000	0.0000	0.0 %

Another parameter that appears to be critical in this system is the population of the Client. The next table contains the sensitivities/elasticities of four software tasks {Term_User_Agent, Term_Call_Setup, Orig_Term_Agent, Orig_User_Agent} with respect to the Client population.

Table 7-2 : Sensitivity/Elasticity of the utilization of some processes to the population of the Client (TM=182638)

Output Parameter		Direct	Perturbation (1.9,2.1)	%diff
Term User Agent Utilization	<i>S</i>	5.400e-07	5.750e-07	6.1 %
	<i>E</i>	1.2355	1.3154	6.1 %
Term Call Setup Utilization	<i>S</i>	5.400e-07	5.750e-07	6.1 %
	<i>E</i>	1.2355	1.3154	6.1 %
Orig Term Agent Utilization	<i>S</i>	5.035e-07	5.250e-07	4.1 %
	<i>E</i>	1.2318	1.2845	4.1 %
Orig User Agent Utilization	<i>S</i>	5.300e-07	5.560e-07	4.7 %
	<i>E</i>	1.2318	1.2929	4.7 %

The results presented in the above table, Table 7-2 are with error less than 10%. This is believed to be good considering the complexity and size of the system.

7.1.3 Discussion

A sensitivity analysis was performed on Call Processing System presented in [ppj91]. A file containing the ranking of parameter elasticities was obtained from the analysis.

Considering the size and complexity of the system, the results obtained are very good. Noticing that both Term User Agent and Term Call Setup are eight layers below the Client and that we are able to get very good sensitivity results really strengthens the concept and its implementation. The Call Processing model contained 182638 clients in the system which indicates that the approach used is scalable.

Using the above example a limitation of the implementation of the approach was discovered. The Call Processing system was analyzed using different configurations such as varying the number of cpu's. In some cases, where there was a large number of clients

or a large number of servers, or both, the submodel derivative files got quite large. The submodel .der files at times can approach over three megabytes. In these cases it might be beneficial to not perform the reduction stage because performing the reduction on a three megabyte file is computationally expensive. By eliminating the reduction the size of the A matrix will increase so the matrix solution stage will take longer. The total time will however be shorter than if the reduction was enabled. However the total time should be much shorter.

The use of the direct approach on this Call Processing Application proved very successful. The direct approach is estimated to be ten times faster than the traditional perturbation approach. It is also very interesting to note that as the models get larger the direct approach is more feasible. By obtaining sensitivities of all output parameters to all input parameters it is not necessary to screen parameters at an earlier stage. Rather, parameters can be screened using the elasticities.

Chapter 8

Using the Sensitivity Information

In this chapter the sensitivity/elasticity information is used to determine which parameters are affecting the performance of the system. The sensitivity/elasticity information is used to locate bottlenecks and the sensitivity/elasticity measures are used in optimizing the design. The goal of this chapter is not to focus on design optimization rather to demonstrate usefulness of the sensitivity information and to motivate future research on how this information can be used. The last two examples presented in Chapter 6 will be used.

8.1 Bottleneck Identification in the Distributed Database Server Case

The Distributed Server example used in the previous chapter is studied further in this chapter. This particular example was designed such that the Dispatcher process would be a software bottleneck. One objective was to determine whether the sensitivity/elasticity information could be used to detect this software bottleneck. The layered model with model parameters of the Distributed Database Server can be found in Figure 6-9.

In Figure 8-1 the parameter elasticities for this model are ranked and presented. From the .rank file it clear that the two input parameters that effect Client throughput the most are the average number of visits from the Client to the Dispatcher, and the population or multiplicity of the Dispatcher. The influence of both these input parameters on Client throughput are studied further in the sections to follow.

8.1.1 Effect of Dispatcher Population on Client Throughput

Analysis of the ranked elasticities highlighted that Client throughput was very sensitive to changes in the multiplicity of the Dispatcher. From [neilson95][sheikh95] it was found that cloning the Dispatcher will help relieve the bottleneck and increase Client

throughput. In this example the multiplicity of the Dispatcher was varied from 1 to 5 Dispatcher clones.

In Table 8-1 Client throughput along with the sensitivity/elasticity of Client throughput with respect to Dispatcher Multiplicity are given. As the number of Dispatcher clones is increased from 1 to 5, Client throughput increases from 0.0332 to 0.0720. A graph of Client throughput vs. Dispatcher Multiplicity is given in Figure 8-4.

Table 8-1 : Sensitivity/Elasticity of Client Throughput wrt Dispatcher Multiplicity

Dispatcher Multiplicity	Client Throughput	Sensitivity	Elasticity
1	0.0332	0.0297	0.8944
2	0.0526	0.0160	0.6079
3	0.0624	0.0138	0.6654
4	0.0682	0.0057	0.3320
5	0.0720	0.0040	0.2745

In addition, as a result of the cloning we are also reducing the sensitivity and elasticity of Client throughput to the number of Dispatcher clones as shown in Figure 8-2 and Figure 8-3. It is interesting to note that in Figure 8-3 the elasticity goes down from 0.8944 to 0.6079 when a Dispatcher clone is added, however when another Dispatcher clone is added the elasticity goes up to 0.6654 instead of going down as one might expect. At 2 and 3 Dispatcher clones, the Dispatcher is completely saturated. Therefore, Client throughput is still very sensitive to Dispatcher multiplicity. Recall that elasticity is the percentage change of a parameter to a percentage change in another parameter. In this case elasticity is:

$$E_{X_{CLIENT}, TM_{DISPATCHER}} = \frac{TM_{DISPATCHER}}{X_{CLIENT}} \frac{\partial X_{CLIENT}}{\partial TM_{DISPATCHER}}$$

The elasticity actually increased because the ratio of the multiplicity of the Dispatcher to the throughput of the Client was greater at 3 Dispatcher clones as compared

0.9985898320+	9.9552665e-02	UTILIZATION_[proc1]<1>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<1>
0.9968381837-	-1.6562915e-02	THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT]->[DISPATCHER][D_ENTRY]<1>
0.9968351690-	-6.6251659e-02	UTILIZATION_[proc1]<1>	VISITS_[CLIENT]->[DISPATCHER][D_ENTRY]<1>
0.8943977732+	1.7832950e-01	THROUGHPUT_[SERVER1]<3>	POPULATION_[DISPATCHER]<1>
0.8943977230+	1.7832949e-01	UTILIZATION_[proc4]<3>	POPULATION_[DISPATCHER]<1>
0.8943975280+	8.9164815e-01	UTILIZATION_[DISPATCHER]<1>	POPULATION_[DISPATCHER]<1>
0.8943966308+	5.9443210e-02	THROUGHPUT_[DISPATCHER]<1>	POPULATION_[DISPATCHER]<1>
0.8943966308+	2.9721605e-02	THROUGHPUT_[CLIENT]<1>	POPULATION_[DISPATCHER]<1>
0.8943965193+	1.7832925e-01	UTILIZATION_[SERVER2]<2>	POPULATION_[DISPATCHER]<1>
0.8943965193+	1.7832925e-01	THROUGHPUT_[SERVER1]<2>	POPULATION_[DISPATCHER]<1>
0.8943959237+	5.9443163e-02	THROUGHPUT_[SERVER2]<3>	POPULATION_[DISPATCHER]<1>
0.8943955052+	3.5665899e-01	UTILIZATION_[proc3]<3>	POPULATION_[DISPATCHER]<1>
0.8943953853+	2.3777233e-01	UTILIZATION_[SERVER3]<2>	POPULATION_[DISPATCHER]<1>
0.8943947200+	5.9443083e-02	THROUGHPUT_[SERVER2]<2>	POPULATION_[DISPATCHER]<1>
0.8943947200+	5.9443083e-02	THROUGHPUT_[DISPATCHER]<2>	POPULATION_[DISPATCHER]<1>
0.8943942764+	3.5665850e-01	UTILIZATION_[SERVER1]<2>	POPULATION_[DISPATCHER]<1>
0.8943939394+	1.1888642e-01	UTILIZATION_[proc1]<1>	POPULATION_[DISPATCHER]<1>
0.8943932623+	1.1888633e-01	THROUGHPUT_[SERVER3]<3>	POPULATION_[DISPATCHER]<1>
0.8943920586+	1.1888617e-01	UTILIZATION_[proc2]<2>	POPULATION_[DISPATCHER]<1>
0.8943920586+	1.1888617e-01	THROUGHPUT_[SERVER3]<2>	POPULATION_[DISPATCHER]<1>
0.8670830305+	4.0339691e-01	UTILIZATION_[proc2]<2>	SERVICE_TIME_[DISPATCHER]@[proc2][D_ENTRY]<2>

Figure 8-1 Ranking of Elasticities for the Distributed Server Case

to 2 Dispatcher clones. The sensitivity term remained constant because the utilization of the Dispatcher was still 100% and was still limiting the performance of the system.

Figure 8-2 Sensitivity of Client throughput wrt to Dispatcher population for increasing number of Dispatcher clones

Figure 8-3 Elasticity of Client throughput wrt to Dispatcher population for increasing number of Dispatcher clones

Figure 8-4 Client throughput for increasing number of Dispatcher clones

8.1.2 Effect of Average Visits from Client to Dispatcher on Client Throughput

From the rank file of the elasticities it was found that output parameters such as the utilization of processor, proc1, and Client throughput were very sensitive to the average number of visits from the Client to the Dispatcher. This is consistent with a heavily congested system when much of client time is spent at the dispatcher.

As the average number of visits from Client to Dispatcher are varied from 0.5 to 2.5, Client throughput decreases from 0.1316 to 0.0266. Client throughput becomes less sensitive to average visits as it is increased, however the elasticity remains the same. Results can be found in Table 8-2 and Figure 8-5, Figure 8-6 and Figure 8-7 below.

Table 8-2 : Sensitivity/Elasticity of Client Throughput wrt Avg Visits from Client to Dispatcher [E1]

Avg Visits	Client Throughput	Sensitivity	Elasticity
0.5	0.1316	-0.2592	-0.9852
1.0	0.0662	-0.0658	-0.9933

Table 8-2 : Sensitivity/Elasticity of Client Throughput wrt Avg Visits from Client to Dispatcher [E1]

Avg Visits	Client Throughput	Sensitivity	Elasticity
1.5	0.0443	-0.0294	-0.9957
2.0	0.0332	-0.0166	-0.9968
2.5	0.0266	-0.0106	-0.9975

Figure 8-5 Client throughput as the avg number of visits from Client to Dispatcher are varied

Figure 8-6 Sensitivity of Client throughput to avg. visits from Client to Dispatcher

Figure 8-7 Elasticity of Client throughput to avg. visits from Client to Dispatcher

8.2 The Multi-level Case Revisited

Similar analysis presented in the previous section was carried out on the Multi-level case. The Multi-level case is described in the Chapter 6, and a Layered Queueing Model can be found in Figure 6-10.

The ranking of the elasticities, given in Figure 8-8 indicate the parameters that Client throughput is most sensitive parameters to are the average visits from Client to server S2, average visits from Client to server S1, and the population of the server, S1. Other parameters that need the attention of architects and/or developers are the Client service time, S2 service time, and the visits from S1 to S3, and some others. Parameters such as server population can sometimes be easily modified, however other parameters might require re-design. The sensitivity information aims at highlighting the so-called “hot spots” in distributed systems. It also gives direction information which can be used in deciding how to deal with these hot-spots

Figure 8-9 Sensitivity of Client throughput to the average visits from Client to server S2.

As the number of visits is increased from 1 to 5, the Client throughput is quite sensitive to the visits. As the visits are increased, the elasticity increases which implies that for a percentage increase in the number of visits one see's a greater decrease in Client throughput. Figure 8-11 confirms that fact that as the number of visits is increased, Client throughput drops. Graphs for the sensitivity and elasticity of server S2 throughput to the average visits from Client to S2 are also given.

0.9987618001	+	1.3031857e-01	UTILIZATION_[proc1]<1>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<1>
0.9733597103	+	1.0886055e-01	UTILIZATION_[proc3]<2>	SERVICE_TIME_[SERVER2]@[proc3][S2_E1]<2>
0.925430189	+	9.1471216e-01	UTILIZATION_[SERVER1]<1>	POPULATION_[SERVER1]<1>
0.8795366416	+	1.6394563e-02	THROUGHPUT_[SERVER2]<1>	VISITS_[CLIENT]->[SERVER2][S2_E1]<1>
0.8795352146	+	8.1972682e-02	UTILIZATION_[proc3]<2>	VISITS_[CLIENT]->[SERVER2][S2_E1]<1>
0.8795351931	+	1.6394536e-02	THROUGHPUT_[SERVER2]<2>	VISITS_[CLIENT]->[SERVER2][S2_E1]<1>
0.8724783798	-	-1.0841998e-02	UTILIZATION_[proc1]<1>	VISITS_[CLIENT]->[SERVER1][S1_E1]<1>
0.8724783476	-	-5.4209988e-03	THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT]->[SERVER1][S1_E1]<1>
0.8724783262	-	-1.6262996e-02	THROUGHPUT_[SERVER2]<1>	VISITS_[CLIENT]->[SERVER1][S1_E1]<1>
0.8724756652	-	-8.1314732e-02	UTILIZATION_[proc3]<2>	VISITS_[CLIENT]->[SERVER1][S1_E1]<1>
0.8693393374	+	1.5479775e-01	UTILIZATION_[SERVER2]<1>	VISITS_[CLIENT]->[SERVER2][S2_E1]<1>
0.8592184069	-	-1.5299558e-01	UTILIZATION_[SERVER2]<1>	VISITS_[CLIENT]->[SERVER1][S1_E1]<1>
0.6773286579	+	2.2483620e-01	UTILIZATION_[SERVER3]<2>	POPULATION_[SERVER1]<1>
0.6708831452	+	1.7913645e-01	UTILIZATION_[SERVER2]<1>	POPULATION_[SERVER1]<1>
0.6237384464	+	2.0126761e-01	UTILIZATION_[SERVER4]<2>	POPULATION_[SERVER1]<1>
0.5425905579	+	3.0341664e-02	THROUGHPUT_[SERVER3]<2>	VISITS_[SERVER1]->[SERVER3][S3_E1]<2>
0.5425897711	+	3.0341620e-02	THROUGHPUT_[SERVER3]<3>	VISITS_[SERVER1]->[SERVER3][S3_E1]<2>
0.5265992950	-	-7.0326151e-02	UTILIZATION_[SERVER2]<1>	SERVICE_TIME_[SERVER3]@[proc4][S3_E1]<3>
0.5255992875	-	-2.8130460e-01	UTILIZATION_[SERVER2]<1>	VISITS_[SERVER1]->[proc4][S3_E1]<3>
0.5255989131	-	-1.4065220e-01	UTILIZATION_[SERVER2]<1>	VISITS_[SERVER1]->[SERVER3][S3_E1]<2>
0.5010738376	-	-5.6040098e-02	THROUGHPUT_[SERVER4]<3>	SERVICE_TIME_[SERVER4]@[proc4][S4_E1]<3>
0.4786665773	-	-2.6767035e-02	THROUGHPUT_[SERVER4]<3>	SERVICE_TIME_[SERVER3]@[proc4][S3_E1]<3>
0.4786665594	-	-5.3534068e-02	THROUGHPUT_[SERVER4]<3>	VISITS_[SERVER1]->[SERVER3][S3_E1]<2>
0.4678852647	+	2.6164144e-02	THROUGHPUT_[SERVER3]<3>	POPULATION_[SERVER1]<1>
0.4678844528	+	4.3606831e-03	THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER1]<1>
0.4678844421	+	1.3082049e-02	THROUGHPUT_[SERVER2]<1>	POPULATION_[SERVER1]<1>
0.4678844421	+	1.3082049e-02	THROUGHPUT_[SERVER1]<1>	POPULATION_[SERVER1]<1>
0.4678830830	+	1.3082011e-02	THROUGHPUT_[SERVER2]<2>	POPULATION_[SERVER1]<1>

Figure 8-8 Ranking of Elasticities for the Multi-level Case

Figure 8-10 Elasticity of Client throughput to the average number of visits from Client to server S2

Figure 8-11 Client throughput as the average visits from Client to server S2 are increased

Figure 8-12 Sensitivity of S2 throughput to the average number of visits from Client to S2

Figure 8-13 Elasticity of S2 to the average visits from Client to S2

Figure 8-14 S2 throughput as the average number of visits from Client to S2 are increased

8.2.1 Improving Client Throughput via Server Cloning

In this section the sensitivity/elasticity information was used to determine how one should decide which software server to clone for an increase in Client throughput. Sensitivity and elasticity information for this particular case are given in Table 8-3 below.

Table 8-3 : Sensitivity/elasticities of software servers

Server	Current Multiplicity	Sensitivity	Elasticity
S1	2	0.0044	0.4679
S2	1	0.0004	0.0218
S3	1	0.0022	0.1162
S4	1	0.0018	0.0988

Looking at both sensitivity and elasticities in the Table 8-3 suggests that cloning server S1 will improve Client throughput the most. The throughput of the Client for the above configuration is 0.186. The multiplicity of each of the software servers in the model were was increased by one. The effect on overall Client throughput is given in Table 8-4

Initial Client Throughput	0.0186
---------------------------	--------

Table 8-4 : Effect on Client throughput as software server multiplicity is increased

SERVER	MULTIPLICITY	Client Throughput	Improvement Factor
S1	3	0.0193	1.04
S2	2	0.0182	0.98
S3	2	0.0191	1.03
S4	2	0.0187	1.00

Even though cloning the servers did not result in much improvement (decrease in one case), increasing the multiplicity of S1 did improve Client throughput the most as indicated by the sensitivity/elasticity measures. The decrease in the throughput as a result of cloning S2 is interesting. Considering possible errors due to the interlocking assumptions in the technique the drop in throughput was possible. In addition the sensitivity of Client throughput to S2 multiplicity is also very small (0.0004).

In this section both sensitivity and elasticity information recommended that server S1 be cloned. What if the sensitivity and elasticity information suggest different servers to be cloned? What is the better measure to use? In the next section this is investigated.

0.8724783476	-	-5.4209988e-03	THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT]>[SERVER1][S1_E1]<1>
0.4678844528	+	4.3606831e-03	THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER1]<1>
0.4574116524	-	2.1315383e-03	THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[SERVER3]@[proc4][S3_E1]<3>
0.4574116470	-	8.5261531e-03	THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER3]·>[proc4][S3_E1]<3>
0.4574115343	-	4.2630755e-03	THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER1]·>[SERVER3][S3_E1]<2>
0.3500391845	-	3.2623652e-03	THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[SERVER4]@[proc4][S4_E1]<3>
0.3483777361	-	2.1645870e-03	THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER1]·>[SERVER4][S4_E1]<2>
0.3151099839	-	5.8736501e-03	THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER3]<3>
0.1204633696	-	7.4847907e-04	THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT]·>[SERVER2][S2_E1]<1>
0.1161867972	+	2.1657219e-03	THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER3]<2>
0.0987588251	+	1.8408645e-03	THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER4]<2>
0.0938229077	-	1.7488590e-03	THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER2]·>[SERVER4][S4_E1]<2>
0.0667207451	-	1.8655111e-03	THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[SERVER1]@[proc2][S1_E1]<2>
0.0649764700	+	2.4223228e-04	THROUGHPUT_[CLIENT]<1>	POPULATION_[CLIENT]<1>
0.0266405633	-	1.9863204e-04	THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[SERVER2]@[proc3][S2_E1]<2>
0.0217588498	+	4.0558496e-04	THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER2]<1>
0.0058225987	-	1.0853324e-05	THROUGHPUT_[CLIENT]<1>	THINK_TIME_[CLIENT]<1>
0.0012372146	-	8.0715959e-05	THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<1>

Figure 8-15 Ranking of elasticities for the Multi-level Case II

8.2.2 Using Sensitivity Information to Determine Cloning Strategy

The Multi-level Case was further studied where the sensitivity and elasticity information was used to determine which software server to clone. The goal was to optimize Client throughput. Some changes were made to the existing Multi-level Case for the following study and is referred to as the Multi-level Case II. The overall architecture of the system remained the same. The task multiplicity of the Client was changed from 5 to 10. Service times and the visits ratios between processes were also changed. A layered model of the system is presented in Figure 8-16.

Figure 8-16 Multi-level Case II

A sequence of eight changes were made such that at each stage a particular software server was cloned. In the first case, the sensitivity information was used as the basis to

determine which software server to clone. In the second case was where the elasticity values were used to determine which server to clone. The goal of this study was to determine which approach resulted in greater Client throughput after eight arbitrarily chosen stages. The reason why this study was carried out was because it was not always clear how to decide which server to clone when looking at both sensitivities and elasticities measures. One would hypothesize that the elasticity based approach is a better guide since it normalizes the sensitivities to their parameter values. To check this each criteria was separately to guide decision making over multiple steps. The objective of this study was not to arrive at some conclusions about which was better, rather to get a feel for the two measures in the context of our problem environment.

8.2.3 Sensitivity based Cloning Strategy

In this section, different software servers in the layered model shown in Figure 8-16 were cloned with the goal of improving client throughput. Sensitivity information was used to determine which software server to clone. For example, if Client throughput was most sensitive to the multiplicity of server S3, then the multiplicity of S3 was increased by one. The layered model was then re-solved with the new clone set. The clone set, $(M_1, M_2, \dots, M_n)$, defines the multiplicity of all the servers in the model where n is the number of servers in the model. Therefore, if every server in this example had a multiplicity of two then the clone set of the system would read as (2,2,2,2).

As mentioned earlier, eight stages of cloning were carried out. The clones sets at each of the stages is given in Figure 8-19.

In Figure 8-21 we see the sensitivity of the Client throughput with respect to the various software servers. As the number of cloning steps increase, the sensitivity of the servers can either go up or down. In this particular case server S2 had the highest sensitivity to start off with. Once its multiplicity was increased, the sensitivity of server S2 decreased and the sensitivity of S1 increased significantly. Server S1 was then selected as the next candidate to clone. In general, Client throughput becomes less sensitive as the number of server clones are increased.

Figure 8-17 Sensitivity based Cloning: Sensitivity of Client throughput to the multiplicity of the servers in the model

Figure 8-18 Sensitivity based Cloning: Elasticity of Client throughput to the multiplicity of the servers in the model

Figure 8-19 Sensitivity based Cloning Decision Stages

In Figure 8-18 the elasticity of Client throughput with respect to the various software servers is given. Changes in Client throughput at each of the eight stages can be found in Figure 8-23. The sensitivity based cloning technique does result in an increase

in Client throughput, however we actually see a decrease in Client throughput at the sixth stage. Apparently, by cloning S4 in the fifth stage, resulted in an increase in waiting time at the processor P4, which pushed back up to the server S1. Another interesting thing to note in the above graphs is that by cloning one server, another server became more sensitive. This indicates the movement of a potential bottleneck from one resource to another. Can sensitivity information be used to track the movement of bottlenecks?

8.2.3.1 Elasticity based Cloning Strategy

In this section, elasticity of Client throughput with respect to server multiplicity is the measure used to determine which server to clone. Figure 8-20 presents the clone sets at the eight stages.

In Figure 8-21 the sensitivity of Client throughput with respect server population at each of the stages is plotted. Note however that elasticity was the measure that was used to determine which server to clone. In the next figure, Figure 8-22, the elasticity of Client throughput for each of the servers at each cloning stage is presented. In general, as the number of clones in the system are increased, Client throughput becomes less sensitive to the population of the servers.

Figure 8-20 Elasticity based Cloning Decision Stages

Figure 8-21 Elasticity based Cloning: Sensitivity of Client throughput to the multiplicity of the servers in the model

Figure 8-22 Elasticity based Cloning: Elasticity of Client throughput to the multiplicity of the servers in the model

In Figure 8-23, a graph of Client throughput for the two approaches at the different cloning stages is presented. The elasticity based cloning results in a smooth and steady increase in Client throughput however it ends up lower after eight stages.

Figure 8-23 Client throughput

8.2.4 Results

In Figure 8-23 the change Client throughput for the two cloning strategies can be compared. After two stages it is clear that the sensitivity based cloning technique yields an overall greater Client throughput than the elasticity based cloning technique. At the sixth stage however, the throughput of the Client is almost the same for the two techniques. Another important note is that Client throughput appears to be levelling off after eight stages when the elasticity based technique is used. On the other hand we still see an upward trend in Client throughput after eight stages for the sensitivity based technique.

8.3 Discussion

In this chapter, sensitivity in combination with elasticities were used to identify and relieve bottlenecks in the distributed systems studied. The client throughput of the system was also improved by cloning server tasks. Based on the analysis presented, it is clear that the sensitivity/elasticity can be used in the optimizing of distributed system designs or systems.

Two approaches for determining which servers to clone was also briefly investigated. From the above analysis it appears that sensitivity based decision making results in greater client throughput. It appears that the elasticity based approach is biased to adding to larger

pools which is why we end up with more S1 clones and still only one S3 clone. Overall, these results are not conclusive and require further investigation.

Chapter 9

Conclusions

A Direct approach for obtaining parameter sensitivities in layered queueing models was designed, implemented and evaluated. This approach determines sensitivities of all model outputs with respect to all model inputs using one model solution. It was found that the direct approach was at least twice as fast and at times ten times faster than the traditional perturbation based approach. In addition, parameter elasticities were determined from the sensitivities. The final output of the analysis ranks the elasticities. Therefore, the approach proved to be both exhaustive and efficient.

Results were very accurate, essentially zero error, for all systems with no interlocking and also zero for some systems with interlocking.

However, cases exist where accuracy results for systems with interlocking were in fact quite disappointing. The error in these systems can be attributed to an assumption made in the partial derivatives of the interlocking equations. The probability of interlocking and the number of sources was assumed to be a constant. For systems with multiple sources the interlocking probability does not remain constant as certain parameters are changed. This results in error in the sensitivities.

Sensitivity calculations were implemented for waiting time equations for servers with First Come First Serve (FCFS), Processor Sharing (PS) and High-Variance First Come First Serve (HV-FCFS). Results for FCFS and PS were very accurate. However, results for HV-FCFS were not accurate. The reason for the error in the sensitivities where HV-FCFS was used was due to an assumption of variance for multiple classes in the waiting time calculation. The assumption was used because of the difficulty in obtaining partial derivatives of the variance equation.

The technique was evaluated using a large industrial application, a Call Processing System Prototype. Sensitivity and elasticities results were very good for this system

despite interlocking. It was also discovered that the approach implemented was determined to be ten times faster than the traditional perturbation approach. It appears that the accuracy is better for larger systems and the time advantage also increases.

A few examples were used to show how the sensitivity information could be used. Sensitivities and elasticities were used to locate bottlenecks in client-server systems. A model was created with an inherent software bottleneck. This model was then run through the sensitivity analysis technique to determine whether this bottleneck could be identified. This proved quite successful because the sensitivity results did highlight the bottleneck. In addition, sensitivities and elasticities were used to make decisions to improve the throughput of another client-server system. Two approaches were used to decide which software servers to clone. The first approach was to use sensitivities to decide which server to clone. The second approach was to use elasticities to decide which servers to clone. In the investigated system, it was found that by using the sensitivities it was possible to achieve a higher client throughput after the same number of stages.

Overall, the approach can help decide which parameters need to be considered in more detail for model building. This provides valuable information to designer and developers on how the performance of the systems can be improved leading to more robust designs.

Future Work

Future work includes eliminating the assumptions about the probability of interlocking and the variance calculations made in this study. All in all, these assumptions proved to be weak and should be included to complete the analysis.

The current implementation can also be extended to incorporate Linearizer, a more accurate Approximate Mean Value Analysis solution technique. It is also possible to use the simple Bard-Schweitzer equations with the results obtained from a Linearizer solution.

Further optimization of technique can also be performed. Currently the technique is quite efficient however there do exist cases where the direct approach does not yield a

clear advantage. Typically these are cases where the size of the submodels are quite large. Options exist where reduction on the large derivative files can be avoided, however these can move the bottleneck from the reduction to writing the data to file.

More studies on how the sensitivity information can be used is required. Using the sensitivity information to optimize distributed system designs is one avenue that needs to be explored. Although using the sensitivity information to detect bottlenecks in systems was touched upon in this thesis, it requires further work.

In distributed and other complex systems combinations of two or more parameters can play a significant role. This is referred to as a two way interactions. Being able to determine sensitivities to combinations of parameters needs to be investigated.

References

- [aky88] Akyildiz I. F., and W. Liu, *Throughput Maximization and Response Time Minimization in a Multiprocessor System*, Technical Report GIT-ICS-88-18, Georgia Institute of Technology, May 15, 1988.
- [andres95] Andres, T. H., *Sampling Methods and Sensitivity Analysis for Large Parameter Sets*, SAMO Web site, <http://www.ei.jrc.it/samo>.
- [banks95] Steve Bankes and Tom Lucas, *Statistical Approaches for the Exploratory Modelling of Large Complex Models*, SAMO Web site, <http://www.ei.jrc.it/samo>.
- [blake88] James Blake, Andrew L. Reibman, and Kishor Trivedi, *Sensitivity analysis of reliability and performability measures for multiprocessor systems*, Performance Evaluation Review, Vol. 16, No. 1, 1988.
- [cao96] Xi-Ren Cao and Dye-Jyun Ma, *Performance Sensitivity formulae, algorithms and estimates for closed queueing networks with exponential servers*, Performance Evaluation, Vol. 26, 1996, pp. 181-199.
- [chandy82] K. Mani Chandy, and Doug Neuse, *Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems*, Communications of the ACM, February 1982, Vol. 25, No. 2, pp. 126-134.
- [franks97] Greg Franks, *Performance Analysis of Distributed Server Systems, PhD Proposal*, Department of Systems and Computer Engineering, Carleton University, February 1997.
- [franks95] Greg Franks, *Traffic Dependencies in Client-Server Systems and Their Effect on Performance Prediction*, IEEE International Computer Performance and Dependability Symposium, Erlangen, Germany, pp. 24-33, April 1995.
- [gordon80] Gordon, Karen D., and Lawrence Dowdy, *The Impact of Certain Parameter Estimation Errors in Queueing Network Models*, ACM Sigmetrics, 1980.

- [hofer95] Hofer, E., and B. Kryzkacz, *On Benefits and Drawbacks of Customary Sensitivity Measures*, SAMO Web site, <http://www.ei.jrc.it/samo>
- [huber95] Klaus-Peter Huber, *Locating Sensitive Regions by Learning from Simulation Data*, SAMO Web Site, <http://www.ei.jrc.it/samo>.
- [jain91] Raj Jain, *The Art of Computer Systems Performance Analysis*, John Wiley and Sons, 1991.
- [ppj91] Prasad Jogalekar, *Scalability Analysis Framework for Distributed Systems*, PhD Thesis, Department of Systems and Computer Engineering, Carleton University, 1997 (in progress).
- [kleijnen95] Jack P. C. Kleijnen, *Sensitivity Analysis and Related Analysis: a survey of Statistical Techniques*, SAMO Web Site, <http://www.ei.jrc.it/samo>.
- [koba83] H. Kobayashi and M. Gerla, *Optimal routing in closed queueing networks*, ACM Transactions on Computer Systems 1, pp. 294-310, 1980.
- [liu91] Zhen Liu and Phillipe Nain. *Sensitivity results in open, closed, and mixed product form queueing networks*, Performance Evaluation, 13:237-251, 1991.
- [neilson95] J. Neilson, C. M. Woodside, D. C. Petriu, S. Majumdar, *Software Bottlenecking in Client-Server Systems and Rendez-vous Networks*, IEEE Transactions on Software Engineering, Vol. 21, No. 9, pp. 776-782, September 1995.
- [opdahl96] Andreas Opdahl, *Sensitivity Analysis of Combined Software and Hardware Performance Models: Open Queueing Networks*, Performance Evaluation, 22(1), pp. 75-92, North-Holland, February 1995.
- [lazow84] Lazowska E., Zahorjan J., Graham G., and Sevcik K., *Quantitative System Performance*, Prentice-Hall, 1984.
- [reilly95] Terence Reilly, *The effect of correlated variables on a one-way sensitivity analysis*, SAMO Web Site, <http://www.ei.jrc.it/samo>.

- [rolia95] J. A. Rolia and K. C. Sevcik, *The Method of Layers*, IEEE Transactions on Software Engineering, Vol. 21, No. 8, pp. 689-700, August 1995.
- [rolia92] J. A. Rolia, *Predicting the Performance of Software Systems*, PhD Thesis, University of Toronto, January 1992.
- [sheikh97a] Fahim Sheikh and Murray Woodside, *Layered Analytic Performance Modelling of Distributed Database System*, ICDCS '97, Baltimore, Maryland, pp. 482-490, May 27-30, 1997.
- [sheikh97b] F. Sheikh, J. Rolia, P. Garg, S. Frolund, and A. Shepherd, *Performance Evaluation of a Large Scale Distributed Application Design*, WCSS '97, Singapore, September 1-4, 1997.
- [sheikh98a] Fahim Sheikh, Murray Woodside, *Sensitivity Analysis of Performance Predictions for Distributed Applications*, Second International Symposium on Sensitivity of Model Output, SAMO '98, Ca' Dolfin, Venice, Italy, April 19-22, 1998.
- [sheikh95] Fahim Sheikh, *Software Bottlenecking in Client-Server Systems*, 4th Year Project, Department of Systems and Computer Engineering, Carleton University, May 1995.
- [triv80] K. S. Trivedi, R. A. Wagner and T. M. Sigmon, *Optimal selection of CPU speed, device capacities, and file assignments*, J. ACM 27 (2) pp. 323-337, 1980.
- [wall96] L. Wall, T. Christiansen, and R. Schwartz, *Programming Perl*, O'Reilly & Associates Inc, 1996.
- [williams76] A. C. Williams and R. A. Bhandiwad, *A generating function approach to queueing network analysis of multiprogrammed computers*, Networks 6, pp. 1-22, 1976.
- [woodside89] C. M. Woodside, *Throughput calculations of basic stochastic rendezvous networks*, Performance Evaluation, v. 9, pp. 143-160, 1989.

- [woodside95] C. M. Woodside, J. E. Neilson, D. C. Petriu and S. Majumdar, *The Stochastic Rendez-vous Network Model for Performance of Synchronous Client-Server-like Distributed Software*, IEEE Transactions on Computer, Vol. 44, No. 1, pp. 20-34, January 1995.

Appendix A

All the examples files included in this appendix correspond to the Simple Client-server case with Interlocking with FCFS scheduling. The following are the example files included in this appendix.

- Original DERIVATIVE .der file
- Reduced DERIVATIVE .der file
- Original INTERACTION .int file
- Translated INTERACTION File .itx file
- IDENTIFICATION (.id) file
- Generated MATLAB (.m) file
- SENSITIVITY (.sens) file
- RANKED ELASTICITY (.rank) file

Original DERIVATIVE .der file

SUBMODEL 1

L_ej_1111	L[1][1][1][2]	0.5
L_ej_1111	N[1]	0.106223
L[1][1][1][0]	L_ej_1111	1.0
U[1][1][1][0]	L_ej_1111	1
U[1][1][1][0]	L[1][1][1][2]	-0.5
U[1][1][1][0]	U[1][1][1][2]	0.5
L_ej_2111	L[2][1][1][2]	0.5
L_ej_2111	N[1]	0.163574
L[2][1][1][0]	L_ej_2111	1.0
U[2][1][1][0]	L_ej_2111	0.753495
U[2][1][1][0]	L[2][1][1][2]	-0.376748
U[2][1][1][0]	U[2][1][1][2]	0.5
W[1][1][0]_1	S(1,1)_1	1.0
sumOf_SL_m_21	S(1,1)_2	0.327148
sumOf_SL_m_21	L[2][1][1][0]	2.14163
util(2,1)	U[2][1][1][0]	1.0
W[1][1][0]_2	S(1,1)_2	1.0
W[1][1][0]_2	sumOf_SL_m_21	1.0
W[1][1][0]_2	M[2]	-0.70063
STEP:sum_1	Z[1]	1.0
R(1,1)_1	V(1,1)_1	3.69147
R(1,1)_1	W[1][1][0]_1	1
STEP:sum_1	R(1,1)_1	1.0
R(1,1)_2	V(1,1)_2	2.84226
R(1,1)_2	W[1][1][0]_2	2
STEP:sum_1	R(1,1)_2	1.0
X[1][2]	STEP:sum_1	-0.00662416
X[1][2]	N[1]	0.0575507
R(1,1)_1	V(1,1)_1	3.69147
R(1,1)_1	W[1][1][0]_1	1
INTERLOCK_ADJ11_1	lambda11_1	1.0
L[1][1][1][2]	INTERLOCK_ADJ11_1	3.69147
L[1][1][1][2]	R(1,1)_1	0.115101
lambda11_1	X[1][2]	1.0
U[1][1][1][2]	X[1][2]	3.69147
U[1][1][1][2]	V(1,1)_1	0.424893
U[1][1][1][2]	S(1,1)_1	0.115101
R(1,1)_2	V(1,1)_2	2.84226
R(1,1)_2	W[1][1][0]_2	2
INTERLOCK_ADJ11_2	lambda11_2	1.0
L[2][1][1][2]	INTERLOCK_ADJ11_2	5.68452
L[2][1][1][2]	R(1,1)_2	0.115101
lambda11_2	X[1][2]	1.0
U[2][1][1][2]	X[1][2]	4.28326
U[2][1][1][2]	V(1,1)_2	0.246505
U[2][1][1][2]	S(1,1)_2	0.230203
D[1]	U[1][1][1][2]	1.0
D[2]	U[2][1][1][2]	1.0

A[1]	B[1][1]	1.0
C[1][1][1]	V(1,1)_1	0.115101
C[1][1][1]	X[1][2]	1
B[1][1]	C[1][1][1]	1.0
A[2]	B[2][1]	1.0
C[2][1][1]	V(1,1)_2	0.115101
C[2][1][1]	X[1][2]	2
B[2][1]	C[2][1][1]	1.0

SUBMODEL 2

L_ej_3111	L[3][1][1][5]	0.5
L_ej_3111	N[1]	0.163574
L[3][1][1][3]	L_ej_3111	1.0
U[3][1][1][3]	L_ej_3111	1
U[3][1][1][3]	L[3][1][1][5]	-0.5
U[3][1][1][3]	U[3][1][1][5]	0.5
L_ej_3112	L[3][1][1][5]	1
L_ej_3112	N[1]	0.163574
L[3][1][1][0]	L_ej_3112	1.0
U[3][1][1][0]	L_ej_3112	1
U[3][1][1][0]	L[3][1][1][5]	-1
U[3][1][1][0]	U[3][1][1][5]	1
L_ej_3121	L[3][1][2][5]	1
L_ej_3121	N[2]	0
L[3][1][2][3]	L_ej_3121	1.0
L_ej_3122	L[3][1][2][5]	0
L_ej_3122	N[2]	0
L[3][1][2][0]	L_ej_3122	1.0
L_ej_4111	L[4][1][1][5]	0.5
L_ej_4111	N[1]	0
L[4][1][1][3]	L_ej_4111	1.0
L_ej_4112	L[4][1][1][5]	1
L_ej_4112	N[1]	0
L[4][1][1][0]	L_ej_4112	1.0
L_ej_4121	L[4][1][2][5]	1
L_ej_4121	N[2]	0
L[4][1][2][3]	L_ej_4121	1.0
L_ej_4122	L[4][1][2][5]	0
L_ej_4122	N[2]	0
L[4][1][2][0]	L_ej_4122	1.0
L_ej_5111	L[5][1][1][5]	0.5
L_ej_5111	N[1]	0.0531117
L[5][1][1][3]	L_ej_5111	1.0
U[5][1][1][3]	L_ej_5111	1.08358
U[5][1][1][3]	L[5][1][1][5]	-0.541789
U[5][1][1][3]	U[5][1][1][5]	0.5
L_ej_5112	L[5][1][1][5]	1
L_ej_5112	N[1]	0.0531117
L[5][1][1][0]	L_ej_5112	1.0
U[5][1][1][0]	L_ej_5112	1.08358
U[5][1][1][0]	L[5][1][1][5]	-1.08358
U[5][1][1][0]	U[5][1][1][5]	1
L_ej_5121	L[5][1][2][5]	1

L_ej_5121	N[2]	0
L[5][1][2][3]	L_ej_5121	1.0
L_ej_5122	L[5][1][2][5]	0
L_ej_5122	N[2]	0
L[5][1][2][0]	L_ej_5122	1.0
L_ej_5211	L[5][2][1][5]	0.5
L_ej_5211	N[1]	0
L[5][2][1][3]	L_ej_5211	1.0
L_ej_5212	L[5][2][1][5]	1
L_ej_5212	N[1]	0
L[5][2][1][0]	L_ej_5212	1.0
L_ej_5221	L[5][2][2][5]	1
L_ej_5221	N[2]	0.246505
L[5][2][2][3]	L_ej_5221	1.0
U[5][2][2][3]	L_ej_5221	1.86774
U[5][2][2][3]	L[5][2][2][5]	-1.86774
U[5][2][2][3]	U[5][2][2][5]	1
L_ej_5222	L[5][2][2][5]	0
L_ej_5222	N[2]	0.246505
L[5][2][2][0]	L_ej_5222	1.0
U[5][2][2][0]	L_ej_5222	1.86774
U[5][2][2][0]	L[5][2][2][5]	-0
U[5][2][2][0]	U[5][2][2][5]	0
W[1][1][0]_3	S(1,1)_3	1.0
W[1][2][0]_4	S(1,2)_4	1.0
sumOf_SL_m_51	S(1,1)_5	0.106223
sumOf_SL_m_51	L[5][1][1][3]	0.666667
sumOf_SL_m_51	S(1,2)_5	0
sumOf_SL_m_51	L[5][1][2][3]	0
sumOf_SL_m_51	S(2,1)_5	0
sumOf_SL_m_51	L[5][2][1][3]	0
sumOf_SL_m_51	S(2,2)_5	0.246505
sumOf_SL_m_51	L[5][2][2][3]	2
util(5,1)	U[5][1][1][3]	1.0
util(5,1)	U[5][1][2][3]	1.0
util(5,1)	U[5][2][1][3]	1.0
util(5,1)	U[5][2][2][3]	1.0
W[1][1][0]_5	S(1,1)_5	1.0
W[1][1][0]_5	sumOf_SL_m_51	1.0
W[1][1][0]_5	M[5]	-0.563825
sumOf_SL_m_52	S(1,1)_5	0.212447
sumOf_SL_m_52	L[5][1][1][0]	0.666667
sumOf_SL_m_52	S(1,2)_5	0
sumOf_SL_m_52	L[5][1][2][0]	0
sumOf_SL_m_52	S(2,1)_5	0
sumOf_SL_m_52	L[5][2][1][0]	0
sumOf_SL_m_52	S(2,2)_5	0
sumOf_SL_m_52	L[5][2][2][0]	2
util(5,1)	U[5][1][1][0]	1.0
util(5,1)	U[5][1][2][0]	1.0
util(5,1)	U[5][2][1][0]	1.0
util(5,1)	U[5][2][2][0]	1.0
W[2][2][0]_5	S(2,2)_5	1.0
W[2][2][0]_5	sumOf_SL_m_52	1.0

W[2][2][0]_5	M[5]	-0.141631
STEP:sum_1	Z[1]	1.0
R(1,1)_3	V(1,1)_3	5.68452
R(1,1)_3	W[1][1][0]_3	1
STEP:sum_1	R(1,1)_3	1.0
R(1,1)_4	V(1,1)_4	0
R(1,1)_4	W[1][1][0]_4	0
STEP:sum_1	R(1,1)_4	1.0
R(1,1)_5	V(1,1)_5	1.23049
R(1,1)_5	W[1][1][0]_5	3
STEP:sum_1	R(1,1)_5	1.0
R(2,1)_5	V(2,1)_5	0
R(2,1)_5	W[2][1][0]_5	0
STEP:sum_1	R(2,1)_5	1.0
X[1][5]	STEP:sum_1	-0.00662416
X[1][5]	N[1]	0.0575507
R(1,1)_3	V(1,1)_3	5.68452
R(1,1)_3	W[1][1][0]_3	1
INTERLOCK_ADJ11_3	lambda11_3	1.0
L[3][1][1][5]	INTERLOCK_ADJ11_3	5.68452
L[3][1][1][5]	R(1,1)_3	0.115101
lambda11_3	X[1][5]	1.0
U[3][1][1][5]	X[1][5]	5.68452
U[3][1][1][5]	V(1,1)_3	0.654296
U[3][1][1][5]	S(1,1)_3	0.115101
R(1,1)_4	V(1,1)_4	0
R(1,1)_4	W[1][1][0]_4	0
INTERLOCK_ADJ11_4	lambda11_4	1.0
L[4][1][1][5]	INTERLOCK_ADJ11_4	0
L[4][1][1][5]	R(1,1)_4	0.115101
lambda11_4	X[1][5]	1.0
U[4][1][1][5]	X[1][5]	0
U[4][1][1][5]	V(1,1)_4	0
U[4][1][1][5]	S(1,1)_4	0
R(1,1)_5	V(1,1)_5	1.23049
R(1,1)_5	W[1][1][0]_5	3
INTERLOCK_ADJ11_5	lambda11_5	0.5
L[5][1][1][5]	INTERLOCK_ADJ11_5	3.69147
L[5][1][1][5]	R(1,1)_5	0.0575507
lambda11_5	X[1][5]	1.0
U[5][1][1][5]	X[1][5]	2
U[5][1][1][5]	V(1,1)_5	0.0767342
U[5][1][1][5]	S(1,1)_5	0.345304
R(2,1)_5	V(2,1)_5	0
R(2,1)_5	W[2][1][0]_5	0
INTERLOCK_ADJ21_5	lambda21_5	0.5
L[5][2][1][5]	INTERLOCK_ADJ21_5	0
L[5][2][1][5]	R(2,1)_5	0.0575507
lambda21_5	X[1][5]	1.0
U[5][2][1][5]	X[1][5]	0
U[5][2][1][5]	V(2,1)_5	0
U[5][2][1][5]	S(2,1)_5	0
STEP:sum_2	Z[2]	1.0
R(1,2)_3	V(1,2)_3	0

R(1,2)_3	W[1][2][0]_3	0
STEP:sum_2	R(1,2)_3	1.0
R(1,2)_4	V(1,2)_4	0
R(1,2)_4	W[1][2][0]_4	1
STEP:sum_2	R(1,2)_4	1.0
R(1,2)_5	V(1,2)_5	0
R(1,2)_5	W[1][2][0]_5	0
STEP:sum_2	R(1,2)_5	1.0
R(2,2)_5	V(2,2)_5	2.14163
R(2,2)_5	W[2][2][0]_5	1
STEP:sum_2	R(2,2)_5	1.0
X[2][5]	STEP:sum_2	-0.0529932
X[2][5]	N[2]	0.230203
R(1,2)_3	V(1,2)_3	0
R(1,2)_3	W[1][2][0]_3	0
INTERLOCK_ADJ12_3	lambda12_3	1.0
L[3][1][2][5]	INTERLOCK_ADJ12_3	0
L[3][1][2][5]	R(1,2)_3	0.230203
lambda12_3	X[2][5]	1.0
U[3][1][2][5]	X[2][5]	0
U[3][1][2][5]	V(1,2)_3	0
U[3][1][2][5]	S(1,2)_3	0
R(1,2)_4	V(1,2)_4	0
R(1,2)_4	W[1][2][0]_4	1
INTERLOCK_ADJ12_4	lambda12_4	1.0
L[4][1][2][5]	INTERLOCK_ADJ12_4	0
L[4][1][2][5]	R(1,2)_4	0.230203
lambda12_4	X[2][5]	1.0
U[4][1][2][5]	X[2][5]	0
U[4][1][2][5]	V(1,2)_4	0
U[4][1][2][5]	S(1,2)_4	0.230203
R(1,2)_5	V(1,2)_5	0
R(1,2)_5	W[1][2][0]_5	0
INTERLOCK_ADJ12_5	lambda12_5	0.5
L[5][1][2][5]	INTERLOCK_ADJ12_5	0
L[5][1][2][5]	R(1,2)_5	0.115101
lambda12_5	X[2][5]	1.0
U[5][1][2][5]	X[2][5]	0
U[5][1][2][5]	V(1,2)_5	0
U[5][1][2][5]	S(1,2)_5	0
R(2,2)_5	V(2,2)_5	2.14163
R(2,2)_5	W[2][2][0]_5	1
INTERLOCK_ADJ22_5	lambda22_5	0.5
L[5][2][2][5]	INTERLOCK_ADJ22_5	2.14163
L[5][2][2][5]	R(2,2)_5	0.115101
lambda22_5	X[2][5]	1.0
U[5][2][2][5]	X[2][5]	2
U[5][2][2][5]	V(2,2)_5	0.460405
U[5][2][2][5]	S(2,2)_5	0.230203
D[3]	U[3][1][1][5]	1.0
D[3]	U[3][1][2][5]	1.0
D[4]	U[4][1][1][5]	1.0
D[4]	U[4][1][2][5]	1.0
D[5]	U[5][1][1][5]	1.0

D[5]	U[5][1][2][5]	1.0
D[5]	U[5][2][1][5]	1.0
D[5]	U[5][2][2][5]	1.0
A[3]	B[3][1]	1.0
C[3][1][1]	V(1,1)_3	0.115101
C[3][1][1]	X[1][5]	1
C[3][1][2]	V(1,2)_3	0.230203
C[3][1][2]	X[2][5]	0
B[3][1]	C[3][1][1]	1.0
B[3][1]	C[3][1][2]	1.0
A[4]	B[4][1]	1.0
C[4][1][1]	V(1,1)_4	0.115101
C[4][1][1]	X[1][5]	0
C[4][1][2]	V(1,2)_4	0.230203
C[4][1][2]	X[2][5]	1
B[4][1]	C[4][1][1]	1.0
B[4][1]	C[4][1][2]	1.0
A[5]	B[5][1]	1.0
A[5]	B[5][2]	1.0
C[5][1][1]	V(1,1)_5	0.115101
C[5][1][1]	X[1][5]	3
C[5][1][2]	V(1,2)_5	0.230203
C[5][1][2]	X[2][5]	0
B[5][1]	C[5][1][1]	1.0
B[5][1]	C[5][1][2]	1.0
C[5][2][1]	V(2,1)_5	0.115101
C[5][2][1]	X[1][5]	0
C[5][2][2]	V(2,2)_5	0.230203
C[5][2][2]	X[2][5]	1
B[5][2]	C[5][2][1]	1.0
B[5][2]	C[5][2][2]	1.0

Reduced DERIVATIVE .der file

SUBMODEL 1

A[1]<1>	B[1][1]<1>	1.0
A[2]<1>	B[2][1]<1>	1.0
B[1][1]<1>	C[1][1][1]<1>	1.0
B[2][1]<1>	C[2][1][1]<1>	1.0
C[1][1][1]<1>	V(1,1)_1<1>	0.115101
C[1][1][1]<1>	X[1][2]<1>	1
C[2][1][1]<1>	V(1,1)_2<1>	0.115101
C[2][1][1]<1>	X[1][2]<1>	2
D[1]<1>	U[1][1][1][2]<1>	1.0
D[2]<1>	U[2][1][1][2]<1>	1.0
L[1][1][1][0]<1>	L[1][1][1][2]<1>	0.5
L[1][1][1][0]<1>	N[1]<1>	0.106223
L[1][1][1][2]<1>	R(1,1)_1<1>	0.115101
L[1][1][1][2]<1>	X[1][2]<1>	3.69147
L[2][1][1][0]<1>	L[2][1][1][2]<1>	0.5
L[2][1][1][0]<1>	N[1]<1>	0.163574
L[2][1][1][2]<1>	R(1,1)_2<1>	0.115101
L[2][1][1][2]<1>	X[1][2]<1>	5.68452
R(1,1)_1<1>	V(1,1)_1<1>	3.69147
R(1,1)_1<1>	W[1][1][0]_1<1>	1
R(1,1)_2<1>	V(1,1)_2<1>	2.84226
R(1,1)_2<1>	W[1][1][0]_2<1>	2
U[1][1][1][0]<1>	L[1][1][1][2]<1>	0
U[1][1][1][0]<1>	N[1]<1>	0.106223
U[1][1][1][0]<1>	U[1][1][1][2]<1>	0.5
U[1][1][1][2]<1>	S(1,1)_1<1>	0.115101
U[1][1][1][2]<1>	V(1,1)_1<1>	0.424893
U[1][1][1][2]<1>	X[1][2]<1>	3.69147
U[2][1][1][0]<1>	L[2][1][1][2]<1>	-5.00000000014378e-07
U[2][1][1][0]<1>	N[1]<1>	0.12325219113
U[2][1][1][0]<1>	U[2][1][1][2]<1>	0.5
U[2][1][1][2]<1>	S(1,1)_2<1>	0.230203
U[2][1][1][2]<1>	V(1,1)_2<1>	0.246505
U[2][1][1][2]<1>	X[1][2]<1>	4.28326
W[1][1][0]_1<1>	S(1,1)_1<1>	1.0
W[1][1][0]_2<1>	L[2][1][1][0]<1>	2.14163
W[1][1][0]_2<1>	M[2]<1>	-0.70063
W[1][1][0]_2<1>	S(1,1)_2<1>	1.327148
X[1][2]<1>	N[1]<1>	0.0575507
X[1][2]<1>	R(1,1)_1<1>	-0.00662416
X[1][2]<1>	R(1,1)_2<1>	-0.00662416
X[1][2]<1>	Z[1]<1>	-0.00662416
Q[1]<1>	Z[1]<1>	1.0

SUBMODEL 2

A[3]<2>	B[3][1]<2>	1.0
A[4]<2>	B[4][1]<2>	1.0
A[5]<2>	B[5][1]<2>	1.0

A[5] <2>	B[5] [2] <2>	1.0
B[3] [1] <2>	C[3] [1] [1] <2>	1.0
B[3] [1] <2>	C[3] [1] [2] <2>	1.0
B[4] [1] <2>	C[4] [1] [1] <2>	1.0
B[4] [1] <2>	C[4] [1] [2] <2>	1.0
B[5] [1] <2>	C[5] [1] [1] <2>	1.0
B[5] [1] <2>	C[5] [1] [2] <2>	1.0
B[5] [2] <2>	C[5] [2] [1] <2>	1.0
B[5] [2] <2>	C[5] [2] [2] <2>	1.0
C[3] [1] [1] <2>	V(1,1)_3<2>	0.115101
C[3] [1] [1] <2>	X[1] [5] <2>	1
C[3] [1] [2] <2>	V(1,2)_3<2>	0.230203
C[3] [1] [2] <2>	X[2] [5] <2>	0
C[4] [1] [1] <2>	V(1,1)_4<2>	0.115101
C[4] [1] [1] <2>	X[1] [5] <2>	0
C[4] [1] [2] <2>	V(1,2)_4<2>	0.230203
C[4] [1] [2] <2>	X[2] [5] <2>	1
C[5] [1] [1] <2>	V(1,1)_5<2>	0.115101
C[5] [1] [1] <2>	X[1] [5] <2>	3
C[5] [1] [2] <2>	V(1,2)_5<2>	0.230203
C[5] [1] [2] <2>	X[2] [5] <2>	0
C[5] [2] [1] <2>	V(2,1)_5<2>	0.115101
C[5] [2] [1] <2>	X[1] [5] <2>	0
C[5] [2] [2] <2>	V(2,2)_5<2>	0.230203
C[5] [2] [2] <2>	X[2] [5] <2>	1
D[3] <2>	U[3] [1] [1] [5] <2>	1.0
D[3] <2>	U[3] [1] [2] [5] <2>	1.0
D[4] <2>	U[4] [1] [1] [5] <2>	1.0
D[4] <2>	U[4] [1] [2] [5] <2>	1.0
D[5] <2>	U[5] [1] [1] [5] <2>	1.0
D[5] <2>	U[5] [1] [2] [5] <2>	1.0
D[5] <2>	U[5] [2] [1] [5] <2>	1.0
D[5] <2>	U[5] [2] [2] [5] <2>	1.0
L[3] [1] [1] [0] <2>	L[3] [1] [1] [5] <2>	1
L[3] [1] [1] [0] <2>	N[1] <2>	0.163574
L[3] [1] [1] [3] <2>	L[3] [1] [1] [5] <2>	0.5
L[3] [1] [1] [3] <2>	N[1] <2>	0.163574
L[3] [1] [1] [5] <2>	R(1,1)_3<2>	0.115101
L[3] [1] [1] [5] <2>	X[1] [5] <2>	5.68452
L[3] [1] [2] [0] <2>	L[3] [1] [2] [5] <2>	0
L[3] [1] [2] [0] <2>	N[2] <2>	0
L[3] [1] [2] [3] <2>	L[3] [1] [2] [5] <2>	1
L[3] [1] [2] [3] <2>	N[2] <2>	0
L[3] [1] [2] [5] <2>	R(1,2)_3<2>	0.230203
L[3] [1] [2] [5] <2>	X[2] [5] <2>	0
L[4] [1] [1] [0] <2>	L[4] [1] [1] [5] <2>	1
L[4] [1] [1] [0] <2>	N[1] <2>	0
L[4] [1] [1] [3] <2>	L[4] [1] [1] [5] <2>	0.5
L[4] [1] [1] [3] <2>	N[1] <2>	0
L[4] [1] [1] [5] <2>	R(1,1)_4<2>	0.115101
L[4] [1] [1] [5] <2>	X[1] [5] <2>	0
L[4] [1] [2] [0] <2>	L[4] [1] [2] [5] <2>	0
L[4] [1] [2] [0] <2>	N[2] <2>	0
L[4] [1] [2] [3] <2>	L[4] [1] [2] [5] <2>	1

L[4][1][2][3]<2>	N[2]<2>	0
L[4][1][2][5]<2>	R(1,2)_4<2>	0.230203
L[4][1][2][5]<2>	X[2][5]<2>	0
L[5][1][1][0]<2>	L[5][1][1][5]<2>	1
L[5][1][1][0]<2>	N[1]<2>	0.0531117
L[5][1][1][3]<2>	L[5][1][1][5]<2>	0.5
L[5][1][1][3]<2>	N[1]<2>	0.0531117
L[5][1][1][5]<2>	R(1,1)_5<2>	0.0575507
L[5][1][1][5]<2>	X[1][5]<2>	1.845735
L[5][1][2][0]<2>	L[5][1][2][5]<2>	0
L[5][1][2][0]<2>	N[2]<2>	0
L[5][1][2][3]<2>	L[5][1][2][5]<2>	1
L[5][1][2][3]<2>	N[2]<2>	0
L[5][1][2][5]<2>	R(1,2)_5<2>	0.115101
L[5][1][2][5]<2>	X[2][5]<2>	0
L[5][2][1][0]<2>	L[5][2][1][5]<2>	1
L[5][2][1][0]<2>	N[1]<2>	0
L[5][2][1][3]<2>	L[5][2][1][5]<2>	0.5
L[5][2][1][3]<2>	N[1]<2>	0
L[5][2][1][5]<2>	R(2,1)_5<2>	0.0575507
L[5][2][1][5]<2>	X[1][5]<2>	0
L[5][2][2][0]<2>	L[5][2][2][5]<2>	0
L[5][2][2][0]<2>	N[2]<2>	0.246505
L[5][2][2][3]<2>	L[5][2][2][5]<2>	1
L[5][2][2][3]<2>	N[2]<2>	0.246505
L[5][2][2][5]<2>	R(2,2)_5<2>	0.115101
L[5][2][2][5]<2>	X[2][5]<2>	1.070815
R(1,1)_3<2>	V(1,1)_3<2>	5.68452
R(1,1)_3<2>	W[1][1][0]_3<2>	1
R(1,1)_4<2>	V(1,1)_4<2>	0
R(1,1)_4<2>	W[1][1][0]_4<2>	0
R(1,1)_5<2>	V(1,1)_5<2>	1.23049
R(1,1)_5<2>	W[1][1][0]_5<2>	3
R(1,2)_3<2>	V(1,2)_3<2>	0
R(1,2)_3<2>	W[1][2][0]_3<2>	0
R(1,2)_4<2>	V(1,2)_4<2>	0
R(1,2)_4<2>	W[1][2][0]_4<2>	1
R(1,2)_5<2>	V(1,2)_5<2>	0
R(1,2)_5<2>	W[1][2][0]_5<2>	0
R(2,1)_5<2>	V(2,1)_5<2>	0
R(2,1)_5<2>	W[2][1][0]_5<2>	0
R(2,2)_5<2>	V(2,2)_5<2>	2.14163
R(2,2)_5<2>	W[2][2][0]_5<2>	1
U[3][1][1][0]<2>	L[3][1][1][5]<2>	0
U[3][1][1][0]<2>	N[1]<2>	0.163574
U[3][1][1][0]<2>	U[3][1][1][5]<2>	1
U[3][1][1][3]<2>	L[3][1][1][5]<2>	0
U[3][1][1][3]<2>	N[1]<2>	0.163574
U[3][1][1][3]<2>	U[3][1][1][5]<2>	0.5
U[3][1][1][5]<2>	S(1,1)_3<2>	0.115101
U[3][1][1][5]<2>	V(1,1)_3<2>	0.654296
U[3][1][1][5]<2>	X[1][5]<2>	5.68452
U[3][1][2][5]<2>	S(1,2)_3<2>	0
U[3][1][2][5]<2>	V(1,2)_3<2>	0

U[3][1][2][5]<2>	X[2][5]<2>	0
U[4][1][1][5]<2>	S(1,1)_4<2>	0
U[4][1][1][5]<2>	V(1,1)_4<2>	0
U[4][1][1][5]<2>	X[1][5]<2>	0
U[4][1][2][5]<2>	S(1,2)_4<2>	0.230203
U[4][1][2][5]<2>	V(1,2)_4<2>	0
U[4][1][2][5]<2>	X[2][5]<2>	0
U[5][1][1][0]<2>	L[5][1][1][5]<2>	0
U[5][1][1][0]<2>	N[1]<2>	0.057550775886
U[5][1][1][0]<2>	U[5][1][1][5]<2>	1
U[5][1][1][3]<2>	L[5][1][1][5]<2>	1.00000000002876e-06
U[5][1][1][3]<2>	N[1]<2>	0.057550775886
U[5][1][1][3]<2>	U[5][1][1][5]<2>	0.5
U[5][1][1][5]<2>	S(1,1)_5<2>	0.345304
U[5][1][1][5]<2>	V(1,1)_5<2>	0.0767342
U[5][1][1][5]<2>	X[1][5]<2>	2
U[5][1][2][5]<2>	S(1,2)_5<2>	0
U[5][1][2][5]<2>	V(1,2)_5<2>	0
U[5][1][2][5]<2>	X[2][5]<2>	0
U[5][2][1][5]<2>	S(2,1)_5<2>	0
U[5][2][1][5]<2>	V(2,1)_5<2>	0
U[5][2][1][5]<2>	X[1][5]<2>	0
U[5][2][2][0]<2>	L[5][2][2][5]<2>	0
U[5][2][2][0]<2>	N[2]<2>	0.4604072487
U[5][2][2][0]<2>	U[5][2][2][5]<2>	0
U[5][2][2][3]<2>	L[5][2][2][5]<2>	0
U[5][2][2][3]<2>	N[2]<2>	0.4604072487
U[5][2][2][3]<2>	U[5][2][2][5]<2>	1
U[5][2][2][5]<2>	S(2,2)_5<2>	0.230203
U[5][2][2][5]<2>	V(2,2)_5<2>	0.460405
U[5][2][2][5]<2>	X[2][5]<2>	2
W[1][1][0]_3<2>	S(1,1)_3<2>	1.0
W[1][1][0]_5<2>	L[5][1][1][3]<2>	0.666667
W[1][1][0]_5<2>	L[5][1][2][3]<2>	0
W[1][1][0]_5<2>	L[5][2][1][3]<2>	0
W[1][1][0]_5<2>	L[5][2][2][3]<2>	2
W[1][1][0]_5<2>	M[5]<2>	-0.563825
W[1][1][0]_5<2>	S(1,1)_5<2>	1.106223
W[1][1][0]_5<2>	S(1,2)_5<2>	0
W[1][1][0]_5<2>	S(2,1)_5<2>	0
W[1][1][0]_5<2>	S(2,2)_5<2>	0.246505
W[1][2][0]_4<2>	S(1,2)_4<2>	1.0
W[2][2][0]_5<2>	L[5][1][1][0]<2>	0.666667
W[2][2][0]_5<2>	L[5][1][2][0]<2>	0
W[2][2][0]_5<2>	L[5][2][1][0]<2>	0
W[2][2][0]_5<2>	L[5][2][2][0]<2>	2
W[2][2][0]_5<2>	M[5]<2>	-0.141631
W[2][2][0]_5<2>	S(1,1)_5<2>	0.212447
W[2][2][0]_5<2>	S(1,2)_5<2>	0
W[2][2][0]_5<2>	S(2,1)_5<2>	0
W[2][2][0]_5<2>	S(2,2)_5<2>	1
X[1][5]<2>	N[1]<2>	0.0575507
X[1][5]<2>	R(1,1)_3<2>	-0.00662416
X[1][5]<2>	R(1,1)_4<2>	-0.00662416

X[1][5]<2>	R(1,1)_5<2>	-0.00662416
X[1][5]<2>	R(2,1)_5<2>	-0.00662416
X[1][5]<2>	Z[1]<2>	-0.00662416
X[2][5]<2>	N[2]<2>	0.230203
X[2][5]<2>	R(1,2)_3<2>	-0.0529932
X[2][5]<2>	R(1,2)_4<2>	-0.0529932
X[2][5]<2>	R(1,2)_5<2>	-0.0529932
X[2][5]<2>	R(2,2)_5<2>	-0.0529932
X[2][5]<2>	Z[2]<2>	-0.0529932
Z[2]<2>	N[2]<2>	2.20237

Original INTERACTION .int file

SUBMODEL 1

SERVICE_TIME_A	CLIENT	CLIENT	1	[1][1][1]_1	
SERVICE_TIME_B	CLIENT	CLIENT	1	2	0
SERVICE_TIME_A	CLIENT	CLIENT	2	[1][1][2]_1	
SERVICE_TIME_B	CLIENT	CLIENT	2	2	3.69147
THINK_TIME	1	CLIENT	IDLE_TIME0		
ELAPSED_TIME_GET	SERVER	SERVER_E1	1	[1][1][1]_2	
ELAPSED_TIME	SERVER	SERVER_E1	1	1	0
ELAPSED_TIME	SERVER	SERVER_E1	1	2	2.14163
UTILIZATION_CALCULATION_BEGIN					
ELAPSED_TIME	CLIENT	CLIENT	1	1	0
ELAPSED_TIME	CLIENT	CLIENT	1	2	0
U	CLIENT	CLIENT	1	THROUGHPUT	0
U	CLIENT	CLIENT	1	ELAPSED_TIME	0.115101
UTILIZATION_CALCULATION_END					
UTILIZATION_CALCULATION_BEGIN					
ELAPSED_TIME	CLIENT	CLIENT	2	1	5.68452
ELAPSED_TIME	CLIENT	CLIENT	2	2	3.69147
U	CLIENT	CLIENT	2	THROUGHPUT	9.376
U	CLIENT	CLIENT	2	ELAPSED_TIME	0.115101
UTILIZATION_CALCULATION_END					
UTILIZATION_CALCULATION_BEGIN					
ELAPSED_TIME	SERVER	SERVER_E11	1		0
ELAPSED_TIME	SERVER	SERVER_E11	2		2.14163
U	SERVER	SERVER_E11		THROUGHPUT	2.14163
U	SERVER	SERVER_E11		ELAPSED_TIME	0.230203
UTILIZATION_CALCULATION_END					
UTILIZATION	SERVER	SERVER_E1	1.0		
RHO	SERVER	UTILIZATION	1		
RHO	SERVER	POPULATION	-0.493009		
IDLE_TIME	SERVER	RHO	-4.344		
IDLE_TIME	SERVER	THROUGHPUT	-9.56709		
IDLE_TIME	SERVER	POPULATION	2.20237		

SUBMODEL 2

SERVICE_TIME_A	CLIENT	CLIENT	1	[1][1][1]_3	
SERVICE_TIME_B	CLIENT	CLIENT	1	1	0
SERVICE_TIME_A	CLIENT	CLIENT	2	[1][1][2]_3	
SERVICE_TIME_B	CLIENT	CLIENT	2	1	5.68452
THINK_TIME	1	CLIENT	IDLE_TIME		1
SERVICE_TIME_A	SERVER	SERVER_E1	1	[1][2][1]_4	
SERVICE_TIME_B	SERVER	SERVER_E1	1	1	0
THINK_TIME	2	SERVER	IDLE_TIME1		
UTILIZATION_CALCULATION_BEGIN					
ELAPSED_TIME	CLIENT	CLIENT	1	1	0
ELAPSED_TIME	CLIENT	CLIENT	1	2	0
U	CLIENT	CLIENT	1	THROUGHPUT	0
U	CLIENT	CLIENT	1	ELAPSED_TIME	0.115101

```

UTILIZATION_CALCULATION_END
UTILIZATION_CALCULATION_BEGIN
ELAPSED_TIME      CLIENT      CLIENT      2      1      5.68452
ELAPSED_TIME      CLIENT      CLIENT      2      2      3.69147
U                  CLIENT      CLIENT      2      THROUGHPUT 9.376
U                  CLIENT      CLIENT      2      ELAPSED_TIME 0.115101
UTILIZATION_CALCULATION_END
UTILIZATION_CALCULATION_BEGIN
ELAPSED_TIME      SERVER      SERVER_E1      1      1      0
ELAPSED_TIME      SERVER      SERVER_E1      1      2      2.14163
U                  SERVER      SERVER_E1      1      THROUGHPUT 2.14163
U                  SERVER      SERVER_E1      1      ELAPSED_TIME
0.230203
UTILIZATION_CALCULATION_END
UTILIZATION      proc1      CLIENT      1.0
UTILIZATION      proc1      SERVER_E1      1.0
RHO              proc1      UTILIZATION      1
RHO              proc1      POPULATION      -0.690608
IDLE_TIME        proc1      RHO      -1.7376
IDLE_TIME        proc1      THROUGHPUT      -0.934133
IDLE_TIME        proc1      POPULATION      0.5376

```

Translated INTERACTION File .itx file

1	S(1,1)_1<1>	R(1,1)_5<2>	2	1.0
1	S(1,1)_2<1>	R(2,2)_5<2>	2	1.0
2	S(1,1)_3<2>	R(1,1)_2<1>	1	1.0
2	Z[1]<2>	Q[1]<1>	1	1.0
2	Z[2]<2>	A[2]<1>	1	-9.56709
2	Z[2]<2>	D[2]<1>	1	-4.344

IDENTIFICATION (.id) file

1	CHAIN	CLIENT	SERVER	1
1	STATION	CLIENT	1	0
1	STATION	SERVER	2	0
2	CHAIN	CLIENT	proc1	1
2	CHAIN	SERVER	proc1	2
2	STATION	CLIENT	1	0
2	STATION	SERVER	2	0
2	STATION	proc1	3	1
1	ENTRY	CLIENT	CLIENT	1
1	ENTRY	SERVER	SERVER_E1	1
2	ENTRY	CLIENT	CLIENT	1
2	ENTRY	SERVER	SERVER_E1	1
2	ENTRY	proc1	CLIENT	1
2	ENTRY	proc1	SERVER_E1	2

Generated MATLAB (.m) file

```

load model2.g1;
gmat = model2;
i1 = gmat(:,1);
j1 = gmat(:,2);
s1 = gmat(:,3);
SG1 = sparse(i1,j1,s1,24,24);
clear gmat;
clear i1 j1 s1;
load model2.u1;
umat = model2;
i1 = umat(:,1);
j1 = umat(:,2);
s1 = umat(:,3);
SU1 = sparse(i1,j1,s1,24,5);
clear umat1
clear i1 j1 s1;
load model2.g2;
gmat = model2;
i2 = gmat(:,1);
j2 = gmat(:,2);
s2 = gmat(:,3);
SG2 = sparse(i2,j2,s2,76,76);
clear gmat;
clear i2 j2 s2;
load model2.u2;
umat = model2;
i2 = umat(:,1);
j2 = umat(:,2);
s2 = umat(:,3);
SU2 = sparse(i2,j2,s2,76,15);
clear umat1
clear i2 j2 s2;

gsizel = size(SG1);
usizel = size(SU1);

a = zeros(usizel(1,2),gsizel(1,1));
b = eye(usizel(1,2));
A1 = [SG1 SU1; a b; ];
SA1 = sparse(A1);
clear A1;
asizel = size(SA1);
d1 = eye(usizel(1,2));
m1 = usizel(1,2);

gsizel2 = size(SG2);
usizel2 = size(SU2);

a = zeros(usizel2(1,2),gsizel2(1,1));
b = eye(usizel2(1,2));
A2 = [SG2 SU2; a b; ];

```

```

SA2 = sparse(A2);
clear A2;
asize2 = size(SA2);
d2 = eye(usize2(1,2));
m2 = usize2(1,2);

A = [ SA1 zeros(asize1(1,1),asize2(1,2)) ;
      zeros(asize2(1,1),asize1(1,2)) SA2 ;
      ];

A(22,74) = 1.0;
A(23,79) = 1.0;
A(101,14) = 1.0;
A(105,24) = 1.0;
A(100,2) = -9.56709;
A(100,8) = -4.344;
p = 0 + m1 + m2 ;
r11 = zeros(m1,m1);
r12 = zeros(m1,m2);
r21 = zeros(m2,m1);
r22 = zeros(m2,m2);
b = [
zeros(gsize1(1,1),p) ;
d1 r12 ;
zeros(gsize2(1,1),p) ;
r21 d2 ;
];
SA = sparse(A);
Sb = sparse(b);
clear A b;
X = SA \ Sb;
FX = full(X);
sz = size(FX)
save model2.res FX -ascii;

```

SENSITIVITY (.sens) file

THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT] -> [CLIENT] [CLIENT]<1>	9.4761442e-02	0.823289476199164
THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT] -> [SERVER] [SERVER_E1]<1>	-2.4583668e-02	-
0.427166888211223			
THROUGHPUT_[CLIENT]<1>	POPULATION_[CLIENT]<1>	4.2994414e-02	0.74707281431091
THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER]<1>	9.7508519e-03	
0.0847156141128226			
THROUGHPUT_[CLIENT]<1>	THINK_TIME_[CLIENT]<1>	-5.2433027e-03	-
0.364431426312543			
THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT] -> [CLIENT] [CLIENT]<2>	1.5153669e-03	
0.0131655406990382			
THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER] -> [CLIENT] [CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT] -> [SERVER] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER] -> [SERVER] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT] -> [proc1] [CLIENT]<2>	-7.9958501e-03	-
0.208404360518154			
THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER] -> [proc1] [CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<1>	VISITS_[CLIENT] -> [proc1] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<1>	VISITS_[SERVER] -> [proc1] [SERVER_E1]<2>	-4.9167328e-02	-
0.427166818707049			
THROUGHPUT_[CLIENT]<1>	POPULATION_[CLIENT]<2>	-3.8191679e-03	-
0.0663620281318147			
THROUGHPUT_[CLIENT]<1>	POPULATION_[SERVER]<2>	-2.5667623e-02	-
0.223000868802182			
THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[CLIENT]@[proc1] [CLIENT]<2>	-2.6442388e-02	-
0.153154772597944			
THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[SERVER]@[proc1] [CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[CLIENT]@[proc1] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<1>	SERVICE_TIME_[SERVER]@[proc1] [SERVER_E1]<2>	-2.7763344e-02	-
0.482417077175698			
THROUGHPUT_[CLIENT]<1>	POPULATION_[proc1]<2>	1.4242928e-02	0.123742869306088
THROUGHPUT_[SERVER]<1>	VISITS_[CLIENT] -> [CLIENT] [CLIENT]<1>	-4.0679117e-02	-
0.176709760515719			
THROUGHPUT_[SERVER]<1>	VISITS_[CLIENT] -> [SERVER] [SERVER_E1]<1>	6.5933664e-02	0.572830623406298
THROUGHPUT_[SERVER]<1>	POPULATION_[CLIENT]<1>	8.5988828e-02	0.74706956903255
THROUGHPUT_[SERVER]<1>	POPULATION_[SERVER]<1>	1.9501704e-02	
0.0847152469776676			
THROUGHPUT_[SERVER]<1>	THINK_TIME_[CLIENT]<1>	-1.0486605e-02	-
0.364429829324553			
THROUGHPUT_[SERVER]<1>	VISITS_[CLIENT] -> [CLIENT] [CLIENT]<2>	3.0307337e-03	0.013165483073635
THROUGHPUT_[SERVER]<1>	VISITS_[SERVER] -> [CLIENT] [CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<1>	VISITS_[CLIENT] -> [SERVER] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<1>	VISITS_[SERVER] -> [SERVER] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<1>	VISITS_[CLIENT] -> [proc1] [CLIENT]<2>	-1.5991700e-02	-
0.208403452604875			
THROUGHPUT_[SERVER]<1>	VISITS_[SERVER] -> [proc1] [CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<1>	VISITS_[CLIENT] -> [proc1] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<1>	VISITS_[SERVER] -> [proc1] [SERVER_E1]<2>	-9.8334656e-02	-
0.427164963097788			
THROUGHPUT_[SERVER]<1>	POPULATION_[CLIENT]<2>	-7.6383358e-03	-
0.0663617398556926			
THROUGHPUT_[SERVER]<1>	POPULATION_[SERVER]<2>	-5.1335246e-02	-
0.222999900088183			
THROUGHPUT_[SERVER]<1>	SERVICE_TIME_[CLIENT]@[proc1] [CLIENT]<2>	-5.2884776e-02	-
0.153154107294831			
THROUGHPUT_[SERVER]<1>	SERVICE_TIME_[SERVER]@[proc1] [CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<1>	SERVICE_TIME_[CLIENT]@[proc1] [SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<1>	SERVICE_TIME_[SERVER]@[proc1] [SERVER_E1]<2>	-5.5526689e-02	-
0.482414990247738			
THROUGHPUT_[SERVER]<1>	POPULATION_[proc1]<2>	2.8485856e-02	0.123742331768048
UTILIZATION_[CLIENT]<1>	VISITS_[CLIENT] -> [CLIENT] [CLIENT]<1>	3.1836041e-01	0.749271957881161
UTILIZATION_[CLIENT]<1>	VISITS_[CLIENT] -> [SERVER] [SERVER_E1]<1>	-4.5844056e-02	-
0.215791062691078			
UTILIZATION_[CLIENT]<1>	POPULATION_[CLIENT]<1>	2.2320660e-01	1.05064851621467
UTILIZATION_[CLIENT]<1>	POPULATION_[SERVER]<1>	5.5042894e-02	0.129545306700746
UTILIZATION_[CLIENT]<1>	THINK_TIME_[CLIENT]<1>	-2.9598086e-02	-
0.557280745976046			
UTILIZATION_[CLIENT]<1>	VISITS_[CLIENT] -> [CLIENT] [CLIENT]<2>	-4.2006495e-03	-
0.00988637021556015			
UTILIZATION_[CLIENT]<1>	VISITS_[SERVER] -> [CLIENT] [CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<1>	VISITS_[CLIENT] -> [SERVER] [SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<1>	VISITS_[SERVER] -> [SERVER] [SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<1>	VISITS_[CLIENT] -> [proc1] [CLIENT]<2>	1.0949055e-01	0.773069102103353
UTILIZATION_[CLIENT]<1>	VISITS_[SERVER] -> [proc1] [CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<1>	VISITS_[CLIENT] -> [proc1] [SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<1>	VISITS_[SERVER] -> [proc1] [SERVER_E1]<2>	-9.1688215e-02	-0.21579130510505
UTILIZATION_[CLIENT]<1>	POPULATION_[CLIENT]<2>	1.4356115e-02	
0.0675752012859708			

UTILIZATION_[CLIENT]<1>	POPULATION_[SERVER]<2>	1.7538204e-01	0.412767543828682
UTILIZATION_[CLIENT]<1>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	2.8620400e-01	0.449060733097509
UTILIZATION_[CLIENT]<1>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<1>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<1>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	2.2990627e-02	0.108218431463921
UTILIZATION_[CLIENT]<1>	POPULATION_[proc1]<2>	-1.4444603e-01	-
0.339958601341985			
UTILIZATION_[SERVER]<1>	VISITS_[CLIENT]->[CLIENT][CLIENT]<1>	-8.8367747e-02	-
0.179241650760939			
UTILIZATION_[SERVER]<1>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<1>	1.3768896e-01	0.558565705697056
UTILIZATION_[SERVER]<1>	POPULATION_[CLIENT]<1>	1.8498193e-01	0.750420093750824
UTILIZATION_[SERVER]<1>	POPULATION_[SERVER]<1>	4.5989121e-02	
0.0932825181690395			
UTILIZATION_[SERVER]<1>	THINK_TIME_[CLIENT]<1>	-2.4729622e-02	-
0.401284714883501			
UTILIZATION_[SERVER]<1>	VISITS_[CLIENT]->[CLIENT][CLIENT]<2>	-4.4979290e-03	-
0.00912342168195713			
UTILIZATION_[SERVER]<1>	VISITS_[SERVER]->[CLIENT][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<1>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<1>	VISITS_[SERVER]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<1>	VISITS_[CLIENT]->[proc1][CLIENT]<2>	-2.5846579e-02	-
0.157278542582387			
UTILIZATION_[SERVER]<1>	VISITS_[SERVER]->[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<1>	VISITS_[CLIENT]->[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<1>	VISITS_[SERVER]->[proc1][SERVER_E1]<2>	2.7537759e-01	0.558565036338079
UTILIZATION_[SERVER]<1>	POPULATION_[CLIENT]<2>	9.1960744e-03	
0.0373059088170804			
UTILIZATION_[SERVER]<1>	POPULATION_[SERVER]<2>	1.7009181e-02	0.034500751507579
UTILIZATION_[SERVER]<1>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	-4.2391900e-02	-
0.0573240666951313			
UTILIZATION_[SERVER]<1>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<1>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<1>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	1.1304958e-01	0.458610613599346
UTILIZATION_[SERVER]<1>	POPULATION_[proc1]<2>	1.7318256e-02	
0.0351276570405611			
THROUGHPUT_[CLIENT]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<1>	4.1133294e-03	
0.0357366955977793			
THROUGHPUT_[CLIENT]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<1>	-2.4583668e-02	-
0.427166888211223			
THROUGHPUT_[CLIENT]<2>	POPULATION_[CLIENT]<1>	-1.4556286e-02	-0.25293066089782
THROUGHPUT_[CLIENT]<2>	POPULATION_[SERVER]<1>	9.7508519e-03	
0.0847156141128226			
THROUGHPUT_[CLIENT]<2>	THINK_TIME_[CLIENT]<1>	-5.2433027e-03	-
0.364431426312543			
THROUGHPUT_[CLIENT]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<2>	7.8961197e-02	0.686016602809706
THROUGHPUT_[CLIENT]<2>	VISITS_[SERVER]->[CLIENT][CLIENT]<2>	2.3020300e-01	0
THROUGHPUT_[CLIENT]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<2>	VISITS_[SERVER]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<2>	VISITS_[CLIENT]->[proc1][CLIENT]<2>	-7.9958501e-03	-
0.208404360518154			
THROUGHPUT_[CLIENT]<2>	VISITS_[SERVER]->[proc1][CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<2>	VISITS_[CLIENT]->[proc1][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<2>	VISITS_[SERVER]->[proc1][SERVER_E1]<2>	-4.9167328e-02	-
0.427166818707049			
THROUGHPUT_[CLIENT]<2>	POPULATION_[CLIENT]<2>	5.3731532e-02	0.933641445339311
THROUGHPUT_[CLIENT]<2>	POPULATION_[SERVER]<2>	-2.5667623e-02	-
0.223000868802182			
THROUGHPUT_[CLIENT]<2>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	-2.6442388e-02	-
0.153154772597944			
THROUGHPUT_[CLIENT]<2>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<2>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[CLIENT]<2>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	-2.7763344e-02	-
0.482417077175698			
THROUGHPUT_[CLIENT]<2>	POPULATION_[proc1]<2>	1.4242928e-02	0.123742869306088
THROUGHPUT_[SERVER]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<1>	-4.0679085e-02	-
0.176709621507973			
THROUGHPUT_[SERVER]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<1>	6.5933897e-02	0.572832647706589
THROUGHPUT_[SERVER]<2>	POPULATION_[CLIENT]<1>	8.5988761e-02	0.747068986937616
THROUGHPUT_[SERVER]<2>	POPULATION_[SERVER]<1>	1.9501690e-02	
0.0847151861617789			
THROUGHPUT_[SERVER]<2>	THINK_TIME_[CLIENT]<1>	-1.0486598e-02	-
0.364429586060998			
THROUGHPUT_[SERVER]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<2>	3.0307267e-03	
0.0131654526656907			
THROUGHPUT_[SERVER]<2>	VISITS_[SERVER]->[CLIENT][CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<2>	1.1510100e-01	0
THROUGHPUT_[SERVER]<2>	VISITS_[SERVER]->[SERVER][SERVER_E1]<2>	2.3020300e-01	1
THROUGHPUT_[SERVER]<2>	VISITS_[CLIENT]->[proc1][CLIENT]<2>	-1.5991684e-02	-
0.208403244093257			
THROUGHPUT_[SERVER]<2>	VISITS_[SERVER]->[proc1][CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<2>	VISITS_[CLIENT]->[proc1][SERVER_E1]<2>	0.0000000e+00	0

THROUGHPUT_[SERVER]<2>	VISITS_[SERVER]->[proc1][SERVER_E1]<2>	-9.8334373e-02	-0.42716373374804
THROUGHPUT_[SERVER]<2>	POPULATION_[CLIENT]<2>	-7.6383190e-03	-0.06636159389756
THROUGHPUT_[SERVER]<2>	POPULATION_[SERVER]<2>	6.2157214e-02	0.270010442956869
THROUGHPUT_[SERVER]<2>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	-5.2884705e-02	-
0.153153901679105			
THROUGHPUT_[SERVER]<2>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<2>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[SERVER]<2>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	-5.5526547e-02	-
0.482413756553998			
THROUGHPUT_[SERVER]<2>	POPULATION_[proc1]<2>	2.8485816e-02	0.123742158008367
THROUGHPUT_[proc1]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<1>	-2.8339097e-02	-
0.0492419675173369			
THROUGHPUT_[proc1]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<1>	-7.8171066e-03	-
0.0271659826900455			
THROUGHPUT_[proc1]<2>	POPULATION_[CLIENT]<1>	4.2319903e-02	0.14706998524779
THROUGHPUT_[proc1]<2>	POPULATION_[SERVER]<1>	4.8754246e-02	-
0.0847152962518267			
THROUGHPUT_[proc1]<2>	THINK_TIME_[CLIENT]<1>	-2.6216506e-02	-0.36443005558577
THROUGHPUT_[proc1]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<2>	-1.0538868e-01	-0.18312319398374
THROUGHPUT_[proc1]<2>	VISITS_[SERVER]->[CLIENT][CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[proc1]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[proc1]<2>	VISITS_[SERVER]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[proc1]<2>	VISITS_[CLIENT]->[proc1][CLIENT]<2>	7.5121766e-02	0.391594364621108
THROUGHPUT_[proc1]<2>	VISITS_[SERVER]->[proc1][CLIENT]<2>	2.3020300e-01	0
THROUGHPUT_[proc1]<2>	VISITS_[CLIENT]->[proc1][SERVER_E1]<2>	1.1510100e-01	0
THROUGHPUT_[proc1]<2>	VISITS_[SERVER]->[proc1][SERVER_E1]<2>	-1.5633357e-02	-
0.0271644949583585			
THROUGHPUT_[proc1]<2>	POPULATION_[CLIENT]<2>	1.5355628e-01	0.533638270255618
THROUGHPUT_[proc1]<2>	POPULATION_[SERVER]<2>	-1.4845655e-02	-
0.0257957852814996			
THROUGHPUT_[proc1]<2>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	-1.3221187e-01	-
0.153154159267029			
THROUGHPUT_[proc1]<2>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
THROUGHPUT_[proc1]<2>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
THROUGHPUT_[proc1]<2>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	-1.3881658e-01	-
0.482414914153955			
THROUGHPUT_[proc1]<2>	POPULATION_[proc1]<2>	7.1214600e-02	0.123742369771349
UTILIZATION_[CLIENT]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<1>	-1.6640941e-02	-
0.0254333527944539			
UTILIZATION_[CLIENT]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<1>	2.4251215e-01	0.741291861787326
UTILIZATION_[CLIENT]<2>	POPULATION_[CLIENT]<1>	1.0568962e-01	0.323063628694047
UTILIZATION_[CLIENT]<2>	POPULATION_[SERVER]<1>	-1.3304921e-01	-
0.203347124237348			
UTILIZATION_[CLIENT]<2>	THINK_TIME_[CLIENT]<1>	-4.3556760e-02	-
0.532563365816083			
UTILIZATION_[CLIENT]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<2>	4.3232222e-01	0.660744097472703
UTILIZATION_[CLIENT]<2>	VISITS_[SERVER]->[CLIENT][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<2>	VISITS_[SERVER]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<2>	VISITS_[CLIENT]->[proc1][CLIENT]<2>	-4.5524161e-02	-
0.208731954650495			
UTILIZATION_[CLIENT]<2>	VISITS_[SERVER]->[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<2>	VISITS_[CLIENT]->[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<2>	VISITS_[SERVER]->[proc1][SERVER_E1]<2>	4.8502433e-01	0.741291907638133
UTILIZATION_[CLIENT]<2>	POPULATION_[CLIENT]<2>	3.4334514e-01	1.04951012997176
UTILIZATION_[CLIENT]<2>	POPULATION_[SERVER]<2>	2.9957637e-02	0.045786061660166
UTILIZATION_[CLIENT]<2>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	-7.4666246e-02	-
0.0760779864496833			
UTILIZATION_[CLIENT]<2>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<2>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[CLIENT]<2>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	1.9911471e-01	0.608638017044274
UTILIZATION_[CLIENT]<2>	POPULATION_[proc1]<2>	3.0503326e-02	-
0.0466200710381846			
UTILIZATION_[SERVER]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<1>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<1>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	POPULATION_[CLIENT]<1>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	POPULATION_[SERVER]<1>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	THINK_TIME_[CLIENT]<1>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[SERVER]->[CLIENT][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[SERVER]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[CLIENT]->[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[SERVER]->[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[CLIENT]->[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	VISITS_[SERVER]->[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	POPULATION_[CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	POPULATION_[SERVER]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0

UTILIZATION_[SERVER]<2>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[SERVER]<2>	POPULATION_[proc1]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<1>	-7.3131512e-02	-
0.105894388712555			
UTILIZATION_[proc1]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<1>	8.2700459e-02	0.239500437295832
UTILIZATION_[proc1]<2>	POPULATION_[CLIENT]<1>	1.4286495e-01	0.413736736325093
UTILIZATION_[proc1]<2>	POPULATION_[SERVER]<1>	5.8505084e-02	
0.0847153291013136			
UTILIZATION_[proc1]<2>	THINK_TIME_[CLIENT]<1>	-3.1459801e-02	-
0.364430194842805			
UTILIZATION_[proc1]<2>	VISITS_[CLIENT]->[CLIENT][CLIENT]<2>	-6.6218153e-02	-
0.0958838487246021			
UTILIZATION_[proc1]<2>	VISITS_[SERVER]->[CLIENT][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	VISITS_[CLIENT]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	VISITS_[SERVER]->[SERVER][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	VISITS_[CLIENT]->[proc1][CLIENT]<2>	2.8759132e-02	0.124929621435025
UTILIZATION_[proc1]<2>	VISITS_[SERVER]->[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	VISITS_[CLIENT]->[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	VISITS_[SERVER]->[proc1][SERVER_E1]<2>	1.6540160e-01	0.239501424831453
UTILIZATION_[proc1]<2>	POPULATION_[CLIENT]<2>	9.2186426e-02	0.266971787178834
UTILIZATION_[proc1]<2>	POPULATION_[SERVER]<2>	7.2979182e-02	0.105673814957255
UTILIZATION_[proc1]<2>	SERVICE_TIME_[CLIENT]@[proc1][CLIENT]<2>	1.8664981e-01	0.180179304154122
UTILIZATION_[proc1]<2>	SERVICE_TIME_[SERVER]@[proc1][CLIENT]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	SERVICE_TIME_[CLIENT]@[proc1][SERVER_E1]<2>	0.0000000e+00	0
UTILIZATION_[proc1]<2>	SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>	6.3623217e-02	0.184252765678938
UTILIZATION_[proc1]<2>	POPULATION_[proc1]<2>	8.5457487e-02	0.123742393658921

RANKED ELASTICITY (.rank) file

```

1.0506485162 - 2.2320660e-01 UTILIZATION_[CLIENT]<1> POPULATION_[CLIENT]<1>
1.0495101300 - 3.4334514e-01 UTILIZATION_[CLIENT]<2> POPULATION_[CLIENT]<2>
1.0000000000 - 2.3020300e-01 THROUGHPUT_[SERVER]<2> VISITS_[SERVER] -
>[SERVER][SERVER_E1]<2>
0.9336414453 - 5.3731532e-02 THROUGHPUT_[CLIENT]<2> POPULATION_[CLIENT]<2>
0.8232894762 - 9.4761442e-02 THROUGHPUT_[CLIENT]<1> VISITS_[CLIENT] ->[CLIENT][CLIENT]<1>
0.7730691021 - 1.0949055e-01 UTILIZATION_[CLIENT]<1> VISITS_[CLIENT] ->[proc1][CLIENT]<2>
0.7504200938 - 1.8498193e-01 UTILIZATION_[SERVER]<1> POPULATION_[CLIENT]<1>
0.7492719579 - 3.1836041e-01 UTILIZATION_[CLIENT]<1> VISITS_[CLIENT] ->[CLIENT][CLIENT]<1>
0.7470728143 - 4.2994414e-02 THROUGHPUT_[CLIENT]<1> POPULATION_[CLIENT]<1>
0.7470695690 - 8.5988828e-02 THROUGHPUT_[SERVER]<1> POPULATION_[CLIENT]<1>
0.7470689869 - 8.5988761e-02 THROUGHPUT_[SERVER]<2> POPULATION_[CLIENT]<1>
0.7412919076 - 4.8502433e-01 UTILIZATION_[CLIENT]<2> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.7412918618 - 2.4251215e-01 UTILIZATION_[CLIENT]<2> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.6860166028 - 7.8961197e-02 THROUGHPUT_[CLIENT]<2> VISITS_[CLIENT] ->[CLIENT][CLIENT]<2>
0.6607440975 - 4.3232222e-01 UTILIZATION_[CLIENT]<2> VISITS_[CLIENT] ->[CLIENT][CLIENT]<2>
0.6086380170 - 1.9911471e-01 UTILIZATION_[CLIENT]<2>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.5728326477 - 6.5933897e-02 THROUGHPUT_[SERVER]<2> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.5728306234 - 6.5933664e-02 THROUGHPUT_[SERVER]<1> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.5585657057 - 1.3768896e-01 UTILIZATION_[SERVER]<1> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.5585650363 - 2.7537759e-01 UTILIZATION_[SERVER]<1> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.5572807460 - 2.9598086e-02 UTILIZATION_[CLIENT]<1> THINK_TIME_[CLIENT]<1>
0.5336382703 - 1.5355628e-01 THROUGHPUT_[proc1]<2> POPULATION_[CLIENT]<2>
0.5325633658 - 4.3556760e-02 UTILIZATION_[CLIENT]<2> THINK_TIME_[CLIENT]<1>
0.4824170772 - 2.7763344e-02 THROUGHPUT_[CLIENT]<2>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.4824170772 - 2.7763344e-02 THROUGHPUT_[CLIENT]<1>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.4824149902 - 5.5526689e-02 THROUGHPUT_[SERVER]<1>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.4824149142 - 1.3881658e-01 THROUGHPUT_[proc1]<2>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.4824137566 - 5.5526547e-02 THROUGHPUT_[SERVER]<2>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.4586106136 - 1.1304958e-01 UTILIZATION_[SERVER]<1>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.4490607331 - 2.8620400e-01 UTILIZATION_[CLIENT]<1> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.4271668882 - 2.4583668e-02 THROUGHPUT_[CLIENT]<2> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.4271668882 - 2.4583668e-02 THROUGHPUT_[CLIENT]<1> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.4271668187 - 4.9167328e-02 THROUGHPUT_[CLIENT]<2> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.4271668187 - 4.9167328e-02 THROUGHPUT_[CLIENT]<1> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.4271649631 - 9.8334656e-02 THROUGHPUT_[SERVER]<1> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.4271637337 - 9.8334373e-02 THROUGHPUT_[SERVER]<2> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.4137367363 - 1.4286495e-01 UTILIZATION_[proc1]<2> POPULATION_[CLIENT]<1>
0.4127675438 - 1.7538204e-01 UTILIZATION_[CLIENT]<1> POPULATION_[SERVER]<2>
0.4012847149 - 2.4729622e-02 UTILIZATION_[SERVER]<1> THINK_TIME_[CLIENT]<1>
0.3915943646 - 7.5121766e-02 THROUGHPUT_[proc1]<2> VISITS_[CLIENT] ->[proc1][CLIENT]<2>
0.3644314263 - 5.2433027e-03 THROUGHPUT_[CLIENT]<2> THINK_TIME_[CLIENT]<1>
0.3644314263 - 5.2433027e-03 THROUGHPUT_[CLIENT]<1> THINK_TIME_[CLIENT]<1>
0.3644301948 - 3.1459801e-02 UTILIZATION_[proc1]<2> THINK_TIME_[CLIENT]<1>
0.3644300556 - 2.6216506e-02 THROUGHPUT_[proc1]<2> THINK_TIME_[CLIENT]<1>
0.3644298293 - 1.0486605e-02 THROUGHPUT_[SERVER]<1> THINK_TIME_[CLIENT]<1>
0.3644295861 - 1.0486598e-02 THROUGHPUT_[SERVER]<2> THINK_TIME_[CLIENT]<1>
0.3399586013 - 1.4444603e-01 UTILIZATION_[CLIENT]<1> POPULATION_[proc1]<2>
0.3230636287 - 1.0568962e-01 UTILIZATION_[CLIENT]<2> POPULATION_[CLIENT]<1>
0.2700104430 - 6.2157214e-02 THROUGHPUT_[SERVER]<2> POPULATION_[SERVER]<2>
0.2669717872 - 9.2186426e-02 UTILIZATION_[proc1]<2> POPULATION_[CLIENT]<2>
0.2529306609 - 1.4556286e-02 THROUGHPUT_[CLIENT]<2> POPULATION_[CLIENT]<1>
0.2395014248 - 1.6540160e-01 UTILIZATION_[proc1]<2> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.2395004373 - 8.2700459e-02 UTILIZATION_[proc1]<2> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.2230008688 - 2.5667623e-02 THROUGHPUT_[CLIENT]<2> POPULATION_[SERVER]<2>
0.2230008688 - 2.5667623e-02 THROUGHPUT_[CLIENT]<1> POPULATION_[SERVER]<2>
0.2229999001 - 5.1335246e-02 THROUGHPUT_[SERVER]<1> POPULATION_[SERVER]<2>
0.2157913051 - 9.1688215e-02 UTILIZATION_[CLIENT]<1> VISITS_[SERVER] ->[proc1][SERVER_E1]<2>
0.2157910627 - 4.5844056e-02 UTILIZATION_[CLIENT]<1> VISITS_[CLIENT] -
>[SERVER][SERVER_E1]<1>
0.2087319547 - 4.5524161e-02 UTILIZATION_[CLIENT]<2> VISITS_[CLIENT] ->[proc1][CLIENT]<2>
0.2084043605 - 7.9958501e-03 THROUGHPUT_[CLIENT]<2> VISITS_[CLIENT] ->[proc1][CLIENT]<2>

```

```

0.2084043605 - -7.9958501e-03 THROUGHPUT_[CLIENT]<1> VISITS_[CLIENT]->[proc1][CLIENT]<2>
0.2084034526 - -1.5991700e-02 THROUGHPUT_[SERVER]<1> VISITS_[CLIENT]->[proc1][CLIENT]<2>
0.2084032441 - -1.5991684e-02 THROUGHPUT_[SERVER]<2> VISITS_[CLIENT]->[proc1][CLIENT]<2>
0.2033471242 - -1.3304921e-01 UTILIZATION_[CLIENT]<2> POPULATION_[SERVER]<1>
0.1842527657 + 6.3623217e-02 UTILIZATION_[proc1]<2>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.1831231940 - -1.0538868e-01 THROUGHPUT_[proc1]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<2>
0.1801793042 + 1.8664981e-01 UTILIZATION_[proc1]<2> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.1792416508 - -8.8367747e-02 UTILIZATION_[SERVER]<1> VISITS_[CLIENT]->[CLIENT][CLIENT]<1>
0.1767097605 - -4.0679117e-02 THROUGHPUT_[SERVER]<1> VISITS_[CLIENT]->[CLIENT][CLIENT]<1>
0.1767096215 - -4.0679085e-02 THROUGHPUT_[SERVER]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<1>
0.1572785426 - -2.5846579e-02 UTILIZATION_[SERVER]<1> VISITS_[CLIENT]->[proc1][CLIENT]<2>
0.1531547726 - -2.6442388e-02 THROUGHPUT_[CLIENT]<2> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.1531547726 - -2.6442388e-02 THROUGHPUT_[CLIENT]<1> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.1531541593 - -1.3221187e-01 THROUGHPUT_[proc1]<2> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.1531541073 - -5.2884776e-02 THROUGHPUT_[SERVER]<1> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.1531539017 - -5.2884705e-02 THROUGHPUT_[SERVER]<2> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.1470699852 + 4.2319903e-02 THROUGHPUT_[proc1]<2> POPULATION_[CLIENT]<1>
0.1295453067 + 5.5042894e-02 UTILIZATION_[CLIENT]<1> POPULATION_[SERVER]<1>
0.1249296214 + 2.8759132e-02 UTILIZATION_[proc1]<2> VISITS_[CLIENT]->[proc1][CLIENT]<2>
0.1237428693 + 1.4242928e-02 THROUGHPUT_[CLIENT]<2> POPULATION_[proc1]<2>
0.1237428693 + 1.4242928e-02 THROUGHPUT_[CLIENT]<1> POPULATION_[proc1]<2>
0.1237423937 + 8.5457487e-02 UTILIZATION_[proc1]<2> POPULATION_[proc1]<2>
0.1237423698 + 7.1214600e-02 THROUGHPUT_[proc1]<2> POPULATION_[proc1]<2>
0.1237423318 + 2.8485856e-02 THROUGHPUT_[SERVER]<1> POPULATION_[proc1]<2>
0.1237421580 + 2.8485816e-02 THROUGHPUT_[SERVER]<2> POPULATION_[proc1]<2>
0.1082184315 + 2.2990627e-02 UTILIZATION_[CLIENT]<1>
SERVICE_TIME_[SERVER]@[proc1][SERVER_E1]<2>
0.1058943887 - -7.3131512e-02 UTILIZATION_[proc1]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<1>
0.1056738150 + 7.2979182e-02 UTILIZATION_[proc1]<2> POPULATION_[SERVER]<2>
0.0958838487 - -6.6218153e-02 UTILIZATION_[proc1]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<2>
0.0932825182 + 4.5989121e-02 UTILIZATION_[SERVER]<1> POPULATION_[SERVER]<1>
0.0847156141 + 9.7508519e-03 THROUGHPUT_[CLIENT]<2> POPULATION_[SERVER]<1>
0.0847156141 + 9.7508519e-03 THROUGHPUT_[CLIENT]<1> POPULATION_[SERVER]<1>
0.0847153291 + 5.8505084e-02 UTILIZATION_[proc1]<2> POPULATION_[SERVER]<1>
0.0847152963 + 4.8754246e-02 THROUGHPUT_[proc1]<2> POPULATION_[SERVER]<1>
0.0847152470 + 1.9501704e-02 THROUGHPUT_[SERVER]<1> POPULATION_[SERVER]<1>
0.0847151862 + 1.9501690e-02 THROUGHPUT_[SERVER]<2> POPULATION_[SERVER]<1>
0.0760779864 - -7.4666246e-02 UTILIZATION_[CLIENT]<2> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.0675752013 + 1.4356115e-02 UTILIZATION_[CLIENT]<1> POPULATION_[CLIENT]<2>
0.0663620281 - -3.8191679e-03 THROUGHPUT_[CLIENT]<1> POPULATION_[CLIENT]<2>
0.0663617399 - -7.6383358e-03 THROUGHPUT_[SERVER]<1> POPULATION_[CLIENT]<2>
0.0663615939 - -7.6383190e-03 THROUGHPUT_[SERVER]<2> POPULATION_[CLIENT]<2>
0.0573240667 - -4.2391900e-02 UTILIZATION_[SERVER]<1> SERVICE_TIME_[CLIENT]@[proc1][CLI-
ENT]<2>
0.0492419675 - -2.8339097e-02 THROUGHPUT_[proc1]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<1>
0.0466200710 + 3.0503326e-02 UTILIZATION_[CLIENT]<2> POPULATION_[proc1]<2>
0.0457860617 + 2.9957637e-02 UTILIZATION_[CLIENT]<2> POPULATION_[SERVER]<2>
0.0373059088 + 9.1960744e-03 UTILIZATION_[SERVER]<1> POPULATION_[CLIENT]<2>
0.0357366956 + 4.1133294e-03 THROUGHPUT_[CLIENT]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<1>
0.0351276670 - 1.7318256e-02 UTILIZATION_[SERVER]<1> POPULATION_[proc1]<2>
0.0345007515 - 1.7009181e-02 UTILIZATION_[SERVER]<1> POPULATION_[SERVER]<2>
0.0271659827 - -7.8171066e-03 THROUGHPUT_[proc1]<2> VISITS_[CLIENT]-
>[SERVER][SERVER_E1]<1>
0.0271644950 - -1.5633357e-02 THROUGHPUT_[proc1]<2> VISITS_[SERVER]->[proc1][SERVER_E1]<2>
0.0257957853 - -1.4845655e-02 THROUGHPUT_[proc1]<2> POPULATION_[SERVER]<2>
0.0254333528 - -1.6640941e-02 UTILIZATION_[CLIENT]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<1>
0.0131655407 - 1.5153669e-03 THROUGHPUT_[CLIENT]<1> VISITS_[CLIENT]->[CLIENT][CLIENT]<2>
0.0131654831 + 3.0307337e-03 THROUGHPUT_[SERVER]<1> VISITS_[CLIENT]->[CLIENT][CLIENT]<2>
0.0131654527 + 3.0307267e-03 THROUGHPUT_[SERVER]<2> VISITS_[CLIENT]->[CLIENT][CLIENT]<2>
0.0098863702 - -4.2006495e-03 UTILIZATION_[CLIENT]<1> VISITS_[CLIENT]->[CLIENT][CLIENT]<2>
0.0091234217 - -4.4979290e-03 UTILIZATION_[SERVER]<1> VISITS_[CLIENT]->[CLIENT][CLIENT]<2>

```

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc.. All Rights Reserved