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

**Performance Bounds for Concurrent Software
Modelled by Multiclass Queueing Networks with
Simultaneous Resource Possession**

by

Alex Hubbard, B.Eng

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for 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

May 4, 1998

© Copyright 1998, Alex Hubbard

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.

0-612-32400-1

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

**Performance Bounds for Concurrent Software Modelled by Multiclass
Queueing Networks with Simultaneous Resource Possession**

submitted by Alex Hubbard, B.ENG (CSE)
in partial fulfillment of the requirements for the degree of
Master of Engineering

Chair, Department of Systems and Computer Engineering

Professor Shikharesh Majumdar, Thesis Supervisor

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
May 4, 1998

Abstract

Multiclass queueing network modelling is an important technique for understanding the performance of large and complex software systems such as those found in concurrent software and distributed systems. A technique for computing the upper and lower bounds on class throughputs for these queueing networks is presented. Since it is an analytic technique of moderate complexity, it provides results more quickly than simulation. However, since it is a bounds analysis technique, it can compute the guaranteed level of performance in the system under study. The underlying assumptions on which the modelling approach is based are weaker than those for many analytic techniques making this technique quite flexible. The technique has many features, such as simultaneous resource possession and non-FIFO scheduling policies, that are well suited to the modelling of real-world concurrent software systems. The implementation of this modelling technique as a performance analysis tool written in Java and CLP(BNR) is described.

Acknowledgements

The funding for this research came in part from the Telecommunications Research Institute of Ontario, a provincial centre of excellence, through its project on Telecom Software methods. Professor Murray Woodside also supported this research in allowing me to work on my Master's degree while working as a full-time Research Engineer for him.

I would like to thank Professor Shikharesh Majumdar, my supervisor, for his guidance and support at every stage of development of this thesis. His patient and thorough editing of the lengthy, and sometimes complicated mathematical derivations was greatly appreciated. I would also like to thank him, and Professor Woodside too, for providing such an interesting and challenging problem to work on.

Working on this thesis would not have been the same without the community and support of the RADS lab regulars. Their suggestions, inspiration, and friendship over the years has made a big difference to me. In particular, I wish to mention Greg Franks, Curtis Hrischuk, Fahim Sheikh and Marc Courtois.

Finally, I want to thank my wife Chris without whose loving patience and support this thesis could not have been written. I thank her for her good natured acceptance of my being away to work on my thesis so many evenings and weekends. And I thank her for her encouragement and belief in me during those times when it seemed like the obstacles in my path were insurmountable. For three years I have been working on this thesis and she has been there with me every step of the way. I dedicate this thesis to Chris.

Contents

1	Introduction	1
1.1	Motivation	1
1.1.1	Queueing Network Modelling	2
1.1.2	Queueing Network Bounds	3
1.1.3	Simultaneous Resource Possession	3
1.2	Contributions	4
1.3	Scope	5
1.4	Outline	6
2	Queueing Networks and Interval Analysis	7
2.1	Introduction	7
2.2	Queueing Network Bounds Analysis	7
2.2.1	Single Class Queueing Networks	8
2.2.2	Multiclass Queueing Networks	8
2.2.3	Rendezvous Networks	9
2.3	Simultaneous Resource Possession	9
2.4	Constraint Logic Programming and Interval Analysis	10
2.4.1	Interval Arithmetic	10
2.4.2	Relational Interval Arithmetic	13
2.4.3	Interval Analysis in Performance Evaluation	14
3	The Model and The Bounds	16
3.1	Introduction	16
3.1.1	Minimal Stochastic Assumptions	17
3.2	The Model	17
3.3	Class Cycle Time	21
3.4	No Contention Upper Bound	21
3.5	Utilization Upper Bound	22
3.6	Lower Bound on Throughput	23
3.6.1	Waiting Time at a Secondary Resource	25
3.6.2	Waiting Time at a Primary Resource	25
3.7	Waiting Time Bounds at Secondary Resources	27

3.7.1	Secondary Resources with FIFO Scheduling	27
3.7.2	Secondary Resources with PP Scheduling	29
3.7.3	Secondary Resources with NPP Scheduling	31
3.7.4	Secondary Resources with PS Scheduling	36
3.8	Waiting Time Bounds at FIFO Primary Resources	38
3.8.1	Secondary Resources with FIFO Scheduling	38
3.8.2	Secondary Resources with PP Scheduling	40
3.8.3	Secondary Resources with NPP Scheduling	42
3.8.4	Secondary Resources with PS Scheduling	46
3.9	Waiting Time Bounds at NPP Primary Resources	48
3.9.1	The General Approach	48
3.9.2	Secondary Resources with FIFO Scheduling	54
3.9.3	Secondary Resources with PS Scheduling	57
3.9.4	Secondary Resources with PP Scheduling	60
3.9.5	Secondary Resources with NPP Scheduling	67
3.10	Summary Of Lower Bounds Expressions	73
3.10.1	General Expression for Lower Bound on Class Frequency	74
3.10.2	Waiting Time at Secondary Resources	74
3.10.3	Waiting Time at Primary Resources	75
3.10.4	Expected Collisions at a Primary Resource	76
3.10.5	Delay at a Secondary Resource	77
3.10.6	Indirect Delay at a Primary Resource	78
3.11	Discussion	79
4	Modeling Software	80
4.1	Introduction	80
4.2	Graphical User Interface Overview	82
4.3	The HTK Package	84
4.4	The MVC Package	86
4.4.1	The Model	88
4.4.2	The View	88
4.4.3	The Control	88
4.4.4	MVC Classes and Inheritance	89
4.4.5	The Model Classes	89
4.4.6	The View Classes	92
4.4.7	The Control Classes	93
4.4.8	The Indication Classes	93
4.4.9	Indication Exceptions	94
4.5	The QNBApp Application	94
4.6	Graphical User Interface Operation	96
4.6.1	Tool Initialization	96
4.6.2	Creating A Class Component	97

4.6.3	Moving a Lock	97
4.6.4	Solving A Model	98
4.7	CLP(BNR) Bounds Analyzer	99
4.7.1	The Prolog Data Structures	100
4.7.2	Analysis Algorithms	105
4.8	Supporting Software	106
4.9	Discussion	107
5	Validation	108
5.1	Introduction	108
5.2	Validation of The Analyzer Software	108
5.2.1	Checking The Inequalities Symbolically	108
5.2.2	Regression Tests	109
5.3	Validation of The Bounds Computation	110
5.3.1	A Client-Server System	110
5.3.2	The Client-Server System with Two CPUs	119
5.3.3	Adding Non-FIFO Scheduling to the System	123
5.3.4	A More Complex Example	125
5.4	Problems Encountered	129
5.4.1	Bounds Looseness	129
5.4.2	CLP(BNR) Bugs	134
5.5	Discussion	135
6	Conclusions	137
6.1	Discussion	137
6.2	Summary of Contributions	138
6.2.1	Formal Model	138
6.2.2	Bounds Analysis Tool	138
6.2.3	Validation of the Model and Tool	139
6.3	Future Research	139
6.3.1	Tightening the Bounds	139
6.3.2	Model Extensions	140
	Appendix	146

List of Figures

3.1	Queueing network model graphical notation	19
4.1	Screen dump of analysis tool interface	81
4.2	Components of the bounds analysis tool	82
4.3	Java packages of bounds analysis tool interface	83
4.4	Classes and inheritance in the HTK package	84
4.5	Model/View/Control abstractions and interactions	87
4.6	Classes and inheritance in the MVC package	90
4.7	Classes and inheritance in the QNB application	95
4.8	Bounds analyzer Prolog data structures	101
4.9	The class/5 data structure	103
4.10	The service/7 data structure	104
5.1	Automatically generated symbolic expressions	109
5.2	Telemarketing System queueing network model	113
5.3	Telemarketing client-server bounds and simulation results	116
5.4	Feasible throughput region for telemarketing system	120
5.5	Model for two-CPU server configuration of Telemarketing System	121
5.6	Single and dual CPU results for telemarketing system	122
5.7	Results for non-preemptive priority dual CPU server	124
5.8	Results for non-preemptive priority dual CPU server, priorities reversed	126
5.9	A more complex queueing network model.	127
5.10	Test programming demonstrating loose bounds	130
5.11	Interval replication and bounds looseness	131
5.12	Two CPUs modeled by a resource type with two instances	132
5.13	Two CPUs modeled by two distinct but identical resources	133

List of Tables

3.1	Summary of modeling terms	20
5.1	Client-server Ethernet parameters	115
5.2	Bound solution time versus simulation time for telemarketing system. . . .	119
5.3	Visit count and service demand parameters	128
5.4	Throughput bounds and primary resource scheduling	128
5.5	Results for replicated and non-replicated primary resources	131

List of Terms

N_c	The number of customers in class c . $c = 1 \dots C$	17
M_r	The number of instances of primary resource r	18
M_k	The number of instances of secondary resource k	18
FIFO	First is first out scheduling	18
NPP	Non-preemptive priority scheduling	18
PP	Preemptive priority scheduling	18
PS	Processor sharing scheduling	18
Z_c	Mean think time per cycle of customers in class c	18
$V_k(c_0)$	Mean number of direct visits a customer in class c makes to any resource k . $k = R + 1 \dots K$	18
$S_k(c_0)$	Mean execution demand for a direct visit of a customer in class c to any resource k	18
$V_k(c_r)$	Mean number of indirect visits a customer in class c makes to a secondary resource k while holding primary resource r . $r = 1 \dots R$	18
$S_k(c_r)$	Mean service demand for an indirect visit of a customer in class c makes to secondary resource k while holding primary resource r	18
$W_k(c_0)$	Expected waiting time for a direct visit of a customer in class c to any resource k	18
$W_k(c_r)$	Expected waiting time for an indirect visit of a customer in class c to secondary resource k while holding primary resource r	18
T_c	Expected time taken by a customer in class c to complete one full cycle	18
f_c	Average throughput of a single customer of class c . $f_c = 1/T_c$	18

X_c	The average throughput of class c given by $X_c = N_c f_c$	19
X_c^-	The lower bound on X_c	19
X_c^+	The upper bounds on X_c	19
$N_m(c)$	The number of customers in class m that can contend with a customer in class c	24
$N_n(c, m)$	The number of customers in class n that can be in contention with a particular class m customer that is delaying a particular class c customer	24
$Q_r(c_0, m_0)$	The average number of times that a class m customer delays a particular class c customer directly on each direct visit to resource r by the class c customer	24
$d_k(c_x, m_y)$	The average delay experienced by a class c customer due to a class m customer on each visit to resource k by the class c customer	24
$d_k(c_r, m_r, n_x)$	The average delay experienced by a class c customer per visit of primary resource r due to a class m customer holding r and blocked at secondary resource k while a class n customer visits k directly ($x = 0$) or indirectly ($x = 1 \dots R$)	24
$\mathcal{P}$	The set of primary resources having either preemptive priority (PP) or non-preemptive priority (NPP) scheduling	24
$\mathcal{S}$	The set of secondary resources having scheduling policies other than preemptive or non-preemptive priority scheduling	24
$\mathcal{L}_k(c)$	The set of classes with <i>lower</i> priority at k than a given class c	25
$\mathcal{E}_k(c)$	The set of classes with <i>equal</i> priority at k than a given class c	25
$\mathcal{H}_k(c)$	The set of classes with <i>higher</i> priority at k than a given class c	25
$\Psi_r(c_0, m_0)$	The expected delay experienced by a customer of class c on a single visit to resource r due to a contending customer of class m visiting r	48

Chapter 1

Introduction

Performance bounds analysis of multiclass queueing networks is important for understanding the performance of concurrent systems. In this chapter, the motivation behind doing research on queueing network modelling with emphasis on performance bounds is given. This is followed by a brief description of the original contributions presented in this thesis and a short discussion on the limits to the scope of this work. Lastly, an outline of remainder of the thesis is provided.

1.1 Motivation

As software systems grow larger and more complex, the ability to make models of high level and detailed designs as well as existing systems becomes increasingly important. Models of software systems aide in the understanding and evaluation of designs and existing systems, helping designers make the right engineering decisions to produce or change implementations so that the given requirements may be met. In many situations the performance of the system in terms of its ability to complete a certain amount of work in a limited period of time is critical. This is particularly true in real-time systems such as telephone switches or in client server systems as in the case of Automated Teller Machines connected to a bank's central accounts database, for example. The cost of fixing performance problems after the

completion of either a new system or after making a significant change to an existing system can be so high that the project runs the risk of being cancelled. For this reason, performance analysis using models is needed to help designers predict whether a system will be successful in terms of its performance requirements.

1.1.1 Queueing Network Modelling

Both simulation and analytic models can be used for performance analysis. Due to the large cost associated with simulation, analytic modelling is often preferred in many situations such as during high level system design, in capacity planning, and during system upgrading. Queueing Network modelling is a well known analytic technique used for the performance evaluation of computer systems and communication networks (see [24] for example). Techniques for using multiclass queueing networks in performance evaluation of client-server systems are presented in [37].

Standard algorithms exist for the exact solution of multiclass queueing networks. A number of assumptions about the stochastic behavior of the system are required however for their deployment. These include:

1. independent routing decisions on departure from a queueing centre.
2. all classes have exponential service distributions with the same mean service time at a queueing centre that uses FIFO service discipline.
3. no priorities be associated with client classes.

As pointed out in [35], these assumptions may not be met in many real systems. Routing may be deterministic on certain systems making assumption (1) invalid. Traffic from a file server has been observed to be dominated by two or three packet sizes [38]. Service times may also vary quite widely depending on the nature of service requested from a server. Assumption (2) is violated in both of these situations. In certain systems a given client class may have priority over others, invalidating assumption (3). Multiclass queueing networks with priorities are discussed in [4] but only approximate mean residence times

are computed. Approximate analyses such as by Reiser and Lavenberg [50], by Stuck and Arthurs for computer networks [53] and by Jenq, Kohler, and Towsley for database systems [21] are available for systems that do not satisfy the assumptions. But the errors associated with the solution are often unknown. Moreover in certain systems (see [37] for example) the largest acceptable value for mean response time is specified making the approximate analysis unusable.

1.1.2 Queueing Network Bounds

Bounds analysis for queueing networks, as has been considered by Majumdar and Woodside in [35], handles many of these problems. In their bounds model, rather than computing an average value for an attribute of interest, such as customer throughput, the upper and lower bounds for the quantity are evaluated instead. The underlying theory for these bounds calculations does not depend on the strong assumptions indicated in Section 1.1.1. Furthermore, the lower bound provides a performance guarantee that is important in the design of conservative systems (as in the case of those systems where the largest acceptable value for mean response time is specified).

Due to the relaxed assumptions on which it is built and the performance guarantees provided by the lower bounds, queueing network bounds analysis is important in the performance modelling of software systems. Also important then are extensions to this modelling technique that improve its flexibility allowing it to be used in a wider range of applications. In [36], Majumdar and Woodside consider systems that can be modelled by multiclass queueing networks where the queueing centres may have scheduling disciplines other than the usual first-in first-out scheduling, including priority scheduling and processor sharing.

1.1.3 Simultaneous Resource Possession

Up to this point, bounds on conventional multiclass queueing networks have been discussed. More complex models such as queueing networks with “simultaneous resource possession” [19] are required in many situations. Consider a database system for example. Locks are

often associated with database records: Both the lock as well as the device containing the record are simultaneously used by a client transaction. In a multithreaded server on a client-server system, a client request has to first queue up for a thread from an available pool of threads and then obtain service from the server. Similarly, in a multiprocessor-based server, both the processor and the server software need to be modeled as queueing centres in the queueing network model for the system. Both the processor as well as the server process are “held simultaneously” by a client while receiving service. Simultaneous resource possession also occurs when modeling logical resources such as semaphores providing mutual exclusion for the code in a critical section used by concurrent processes. In certain client-server systems (described in [16]), a client request can engage multiple servers which are nested in a multilayered architecture. In the I/O subsystem used in a server, both the disk as well as the I/O channel may be held during the service of a request.

A theory for extending the work of [35] to handle systems with a single layer of shared resources with FIFO queueing discipline has been proposed by Majumdar (but not implemented) [29].

1.2 Contributions

This thesis begins with the bounds computation technique for queueing networks with simultaneous resource possession proposed by Majumdar [29] and extends it to include non-preemptive priority scheduling at shared resources as well as preemptive, non-preemptive, and processor sharing scheduling policies at unshared resources. A complete formal mathematical derivation of this extended queueing network model is presented in this thesis.

The bounds analysis software that computes the upper and lower throughput bounds for the extended queueing network model was developed for this thesis. This software is based on earlier software, previously developed by Hubbard, for computing the bounds models described by [35] and [36]. Due to the nature of the changes required to support simultaneous resource possession, the earlier solver software had to be nearly completely rewritten. The new software consists of approximately 4000 lines of CLP(BNR).

A graphical user interface written in Java and based on a variant of the well-known Model View Control application framework [48] was developed for this thesis to provide an iconic interface to the bounds analysis engine. The graphical tool is composed of a generic framework and some application specific code. The generic graphical tool framework consists of approximately 11000 lines of Java and is suitable for use in the development of other iconic tools. The application specific code that makes this tool specific to the queueing network model graphical language consists of approximately 3400 lines of Java code.

The validity of the bounds analysis technique presented in this thesis was demonstrated through two case studies. In one case study, a client-server system is modelled in a capacity planning exercise as various hardware and software architecture design options are explored. The other case study further demonstrates the ability of the extended model to handle systems with shared resources and various scheduling disciplines.

In [31] and [32], Majumdar and Hubbard present the simultaneous resource possession extensions to [35] (but not the scheduling extensions) including a demonstration of the bounds analysis technique through a version of the client-server case study previously mentioned.

1.3 Scope

Although the technique presented here is useful for analyzing a variety of different systems, it is not applicable to systems that exhibit high variances in services times. The reason for this is related to an assumption about the general type of probability distribution for service times at queueing centers in the model. The assumption underlying the bounds computation is discussed in detail in Chapter 3.

Only systems with zero or one layer of shared resources can be analyzed using the technique presented here. That is, only systems where a customer accesses unshared resources directly or accesses unshared resources while holding a single shared resource can be modelled. The work on stochastic rendezvous network bounds [34] does handle systems that have more than a single layer of shared resources but are limited to a single class of

customer. Future research could consider multiclass systems in which the customer can consecutively access and hold an arbitrary number of shared resources.

The implementation of the bounds analysis engine is dependent on use of the CLP(BNR) environment found in BNR Prolog or ALS Prolog. Certain inequalities defined by the model are complicated to solve in conventional ways. Thus solving the models without using CLP(BNR) or some other CLP system could prove to be unreasonably difficult.

1.4 Outline

Chapter 2 describes the previous work in the field of bounds analysis of queueing network models. Brief descriptions of relational interval arithmetic and constraint logic programming are also given. In Chapter 3, a complete formal derivation of the inequalities comprising the queueing network bounds analysis is given. The derivation begins with the upper bounds inequalities and then considers the lower bounds inequalities in a series of sections covering the various different combinations of scheduling policies at shared and unshared resources. Chapter 4 describes the design and function of both the bounds analysis engine and the graphical user interface. In addition to the two case studies mentioned above, Chapter 5 also describes the methods by which the bounds analysis software was tested and validated. Conclusions and ideas for future research are presented in Chapter 6.

Chapter 2

Queueing Networks and Interval Analysis

2.1 Introduction

In this chapter, the topics of queueing network bounds analysis and interval arithmetic based constraint logic programming are described and the relevant work that has led to the state of the art in the areas is summarized.

Although much important research work has been done in the area of queueing network modelling, only the research that has led to the development of the bounds analysis model presented in this thesis is mentioned. As with queueing modelling, only a representative selection of the work on interval analysis is mentioned here. However, the underlying principles of interval analysis are described in more detail to provide some familiarity with the issues involved in the implementation of the model that is presented in later chapters.

2.2 Queueing Network Bounds Analysis

Bounds analysis for queueing networks is considered by a number of researchers. A substantial literature attests to the importance of the topic: only a representative set of work

is mentioned here.

2.2.1 Single Class Queueing Networks

Previous research began with single class queueing networks. Muntz and Wong derived *asymptotic* bounds applicable to separable and non-separable queueing networks [40]. Balanced Job Bounds that are tighter than the asymptotic bounds but applicable to separable queueing networks only were developed by Zahorjan *et. al.* [59]. Eager and Sevcik have proposed a hierarchy of bounds with different accuracy levels and computational complexity for separable single class networks [12]. Bound hierarchies that offer a tradeoff between computational cost and accuracy are also considered by Kriz [23] and Suri [54], who base their work on the mean value analysis recursions, and by Stephens and Dowdy [52] whose work is based on the convolution algorithm recursion.

2.2.2 Multiclass Queueing Networks

Bounds analysis for multiclass queueing networks have also been considered in past work. For example, Zahorjan *et. al.* derived balanced job bounds in [59] and Eager and Sevcik developed hierarchies of bounds in [13]. Also, a Composite Bound Method was outlined by Kerola in [22]. Extension of the well-known upper bounds on throughput for single class networks to multiclass queueing nets are considered by a number of researchers (see for example [35] and [36]). Lower bounds on throughput for multiclass queueing networks are derived in [35] and [36]. The first paper handles FIFO queueing centres only whereas processor sharing and priority scheduling are also considered in the second. Asymptotic analysis of closed multiclass queueing networks is also considered by Balbo and Serazzi in [2] in which asymptotic expressions for performance measures such as throughput are obtained for product-form networks with multiple bottleneck resources.

2.2.3 Rendezvous Networks

In addition to the work on bounds analysis of general multiclass queueing networks, research has also been done on bounds analysis of rendezvous networks [34]. The Stochastic Rendezvous Network Model [57] of Woodside, et. al. is a model based on queueing networks but which takes into consideration some special properties of networks having components that communicate through *rendezvous*. These properties are that components are arranged in layers, the top most layer being a layer of pure clients and the bottom most layer being a layer of pure servers. The middle layers are then both servers to higher layers and clients to lower layers. Due to the blocking nature of the rendezvous style of communication that exists between a component fulfilling the role of a client and a second component fulfilling the role of a server, the client component waits for a response from the server component. After replying, a server component may do some independent processing before it becomes ready to handle another request. This independent processing is called Phase 2 processing by Woodside *et. al.* The notion that a server may be busy in its role as a client of a lower layer and hence be unable to handle requests from upper layers is similar to the concept of simultaneous resource possession discussed in this thesis. This server blocking and the phase 2 processing make rendezvous networks difficult to model using conventional techniques. A bounds approach gives similar advantages for rendezvous networks as is the case for general queueing networks. A tool was developed for solving rendezvous network bounds and is reported in [14].

2.3 Simultaneous Resource Possession

Previous work in the area of modelling systems that involve simultaneous resource possession include techniques that are devised for modeling specific I/O subsystems and are reported in [55, 58]. More general approximate solutions, that may be applicable to a more varied class of queueing network models are reported in [19]. None of these papers provide a method for computing performance guarantees that is described in this thesis.

2.4 Constraint Logic Programming and Interval Analysis

The bounds inequalities derived in this thesis are difficult to solve in closed form due to their complexity and non-linearity. The techniques of constraint logic programming and interval analysis provide a way of solving these inequalities that is natural and declarative making the solutions much easier to implement and verify.

Constraint logic programming consists of the marriage of the concepts of *logic programming*, as embodied in the family of Prolog and Prolog-like languages, and the mathematical system known as *interval arithmetic*. The following sections describe how these concepts work together to provide a very powerful programming environment for solving mathematical models.

2.4.1 Interval Arithmetic

A general introduction to interval arithmetic is given first. The significance of interval arithmetic in the context of performance bounds computations is provided in Section 2.4.3.

When computers are used to perform computations on real numbers, the real numbers are represented in the program using some form of finite approximation such as the standard IEEE single precision or double precision floating point representation. Rarely do these representations exactly correspond to the real number they represent. Most often the approximation is either too high or too low and it is usually impossible to predict which.

Suppose that instead of representing a theoretical real number by a single finite precision floating point number, two finite precision floating point numbers are used where one approximate value is less than the theoretical real number and the other approximate value is greater than the theoretical real number. This pair of floating point approximations form an *interval* containing the theoretical real number. If care is taken to ensure that as a result of any mathematical operation, the lower bound of the interval is always less than or equal to the theoretical real number and the upper bound is always greater than or equal to the theoretical real number, then when the result of the computation is obtained, the theoretical result will be contained in an interval and the error on the result will be known.

The system of mathematical operations on intervals such as addition, subtraction, and the like, in which the upper and lower bounds are treated correctly to maintain the theoretical value within the interval bounds is known as *interval arithmetic* and was first used in a practical way by Ramon Moore in 1966 [39].

The following are some examples of elementary algebraic operations on intervals demonstrating how the bounds are handled to ensure that the final interval contains the resulting theoretical real value. Let the notation $[x^-, x^+]$ be used to represent an interval X where x^- is the lower bound and x^+ is the upper bound. Assume for now that x^+ and x^- are bound values with infinite precision. Then if A and B are two such intervals, then the following equations define the addition, subtraction, multiplication, and division operations over the interval domain.

Addition: The rule for addition is the following:

$$[a^-, a^+] + [b^-, b^+] = [a^- + b^-, a^+ + b^+]$$

The addition of intervals is like vector addition in the plane and the rules of commutativity and associativity hold (for infinite precision).

Subtraction: The rule for subtraction is the following:

$$[a^-, a^+] - [b^-, b^+] = [a^- - b^+, a^+ - b^-]$$

Note that subtraction *is not* the inverse of addition as the following expansion of $A + B - B$ demonstrates:

$$[a^-, a^+] + [b^-, b^+] - [b^-, b^+] = [a^- + b^- - b^+, a^+ + b^+ - b^-] \neq [a^-, a^+]$$

Thus, $A + B - B \neq A$.

Multiplication; The rule for multiplication is the following:

$$[a^-, a^+] \times [b^-, b^+] = [\min(a^-b^-, a^-b^+, a^+b^-, a^+b^+), \max(a^-b^-, a^-b^+, a^+b^-, a^+b^+)]$$

It can be shown that multiplication is commutative and associative (in infinite precision) but that the distributive law is weakened to be:

$$X(Y + Z) \supset XY + XZ$$

where X , Y , and Z are intervals and the relation $\supset$ indicates that the interval on the left hand side is included in the interval on the right. This weakness in the distributive law results in tighter bounds for product over sum expressions than for sum over product expressions.

Division; If $0 \notin [b^-, b^+]$ then the rule for division is the following:

$$[a^-, a^+] \div [b^-, b^+] = [a^-, a^+] \times [\frac{1}{b^+}, \frac{1}{b^-}]$$

It can be seen that unless X is a point interval (that is, $x^- = x^+$), X/X is not 1.

It can be seen from the above examples that there are some fundamental differences between arithmetic over the reals and interval arithmetic: subtraction is not the inverse of addition, multiplication is not distributive, and division of an interval by itself is not, in general, 1. Note that the above examples assume infinite precision for the bounds. In practical interval arithmetic, intervals must be represented by finite precision floating point numbers and so, to preserve the validity of the bounds of an interval during a calculation, outward rounding (the lower bound rounded downwards and the upper bound rounded upwards) is required. However, the associative laws of addition and multiplication are lost as a result of outward rounding. Thus in practical interval arithmetic, most of the axioms of the real domain no longer hold and, hence, most of the algebraic derivations which are formally justified in the domain of the reals are not justified in the domain of intervals

making mathematical analysis very difficult.

2.4.2 Relational Interval Arithmetic

In *relational interval arithmetic*, the advantages of classical interval arithmetic are retained and the problems of the failure of fundamental axioms are eliminated. This is accomplished by treating the operations on the intervals as a network of *relations* constraining the constituent intervals. As constraints are added to the network, the effects of the new constraints are propagated through the entire network as a narrowing of the intervals. This is similar to the Prolog concept of unification. If a constraint is inconsistent with its predecessors, that constraint is said to “fail” and backtracking occurs to the last Prolog choice point. This backtracking undoes any narrowing that had taken place as a result of the constraint that failed.

The key points to relational interval arithmetic are that relations are used instead of functions and that narrowing operators are applied everywhere. Take the following sample of CLP(BNR) code:

```
[V, D, T]:real, {D = V*T}
```

This line of code declares three interval variables, velocity (V), distance (D), and time (T), and applies a *constraint relation* equating the distance to the product of velocity and time. Even though the intervals V, D, and T have not been given specific values, the relationship applied to them holds. In subsequent CLP(BNR) statements, values may be given to *any* of the three components and the others will be constrained — their intervals narrowed — to maintain the “truth” of the constraint. This single constraint acts like three classical interval arithmetic functions all held to be simultaneously true:

$$\begin{aligned} V &= \frac{D}{T} \\ D &= VT \\ T &= \frac{D}{V} \end{aligned}$$

Any time that a change is made to any of the interval variables a narrowing operation takes place to make all the intervals as narrow as possible but still formally satisfying constraints in the constraint network.

The underlying concept *relational interval arithmetic* was introduced by John Cleary in [6] as a means of seamlessly integrating mathematics with the Prolog programming language. The concept was first fully developed in BNR Prolog in 1987 [8, 9]. An excellent discussion of relational interval arithmetic in the context of CLP(BNR) is given in [47]. Other papers describing CLP(BNR) are [46, 44, 45]. The CLP(BNR) system is also available commercially as part of the ALS Prolog distribution [1].

Other constraint logic systems include CLP $\mathbb{R}$, [20], Prolog-III [7], CHIP [11], and the system described by Hyvonen in [18]. CLP $\mathbb{R}$ dealing with reals and Prolog-III with rationals use symbolic solving and term rewriting techniques rather than the relational interval arithmetic — primarily a numeric technique — of CLP(BNR). CHIP dealing with integer domains and the system described by Hyvonen for reals use local propagation techniques similar to that of relational interval arithmetic. In [17], Hickey discusses CLP(F), a constraint logic programming systems that is written in CLP(BNR) to provide an enhanced programming interface for dealing with interval related problems in mathematics and the natural sciences. Another interval constraint solving system, UniCalc [51], manages systems of equality and inequality constraints through a text-based windowing interface. Some simple queueing network bounds were computed using UniCalc but since it does not provide a general purpose programming interface its use was not explored further.

2.4.3 Interval Analysis in Performance Evaluation

Interval analysis has been successfully used in research with queueing network models. In her M.Eng. thesis [49], Revathy Ramadoss explored the feasibility of using interval techniques in the solution of existing well-known models of computing systems with interval parameters including single and multi-class queueing networks and other more complex models. Some of the results are described in [33]. Sensitivity analysis and performance bot-

tleneck identification using interval inputs were presented. Majumdar, Lüthi and Haring [26], and also Kotsis [28, 27] discuss the use of histograms as inputs to performance models when there are parameter variabilities or uncertainties. These histograms are composed of groups of intervals and associated probabilities. The performance results reported by their modelling technique are also in the form of histograms. And, as indicated above, interval analysis has been used in queueing network bounds analysis [34, 56, 35, 36]. A survey of performance analysis research that uses interval arithmetic is presented in [30].

Some other interesting applications of constraint logic programming and relational interval arithmetic have been made. For example, Older has used CLP(BNR) to solve problems in non-linear optimization [43], in X-Ray crystallography [41], and in differential equations [42]. Work on using interval analysis on ordinary differential equations was also done by Hickey [17].

Chapter 3

The Model and The Bounds

3.1 Introduction

In this chapter, the queueing network model and technique for the computation of bounds are formally presented. The graphical notation and mathematical symbols of the model are defined first, in Section 3.2, and the formulae for the computation of the throughput bounds are derived in the remaining sections. A general expression for the cycle time of a customer, derived from the cycle time expression of [36], is given in Section 3.3. This is followed by the derivation of two inequalities that upper bound the class throughputs: in Section 3.4, the no-contention bound of [36] is extended to handle shared resources; in Section 3.5, the utilization based bound of [36] is similarly extended. This is followed by the derivation of lower bound which is broken down into smaller topics. The general expressions for lower bounds are given in Section 3.6, the bound on the waiting time at a secondary resource is given in Section 3.7, the bound on the waiting time at a primary resource with *first in, first out* scheduling is given in Section 3.8, the bound on the waiting time at a primary resource with priority scheduling is given in Section 3.9 and a summary of the lower bound expressions is given in Section 3.10. A discussion of the derivation is presented in section 3.11.

3.1.1 Minimal Stochastic Assumptions

Some of the theorems and lemmas described in this thesis are based on derivations provided in [36].

Only the following assumptions enumerated in [36] are required for the validity of the bounds.

- The resource service demands are independent identically distributed (i.i.d.) with any arbitrary new better than used in expectation (NBUE) distribution.
- The number of visits made by a customer to a device per cycle is random with known mean. Any routing strategy that has this property is possible.

A resource service demand with NBUE distribution [3] implies that at the time that a customer arrives at the resource for service, the mean time remaining to complete the work already in progress is less than or equal to the mean demand at the resource for the service in progress. Although the technique presented here is apt for analyzing a variety of different systems, the assumption regarding the service time distributions restricts its use to systems that do not exhibit high variances in service times.

All the model parameters discussed below (service demands, number of visits, waiting times, throughputs, etc.), unless indicated otherwise, are expected values.

3.2 The Model

The queueing network model components, which are classes of customer, primary resources, secondary resources, and visits, are simple abstractions that model both the structure and the average steady-state behavior of a system. The customer classes are numbered $c = 1 \dots C$ and there are N_c customers in class c .

The processes, threads or other active devices in the system such as locks are represented by *primary resources* in the model. *Secondary resources* represent hardware devices such as CPU's, disks, and networks, or hardware-software combinations modelled as single

resources. A primary resource is “held” while one or more secondary resources are used. The primary resources are labeled $r = 1 \dots R$ and the secondary resources are labeled $k = R + 1 \dots K$. The factors M_r and M_k indicate the number of multiple instances of a given primary resource r and a given secondary resource k , respectively. The multiple instances of a given resource are identical and independent with a separate queue for each instance. Primary resources can have either *first in first out* (FIFO) scheduling or *non-preemptive priority* (NPP) scheduling. Secondary resources may have *preemptive priority* (PP) scheduling and *processor sharing* (PS) scheduling in addition to FIFO and NPP. Class priorities can be specified separately for each resource.

Customers behave cyclically. Each cycle of a class c customer is characterized by a *think time* of an average duration Z_c and *visits* to resources. Customer think time models the time the customer spends each cycle that does not involve the use of any primary or secondary resource. This can be some computation local to the customer or the average time delay between the commands generated by a human user, for example. The symbols $V_k(c_0)$ and $S_k(c_0)$ are used to refer to the mean number of *direct* visits and mean execution demand respectively for a visit of a class c customer to the secondary resource k . Similarly, the symbols $V_k(c_r)$ and $S_k(c_r)$ refer to the mean number of visits and mean execution demand per visit respectively for a visit of a class c customer to the secondary resource k while the customer holds the primary resource r . This is called an *indirect* visit. If a class c does not visit a resource k directly then $V_k(c_0) = 0$. similarly, $V_k(c_r) = 0$ signifies that class c customers make no indirect visits to secondary resource k while holding primary resource r .

On each visit that a customer makes to a resource, there is a potential for the customer to wait while some other customer is using the resource. This waiting time or queueing delay experienced by a class c customer is represented by the symbols $W_k(c_0)$ and $W_k(c_r)$ referring to direct and indirect visits to k respectively.

The mean cycle time of a class c customer is given the symbol T_c . the throughput f_c

of a single customer of class c is related to the class cycle time by Little's law [25]:

$$f_c = \frac{1}{T_c} \quad \text{for all } c = 1 \dots C \quad (3.1)$$

The throughput X_c for class c is the combined throughputs of all the customers in the class and is given by:

$$X_c = N_c f_c \quad \text{for all } c = 1 \dots C \quad (3.2)$$

A summary of the model terminology is given in table 3.1. Figure 3.1 shows the graphical notation used to describe a model. Although, only one primary resource and two secondary resources are shown in the figure, there is no theoretical restriction to the number of the resources that can be handled by the model.

Figure 3.1: A description of the graphical notation for the queueing network model.

Table 3.1: A summary of the modeling terminology.

<i>Term</i>	<i>Description</i>
N_c	The number of customers in class c . $c = 1 \dots C$
Z_c	Mean think time per cycle of customers in class c
$V_k(c_0)$	Mean number of direct visits a customer in class c makes to any resource k . $k = R + 1 \dots K$.
M_k	The number of instances of server k .
$S_k(c_0)$	Mean execution demand for a direct visit of a customer in class c to any resource k
$W_k(c_0)$	Expected waiting time for a direct visit of a customer in class c to any resource k
$V_k(c_r)$	Mean number of indirect visits a customer in class c makes to a secondary resource k while holding primary resource r , $r = 1 \dots R$
$S_k(c_r)$	Mean service demand for an indirect visit of a customer in class c makes to secondary resource k while holding primary resource r
$W_k(c_r)$	Expected waiting time for an indirect visit of a customer in class c to secondary resource k while holding primary resource r
T_c	Expected time taken by a customer in class c to complete one full cycle.
f_c	Average throughput of a single customer of class c . Note that $f_c = \frac{1}{T_c}$
X_c	The average throughput of class c given by $X_c = N_c f_c$
X_c^- and X_c^+	the lower and upper bounds on X_c , respectively.

3.3 Class Cycle Time

The cycle time of a particular class of customers is composed of three components: the customer think time, the total delay due to the direct visits the customer makes to primary and secondary resources, $D_{c,\text{direct}}$, and the total delay due to the indirect visits the customer makes to secondary resources, $D_{c,\text{indirect}}$. The think time for a customer of class c is Z_c . The total delay due to all direct visits is the sum of the direct delays experienced at all the resources visited by the customer. This includes both service demand and queueing delays. Thus, recalling that $V_k(c_0) = 0$ for customers of class c that do not visit resource k directly, we have, for all classes $c = 1 \dots C$:

$$D_{c,\text{direct}} = \sum_{k=1}^K M_k V_k(c_0) (S_k(c_0) + W_k(c_0))$$

The total delay due to all indirect visits is the sum of the delays experienced at all the secondary resources visited indirectly by the customer. Recalling that $V_k(c_r) = 0$ for customers of class c that do not visit k indirectly through r , we have for all classes $c = 1 \dots C$:

$$D_{c,\text{indirect}} = \sum_{r=1}^R M_r V_r(c_0) \sum_{k=R+1}^K M_k V_k(c_r) (S_k(c_r) + W_k(c_r))$$

Combining these three components, a general expression for the expected class cycle time for a class c , $c = 1 \dots C$, is obtained:

$$\begin{aligned} T_c = & Z_c + \sum_{r=1}^R M_r V_r(c_0) \left[S_r(c_0) + W_r(c_0) + \sum_{k=R+1}^K M_k V_k(c_r) (S_k(c_r) + W_k(c_r)) \right] \\ & + \sum_{k=R+1}^K M_k V_k(c_0) (S_k(c_0) + W_k(c_0)) \end{aligned} \quad (3.3)$$

3.4 No Contention Upper Bound

A lower bound on the class cycle time and hence an upper bound on the class throughput is obtained as in [36] by assuming that there is no contention between customers for resources.

That is, for all classes and all resources, the waiting times $W_k(c_0)$ and $W_k(c_r)$ are zero. Eliminating these terms from equation 3.3 yields an inequality for the lower bound on the cycle time for each class $c = 1 \dots C$:

$$T_c \geq T_c^- = Z_c + \sum_{r=1}^R M_r V_r(c_0) \left(S_r(c_0) + \sum_{k=R+1}^K M_k V_k(c_r) S_k(c_r) \right) + \sum_{k=R+1}^K M_k V_k(c_0) S_k(c_0) \quad (3.4)$$

The upper bound on class throughput is derived from this using equations 3.1 and 3.2 giving for all classes $c = 1 \dots C$:

$$X_c \leq X_c^+ = \frac{N_c}{T_c^-} \quad (3.5)$$

where T_c^- is given by equation 3.4.

3.5 Utilization Upper Bound

A second set of expressions limiting the upper bound on class throughputs is obtained as in [36] by considering the utilization of each resource in the system. For each resource, the amount of work to be done at that resource is limited by the amount of time available to do the work. No resource can be used more than 100%.

A primary resource is busy during a direct visit by a customer as well as when it is held by a customer that is making indirect visits to secondary resources. The total work at primary resource r due to direct visits to r by a customer class m is given by $X_m V_r(m_0) S_r(m_0)$. The total time during which the primary resource r is held by the customer class for indirect visits is given by $\sum_{k=R+1}^K X_m V_r(m_0) M_k V_k(m_r) S_k(m_r)$. The utilization of the primary resource is obtained by summing these two quantities over all classes $m = 1 \dots C$. Thus, for all primary resources $r = 1 \dots R$, we have:

$$\sum_{m=1}^C X_m V_r(m_0) \left(S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) S_k(m_r) \right) \leq 1 \quad (3.6)$$

A secondary resource is busy during both direct and indirect visits by a particular customer. The total work done due to direct visits to a secondary resource k from a customer class m is given by $X_m V_k(m_0) S_k(m_0)$. The total work at the secondary resource by all indirect visits of the customer class is given by $\sum_{r=1}^R X_m V_r(m_0) V_k(m_r) S_k(m_r)$. The utilization of the secondary resource is then the summation of these quantities over all classes. As with the primary resources, the utilization must be less than or equal to 1. Thus for all secondary resources $k = R + 1 \dots K$, we have:

$$\sum_{m=1}^C X_m \left(V_k(m_0) S_k(m_0) + \sum_{r=1}^R M_r V_r(m_0) V_k(m_r) S_k(m_r) \right) \leq 1 \quad (3.7)$$

3.6 Lower Bound on Throughput

The lower bound on throughput for a class is determined by computing the upper bound on the class cycle time. This worst case class cycle time is based on the worst case waiting time $W_r^+(c_0)$, $W_k^+(c_r)$, and $W_k^+(c_0)$ at each resource. The expression for the worst case cycle time for a class c customer is obtained from equation 3.3:

$$\begin{aligned} T_c \leq T_c^+ &= Z_c + \sum_{r=1}^R M_r V_r(c_0) \left[S_r(c_0) + W_r^+(c_0) \right. \\ &\quad \left. + \sum_{k=R+1}^K M_k V_k(c_0) (S_k(c_r) + W_k^+(c_r)) \right] \\ &\quad + \sum_{k=R+1}^K M_k V_k(c_0) (S_k(c_0) + W_k^+(c_0)) \end{aligned} \quad (3.8)$$

The lower bound on class throughput is derived from this using equation 3.1 and 3.2 so that for all classes $c = 1 \dots C$, we have:

$$X_c \geq X_c^- = \frac{V_c}{T_c^+} \quad (3.9)$$

In the subsequent sections the expressions for the waiting times are derived. Additional terminologies required for the purpose are introduced first.

Let $N_m(c)$ be the number of customers in class m that can contend with a customer in class c . The maximum number of class c customers that can contend at a resource with a particular customer in the same class c is $N_c - 1$ since the tagged customer cannot contend with itself. Defining $\delta_{xy} = 1$ for $x = y$ and 0 otherwise, we have $N_m(c) = N_m - \delta_{mc}$. In a similar way, consider the case of contention for indirect visits. Suppose a customer of class c is delayed at a primary resource by a customer of class m that is holding that primary resource. Suppose further that the class m customer is delayed at a secondary resource by customer of class n . Let $N_n(c, m)$ be the number of customers in class n that can be in contention with the class m customer that is delaying the class c customer in this way. Since a customer can never contend with itself, if $n = m$ then the number of class n customers must be $N_n - 1$. also, if $n = c$ then again the number of contenders is $N_n - 1$. Finally, if $n = m = c$ the the number of contenders is $N_n - 2$. This can be written as $N_n(c, m) = N_n - \delta_{nm} - \delta_{nc}$.

When a class c customer makes a visit to primary resource r it may be delayed by a class m customer in service or in queue at resource r . The quantity $Q_r(c_0, m_0)$ is defined to be the average number of times that the class m customer delays the class c customer directly on each direct visit to resource r by the class c customer.

When a class c customer makes a visit to a secondary resource k it may be delayed by a class m customer in service or in queue at resource k . The quantity $d_k(c_x, m_y)$ is defined to be the average delay experienced by the class c customer due to the class m customer on each visit to resource k by the class c customer. The indices x and y are in the range $0 \dots R$ indicating direct visits (x or y is zero) or indirect visits (x or y in the range $1 \dots R$) for c or m . Similarly, the term $d_k(c_r, m_r, n_x)$ is the average delay experienced by a class c customer blocked at primary resource r due to a class m customer holding r and blocked at secondary resource k while a class n customer visits k directly ($x = 0$) or indirectly ($y = 1 \dots R$) for each visit of c to r .

The term $\mathcal{P}$ refers to the set of primary resources having either preemptive priority (PP) or non-preemptive priority (NPP) scheduling. Similarly, the term $\mathcal{S}$ refers to the set

of secondary resources having either preemptive or non-preemptive priority scheduling. For any primary or secondary resource $k \in \{\mathcal{P} \cup \mathcal{S}\}$, the set of classes with *lower* priority at k than a given class c is denoted by $\mathcal{L}_k(c)$. Similarly, $\mathcal{E}_k(c)$ and $\mathcal{H}_k(c)$ are the sets of equal priority classes and higher priority classes at k compared to c , respectively.

3.6.1 Waiting Time at a Secondary Resource

A customer of class c can be delayed at a secondary resource k by any customer of class m that visits the resource either directly or indirectly. Now m can take on any value from 1 to C . For each class $m = 1 \dots C$, the average amount of delay the class c customer experiences is given by the product of the number of potential class m contenders, $N_m(c)$, and the expected delay due to each class m contender at k experienced per visit by the class c customer. There are four distinct cases for which the average per visit delay is required: the class c and class m customers visits k directly ($d_k(c_0, m_0)$), c visits directly while m visits indirectly ($d_k(c_0, m_{r'})$), c visits indirectly but m visits directly ($d_k(c_r, m_0)$), and both c and m visit k indirectly ($d_k(c_r, m_{r'})$, $r' \neq r$). Thus, the average secondary resource waiting time for direct visits of a class c customer to secondary resource k may be written:

$$W_k(c_0) = \sum_{m=1}^C N_m(c) \left(d_k(c_0, m_0) + \sum_{r'=1}^R M_{r'} d_k(c_0, m_{r'}) \right) \quad (3.10)$$

And the average secondary resource waiting time for an indirect visit of a class c customer to secondary resource k through any primary resource r , $r = 1 \dots R$, may be written:

$$W_k(c_r) = \sum_{m=1}^C N_m(c) \left(d_k(c_r, m_0) + \sum_{r'=1: r' \neq r}^R M_{r'} d_k(c_r, m_{r'}) \right) \quad (3.11)$$

3.6.2 Waiting Time at a Primary Resource

A customer of class c can be delayed at a primary resource r by any customer of class m , $m = 1 \dots C$. The duration of this delay depends on the following:

1. the service demands of the class m customer at primary resource r , $S_r(m_0)$

2. the service demands that class m customer requires at all the secondary resources it visits while holding the primary resource r . $\sum_{k=R+1}^K M_k V_k(m_r) S_k(m_r)$
3. the waiting times that the class m customer experiences at all the secondary resources it visits while holding the primary resource r at which the class c customer waits. This waiting time is given by $\sum_{k=R+1}^K V_k(m_r) W_k(m_r, c)$. where $W_k(m_r, c)$ is the queueing delay at k experienced per visit by the class m customer holding r at which c waits.

Given that $N_m(c)$ is the number of potential contending customers to the customer of class c and that $Q_r(c_0, m_0)$ is the number of visits that a customer of class m makes to resource r that actually delay the class c customer at r , the expression for the waiting time $W_r(c_0)$ at resource r for a customer of class c may be written as:

$$W_r(c_0) = \sum_{m=1}^C N_m(c) Q_r(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) (S_k(m_r) + W_k(m_r, c)) \right] \quad (3.12)$$

The waiting time $W_k(m_r, c)$ may be calculated by considering the delay at each secondary resource k for every customer that contends with the class m customer while the class m customer holds the primary resource r blocking the class c customer at r . For a contending customer of class n , $d_k(c_r, m_r, n_{r'})$ is the delay experienced by the class c customer per visit to resource r due to a class m customer holding r while being blocked at secondary resource k by a direct visit ($r' = 0$) or indirect visit ($r' = 1 \dots R$) of the class n customer. Recall that the number of possible contenders in class n to a customer in class m that is contending with a customer is class c is $N_n(c, m)$. The expression for $W_k(m_r, c)$ may then be written as:

$$W_k(m_r, c) = \sum_{n=1}^C N_n(c, m) \left(d_k(c_r, m_r, n_0) + \sum_{r'=1: r' \neq r}^R M_{r'} d_k(c_r, m_r, n_{r'}) \right) \quad (3.13)$$

where as indicated above, the first d_k term in the outermost summation, $d_k(c_r, m_r, n_0)$, represents the delay due to direct visits by the class n customer and the second d_k term, $d_k(c_r, m_r, n_{r'})$, represents delay due to indirect visits by the class n customer through the primary resource r' .

We get the upper bound on the delay at a primary resource r by substituting Equation 3.13 into Equation 3.12:

$$W_r(c_0) = \sum_{m=1}^C V_m(c) Q_r(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) + \sum_{n=1}^C V_n(c, m) \left(d_k(c_r, m_r, n_0) + \sum_{r'=1: r' \neq r}^R M_{r'} d_k(c_r, m_r, n_{r'}) \right) \right) \right] \quad (3.14)$$

Upper bounds on the waiting time expressions are derived in the next sections. These upper bounds are used in Inequality 3.8 to determine the upper bound on the class cycle time.

3.7 Waiting Time Bounds at Secondary Resources

In this section, the upper bound on the waiting times, $W_k^+(c_0)$, and $W_k^+(c_r)$ are derived for systems having secondary resources $k = R + 1 \dots K$ with FIFO, NPP, PP and PS scheduling. The scheduling policy at any primary resource r does not affect the waiting time at a secondary resource.

3.7.1 Secondary Resources with FIFO Scheduling

Theorem 1, the proof of which may be found in the appendix, gives the upper bounds on the average per visit delay due to contention for secondary resources with FIFO scheduling.

Theorem 1 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1.*

If the secondary resources indicated below have FIFO scheduling, then:

- (a) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_0) \leq \min(1, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) S_k(m_0)$$

- (b) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_{r'}) \leq \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) S_k(m_{r'})$$

- (c) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_0) \leq \min(1, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(m_0)$$

- (d) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_{r'}) \leq \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(m_{r'})$$

Proof: Consult the appendix.

Taking Equation 3.10 and applying statements (a) and (b) of Theorem 1, the upper

bound on the direct waiting time. $W_k^-(c_0) \leq W_k^+(c_0)$, is obtained:

$$W_k^+(c_0) = \sum_{m=1}^C N_m(c) \left(\min\left(1, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}\right) S_k(m_0) \right. \\ \left. + \sum_{r'=1}^R M_{r'} \min\left(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}\right) S_k(m_{r'}) \right) \quad (3.15)$$

Applying statements (c) and (d) of Theorem 1 to Equation 3.11, the upper bound on the secondary resource waiting time for an indirect visit of a class c customer to secondary resource k , $W_k^-(c_r) \leq W_k^+(c_r)$, may be written:

$$W_k^+(c_r) = \sum_{m=1}^C N_m(c) \left(\min\left(1, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}\right) S_k(m_0) \right. \\ \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \min\left(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}\right) S_k(m_{r'}) \right) \quad (3.16)$$

3.7.2 Secondary Resources with PP Scheduling

In this section, the upper bound on the waiting times $W_k^+(c_0)$, and $W_k^+(c_r)$ are derived for secondary resources ($k = R+1 \dots K$) with preemptive priority (PP) scheduling. Theorem 2, the proof of which may be found in the appendix, gives the upper bounds on the average per visit delays due to contention for PP secondary resources.

Theorem 2 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have PP scheduling, then:*

- (a) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the*

following inequality:

$$d_k(c_0, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_k(c_0)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) S_k(m_0) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

- (b) The average delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:

$$d_k(c_0, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

- (c) The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:

$$d_k(c_r, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(m_0) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

- (d) The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:

$$d_k(c_r, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

Proof: Consult the appendix.

Taking Equation 3.10, splitting the summation over $m = 1 \dots C$ into two sums, one over $h \in \mathcal{H}_k(c)$ and the other over $e \in \mathcal{E}_k(m)$, and applying statements (a) and (b) of Theorem 2, the upper bound on the direct waiting time, $W_k(c_0) \leq W_k^+(c_0)$, may be written:

$$\begin{aligned}
 W_k^+(c_0) = & \sum_{h \in \mathcal{H}_k(c)} N_h(c) \left(\frac{f_h V_k(h_0)}{f_c V_k(c_0)} S_k(h_0) \right. \\
 & + \sum_{r'=1}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_k(c_0)} S_k(h_{r'}) \Big) \\
 & + \sum_{e \in \mathcal{E}_k(c)} N_e(c) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_k(c_0)}) S_k(e_0) \right. \\
 & \left. + \sum_{r'=1}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_k(c_0)}) S_k(e_{r'}) \right) \quad (3.17)
 \end{aligned}$$

Taking Equation 3.11, splitting the summation over $m = 1 \dots C$ into two sums, one over $h \in \mathcal{H}_k(c)$ and the other over $e \in \mathcal{E}_k(m)$ and applying statements (c) and (d) of Theorem 2, the upper bound on the secondary resource waiting time for an indirect visit of a class c customer to secondary resource k , $W_k(c_r) \leq W_k^+(c_r)$, may be written:

$$\begin{aligned}
 W_k^+(c_r) = & \sum_{h \in \mathcal{H}_k(c)} N_h(c) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(c_r)} S_k(h_0) \right. \\
 & + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(h_{r'}) \Big) \\
 & + \sum_{e \in \mathcal{E}_k(c)} N_e(c) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(e_0) \right. \\
 & \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(e_{r'}) \right) \quad (3.18)
 \end{aligned}$$

3.7.3 Secondary Resources with NPP Scheduling

In this section, the upper bound on the waiting times $W_k^+(c_0)$ and $W_k^+(c_r)$ are derived for secondary resources ($k = R + 1 \dots K$) with non-preemptive priority (NPP) scheduling.

Theorem 3. the proof of which may be found in the appendix. gives the upper bounds on the average per visit delays due to contention for NPP secondary resources.

Theorem 3 Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. Assume that the secondary resources indicated below have NPP scheduling. Without loss of generality, assume further that the lower priority contenders $m \in \mathcal{L}_k(c)$ are labeled $m = 1 \dots L$ and are sorted such that for $a, b \in \mathcal{L}_k(c)$, $a < b \Rightarrow S_k(a_0) \geq S_k(b_0)$.

- (a) The delay experienced per direct visit of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following:

$$d_k(c_0, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_k(c_0)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) S_k(m_0) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $m \in \mathcal{L}_k(c)$.

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_0, m_0) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_0, m_0) S_k(m_0)$$

where

$$\alpha_k(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)), \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) & m > 1 \end{cases}$$

- (b) The delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following:

$$d_k(c_0, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $m \in \mathcal{L}_k(c)$.

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_0, m_{r'}) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_0, m_{r'}) S_k(m_{r'})$$

where

$$\alpha_k(c_0, m_{r'}) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_{r'})), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) & m > 1 \end{cases}$$

- (c) The delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following:

$$d_k(c_r, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(m_0) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $m \in \mathcal{L}_k(c)$.

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_r, m_0) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_r, m_0) S_k(m_0)$$

where

$$\alpha_k(c_r, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_r, j_0)), \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_0)}) & m > 1 \end{cases}$$

- (d) The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary re-

source r' of a class m customer to k is upper bounded by the following:

$$d_k(c_r, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $m \in \mathcal{L}_k(c)$.

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_r, m_{r'}) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_r, m_{r'}) S_k(m_{r'})$$

where

$$\alpha_k(c_r, m_{r'}) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_r, j_{r'})), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) & m > 1 \end{cases}$$

Proof: Consult the appendix.

Taking Equation 3.10, splitting the summation over $m = 1 \dots C$ into three sums, one over $h \in \mathcal{H}_k(c)$, one over $e \in \mathcal{E}_k(c)$, and the last over $l \in \mathcal{L}_k(c)$, applying statements (a) and (b) of Theorem 3, the upper bound on the waiting time for direct visits to k of the tagged class c customer, $W_k(c_0) \leq W_k^+(c_0)$, may be written:

$$\begin{aligned} W_k^+(c_0) = & \sum_{h \in \mathcal{H}_k(c)} N_h(c) \left(\frac{f_h V_k(h_0)}{f_c V_k(c_0)} S_k(h_0) \right. \\ & \left. + \sum_{r'=1}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_k(c_0)} S_k(h_{r'}) \right) \\ & + \sum_{e \in \mathcal{E}_k(c)} N_e(c) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_k(c_0)}) S_k(e_0) \right. \\ & \left. + \sum_{r'=1}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_k(c_0)}) S_k(e_{r'}) \right) \\ & + \sum_{l \in \mathcal{L}_k(c)} N_l(c) \alpha_k(c_0, l_0) S_k(l_0) \end{aligned}$$

$$+ \sum_{r'=1}^R M_{r'} \sum_{l \in \mathcal{L}_k(c)} N_l(c) \alpha_k(c_0, l_{r'}) S_k(l_{r'}) \quad (3.19)$$

where

$$\alpha_k(c_0, l_0) = \begin{cases} \min(\frac{1}{N_l}, \frac{f_l V_k(l_0)}{f_c V_k(c_0)}) & l = 1 \\ \min(\frac{1}{N_l}(1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_0, j_0)), \frac{f_l V_k(l_0)}{f_c V_k(c_0)}) & l > 1 \end{cases} \quad (3.20)$$

and

$$\alpha_k(c_0, l_{r'}) = \begin{cases} \min(\frac{1}{N_l}, \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_k(c_0)}) & l = 1 \\ \min(\frac{1}{N_l}(1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_0, j_{r'})), \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_k(c_0)}) & l > 1 \end{cases} \quad (3.21)$$

Taking Equation 3.11, splitting the summation over $m = 1 \dots C$ into three sums, one over $h \in \mathcal{H}_k(c)$, one over $e \in \mathcal{E}_k(m)$ and the last over $l \in \mathcal{L}_k(c)$, applying statements (c) and (d) of Theorem 3, the upper bound on the secondary resource waiting time for an indirect visit of a class c customer to secondary resource k , $W_k^-(c_r) \leq W_k^+(c_r)$, may be written:

$$\begin{aligned} W_k^+(c_r) = & \sum_{h \in \mathcal{H}_k(c)} N_h(c) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(c_r)} S_k(h_0) \right. \\ & \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(h_{r'}) \right) \\ & + \sum_{e \in \mathcal{E}_k(c)} N_e(c) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(e_0) \right. \\ & \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(e_{r'}) \right) \\ & + \sum_{l \in \mathcal{L}_k(c)} N_l(c) \alpha_k(c_r, l_0) S_k(l_0) \\ & + \sum_{r'=1; r' \neq r}^R M_{r'} \sum_{l \in \mathcal{L}_k(c)} N_l(c) \alpha_k(c_r, l_{r'}) S_k(l_{r'}) \end{aligned} \quad (3.22)$$

where

$$\alpha_k(c_r, l_0) = \begin{cases} \min(\frac{1}{N_l}, \frac{f_l V_k(l_0)}{f_c V_r(c_0) V_k(c_r)}) & l = 1 \\ \min(\frac{1}{N_l}(1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_r, j_0)), \frac{f_l V_k(l_0)}{f_c V_r(c_0) V_k(c_r)}) & l > 1 \end{cases} \quad (3.23)$$

and

$$\alpha_k(c_r, l_{r'}) = \begin{cases} \min(\frac{1}{N_l}, \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) V_k(c_r)}) & l = 1 \\ \min(\frac{1}{N_l}(1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_r, j_{r'})), \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) V_k(c_r)}) & l > 1 \end{cases} \quad (3.24)$$

3.7.4 Secondary Resources with PS Scheduling

In this section, the upper bound on the waiting times $W_k^+(c_0)$, and $W_k^+(c_r)$ are derived for secondary resources with processor sharing (PS) scheduling. Theorem 4, the proof of which may be found in the appendix, gives the upper bounds on the average per visit delays due to contention for PS secondary resources.

Theorem 4 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources have PS scheduling, then:*

- (a) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_0) \leq \min(S_k(c_0), \frac{f_m V_k(m_0)}{f_c V_k(c_0)} S_k(m_0))$$

- (b) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_{r'}) \leq \min(S_k(c_0), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)} S_k(m_{r'}))$$

- (c) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_0) \leq \min(S_k(c_r), \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)} S_k(m_0))$$

- (d) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_{r'}) \leq \min(S_k(c_r), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(m_{r'}))$$

Proof: Consult the appendix.

Taking Equation 3.10 and applying statements (a) and (b) of Theorem 4, the upper bound on the direct waiting time, $W_k(c_0) \leq W_k^+(c_0)$, is obtained:

$$\begin{aligned} W_k^+(c_0) &= \sum_{m=1}^C N_m(c) \left(\min(S_k(c_0), \frac{f_m V_k(m_0)}{f_c V_k(c_0)} S_k(m_0)) \right. \\ &\quad \left. + \sum_{r'=1}^R M_{r'} \min(S_k(c_0), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)} S_k(m_{r'})) \right) \end{aligned} \quad (3.25)$$

Applying statements (c) and (d) of Theorem 4 to Equation 3.11, the upper bound on the secondary resource waiting time for an indirect visit of a class c customer to secondary resource k , $W_k(c_r) \leq W_k^+(c_r)$, may be written:

$$\begin{aligned} W_k^+(c_r) &= \sum_{m=1}^C N_m(c) \left(\min(S_k(c_r), \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)} S_k(m_0)) \right. \\ &\quad \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(S_k(c_r), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(m_{r'})) \right) \end{aligned} \quad (3.26)$$

3.8 Waiting Time Bounds at FIFO Primary Resources

In this section, the upper bounds on the primary resource waiting times, $W_r^+(c_0)$, are derived for systems having primary resources with FIFO scheduling. Lemma 10 of the appendix¹ gives the upper bound on the average number of times a customer of a particular class can expect to be delayed at a primary resource by a given contending customer at that resource. The proof for it may be found in the appendix.

Lemma 10 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. The average number of direct visits of a class m customer to a primary resource r that delay a class c customer at r on each direct visit to r by the class c customer is upper bounded by the following inequality:*

$$Q_r(c_0, m_0) \leq \min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)})$$

Proof: Consult Appendix 6.3.2.

3.8.1 Secondary Resources with FIFO Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a FIFO scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only FIFO secondary resources.

Theorem 5, the proof of which may be found in the appendix, gives the upper bounds on the average per visit delays due to contention, $d_w^+(c_x, m_y, n_z)$, for secondary resources with FIFO scheduling.

Theorem 5 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have FIFO scheduling, then:*

¹Lemmas 1 through 10 are also given in the appendix. Some of these lemmas are only referenced in the appendix where they appear in the order in which they are used.

- (a) *The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by direct visits of a class n customer at k is upper bounded by the inequality:*

$$d_k(c_r, m_r, n_0) \leq \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0)$$

- (b) *The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:*

$$d_k(c_r, m_r, n_{r'}) \leq \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'})$$

Proof: Consult the appendix.

We get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, by first applying statements (a) and (b) of Theorem 5 to Inequality 3.14, giving:

$$\begin{aligned} W_r(c_0) \leq & \sum_{m=1}^C N_m(c) Q_r(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \right. \\ & + \sum_{n=1}^C N_n(c, m) \left(\min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) \right. \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'}) \right) \right) \left. \right] \end{aligned}$$

Now, multiply through by $Q_r(c_0, m_0)$, cancelling $Q_r(c_0, m_0)$ wherever it appears in the denominator. This gives:

$$\begin{aligned} W_r(c_0) \leq & \sum_{m=1}^C N_m(c) \left[Q_r(c_0, m_0) S_r(m_0) \right. \\ & \left. + \sum_{k=R+1}^K M_k V_k(m_r) \left(Q_r(c_0, m_0) S_k(m_r) \right. \right. \end{aligned}$$

$$\begin{aligned}
& + \sum_{n=1}^C N_n(c, m) \left(\min(Q_r(c_0, m_0), \frac{f_n V_k(n_0)}{f_c V_r(c_0) V_k(m_r)}) S_k(n_0) \right. \\
& \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(Q_r(c_0, m_0), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) V_k(m_r)}) S_k(n_{r'}) \right) \Bigg]
\end{aligned}$$

Then, Lemma 10 is applied to get the upper bound on the remaining $Q_r(c_0, m_0)$ terms, giving:

$$\begin{aligned}
W_r^+(c_0) = & \sum_{m=1}^C N_m(c) \left[\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_r(m_0) \right. \\
& + \sum_{k=R+1}^K M_k V_k(m_r) \left(\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_k(m_r) \right. \\
& + \sum_{n=1}^C N_n(c, m) \left(\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_n V_k(n_0)}{f_c V_r(c_0) V_k(m_r)}) S_k(n_0) \right. \\
& \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) V_k(m_r)}) S_k(n_{r'}) \right) \right) \Bigg] \quad (3.27)
\end{aligned}$$

3.8.2 Secondary Resources with PP Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a FIFO scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only PP secondary resources.

Theorem 6, the proof of which may be found in the appendix, gives the upper bounds on the average per visit delays due to contention for PP secondary resources.

Theorem 6 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have PP scheduling, then:*

- (a) *The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k*

by direct visits of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_0) \leq \begin{cases} \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

(b) The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_{r'}) \leq \begin{cases} \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'}) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

Proof: Consult the appendix.

We get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, by first splitting each summation over $n = 1 \dots C$ from Inequality 3.14 into summations over $h \in \mathcal{H}_k(m)$ and $e \in \mathcal{E}_k(m)$, and by applying statements (a) and (b) of Theorem 6 to get:

$$\begin{aligned} W_r(c_0) &\leq \sum_{m=1}^C N_m(c) Q_r(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \right. \\ &\quad + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_0) \right. \\ &\quad + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_{r'}) \Big) \\ &\quad \left. \left. + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_0) \right) \right) \right] \end{aligned}$$

$$+ \sum_{r'=1:r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_{r'})) \Bigg] \Bigg]$$

This result is then multiplied through by the $Q_r(c_0, m_0)$ to cancel $Q_r(c_0, m_0)$ wherever it occurs in the denominator, and finally, Lemma 10 is applied to get the upper bound on the remaining $Q_r(c_0, m_0)$ terms:

$$\begin{aligned} W_r^+(c_0) = & \sum_{m=1}^C N_m(c) \left[\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_r(m_0) \right. \\ & + \sum_{k=R+1}^K M_k V_k(h_r) \left(\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_k(m_r) \right. \\ & + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(m_r)} S_k(h_0) \right. \\ & \quad \left. \left. + \sum_{r'=1:r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(h_{r'}) \right) \right. \\ & + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_e V_k(e_0)}{f_c V_r(c_0) V_k(m_r)}) S_k(e_0) \right. \\ & \quad \left. \left. + \sum_{r'=1:r' \neq r}^R M_{r'} \min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) V_k(m_r)}) S_k(e_{r'}) \right) \right] \quad (3.28) \end{aligned}$$

Since this derivation is similar to the one for FIFO scheduled secondary resources in Section 3.8.1, the intermediate steps have not been shown.

3.8.3 Secondary Resources with NPP Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a FIFO scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only NPP secondary resources.

Theorem 7, the proof of which may be found in the appendix, gives the upper bounds on the average per visit delays due to contention for NPP secondary resources.

Theorem 7 *Consider a multiclass queueing network whose parameters are described in*

Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. Assume that the secondary resources indicated below have NPP scheduling. Without loss of generality, assume further that the lower priority contenders $n \in \mathcal{L}_k(m)$ are labeled $n = 1 \dots L$ and are sorted such that for $a, b \in \mathcal{L}_k(c)$, $a < b \Rightarrow S_k(a_0) \geq S_k(b_0)$ then:

- (a) The delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by direct visits of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_0) \leq \begin{cases} \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0) & n \in \mathcal{H}_k(m) \\ \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) & n \in \mathcal{E}_k(m) \end{cases}$$

and for all lower priority contenders, $n \in \mathcal{L}_k(m)$,

$$\sum_{n \in \mathcal{L}_k(m)} V_n(c, m) d_k(c_r, m_r, n_0) \leq \sum_{n \in \mathcal{L}_k(m)} V_n(c, m) \alpha_k(c_r, m_r, n_0) S_k(n_0)$$

where

$$\alpha_k(c_r, m_r, n_0) = \begin{cases} \min(\frac{1}{V_n} \cdot \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & n = 1 \\ \min(\frac{1}{V_n} (1 - \sum_{j=1}^{n-1} V_j \alpha_k(c_r, m_r, j_0)), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & n > 1 \end{cases}$$

- (b) The delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_{r'}) \leq \begin{cases} \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'}) & n \in \mathcal{H}_k(c) \\ \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'}) & n \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $n \in \mathcal{L}_k(m)$.

$$\sum_{n \in \mathcal{L}_k(m)} N_n(c, m) d_k(c_r, m_r, n_{r'}) \leq \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \alpha_k(c_r, m_r, n_{r'}) S_k(n_{r'})$$

where

$$\alpha_k(c_r, m_r, n_{r'}) = \begin{cases} \min\left(\frac{1}{N_n} \cdot \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) & n = 1 \\ \min\left(\frac{1}{N_n} \left(1 - \sum_{j=1}^{n-1} N_j \alpha_k(c_r, m_r, j_{r'})\right) \cdot \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) & n > 1 \end{cases}$$

Proof: Consult the appendix.

We get the upper bound on the delay at a primary resource r . $W_r(c_0) \leq W_r^+(c_0)$, by first splitting each summation over $n = 1 \dots C$ from Inequality 3.14 into summations over $h \in \mathcal{H}_k(m)$, $e \in \mathcal{E}_k(m)$, and $l \in \mathcal{L}_k(m)$, and by applying statements (a) and (b) of Theorem 7 to get:

$$\begin{aligned} W_r(c_0) \leq & \sum_{m=1}^C N_m(c) Q_r(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \right. \\ & + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_0) \right. \\ & + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_{r'}) \Big) \\ & + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min\left(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) S_k(e_0) \right. \\ & + \sum_{r'=1: r' \neq r}^R M_{r'} \min\left(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) S_k(e_{r'}) \Big) \\ & + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_0) S_k(l_0) \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_{r'}) S_k(l_{r'}) \right) \right] \end{aligned}$$

where $\alpha_k(c_r, m_r, l_0)$ and $\alpha_k(c_r, m_r, l_{r'})$ are given by Theorem 7 above. Then, by multiplying through by $Q_r(c_0, m_0)$, cancelling $Q_r(c_0, m_0)$ wherever it occurs in the denominator, and by applying Lemma 10 to get the upper bound on the remaining $Q_r(c_0, m_0)$ terms, we get:

$$\begin{aligned}
W_r^+(c_0) = & \sum_{m=1}^C N_m(c) \left[\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_r(m_0) \right. \\
& + \sum_{k=R+1}^K M_k V_k(h_r) \left(\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_k(m_r) \right. \\
& + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(m_r)} S_k(h_0) \right. \\
& \quad \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(h_{r'}) \right) \right. \\
& + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) \cdot \frac{f_e V_k(e_0)}{f_c V_r(c_0) V_k(m_r)} S_k(e_0) \right. \\
& \quad \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) V_k(m_r)}) S_k(e_{r'}) \right) \right. \\
& + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k^+(c_r, m_r, l_0) S_k(l_0) \\
& \quad \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k^+(c_r, m_r, l_{r'}) S_k(l_{r'}) \right) \right] \quad (3.29)
\end{aligned}$$

where

$$\alpha_k^+(c_r, m_r, l_0) = \begin{cases} \min(\frac{f_m V_r(m_0)}{N_l f_c V_r(c_0)}, \frac{f_l V_k(l_0)}{f_c V_r(c_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_l} (\frac{f_m V_r(m_0)}{f_c V_r(c_0)} - \sum_{j=1}^{l-1} N_j \alpha_k^+(c_r, m_r, j_0)), \frac{f_l V_k(l_0)}{f_c V_r(c_0) V_k(m_r)}) & l > 1 \end{cases} \quad (3.30)$$

and

$$\alpha_k^+(c_r, m_r, l_{r'}) = \begin{cases} \min(\frac{f_m V_r(m_0)}{N_l f_c V_r(c_0)}, \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_l} (\frac{f_m V_r(m_0)}{f_c V_r(c_0)} - \sum_{j=1}^{l-1} N_j \alpha_k^+(c_r, m_r, j_{r'})), \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) V_k(m_r)}) & l > 1 \end{cases} \quad (3.31)$$

The α^* terms are obtained by multiplying the corresponding α term by $Q_r(c_0, m_0)$, cancelling all the $Q_r(c_0, m_0)$ terms from the denominator and replacing the remaining $Q_r(c_0, m_0)$ with $Q_k^+(c_0, m_0)$ given by Lemma 10.

Since, this derivation is similar to the one for FIFO scheduled secondary resources in Section 3.8.1, the intermediate steps have not been shown.

3.8.4 Secondary Resources with PS Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a FIFO scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only PS secondary resources.

Theorem 8, the proof of which may be found in the appendix, gives the upper bounds on the average per visit delays due to contention for processor sharing secondary resources.

Theorem 8 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have NPP scheduling, then:*

- (a) *The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by direct visits of a class n customer at k is upper bounded by the inequality:*

$$d_k(c_r, m_r, n_0) \leq \min(S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0))$$

- (b) *The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:*

$$d_k(c_r, m_r, n_{r'}) \leq \min(S_k(m_r), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'}))$$

Proof: Consult the appendix.

We get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, by first applying statements (a) and (b) of Theorem 8 to Inequality 3.14, giving:

$$\begin{aligned}
 W_r(c_0) \leq & \sum_{m=1}^C N_m(c) Q_r(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \right. \\
 & + \sum_{n=1}^C N_n(c, m) \left(\min(S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) \right. \\
 & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(S_k(m_r), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'}) \right) \right) \left. \right]
 \end{aligned}$$

Then, by multiplying through by $Q_r(c_0, m_0)$, cancelling $Q_r(c_0, m_0)$ wherever it occurs in the denominator, and by applying Lemma 10 to get the upper bound on the remaining $Q_r(c_0, m_0)$ terms, we get:

$$\begin{aligned}
 W_r^+(c_0) = & \sum_{m=1}^C N_m(c) \left[\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_r(m_0) \right. \\
 & + \sum_{k=R+1}^K M_k V_k(m_r) \left(\min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_k(m_r) \right. \\
 & + \sum_{n=1}^C N_n(c, m) \left(\min(S_k(m_r), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) V_k(m_r)} S_k(n_0) \right) \\
 & + \sum_{r'=1: r' \neq r}^R M_{r'} \min(S_k(m_r), \frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_k(m_r), \\
 & \left. \left. \left. \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(n_{r'}) \right) \right) \right) \left. \right] \tag{3.32}
 \end{aligned}$$

Since this derivation is similar to the one for FIFO scheduled secondary resources in Section 3.8.1, the intermediate steps have not been shown.

3.9 Waiting Time Bounds at NPP Primary Resources

In this section the calculations for the upper bounds on the waiting time at a primary resource, $W_r(c_0)$, are presented for primary resources having non-preemptive priority scheduling. In Section 3.9.1, the general approach that was taken to derive these bounds expressions is described. The subsequent sections, Sections 3.9.2 through 3.9.5, consist of the derivations specific to each of the four types of secondary resource scheduling, FIFO, PS, PP, and NPP.

3.9.1 The General Approach

As discussed in Section 3.6.2, the expression for the waiting time at a primary resource r for a particular customer of class c due to the delay caused by a contending customer of class m at r is given by Equation 3.14 (repeated here):

$$W_r(c_0) = \sum_{m=1}^C N_m(c) Q_r(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) + \sum_{n=1}^C N_n(c, m) \left(d_k(c_r, m_r, n_0) + \sum_{r'=1: r' \neq r}^R M_{r'} d_k(c_r, m_r, n_{r'}) \right) \right) \right]$$

Recall that $Q_r(c_0, m_0)$ is the expected number of times that the class c customer can be delayed by a class m customer at r per visit of c to r . Let us call the quantity in square brackets $\Psi_r(c_0, m_0)$ such that:

$$\begin{aligned} \Psi_r(c_0, m_0) = & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) + \sum_{n=1}^C N_n(c, m) \left(d_k(c_r, m_r, n_0) \right. \right. \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} d_k(c_r, m_r, n_{r'}) \right) \right) \end{aligned} \quad (3.33)$$

This quantity represents the expected delay per visit at primary resource r experienced by a customer of class c due to a single contending customer of class m . Included in this delay are any delays experienced by the class m customer at the secondary resources it visits while it

holds the primary resource r . The total delay experienced by the class c customer per visit of r due to a class m customer is given by the product $Q_r(c_0, m_0)\Psi_r(c_0, m_0)$. Rewriting Equation 3.14, the total delay experienced by the class c customer at r per visit for all possible contenders is then given by:

$$W_r(c_0) = \sum_{m=1}^C N_m(c) Q_r(c_0, m_0) \Psi_r(c_0, m_0) \quad (3.34)$$

To find the upper bound on $W_r(c_0)$, we must first find the upper bounds on $Q_r(c_0, m_0)$ and $\Psi_r(c_0, m_0)$ for $m = 1 \dots C$. Looking first at $\Psi_r(c_0, m_0)$, it can be seen from Equation 3.33 that the upper bound on $\Psi_r(c_0, m_0)$ depends on the upper bounds on the total delay experienced by each contending class m customer, $m = 1 \dots C$, at all secondary resources k , $k = R + 1 \dots K$, due to all possible customer classes n that can contend directly or indirectly with the class m customer at k : $\sum_{n=1}^C N_n(c, m) d_k(c_r, m_r, n_0)$ and $\sum_{n=1}^C N_n(c, m) d_k(c_r, m_r, n_{r'})$ for $k = R + 1 \dots K$, $m = 1 \dots C$, and $r' = 1 \dots R$. The bounds on these sums have already been found in Section 3.8 for primary resources with FIFO scheduling by application of Theorems 5 through 8 and can be reused here for primary resources with NPP scheduling since these bounds on secondary resource waiting times do not depend on the scheduling policy at primary resources.

However, the bounds on $Q_r(c_0, m_0)$ *do* depend on the scheduling policies employed at primary resources. In the case of a NPP scheduled primary resource r , the bounds expressions for $Q_r(c_0, m_0)$ depend on relative priority of classes c and m at the primary resource r . There are three cases to consider: the case where m has a higher priority at r than c , the case where m and c have equal priority at r and the case where m has lower priority at r than c . General upper bound expressions for $W_r(c_0)$ that are independent of the scheduling policies of secondary resources are derived for each of these three cases in the remainder of this section. Then, in Sections 3.9.2 through 3.9.5, these general expressions for the upper bounds on $W_r(c_0)$ are expanded to give the bounds specific to each of the secondary resource scheduling types.

Case 1: Priority of m higher than that of c

The upper bound on $Q_r(c_0, m_0)$, the expected number of times that a customer of class c can be delayed by a customer of a higher priority class m at a NPP scheduled primary resource r may be determined by application of Lemma 3 of the appendix.

Lemma 3 *Consider a resource r that has a priority scheduling queueing discipline. If u and v are two traffic sources that generate visits to r with rates λ_u and λ_v , respectively, and source v has a higher priority at r than u , then the maximum number of times that u can be delayed by v at r for a single visit of u to r is upper bounded by:*

$$Q_r(u, v) \leq \frac{\lambda_v}{\lambda_u}$$

Proof: Consult Appendix 6.3.2.

Recognizing the class c customer as traffic source u with visit rate $\lambda_u = f_c V_r(c_0)$ and the higher priority class m customer, $m \in \mathcal{H}_r(c)$, as traffic source v with visit rate $\lambda_v = f_m V_r(m_0)$, Lemma 3 may be applied, yielding an upper bound on $Q_r(c_0, m_0)$:

$$Q_r(c_0, m_0) \leq Q_r^+(c_0, m_0) = \frac{f_m V_r(m_0)}{f_c V_r(c_0)} \quad (3.35)$$

It will be seen later that by applying Theorems 5 through 8 to Equation 3.33, the upper bounds on the $\Psi_r(c_0, m_0)$, $\Psi_r^+(c_0, m_0)$, may be determined (the $\Psi_r^+(c_0, m_0)$ are given explicitly in Sections 3.9.2 through 3.9.5). Using $\Psi_r^+(c_0, m_0)$ and Equation 3.35, the upper bound on the waiting time for a class c customer due to all higher priority class m customers at the NPP scheduled resource r may be written:

$$W_r(c_0) \leq \sum_{m \in \mathcal{H}_r(c)} V_m(c) \frac{f_m V_r(m_0)}{f_c V_r(c_0)} \Psi_r^+(c_0, m_0) \quad (3.36)$$

Case 2: Priority of m equal to that of c

When a class c customer competes at a NPP scheduled primary resource with a class m customer of equal priority, $m \in \mathcal{E}_r(c)$, the tie is broken by serving the customers in first come, first served order. In this case the bounding expression for $Q_r(c_0, m_0)$ is the same as that for the case of a FIFO scheduled primary resource (see Section 3.8). But the bounds on the $\Psi_r(c_0, m_0)$ given by applying Theorems 5 through 8 to Equation 3.33 are not affected by the primary resource scheduling policy and so they are the same as in the FIFO primary resource case too. Thus, the waiting time here for equal priority contenders at NPP scheduled primary resources is exactly the same as that for FIFO scheduled primary resources and we may reuse the secondary resource scheduling-specific waiting time expressions of Sections 3.8.1 through 3.8.4 for the FIFO, PP, NPP, and PS secondary resource scheduling policies, respectively. In Sections 3.9.2 through 3.9.5, rather than restating these lengthy waiting time inequalities, references are made back to the appropriate inequalities of Section 3.8.

Case 3: Priority of m lower than that of c

In order to determine the upper bound on the expected contention for a customer of class c at a NPP scheduled primary resource r due to the lower priority contenders m , $m \in \mathcal{L}_r(c)$, we apply the principles of Lemma 7 of the appendix (a similar argument is made in the appendix to prove Theorem 3 applied in Section 3.7.3 to NPP scheduled *secondary* resources and lower priority contenders).

Lemma 7 *This lemma is based on Lemma 5 of [36]. Let u denote a source of traffic that generates visits to a resource k that has a non-preemptive priority queueing discipline. Assume that service demands at k are NBUE distributed. Let $l = 1 \dots L$ denote a set of classes of traffic sources that generate visits to k at a lower priority than u . Let N_l be the number of identical traffic sources in each class l . Let λ_u and λ_l be the expected rates of visits to k from u and the classes l . Let $S_k(l)$ be the expected value of the service demands at k for the sources in class l . Let $d_k(u, l)$ be the delay that source u experiences at resource k*

due to a visit of k by source l . Without loss of generality, let us assume that these classes are sorted such that for $a, b \in \{1 \dots L\}$ and $a < b$ we have $S_k(a) \geq S_k(b)$. Then the average delay per visit of u to k due to all the lower priority traffic sources is upper bounded as follows:

$$\sum_{l=1}^L N_l d_k(u, l) \leq \sum_{l=1}^L N_l \alpha_k(u, l) S_k(l)$$

where

$$\alpha_k(u, l) = \begin{cases} \min(\frac{1}{N_l}, \frac{\lambda_l}{\lambda_u}) & l = 1 \\ \min(\frac{1}{N_l}(1 - \sum_{j=1}^{l-1} N_j \alpha_k(u, j)), \frac{\lambda_l}{\lambda_u}) & l > 1 \end{cases}$$

Proof: Consult Appendix 6.3.2.

The results of Lemma 7 are based on the following. In order to compute an upper bound on the delay produced by the lower priority customers, the worst case scenario is considered. A visit of the tagged class c customer can be delayed by the service of at most a single lower priority customer, a customer that is in service when the class c customer arrives. If λ_c is the number of visits per second made by the tagged class c customer, then in the worst case, the class c customer must be delayed as many times as possible by the customers of the class with the largest per visit demand at the resource, say class a . If the expected number of visits per second of the class a customer, λ_a , is less than λ_c then as many as possible of the remaining $(\lambda_c - N_a \lambda_a)$ visits per second must be delayed by customers of the class with the next highest demand at the resource, say class b . If again there are visits per second of the tagged class c customer left unmatched by contending classes a and b , then the visits of customers of the class with the next highest demand are considered, and so on.

Consider now a NPP scheduled primary resource r . As defined above, $Q_r(c_0, m_0)$ represents the expected number of times per visit to resource r that the class c customer is delayed by a lower priority class m customer, $\Psi_r(c_0, m_0)$ represents the expected delay experienced by a customer of class c per visit of resource r due to a single visit of r by a customer of class m , and the product $Q_r(c_0, m_0)\Psi_r(c_0, m_0)$ is the total expected delay experienced per visit by the class c customer due to a class m customer. To find the maximum of the delay

at the primary resource r experienced by the class c customer, we wish to maximize $W_r(c_0)$ given by Equation 3.34 repeated here:

$$W_r(c_0) = \sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \Psi_r(c_0, m_0)$$

We now apply Lemma 7 as follows. Let the label k in Lemma 7 stand for the NPP scheduled resource r in Equation 3.34. Consider the class c customer of Equation 3.34 to be the traffic source u in Lemma 7 with a traffic rate to resource r of $\lambda_u = f_c V_r(c_0)$ visits per second. Similarly, consider the set of lower priority classes $m \in \mathcal{L}_r(c)$ of Equation 3.34 to be the set of classes of lower priority traffic sources labeled $1 \dots L$ in Lemma 7 where each lower priority traffic source l has a rate of visits to r of $\lambda_l = f_m V_r(m_0)$ visits per second. It can be seen that the number of contenders in each class of lower priority traffic source, N_l , is equal to $N_m(c)$. The delay per visit experienced by traffic source u due to a lower priority traffic source l , $d_k(u, l)$, is equal to $Q_r(c_0, m_0) \Psi_r(c_0, m_0)$. Lastly, assuming we have the upper bounds on $\Psi_r(c_0, m_0)$, $\Psi_r^+(c_0, m_0)$, let the $S_k(l)$ of Lemma 7 represent the $\Psi_r^+(c_0, m_0)$.

Note that there is a requirement for the $S_k(l)$ of Lemma 7 to be NBUE distributed. This NBUE requirement allows $S_k(l)$, an expected value, to be used as an upper bound on the expected time to completion of a service in progress at k upon arrival of a new visit to k (see the proof for Lemma 7 given in the appendix). Now, by definition, $\Psi_r^+(c_0, m_0)$ is already the upper bound on the time remaining for a service of a class m customer at r upon a new visit of k by the class c customer. Thus, $\Psi_r^+(c_0, m_0)$ may be substituted for $S_k(l)$.

Finally, in order to apply Lemma 7, we must sort the $S_k(l)$ for $l = 1 \dots L$ such that for $a, b \in \{1 \dots L\}$ and $a < b$ we have $S_k(a) \geq S_k(b)$. This corresponds to sorting the $\Psi_r^+(c_0, m_0)$ such that for $a, b \in \mathcal{L}_r(c)$ and $a < b$ we have $\Psi_r^+(c_0, a_0) \geq \Psi_r^+(c_0, b_0)$. This last requirement poses some challenges. Looking at Equation 3.33 and Theorems 5 through 8, it is apparent that the $\Psi_r^+(c_0, m_0)$ depends on the d_k^+ terms which further depend on the f_c , for $c = 1 \dots C$. But the f_c depend on the $W_r(c_0)$ (see Equations 3.3 and 3.1) which depend

on the $\Psi_r(c_0, m_0)$ (see Equation 3.34). In addition to this, the expressions for $\Psi_r^+(c_0, m_0)$ also contain terms involving $Q_r(c_0, m_0)$, an unknown (see Equation 3.33 and Statement (a) of Theorem 5, for example). These problems are solved in Sections 3.9.2 through 3.9.5. For now, assume that the $\Psi_r^+(c_0, m_0)$ are suitably sorted.

We may now maximize $\sum_{m \in \mathcal{L}_r(c)} V_m(c) Q_r(c_0, m_0) \Psi_r(c_0, m_0)$ by applying Lemma 7 with the substitutions described above:

$$W_r(c_0) = \sum_{m \in \mathcal{L}_r(c)} V_m(c) Q_r(c_0, m_0) \Psi_r(c_0, m_0) \leq \sum_{m \in \mathcal{L}_r(c)} V_m(c) \alpha_r(c_0, m_0) \Psi_r^+(c_0, m_0) \quad (3.37)$$

where

$$\alpha_r(c_0, m_0) = \begin{cases} \min(\frac{1}{V_m}, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m = 1 \\ \min(\frac{1}{V_m} (1 - \sum_{j=1}^{m-1} V_j \alpha_r(c_0, j_0)), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m > 1 \end{cases}$$

The bounds results presented in this section for a non-preemptive priority primary resource r for each of the three cases of higher, equal, and lower priority contenders at resource r are expanded in the next four sections to deal with the four different secondary resource scheduling types. The FIFO and PS scheduling policies are considered first because they are easier to handle. The more complicated cases of priority scheduled secondary resources are presented afterwards.

3.9.2 Secondary Resources with FIFO Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a NPP scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only FIFO secondary resources.

Competing Classes with Higher Priority

We get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, due to higher priority classes m , $m \in \mathcal{H}_r(c)$, by first applying statements (a) and (b) of Theorem 5

to Equation 3.33 to get:

$$\begin{aligned} \Psi_r(c_0, m_0) \leq & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\ & + \sum_{n \in \mathcal{H}_k(m)} N_n(c, m) \left(\min\left(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) S_k(n_0) \right. \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min\left(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) S_k(n_{r'}) \right) \right) \end{aligned}$$

This is then substituted into Inequality 3.34 to get:

$$\begin{aligned} W_r(c_0) \leq & \sum_{m \in \mathcal{H}_r(c)} N_m(c) Q_r(c_0, m_0) \left[S_r(m_0) \right. \\ & + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\ & + \sum_{n \in \mathcal{H}_k(m)} N_n(c, m) \left(\min\left(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) S_k(n_0) \right. \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min\left(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) S_k(n_{r'}) \right) \right) \right] \end{aligned}$$

Next, multiply through by $Q_r(c_0, m_0)$ to get:

$$\begin{aligned} W_r(c_0) \leq & \sum_{m \in \mathcal{H}_r(c)} N_m(c) \left[Q_r(c_0, m_0) S_r(m_0) \right. \\ & + \sum_{k=R+1}^K M_k V_k(m_r) \left(Q_r(c_0, m_0) S_k(m_r) \right. \\ & + \sum_{n \in \mathcal{H}_k(m)} N_n(c, m) \left(\min(Q_r(c_0, m_0), \frac{f_n V_k(n_0)}{f_c V_r(c_0) V_k(m_r)}) S_k(n_0) \right. \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(Q_r(c_0, m_0), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) V_k(m_r)}) S_k(n_{r'}) \right) \right) \right] \end{aligned}$$

Finally, the $Q_r(c_0, m_0)$ terms are replaced by their upper bounds, $Q_r^+(c_0, m_0)$ from Equation 3.35, to get:

$$\begin{aligned}
 W_r^+(c_0) = & \sum_{m=1}^C N_m(c) \left[\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_r(m_0) \right. \\
 & + \sum_{k=R+1}^K M_k V_k(m_r) \left(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_k(m_r) \right. \\
 & + \sum_{n \in \mathcal{H}_k(m)} N_n(c, m) \left(\min \left(\frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_n V_k(n_0)}{f_c V_r(c_0) V_k(m_r)} \right) S_k(n_0) \right. \\
 & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min \left(\frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) V_k(m_r)} \right) S_k(n_{r'}) \right) \right] \quad (3.38)
 \end{aligned}$$

Competing Classes with Equal Priority

Recall that for classes of equal priority contending at a priority scheduled primary resource, the tie is broken by serving the customers in first come, first served order. Therefore, the upper bound on the delay at a NPP primary resource r , $W_r(c_0)$, for systems with FIFO scheduled secondary resources is given by Inequality 3.27, the same expression as that for a FIFO scheduled primary resource with FIFO scheduled secondary resources.

Competing Classes with Lower Priority

We get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, due to lower priority classes m , $m \in \mathcal{L}_r(c)$ from Equation 3.37 *provided* that the $\Psi_r(c_0, m_0)$ can be sorted in non-increasing order of m such that for $a, b \in \mathcal{L}_r(c)$ and $a < b$ we have $\Psi_r^+(c_0, a_0) \geq \Psi_r^+(c_0, b_0)$. This can be accomplished by making certain simplifications to the expression for $\Psi_r^+(c_0, m_0)$. Looking at statement (a) of Theorem 5, we can simplify $d_k^+(c_r, m_r, n_0)$ by ignoring the $\min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)})$ which is guaranteed to be less than or equal to 1. Thus we have:

$$d_k^+(c_r, m_r, n_0) \leq \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) \leq S_k(n_0)$$

By a similar argument concerning statement (b) of Theorem 5. we also have:

$$d_k^+(c, m_r, n_{r'}) \leq \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'}) \leq S_k(n_{r'})$$

Substituting these into Equation 3.33 we get:

$$\begin{aligned} \Psi_r(c_0, m_0) \leq \Psi_r^+(c_0, m_0) = & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\ & \left. + \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \left(S_k(n_0) + \sum_{r'=1: r' \neq r}^R M_{r'} S_k(n_{r'}) \right) \right) \end{aligned}$$

All the terms on the R.H.S. can be computed. Now, without loss of generality, we may sort the lower priority classes $m \in \mathcal{L}_r(c)$ according to the requirement indicated above. Substituting this simplified $\Psi_r^+(c_0, m_0)$ into Equation 3.37, we get the upper bound on the waiting time at a NPP primary resource due to all lower priority contenders:

$$\begin{aligned} W_r^+(c_0) = & \sum_{m \in \mathcal{L}_r(c)} N_m(c) \alpha_k(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \right. \\ & \left. \left. + \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \left(S_k(n_0) + \sum_{r'=1: r' \neq r}^R M_{r'} S_k(n_{r'}) \right) \right) \right] \end{aligned} \quad (3.39)$$

where

$$\alpha_k(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m > 1 \end{cases}$$

3.9.3 Secondary Resources with PS Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a NPP scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only PS secondary resources.

Competing Classes with Higher Priority

We get the upper bound on the delay at a primary resource r . $W_r(c_0) \leq W_r^+(c_0)$, due to higher priority classes m . $m \in \mathcal{H}_r(c)$, by first applying statements (a) and (b) of Theorem 8 to Equation 3.33, giving:

$$\begin{aligned} \Psi_r(c_0, m_0) \leq & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\ & + \sum_{n \in \mathcal{H}_k(m)} N_n(c, m) \left(\min(S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0)) \right. \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(S_k(m_r), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'})) \right) \right) \end{aligned}$$

Then, substituting this result into Inequality 3.34, multiplying through by $Q_r(c_0, m_0)$, and finally by replacing the remaining $Q_r(c_0, m_0)$ terms with $Q_r^+(c_0, m_0)$ given by Equation 3.35, we get:

$$\begin{aligned} W_r^+(c_0) = & \sum_{m \in \mathcal{H}_r(c)} N_m(c) \left[\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_r(m_0) \right. \\ & + \sum_{k=R+1}^K M_k V_k(m_r) \left(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_k(m_r) \right. \\ & + \sum_{n \in \mathcal{H}_k(m)} N_n(c, m) \left(\min(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) V_k(m_r)} S_k(n_0)) \right. \\ & \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_k(m_r), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(n_{r'})) \right) \right) \left. \right] \quad (3.40) \end{aligned}$$

Since this derivation is similar in form to the one for FIFO scheduled secondary resources in Section 3.9.2, the intermediate steps have not been shown.

Competing Classes with Equal Priority

Recall that for classes of equal priority contending at a priority scheduled primary resource, the tie is broken by serving the customers in first come, first served order. Therefore, the upper bound on the delay at a primary resource r , $W_r(c_0)$, for systems with PS scheduled secondary resources is given by Inequality 3.32, the same expression as that for a FIFO scheduled primary resource with PS scheduled secondary resources.

Competing Classes with Lower Priority

As in Section 3.9.2, we get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, due to lower priority classes m , $m \in \mathcal{L}_r(c)$ from Equation 3.37. Again, we must be able to sort the $\Psi_r^+(c_0, m_0)$ in non-increasing order which requires simplification of the expressions for $\Psi_r^+(c_0, m_0)$. Looking at statement (a) of Theorem 8, we can simplify $d_k^+(c_r, m_r, n_0)$ by arbitrarily choosing the left argument of the $\min(S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0))$ term. We may do this and still have the result be an upper bound because if the arbitrary choice turns out to have selected the smaller of the two arguments of the min term, then we have performed the same function as the min statement itself. If we choose the larger of the two, then the result is larger than that what the min function would have produced thus making the bound looser than necessary. However, it is still an upper bound. Therefore, we have:

$$d_k^+(c_r, m_r, n_0) \leq \min(S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0)) \leq S_k(m_r).$$

By a similar argument concerning statement (b) of Theorem 5, we also have:

$$d_k^+(c_r, m_r, n_{r'}) \leq \min(S_k(m_r), \frac{f_n V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'})) \leq S_k(m_r).$$

Substituting these into Equation 3.33 we get:

$$\begin{aligned} \Psi_r(c_0, m_0) \leq \Psi_r^+(c_0, m_0) = & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\ & \left. + \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \left(S_k(m_r) + \sum_{r'=1: r' \neq r}^R M_{r'} S_k(m_r) \right) \right) \end{aligned}$$

Now, without loss of generality, we may sort the lower priority classes $m \in \mathcal{L}_r(c)$ according to the requirement indicated above. Substituting this simplified $\Psi_r^+(c_0, m_0)$ into Equation 3.37, we get the upper bound on the waiting time at a NPP primary resource due to all lower priority contenders:

$$\begin{aligned} W_r^+(c_0) = & \sum_{m \in \mathcal{L}_r(c)} N_m(c) \alpha_k(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \right. \\ & \left. \left. + \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \left(S_k(m_r) + \sum_{r'=1: r' \neq r}^R M_{r'} S_k(m_r) \right) \right) \right] \quad (3.41) \end{aligned}$$

where

$$\alpha_k(c_0, m_0) = \begin{cases} \min\left(\frac{1}{N_m}, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}\right) & m = 1 \\ \min\left(\frac{1}{N_m} \left(1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)\right), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}\right) & m > 1 \end{cases}$$

3.9.4 Secondary Resources with PP Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a NPP scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only PP secondary resources.

Competing Classes with Higher Priority

We get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, due to higher priority classes m , $m \in \mathcal{H}_r(c)$, by first applying statements (a) and (b) of Theorem 6

to Equation 3.33. giving:

$$\begin{aligned}
\Psi_-(c_0, m_0) \leq & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\
& + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_0) \right. \\
& \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_{r'}) \right) \\
& + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_0) \right. \\
& \left. \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_{r'}) \right) \right)
\end{aligned}$$

Then, by substituting this result into Inequality 3.34, multiplying through by $Q_r(c_0, m_0)$, and finally by replacing the remaining $Q_r(c_0, m_0)$ terms by $Q_r^+(c_0, m_0)$ given by Equation 3.35. get get:

$$\begin{aligned}
W_r^+(c_0) = & \sum_{m \in \mathcal{H}_r(c)} N_m(c) \left[\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_r(m_0) \right. \\
& + \sum_{k=R+1}^K M_k V_k(m_r) \left(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_k(m_r) \right. \\
& + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(m_r)} S_k(h_0) \right. \\
& \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(h_{r'}) \right) \\
& + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(\frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_e V_k(e_0)}{f_c V_r(c_0) V_k(m_r)}) S_k(e_0) \right. \\
& \left. \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \min(\frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) V_k(m_r)}) S_k(e_{r'}) \right) \right) \Big] \quad (3.42)
\end{aligned}$$

Since this derivation is similar in form to the one for FIFO scheduled secondary resources in Section 3.9.2, the intermediate steps have not been shown.

Competing Classes with Equal Priority

Recall that for classes of equal priority contending at a priority scheduled primary resource, the tie is broken by serving the customers in first come, first served order. Therefore, the upper bound on the delay at a primary resource r , $W_r(c_0)$, for systems with PP scheduled secondary resources is given by Inequality 3.28, the same expression as that for a FIFO scheduled primary resource with PP scheduled secondary resources.

Competing Classes with Lower Priority

To get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, due to lower priority classes m , $m \in \mathcal{L}_r(c)$, we would like to follow the examples given in Sections 3.9.2 and 3.9.3. However, in those cases, we were able to simplify the $d_k^+(c_r, m_r, n_x)$ terms ($x = 0$ or $x = 1 \dots R$) that appear in the expressions for $\Psi_r^+(c_0, m_0)$, making it possible to sort the $\Psi_r^+(c_0, m_0)$ as required for using Equation 3.37. For example, in Section 3.9.2, in which FIFO scheduled secondary resources are handled, the following simplification was made to statement (a) of Theorem 5:

$$d_k^+(c_r, m_r, n_0) \leq \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) \leq S_k(n_0)$$

This simplification eliminates all the f_c terms and the $Q_r(c_0, m_0)$ term as well, leaving only the $S_k(n_0)$ which may easily be sorted. But in the case of a PP scheduled secondary resource, the corresponding simplification must be made to statement (a) of Theorem 6, which states:

$$d_k^+(c_r, m_r, n_0) \leq \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0)$$

There is no simplification that can be made to this expression to eliminate the f_c and $Q_r(c_0, m_0)$ terms.

However, recall that the reason we need to make this simplification is to be able to sort the $\Psi_r^+(c_0, n_0)$ so that Lemma 7 may be applied to maximize Inequality 3.34 (repeated here):

$$W_r(c_0) = \sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \Psi_r(c_0, m_0)$$

But, looking at $d_k^+(c_r, m_r, n_0)$ given by statement (a) of Theorem 6, it can be seen that a factor of $1/Q_r(c_0, m_0)$ may be extracted from $d_k^+(c_r, m_r, n_0)$. Suppose that $\Psi_r(c_0, m_0)$ was expanded and rearranged in such a way that *all* the $d_k^+(c_r, m_r, n_x)$ terms with the factor $1/Q_r(c_0, m_0)$ were grouped together. Then the $Q_r(c_0, m_0)$ of Inequality 3.34 could be multiplied through and cancelled in these terms. We could then rewrite Inequality 3.34 as:

$$\begin{aligned} W_r(c_0) \leq & \sum_{m \in \mathcal{L}_r(c)} N_m(c) [\text{terms with } 1/Q \text{ cancelled}] \\ & + \sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) [\text{remaining terms}] \end{aligned}$$

Now, to maximize this, we could find the maximum of

$$\sum_{m \in \mathcal{L}_r(c)} N_m(c) [\text{terms with } 1/Q \text{ cancelled}]$$

directly, without using Lemma 7 and hence without needing to sort the “terms with $1/Q$ cancelled”. As is discussed below, the so-called “remaining terms” *can* be sorted and so we may apply Lemma 7 to maximize

$$\sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) [\text{remaining terms}]$$

A more formal presentation of this argument is given in the following discussion. But first, note that as discussed in Section 3.9.1, a visit of the class c customer can be delayed by the service of at most a single lower priority customer. However, by splitting the single sum of Inequality 3.34 into two separate sums as we have done, and by applying Lemma 7 to only

one of these two sums, we have relaxed this assertion, allowing more than one lower priority customer to delay the class c customer per visit. The resulting inequalities still give upper bounds on waiting times but these bounds are looser than they would otherwise be.

Applying statements (a) and (b) of Theorem 6 to Equation 3.33, we may write:

$$\begin{aligned}
\Psi_r(c_0, m_0) \leq & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\
& + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_0) \right. \\
& \quad \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_{r'}) \right) \\
& + \sum_{e \in \mathcal{E}_k(c)} N_e(c, m) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_0) \right. \\
& \quad \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_{r'}) \right) \Bigg)
\end{aligned}$$

Let us define $\Phi_r(c_0, m_0)$ as the sum of terms in the above expression for $\Psi_r(c_0, m_0)$ that have $1/Q_r(c_0, m_0)$ as a factor and in which the $1/Q$ factor has been removed. Then we have:

$$\begin{aligned}
\Phi_r(c_0, m_0) = & \sum_{k=R+1}^K M_k V_k(m_r) \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(m_r)} S_k(h_0) \right. \\
& \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(h_{r'}) \right) \quad (3.43)
\end{aligned}$$

Let us then define $\Pi_r(c_0, m_0)$ to be the remaining terms in the above expression for $\Psi_r(c_0, m_0)$ that have not been included in $\Phi_r(c_0, m_0)$:

$$\Pi_r(c_0, m_0) = S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right.$$

$$\begin{aligned}
& + \sum_{e \in \mathcal{E}_k(c)} N_e(c, m) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_0) \right. \\
& \quad \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_{r'}) \right)
\end{aligned}$$

By the definitions of $\Pi_r(c_0, m_0)$ and $\Phi_r(c_0, m_0)$, we have:

$$\Psi_r(c_0, m_0) \leq \Pi_r(c_0, m_0) + \frac{\Phi_r(c_0, m_0)}{Q_r(c_0, m_0)}$$

Therefore we may rewrite Inequality 3.34 as:

$$W_r(c_0) \leq \sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \left(\Pi_r(c_0, m_0) + \frac{\Phi_r(c_0, m_0)}{Q_r(c_0, m_0)} \right)$$

Rewriting, we get:

$$W_r(c_0) \leq \sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \Pi_r(c_0, m_0) + \sum_{m \in \mathcal{L}_r(c)} N_m(c) \Phi_r(c_0, m_0) \quad (3.44)$$

We now apply the simplifications to the expression for $\Pi_r(c_0, m_0)$ so that we may sort the $\Pi_r(c_0, m_0)$ as required for Lemma 7. The simplifications are identical to those that were made in Section 3.9.2 for FIFO secondary resources:

$$d_k^+(c_r, m_r, e_0) \leq \min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_0) \leq S_k(e_0)$$

and

$$d_k^+(c_r, m_r, e_{r'}) \leq \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_{r'}) \leq S_k(e_{r'})$$

Substituting these into the expression for $\Pi_r(c_0, m_0)$ we have:

$$\Pi_r(c_0, m_0) \leq \Pi_r^+(c_0, m_0) = S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right)$$

$$+ \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(S_k(e_0) + \sum_{r'=1: r' \neq r}^R M_{r'} S_k(e_{r'}) \right) \right) \quad (3.45)$$

We may now apply Lemma 7 to maximize

$$\sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \Pi_r(c_0, m_0)$$

exactly as was done in Case 3 of Section 3.9.1 except that here we have substituted $\Pi_r(c_0, m_0)$ for $\Psi_r(c_0, r_0)$. Thus, we get:

$$\sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \Pi_r(c_0, m_0) \leq \sum_{m \in \mathcal{L}_r(c)} N_m(c) \alpha_r(c_0, m_0) \Pi_r^+(c_0, m_0) \quad (3.46)$$

where

$$\alpha_r(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m > 1 \end{cases}$$

Substituting Inequality 3.46 and Equations 3.45 and 3.43 into Equation 3.44 we get the upper bound on the waiting time for a customer of class c at a NPP scheduled primary resource r for a system with PP scheduled secondary resources:

$$\begin{aligned} W_r^+(c_0) = & \sum_{m \in \mathcal{L}_r(c)} N_m(c) \alpha_k(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \right. \\ & \left. \left. + \sum_{e \in \mathcal{H}_k(m)} N_e(c, m) \left(S_k(e_0) + \sum_{r'=1: r' \neq r}^R M_{r'} S_k(e_{r'}) \right) \right) \right] \\ & + \sum_{k=R+1}^K M_k V_k(m_r) \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(m_r)} S_k(h_0) \right. \\ & \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(h_{r'}) \right) \end{aligned} \quad (3.47)$$

where

$$\alpha_k(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m = 1 \\ \min(\frac{1}{N_m}(1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m > 1 \end{cases}$$

3.9.5 Secondary Resources with NPP Scheduling

In this section, the upper bound on the delay experienced by a particular customer of class c at a NPP scheduled primary resource r due to contention from all competing customer classes m , $m = 1 \dots C$, is derived for systems with only NPP secondary resources.

Competing Classes with Higher Priority

We get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, due to higher priority classes m , $m \in \mathcal{H}_r(c)$, by first applying statements (a) and (b) of Theorem 7 to Equation 3.33, giving:

$$\begin{aligned} \Psi_r(c_0, m_0) \leq & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\ & + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_0) \right. \\ & \quad \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_{r'}) \right) \\ & + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_0) \right. \\ & \quad \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_{r'}) \right) \\ & + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_0) S_k(l_0) \\ & \left. + \sum_{r'=1; r' \neq r}^R M_{r'} \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_{r'}) S_k(l_{r'}) \right) \end{aligned}$$

where

$$\alpha_k(c_r, m_r, l_0) = \begin{cases} \min(\frac{1}{N_i}, \frac{f_l V_k(l_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_i} (1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_r, m_r, j_0)), \frac{f_l V_k(l_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l > 1 \end{cases}$$

and

$$\alpha_k(c_r, m_r, l_{r'}) = \begin{cases} \min(\frac{1}{N_i}, \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_i} (1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_r, m_r, j_{r'})), \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l > 1 \end{cases}$$

Then, by substituting the result into Inequality 3.34, multiplying through by $Q_r(c_0, m_0)$, and finally by replacing the remaining $Q_r(c_0, m_0)$ terms by $Q_r^+(c_0, m_0)$ given by Equation 3.35, we get:

$$\begin{aligned} W_r^+(c_0) &= \sum_{m \in \mathcal{H}_r(c)} N_m(c) \left[\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_r(m_0) \right. \\ &+ \sum_{k=R+1}^K M_k V_k(m_r) \left(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} S_k(m_r) \right. \\ &+ \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(m_r)} S_k(h_0) \right. \\ &\quad \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(h_{r'}) \right) \right. \\ &+ \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(\frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_e V_k(e_0)}{f_c V_r(c_0) V_k(m_r)}) S_k(e_0) \right. \\ &\quad \left. \left. + \sum_{r'=1: r' \neq r}^R M_{r'} \min(\frac{f_m V_r(m_0)}{f_c V_r(c_0)}, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) V_k(m_r)}) S_k(e_{r'}) \right) \right. \\ &\left. + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k^+(c_r, m_r, l_0) S_k(l_0) \right] \end{aligned}$$

$$+ \sum_{r'=1:r' \neq r}^R M_{r'} \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k^+(c_r, m_r, l_{r'}) S_k(l_{r'}) \Bigg] \quad (3.48)$$

where $\alpha_k^+(c_r, m_r, l_0)$ is given by multiplying the $\alpha_k(c_r, m_r, l_0)$ by $Q_r(c_0, m_0)$ and then replacing the remaining $Q_r(c_0, m_0)$ terms by $Q_r^+(c_0, m_0)$ so that we have:

$$\alpha_k^+(c_r, m_r, l_0) = \begin{cases} \min(\frac{f_m V_r(m_0)}{N_l f_c V_r(c_0)}, \frac{f_l V_k(l_0)}{f_c V_r(c_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_l}(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} - \sum_{j=1}^{l-1} N_j \alpha_k^+(c_r, m_r, j_0)), \frac{f_l V_k(l_0)}{f_c V_r(c_0) V_k(m_r)}) & l > 1 \end{cases}$$

and $\alpha_k^+(c_r, m_r, l_{r'})$ is given by multiplying the $\alpha_k(c_r, m_r, l_{r'})$ by $Q_r(c_0, m_0)$ and then replacing the remaining $Q_r(c_0, m_0)$ terms by $Q_r^+(c_0, m_0)$ to get:

$$\alpha_k^+(c_r, m_r, l_{r'}) = \begin{cases} \min(\frac{f_m V_r(m_0)}{N_l f_c V_r(c_0)}, \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_n}(\frac{f_m V_r(m_0)}{f_c V_r(c_0)} - \sum_{j=1}^{l-1} N_j \alpha_k^+(c_r, m_r, j_{r'})), \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) V_k(m_r)}) & l > 1 \end{cases}$$

Since this derivation is similar in form to the one for FIFO scheduled secondary resources in Section 3.9.2, the intermediate steps have not been shown.

Competing Classes with Equal Priority

Recall that for classes of equal priority contending at a priority scheduled primary resource, the tie is broken by serving the customers in first come, first served order. Therefore, the upper bound on the delay at a primary resource r , $W_r(c_0)$, for systems with NPP scheduled secondary resources is given by Inequality 3.29, the same expression as that for a FIFO scheduled primary resource with NPP scheduled secondary resources.

Competing Classes with Lower Priority

To get the upper bound on the delay at a primary resource r , $W_r(c_0) \leq W_r^+(c_0)$, due to lower priority classes m , $m \in \mathcal{L}_r(c)$, we follow the same derivation given in Section 3.9.4. The reason we still cannot use the simpler derivation used in Sections 3.9.2 and 3.9.3 for FIFO and PS secondary resources is that again we are not able to simplify the $d_k^+(c_r, m_r, n_x)$

terms ($x = 0$ or $x = 1 \dots R$) that appear in the expressions for $\Psi_r^+(c_0, m_0)$ to permit the sorting of the $\Psi_r^+(c_0, m_0)$ as required for using Equation 3.37.

In a similar way to what was done in Section 3.9.4. statements (a) and (b) of Theorem 7 are applied to Equation 3.33. giving:

$$\begin{aligned}
\Psi_r(c_0, m_0) \leq & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\
& + \sum_{h \in \mathcal{H}_k(m)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_0) \right. \\
& \quad \left. + \sum_{r'=1:r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(h_{r'}) \right) \\
& + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_0) \right. \\
& \quad \left. + \sum_{r'=1:r' \neq r}^R M_{r'} \min(1, \frac{f_e V_{r'}(e_0) V_k(e_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(e_{r'}) \right) \\
& + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_0) S_k(l_0) \\
& \quad \left. + \sum_{r'=1:r' \neq r}^R M_{r'} \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_{r'}) S_k(l_{r'}) \right)
\end{aligned}$$

where

$$\alpha_k(c_r, m_r, l_0) = \begin{cases} \min(\frac{1}{N_l}, \frac{f_l V_k(l_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_n} (1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_r, m_r, j_0)), \frac{f_l V_k(l_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l > 1 \end{cases}$$

and

$$\alpha_k(c_r, m_r, l_{r'}) = \begin{cases} \min(\frac{1}{N_l}, \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l = 1 \\ \min(\frac{1}{N_l} (1 - \sum_{j=1}^{l-1} N_j \alpha_k(c_r, m_r, j_{r'})), \frac{f_l V_{r'}(l_0) V_k(l_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & l > 1 \end{cases}$$

Let us define $\Phi_r(c_0, m_0)$ as the sum of terms in the above expression for $\Psi_r(c_0, m_0)$ that

have $1/Q_\tau(c_0, m_0)$ as a factor and from which the $1/Q$ factor has been removed. Then we have:

$$\begin{aligned} \Phi_\tau(c_0, m_0) = & \sum_{k=R+1}^K M_k V_k(m_\tau) \sum_{h \in \mathcal{H}_k(m)} N_n(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_\tau(c_0) V_k(m_\tau)} S_k(h_0) \right. \\ & \left. + \sum_{r'=1:r' \neq \tau}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_\tau(c_0) V_k(m_\tau)} S_k(h_{r'}) \right) \end{aligned} \quad (3.49)$$

Let us then define $\Pi_\tau(c_0, m_0)$ to be the remaining terms in the above expression for $\Psi_\tau(c_0, m_0)$ that have not been included in $\Phi_\tau(c_0, m_0)$:

$$\begin{aligned} \Pi_\tau(c_0, m_0) = & S_\tau(m_0) + \sum_{k=R+1}^K M_k V_k(m_\tau) \left(S_k(m_\tau) \right. \\ & + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(\min(1, \frac{f_e V_k(e_0)}{f_c V_\tau(c_0) Q_\tau(c_0, m_0) V_k(m_\tau)}) S_k(e_0) \right. \\ & \left. + \sum_{r'=1:r' \neq \tau}^R M_{r'} \min(1, \frac{f_e V_{r'}(n_0) V_k(e_{r'})}{f_c V_\tau(c_0) Q_\tau(c_0, m_0) V_k(m_\tau)}) S_k(e_{r'}) \right) \left. \right) \\ & + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_\tau, m_\tau, l_0) S_k(l_0) \\ & + \sum_{r'=1:r' \neq \tau}^R M_{r'} \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_\tau, m_\tau, l_{r'}) S_k(l_{r'}) \end{aligned}$$

where $\alpha_k(c_\tau, m_\tau, l_0)$ and $\alpha_k(c_\tau, m_\tau, l_{r'})$ are defined as above. Again, we apply the simplifications to the expression for $\Pi_\tau(c_0, m_0)$ so that we may sort the $\Pi_\tau(c_0, m_0)$ as required for Lemma 7. For the equal priority contenders $e \in \mathcal{E}_k(m)$, the simplifications are identical to those that were made in Section 3.9.2 for FIFO secondary resources:

$$d_k^+(c_\tau, m_\tau, n_0) \leq \min(1, \frac{f_n V_k(n_0)}{f_c V_\tau(c_0) Q_\tau(c_0, m_0) V_k(m_\tau)}) S_k(n_0) \leq S_k(n_0)$$

and

$$d_k^+(c_\tau, m_\tau, n_{r'}) \leq \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_\tau(c_0) Q_\tau(c_0, m_0) V_k(m_\tau)}) S_k(n_{r'}) \leq S_k(n_{r'})$$

For lower priority contenders $l \in \mathcal{L}_k(m)$, the simplification is made by eliminating $\alpha_k(c_r, m_r, l_0)$ and $\alpha_k(c_r, m_r, l_{r'})$ which, by definition, are guaranteed to be less than or equal to one. Thus we have:

$$\sum_{l \in \mathcal{L}_k(m)} N_l(c, m) d_k^+(c_r, m_r, l_0) \leq \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_0) S_k(l_0) \leq \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) S_k(l_0)$$

and

$$\sum_{l \in \mathcal{L}_k(m)} N_l(c, m) d_k^+(c_r, m_r, l_{r'}) \leq \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \alpha_k(c_r, m_r, l_{r'}) S_k(l_{r'}) \leq \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) S_k(l_{r'})$$

Substituting these into the expression for $\Pi_r(c_0, m_0)$ we have:

$$\begin{aligned} \Pi_r(c_0, m_0) \leq \Pi_r^+(c_0, m_0) = & S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) \right. \\ & + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(S_k(e_0) + \sum_{r'=1:r' \neq r}^R M_{r'} S_k(e_{r'}) \right) \\ & \left. + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \left(S_k(l_0) + \sum_{r'=1:r' \neq r}^R M_{r'} S_k(l_{r'}) \right) \right) \quad (3.50) \end{aligned}$$

We may now apply Lemma 7 to maximize

$$\sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \Pi_r(c_0, m_0)$$

exactly as was done in Section 3.9.4:

$$\sum_{m \in \mathcal{L}_r(c)} N_m(c) Q_r(c_0, m_0) \Pi_r(c_0, m_0) \leq \sum_{m \in \mathcal{L}_r(c)} N_m(c) \alpha_r(c_0, m_0) \Pi_r^+(c_0, m_0) \quad (3.51)$$

where

$$\alpha_k(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m > 1 \end{cases}$$

Substituting Inequality 3.51 and Equations 3.50 and 3.49 into Equation 3.44 of Section 3.9.4 we get the upper bound on the waiting time for a customer of class c at a NPP scheduled primary resource r for a system with NPP scheduled secondary resources:

$$W_r^+(c_0) = \sum_{m \in \mathcal{L}_r(c)} N_m(c) \alpha_k(c_0, m_0) \left[S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) \left(S_k(m_r) + \sum_{e \in \mathcal{E}_k(m)} N_e(c, m) \left(S_k(e_0) + \sum_{r'=1; r' \neq r}^R M_{r'} S_k(e_{r'}) \right) + \sum_{l \in \mathcal{L}_k(m)} N_l(c, m) \left(S_k(l_0) + \sum_{r'=1; r' \neq r}^R M_{r'} S_k(l_{r'}) \right) \right) \right] \quad (3.52)$$

$$+ \sum_{k=R+1}^K M_k V_k(m_r) \sum_{h \in \mathcal{H}_k(c)} N_h(c, m) \left(\frac{f_h V_k(h_0)}{f_c V_r(c_0) V_k(m_r)} S_k(h_0) + \sum_{r'=1; r' \neq r}^R M_{r'} \frac{f_h V_{r'}(h_0) V_k(h_{r'})}{f_c V_r(c_0) V_k(m_r)} S_k(h_{r'}) \right) \quad (3.53)$$

where

$$\alpha_k(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)), \frac{f_m V_r(m_0)}{f_c V_r(c_0)}) & m > 1 \end{cases}$$

In Section 3.9.1, it was indicated that a visit of the class c customer at a NPP scheduled primary resource can be delayed by the service of at most a single lower priority customer. However, as discussed in Section 3.9.4, this technique of splitting the sum over Ψ into two parts, a sum over Π and a sum over Φ , and applying Lemma 7 to some of the terms but not others, relaxes this assertion. This results in inequalities that give looser upper bounds on the waiting times.

3.10 Summary Of Lower Bounds Expressions

Expressions for upper bounds on the waiting times, $W_r(c_0)$, $W_k(c_0)$, and $W_k(c_r)$ have been derived in Sections 3.7 through 3.9.3 for all possible combinations of scheduling policies and priorities for both primary and secondary resources. The following summarizes these

results in a form that lends itself to implementing the lower bounds calculation in software.

To make the notation less cumbersome, the following shorthands have been used:

$$\lambda_k(c_0) = f_c V_k(c_0) \quad (3.54)$$

$$\lambda_k(c_{r'}) = f_c V_{r'}(c_0) V_k(c_{r'}) \quad (3.55)$$

$$\lambda_k(c_r, m_r) = f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r) \quad (3.56)$$

$$\lambda_k^*(c_r, m_r) = f_c V_r(c_0) V_k(m_r) \quad (3.57)$$

$$\Gamma_k(c_r, m_{r'}) = \frac{\lambda_k(m_{r'})}{\lambda_k(c_r)} \quad (3.58)$$

3.10.1 General Expression for Lower Bound on Class Frequency

$$\begin{aligned} 1 \leq & f_c(Z_c + \sum_{r=1}^R M_r V_r(c_0)[S_r(c_0) + W_r^+(c_0)] \\ & + \sum_{k=R+1}^K M_k V_k(c_r)(S_k(c_r) + W_k^+(c_r))) \\ & + \sum_{k=R+1}^K M_k V_k(c_0)(S_k(c_0) + W_k^+(c_0))) \end{aligned} \quad (3.59)$$

$$V_k(c_r) = 0 \quad k = R+1 \dots K \text{ and } r > R \quad (3.60)$$

3.10.2 Waiting Time at Secondary Resources

Note that in the following, FIFO and PS scheduled secondary resources are handled with the equal priority cases ($m \in \mathcal{E}_r(c)$).

$$\begin{aligned} W_k^+(c_r) = & \sum_{m \notin \mathcal{L}_k(c)} N_m(c) d_k^+(c_r, m_0) \\ & + \sum_{r'=1, r' \neq r}^R M_{r'} \sum_{m=1}^C N_m(c) d_k^+(c_r, m_{r'}) \end{aligned}$$

$$\begin{aligned}
& + \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_r, m_0) S_k(m_0) \\
& + \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_r, m_{r'}) S_k(m_{r'}) \quad \begin{array}{l} k = R+1 \dots K \\ r = 0 \dots R \end{array} \quad (3.61)
\end{aligned}$$

3.10.3 Waiting Time at Primary Resources

Note that in the following, FIFO scheduled primary resources are treated identically to NPP scheduled primary resources with equal priority contending classes ($m \in \mathcal{E}_r(c)$). Let $\mathcal{N}$ be the set of NPP scheduled secondary resources.

$$\begin{aligned}
W_r^+(c_0) = & \sum_{m \notin \mathcal{L}_r(c)} N_m(c) \left\{ S_r(m_0) Q_r^+(c_0, m_0) \right. \\
& + \sum_{k \notin \mathcal{N}} M_k V_k(m_r) \left[S_k(m_r) Q_r^+(c_0, m_0) \right. \\
& + \sum_{n=1}^C N_n(c, m) d_k^+(c_r, m_r, n_0) \\
& + \left. \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{n=1}^C N_n(c, m) d_k^+(c_r, m_r, n_{r'}) \right] \\
& + \sum_{k \in \mathcal{N}} M_k V_k(m_r) \left[S_k(m_r) Q_r^+(c_0, m_0) \right. \\
& + \sum_{n \notin \mathcal{L}_k(m)} N_n(c, m) d_k^+(c_r, m_r, n_0) \\
& + \left. \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{n \notin \mathcal{L}_k(m)} N_n(c, m) d_k^+(c_r, m_r, n_{r'}) \right] \\
& + \sum_{k \in \mathcal{N}} M_k V_k(m_r) \left[S_k(m_r) Q_r^+(c_0, m_0) \right. \\
& + \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \alpha_k^+(c_r, m_r, n_0) S_k(n_0) \\
& + \left. \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \alpha_k^+(c_r, m_r, n_{r'}) S_k(n_{r'}) \right] \left. \right\}
\end{aligned}$$

$$\begin{aligned}
& + \sum_{m \in \mathcal{L}_r(c)} N_m(c) \alpha_r(c_0, m_0) \left\{ S_r(m_0) \right. \\
& + \sum_{k \in \mathcal{P}} M_k V_k(m_r) \left[S_k(m_r) \right. \\
& + \sum_{n=1}^C N_n(c, m) d_k(c_r, m_r, n_0) \\
& + \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{n=1}^C N_n(c, m) d_k(c_r, m_r, n_{r'}) \left. \right] \\
& + \sum_{k \in \mathcal{P}} M_k V_k(m_r) \left[S_k(m_r) \right. \\
& + \sum_{n \notin \mathcal{L}_k(m)} N_n(c, m) d_k(c_r, m_r, n_0) \\
& + \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{n \notin \mathcal{L}_k(m)} N_n(c, m) d_k(c_r, m_r, n_{r'}) \left. \right] \left. \right\} \\
& + \sum_{k \in \mathcal{P}} M_k V_k(m_r) \left[\sum_{n \in \mathcal{L}_k(m)} N_n(c, m) d_k^*(c_r, m_r, n_0) \right. \\
& + \sum_{r'=1: r' \neq r}^R M_{r'} \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) d_k^*(c_r, m_r, n_{r'}) \left. \right] \quad r = 1 \dots R \quad (3.62)
\end{aligned}$$

3.10.4 Expected Collisions at a Primary Resource

Let $\mathcal{P}$ be the set of primary resources having NPP scheduling. Then for each $r \in \mathcal{P}$ we sort the $m \in \mathcal{L}_r(c)$ in non-increasing order of $\Pi_r(c_0, m_0)$:

$$\begin{aligned}
\Pi_r(c_0, m_0) &= S_r(m_0) + \sum_{k=R+1}^K M_k V_k(m_r) [S_k(m_r) \\
&+ \sum_{n \notin \mathcal{H}_k(m)} N_n(c, m) (d_k^+(c_r, m_r, n_0) \\
&+ \sum_{r'=1: r' \neq r}^R M_{r'} d_k^+(c_r, m_r, n_{r'}))] \quad (3.63)
\end{aligned}$$

For primary resources with FIFO scheduling or with NPP scheduling and but with $m \notin \mathcal{L}_k(c)$, we have:

$$Q_r^+(c_0, m_0) = \begin{cases} \Gamma_r(c_0, m_0) & \begin{array}{l} r \text{ is NPP} \\ m \in \mathcal{H}_k(c) \end{array} \\ \min(1, \Gamma_r(c_0, m_0)) & \begin{array}{l} r \text{ is NPP} \\ \text{and } m \in \mathcal{E}_k(c) \\ \text{or } r \text{ is FIFO} \end{array} \end{cases} \quad (3.64)$$

For primary resources with NPP scheduling and $m \in \mathcal{L}_k(c)$, we have:

$$\alpha_r(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \Gamma_r(c_0, m_0)) & m = 1 \\ \min(\frac{1}{N_m}(1 - \sum_{j=1}^{m-1} N_j \alpha_r(c_0, m_0)), \Gamma_r(c_0, m_0)) & m > 1 \end{cases} \quad (3.65)$$

3.10.5 Delay at a Secondary Resource

$$d_k^+(c_r, m_{r'}) = \begin{cases} \Gamma_k(c_r, m_{r'}) S_k(m_{r'}) & \begin{array}{l} k \in \mathcal{P} \\ m \in \mathcal{H}_k(c) \end{array} \\ \min(1, \Gamma_k(c_r, m_{r'})) S_k(m_{r'}) & \begin{array}{l} k \text{ is FIFO} \\ \text{or } k \in \mathcal{P}, m \in \mathcal{E}_k(c) \end{array} \\ 0 & \begin{array}{l} k \text{ is PP} \\ m \in \mathcal{L}_k(c) \end{array} \\ \min(S_k(c_r), \Gamma_k(c_r, m_{r'}) S_k(m_{r'})) & k \text{ is PS} \end{cases} \quad (3.66)$$

For NPP scheduled secondary resources, $k \in \mathcal{N}$, with $m \in \mathcal{L}_k(c)$ we have:

$$\alpha_k(c_r, m_{r'}) = \begin{cases} \min(\frac{1}{N_n}, \Gamma_k(c_r, m_{r'})) & n = 1 \\ \min(\frac{1}{N_n}(1 - \sum_{j=1}^{n-1} N_j \alpha_k(c_r, j_{r'})), \Gamma_k(c_r, m_{r'})) & n > 1 \end{cases} \quad (3.67)$$

3.10.6 Indirect Delay at a Primary Resource

Let $\mathcal{P}$ be the set of secondary resources having priority scheduling. that is NPP or PP scheduling.

$$d_k^-(c_r, m_r, n_{r'}) = \begin{cases} \frac{\lambda_k(n_{r'})}{\lambda_k^*(c_r, m_r)} S_k(n_{r'}) & \begin{array}{l} r \text{ is FIFO} \\ \text{or } r \text{ is NPP. } m \notin \mathcal{L}_r(c) \\ k \in \mathcal{P}. n \in \mathcal{H}_k(m) \end{array} \\ \min \left(Q_r^+(c_0, m_0), \frac{\lambda_k(n_{r'})}{\lambda_k^*(c_r, m_r)} \right) S_k(n_{r'}) & \begin{array}{l} r \text{ is FIFO} \\ \text{or } r \text{ is NPP. } m \notin \mathcal{L}_r(c) \\ k \text{ is FIFO} \\ \text{or } k \in \mathcal{P}. n \in \mathcal{E}_k(m) \end{array} \\ \min(Q_r^+(c_0, m_0) S_k(m_r), & \\ \frac{\lambda_k(n_{r'})}{\lambda_k^*(c_r, m_r)} S_k(n_{r'})) & \begin{array}{l} r \text{ is FIFO} \\ \text{or } r \text{ is NPP. } m \notin \mathcal{L}_r(c) \\ k \text{ is PS} \end{array} \end{cases} \quad (3.68)$$

For NPP scheduled secondary resources, $k \in \mathcal{N}$. with $n \in \mathcal{L}_k(m)$ we have:

$$\alpha_k^-(c_r, m_r, n_{r'}) = \begin{cases} \min \left(\frac{Q_r^+(c_0, m_0)}{N_n}, \frac{\lambda_k(n_{r'})}{\lambda_k^*(c_r, m_r)} \right) & n = 1 \\ \min \left(\frac{Q_r^+(c_0, m_0)}{N_n} \left(1 - \sum_{j=1}^{n-1} N_j \alpha_k^-(c_r, n_r, j_{r'}) \right), \right. & \\ \left. \frac{\lambda_k(n_{r'})}{\lambda_k^*(c_r, m_r)} \right) & n > 1 \end{cases} \quad (3.69)$$

$$d_k^+(c_r, m_r, n_{r'}) = \begin{cases} S_k(n_{r'}) & \begin{array}{l} r \text{ is NPP. } m \in \mathcal{L}_r(c) \\ k \in \mathcal{P}. n \in \mathcal{E}_k(m) \\ \text{or } k \text{ is FIFO} \end{array} \\ 0 & \begin{array}{l} r \text{ is NPP. } m \in \mathcal{L}_r(c) \\ k \text{ is PP. } n \in \mathcal{L}_k(m) \end{array} \\ S_k(n_{r'}) & \begin{array}{l} r \text{ is NPP. } m \in \mathcal{L}_r(c) \\ k \text{ is NPP. } n \in \mathcal{L}_k(m) \end{array} \\ S_k(m_r) & \begin{array}{l} r \text{ is NPP. } m \in \mathcal{L}_r(c) \\ k \text{ is PS} \end{array} \end{cases} \quad (3.70)$$

3.11 Discussion

The research that culminated in the derivations presented in Sections 3.7 through 3.9.3 began with many smaller derivations, one for each individual case of primary resource scheduling type and secondary resource scheduling type. Each of these individual cases was considered in isolation and the inequalities representing the performance bounds for each case were derived. Once all the cases had been considered, the results were combined, resulting in the more general derivations presented in Sections 3.7 through 3.9.3.

As a test of the validity of the bounds inequalities, wherever possible the bounds expressions derived in this thesis were compared to those produced by earlier, less general techniques, for a particular test system. For example, a system consisting of only secondary resources was considered and the bounds expressions produced by the inequalities derived in this thesis were compared to those of the technique presented in [36] that handle only secondary resources. Similarly, a system consisting of FIFO scheduled primary and secondary resources was analyzed and the bounds expressions were compared to those obtained from the technique proposed by Majumdar in [29] and discussed in Section 1.1.3. In these cases and others, the more general bounds expressions of this thesis reduced to match those of the preceding techniques.

The CLP(BNR) software that was developed to compute the bounds was based on the form of the inequalities given in Sections 3.4 and 3.5 for the upper bounds and Section 3.10 for the lower bounds.

Chapter 4

Modeling Software

4.1 Introduction

This chapter describes the software tool that was designed and implemented for the purpose of the specification and the computation of queueing network performance bounds. Figure 4.1 is a screen dump showing the the tool in action. In the figure a small queueing network model is being edited. There are two customer classes, *c1* and *c2*, two ordinary servers, *s1* and *s2*, and one shared resource, *p1*. The arcs connecting the classes and resources denote the visits and are annotated with the visit parameters. Figure 3.1 gives the exact meaning of all the objects and parameters in the queueing network notation.

The software is formed from two distinct components, a graphical user interface and a bounds analysis engine. Figure 4.2 shows these two components and how they work together. The graphical user interface, represented in the figure by the upper rounded box, provides the interface for a user to create and edit a model graphically. This model can be saved and read from a *model graphics* file as indicated. When a model is ready to be analyzed, the user invokes the solve command. The graphical user interface then creates a *model description* file which is in turn read by the bounds analyzer. The bounds analyzer computes the throughput bounds and prints them to the *model results* file. The dashed lines for the model description and model results files indicate the temporary nature of these

Figure 4.1: A screen dump of the queueing network modelling tool interface.

Figure 4.2: The components of the queueing network bounds analysis tool.

two files. Neither of them is needed once the bounds have been computed and they may be deleted. The graphical user interface reads the bounds from the model results and displays them graphically for the user.

The graphical user interface is a Java program and is described in Section 4.2. Java was selected as the development platform for its ease of use in interface development and for its portability. The bounds analyzer is a CLP(BNR) program (see Chapter 2) with a wrapper consisting of a C program and a shell script. The bounds analyzer is described in Section 4.7 and the glue (C program and script) are described in Section 4.8.

4.2 Graphical User Interface Overview

The graphical user interface (GUI) software is made up of several major components. These are:

- the Java Core API packages [5] supplied by Sun Microsystems Inc. as part of the

standard Java Development Kit version 1.1 (JDK 1.1) distribution.

- the HTK package, a collection of utilities and extensions to the Java Core written for this project.
- the MVC package, an application framework for GUI interfaces implementing a variation of the model/view/control abstraction [48], also written for this project.
- the application specific code, which is further subdivided into:
 - the qnbmodel package implementing the model oriented application specific behaviour
 - the qnbview package implementing the application specific view of the model.
 - the qnbcontrol package implementing the interface between the application specific view and model, and
 - the QNBApp class, implementing the entry point to the application.

These components and their dependencies on one another are shown in Figure 4.3. In

Figure 4.3: The Java package components that make up the queueing network bounds analysis graphical user interface software.

the figure, the components that depend on one another are shown with adjacent edges. The HTK package uses the Java Core, the MVC framework package uses both the HTK and the Java Core, the QNBView package uses MVC, HTK, and Java Core. etc. In the following subsections, each of these packages (but not the Java Core) is described.

4.3 The HTK Package

The HTK is a package of various classes, some of which are convenient or visually pleasing extensions to the Java Core, and some of which are vital components to higher level packages like MVC. Figure 4.4 shows the classes of the HTK package and their inheritance

Figure 4.4: The classes and their inheritance relationships in the HTK package.

relationships. In the figure, the inheritance hierarchy builds downwards with the super-classes on top and the subclasses under the superclasses and connected to them by arcs. Classes labeled in a bold font are part of the HTK package whereas non-bold class names refer to classes in other packages (in this case the Java Core). Names written in a slanted

font are Java interfaces, not classes.

At the top of the figure, there is a tree of classes inheriting from the Canvas class. The Canvas class is a widget belonging to the Java Core that is designed to be a starting point from which custom widgets may be made. The ModeButton class inherits from Canvas to implement a *modal button*. This is a button widget that remembers its state, i.e. when it is pressed, it stays pressed and does not release immediately. The ModeButtons can be grouped together to implement *radio buttons* — a set of buttons in which a single button only may be selected at a time. The ReliefNodeButton class inherits from the ModeButton class to implement the ModeButton functionality but with a three dimensional appearance. The ReliefModeButton class uses the Relief class to achieve this three dimensional look. The MultiLineLabel class inherits from Canvas to implement a type of label widget that can display more than just one line of text. The ReliefLabel class extends Canvas to implement a label widget with a three dimensional appearance. This is accomplished as before through the use of the Relief class. The ReliefCanvas is a three dimensional canvas widget. The DrawingArea class inherits from Canvas to implement a widget suitable for general drawing activities. This is extended further by ScrollableDrawingArea to allow the drawing area to be scrolled using vertical and horizontal scroll bars.

In the top right hand corner of Figure 4.4 is the Java Core class Label. This is a widget that displays a single line of text. The size of the widget is set to be exactly the size of the text it is displaying. The SizedLabel that inherits from Label implements a label whose size can be set independently of the text being displayed. This widget is useful in situations where a group of widgets would otherwise get out of alignment if the size of the label were changed.

In the middle on the left of the figure is the two-class inheritance diagram showing Dialog and ErrorDialog. The Dialog class is a Java Core class which provides a basic dialog box for interacting with the user. The ErrorDialog is an extension designed to provide a streamlined error reporting mechanism.

To the right of the Dialog class, but skipping the Relief class already described, is the

Panel class. This is a Core Java class for assembling widgets together in a window. The ReliefPanel is a three dimensional version of the Panel. The DrawingPanel is a special extension of a three dimensional panel designed to handle generic drawing activities. The DrawingPanel contains a scrollable, three dimensional drawing area and can print the drawing area in PostScript.

To the right again, is the Graphics class. This class is the Core Java graphics context class for handling all drawing to widgets and windows. The PSGraphics re-implements all the graphics methods but for a PostScript target, not a graphical window. This PSGraphics class is used by DrawingPanel for PostScript printing.

In the lower left corner of Figure 4.4 is a grouping of classes including Rectangle, Drawing, AlignedText, CompoundDrawing. These classes are useful for generic drawing activities on a canvas such as the DrawingArea class. The Rectangle class is a Java Core class that implements a basic drawing component: the rectangle. The Drawing class extends the Rectangle to include functionality specifically tailored for use with drawing canvas components like DrawingArea. The CompoundDrawing class further extends Drawing to handle collections of Drawing objects. The class AlignedText is a specific kind of Drawing suitable for printing text with various alignment styles on a drawing canvas. The Drawable interface consists of the important interface methods of Drawing. It is to be used in situations where a class must provide the same interface as Drawing but cannot inherit from Drawing. The DrawingOwner interface must be implemented by any class that is to contain any subclass of Drawing.

The Util class contains some useful methods that did not fit in any other class in the HTK package. These methods include methods for calculating the distance between two points, solving the quadratic formula, etc.

4.4 The MVC Package

The MVC package is a graphical user interface framework. It contains the data structures and functionality for a generic graphical user interface application. To produce a domain

specific application, the developer needs only to extend the class inheritance hierarchy defined in MVC to implement the application specific data structures and behaviour. The amount of code the developer must write varies from application to application but in general it should be small.

The MVC framework presents a Model/View/Control abstraction for understanding and implementing graphical user interface applications. In their book *Object-Oriented Programming: an Evolutionary Approach*, Cox and Novobilski indicate that MVC abstractions are common in graphical user interfaces [10]. Examples include the SmallTalk-80 system[48] and also Apple's Macintosh development environment. According to Cox, the idea was first developed by Trygve Reebkaug of the Norwegian Central Institute during an extended visit to Xerox PARC. The particular implementation used in this thesis has some design elements that are loosely based on ideas developed by Marc Courtois in 1997.

Figure 4.5 shows the interactions between the Model, View, and Control abstractions.

Figure 4.5: The Model, View, and Control abstractions and interactions.

Each of these components is discussed below including a description of the interactions

between the components. Examples of how the MVC framework functions are given in Section 4.6.

4.4.1 The Model

The Model abstraction contains the data structures and behaviour that implement the core program functionality. It is the Model that knows about filing data in and out of the program, about searching, sorting, or whatever else the program may be required to do. The Model, however, knows nothing at all about presenting the data for display or any other interface detail. The model has *aspects* which it publishes. Aspects refer to data structure components like file names, or database records or fields.

4.4.2 The View

The View abstraction is responsible for the interface. The view provides buttons, menus, panels, forms, dialogs and anything else the interface may require. The View knows nothing about the model. The View has features which it publishes. Features represent widgets like file menu selections, status line labels, and drawing canvas scroll bars.

4.4.3 The Control

The Control abstraction is responsible for interfacing the Model to the View and vice versa. Since the Model knows nothing of the View and the View knows nothing of the Model, the Control has to know all about both the Model and the View. As indicated above, the Model publishes aspects and the View publishes features. The Control registers model aspects with view features so that when a view feature experiences some type of occurrence, the occurrence indication is automatically delivered to the model aspect. The Control also registers view features with model aspects so that when a model aspect changes, an aspect indication is automatically delivered to the interested view feature. In this way, the model and view automatically keep each other updated but without requiring tight coupling.

Both the Model and the View may request that the Control take some sort of action.

These requests are made in the form of command indications delivered to the Control from either the Model or the View. These commands are often commands to create model or view components, register aspects and features, or destroy model or view components. The Control has direct control over both the Model and the View and causes actions to be performed without requiring the delivery of indications.

4.4.4 MVC Classes and Inheritance

The classes and their inheritance relationships are shown in Figure 4.6. As in Figure 4.4, boldface names are the names of classes and interfaces defined in the MVC package and non-bold names are defined elsewhere. Slanted names are interface names whereas normal names refer to class names.

It is apparent in Figure 4.6 that the MVC package consists of five general categories of classes and interfaces. These are

- the classes and interfaces supporting the Model abstraction,
- the classes and interfaces for the Control abstraction,
- the classes and interfaces for the View abstraction,
- classes and interfaces supporting the indication mechanism, and
- the exception classes for handling indication exceptions.

4.4.5 The Model Classes

The Model classes present the software designer with the following programming interface. The application sees the model framework as being a tool object containing a graph of nodes and arcs. The Tool class provides the functionality for handling the following types of attributes:

- The title and version number of the tool

Figure 4.6: The classes and their inheritance relationships in the MVC package.

- A status message and error message facility
- File input and output
- Drawing canvas attributes
- Canvas event handling
- Graph creation and manipulation

The `Tool` class is an abstract class and must be extended in the application specific code to provide application specific behaviour for the tool.

There are model components to represent the graph objects: the `Node` class provides the basic functionality for node type objects, the `Arc` class provides the basic functionality of arc type objects and the `Label` class provides the functionality for annotations on nodes or arcs. The `Comment` class provides the functionality to support textual comments that document the graph. The `Node` and `Arc` classes are abstract classes and must be overridden in the application specific code to provide application specific nodes and arcs. The `Comment` class may be used as is or may be enhanced in the application specific code. The `Label` class is not normally extended but is used implicitly through the `Arc` and `Node` classes. Labels, nodes, arcs, and comments are all subclasses of the `Component` class.

It is intended that the application specific Model subclasses should implement the data structures necessary to support whatever model data or model parameters are required by the application. For example, the subclass of an arc could contain one or more double precision values to represent various types of arc parameters. Model subclasses should also implement the methods that are needed to store and retrieve instances of the subclass to and from files.

When any of these components is created, it must be registered with an object that *owns* it. For an object to own a component, that object must implement the `ComponentOwner` interface. The `Tool` classes implements this interface and so any application specific class that inherits from `Tool` will automatically be a component owner. However, if any other

application specific class needs to manage components, it must implement `ComponentOwner`. Interestingly, components inheriting from `Component` are also `ComponentOwners` and therefore may contain nested components.

All of the model classes inherit their basic model behaviour from `Model`. It is this class which implements the capability to publish aspects and to handle occurrence indications. Each subclass of `Model` adds new published aspects which are then inherited by further subclasses. In addition to inheriting aspects, subclasses may override aspects as well.

In summary, the three main `Model` supports on which the application specific code will rest are the `Tool` class which is a `ComponentOwner`, and the `Node`, and `Arc` classes which are components.

4.4.6 The View Classes

The `View` classes provide the software designer with a set of widgets for the purpose of displaying the model. Whereas the model has aspects and responds to occurrence indications, the view has features and responds to aspect indications. Ordinary Java Core widgets do not have “features” (in the sense discussed so far) and do not respond to aspect indications. For this reason, any widget from the Java Core that is to be used as part of the `View` abstraction must be wrapped so that it implements the `View` interface. This guarantees that the widget then has features and responds to aspect indications.

As in the model abstraction, the view abstraction presents the concept of the view of a tool and the view of graphical components within the tool. The tool view, implemented in the `ToolView` class, consists of a main window with a menu bar across the top, and status bar across the bottom. The center of the window is used to display the application specific widgets – usually these include a drawing canvas for drawing nodes and arcs. The graphical components are the view’s counterparts to the model’s node, arc, and comment, the `NodeGraphic`, `ArcGraphic`, and `CommentGraphic`. As in the case of the model, the `ToolView`, `NodeGraphic`, and `ArcGraphic` classes are abstract and must be extended to have application specific behaviour added. The `CommentGraphic` and other view classes

described below are provided to be used as is or to be extended as required.

The `DrawingCanvas` class is an extension of the HTK class `CompoundDrawing` providing the basic drawing canvas used in the `ViewDrawingPanel` class derived from the HTK's `DrawingPanel`. Together, these two classes form the basis for drawing.

The following classes:

```

    AlignedText           ViewTextField  ViewButton    ViewSizedLabel
    ViewCheckboxMenuItem  ViewModeButton ViewMenuItem

```

provide simple widgets that mimic their superclasses but which have the `View` interface. This is obviously not a complete widget set and should be added to as other widgets are wrapped to have the `View` interface.

4.4.7 The Control Classes

The `Control` abstraction presents the control counterpart to the model and view abstractions. There is a tool control and controls for components within the tool. The `ToolControl` class provides the functionality to register the tool model with the tool view and vice versa. It also contains generic code for handling command indications from both the tool model and tool view. Analogous to the model node and arc and view node and arc are the `NodeControl` and `ArcControl` classes. These are designed to map the model arc and node classes onto the view arc and node and vice versa. The `ToolControl`, `NodeControl`, and `ArcControl` classes are abstract classes and must be extended to have application specific functionality added.

4.4.8 The Indication Classes

The model and view interact with each other through the delivery of and response to indications. The model aspects generate `Aspect` instances, derived from `Indication`, which are sent to features of the view. The view's features generate `Occurrence` instances, also derived from `Indication`, which are delivered to the model's aspects. These indications

identify the source of the indication and carry a payload of information concerning the status of the source or a change to the source.

4.4.9 Indication Exceptions

The indication mechanism is very general but does not have the desirable feature of “strong typing”. It is possible for a model aspect to deliver an aspect indication to a view feature that is either not expecting an indication from that model aspect or that is expecting a payload of some other type. The first type of error raises an `AspectIDException`, the second raises a `IndicationValueException`. If the view does not have a feature that the model aspect expects, the `FeatureIDException` is raised. Similarly, the view may encounter an `AspectIDException` if an occurrence indication is attempted to be delivered to a non-existent aspect, or it may encounter an `OccurrenceIDException` if the aspect rejects the feature source. If the payload data type is wrong, a `IndicationValueException` is raised.

4.5 The QNBApp Application

The Queueing Network Bounds application is composed of three packages and one stand-alone class. There is a package each for the the model, view, and control abstractions and the remaining stand-alone class is the main application class that contains the application entry point. Figure 4.7 shows the inheritance relationships between the classes of the application specific code. As can be seen from the figure, the class structure for the application code is quite simple. The `QNBApp` class is the main class of the application which contains the

```
public static void main( String[] args )
```

that all Java applications must implement. The sole purpose of this method is to start the MVC framework by creating the application specific control object embodied in the `QNBControl` class. This class and the other application specific class are described in the following.

Figure 4.7: The classes and inheritance relationships in the QNB application.

The application specific functionality of the tool is incorporated into the `QNBTool` class which is an extension of the generic `Tool` class of the MVC package. The other model classes are abstractions for the queueing network model classes: locks and servers (resources), and visits. There are corresponding view and control versions of these as well, as is indicated in Figure 4.7.

The model parameters such as think time as well as customer priority, visits, and service demands are stored in the model classes, `Class`, `Visit`, and `Server`, respectively. The corresponding view classes display the parameters appropriately on the view and provide a means for the user to edit them. This is all done through application specific specialization of generic framework structure and behaviour.

As described in Section 3.2, a customer can have different priorities at different resources which requires a vector of priorities to be associated with each customer class. Because this complicates the parameter specification process, the current version of the tool supports the association of a single priority for a class at all resources.

4.6 Graphical User Interface Operation

In this section, some scenarios are followed to show how the design of the MVC framework and QNB application functions. The first scenario is that of the initialization of the application. Next, the creation of a customer class is presented. Lastly, a scenario where a primary resource, or lock, is moved on the drawing canvas is described.

4.6.1 Tool Initialization

The flow of control in any Java program begins with the invocation of the class method `main` in the main application class of the Java program. In this case, that main method creates the one and only instance (a singleton) of the `QNBControl` class. This instance then creates singletons of both the `QNBModel` and `QNBView` classes. The `QNBControl` singleton then registers all of the `QNBModel` instance's aspects to receive the `QNBView` instance's published occurrence indications. It also registers all of the `QNBView` instance's features to receive the

QNBModel instance's aspect indications. The QNBControl singleton then forces the model and view instances to broadcast their respective indications so that the view and model each have the most up to date information about each other. The QNBControl singleton then passes the flow of control over to the QNBView instance which starts the graphical user interface event handling. From this time on, all scenarios are initiated by events from user interface.

4.6.2 Creating A Class Component

When the user puts the application into "add class" mode by selecting the Add Class mode button, and then clicks the left mouse button on the drawing canvas, a new customer class is added to the queueing model being developed by the user. The scenario begins with the "mouse click" event being interpreted by the QNBView singleton and transformed into an Occurrence indication which is then delivered to the QNBModel instance. The indication carries as its payload the location on the drawing canvas of the event. This location is saved in the QNBModel instance as the *selected point*. The QNBModel then issues a command indication to the QNBControl instance telling it that a new customer class is to be created. The QNBControl instance uses the appropriate methods in the QNBModel and QNBView instances to create the model and view components, instances of Class and ClassGraphic, respectively. As the ClassGraphic instance is created by the QNBView singleton, it is added to the current drawing and it appears on the drawing canvas. The QNBModel, during the creation of the Class instance, records the instance in its internal data structures. These instances are then passed as arguments by the QNBControl to the creation of an instance of ClassControl which registers the model and view components with each other for the delivery of indications. The new class is now ready to be used.

4.6.3 Moving a Lock

Suppose that a lock has been added to a queueing model such that it is connected to one or more classes and servers. An example of this is the lock p1 shown in Figure 4.1. If

the user places the tool into “move mode” by selecting the Move mode button and then clicks the left mouse button over the lock object on the drawing canvas. the following series of actions takes place. The location of the mouse pointer is transformed by the `QNBView` instance into an indication that is delivered to the `QNBModel` instance where it is recorded as the *selected point*. The collection of components within the `QNBModel` instance is then searched for a member whose coordinates match the selected point. This object is the lock. p1. That object is marked as the selected component. Now if the user moves the mouse the new location is again delivered to the `QNBModel` instance. This time, the selected object’s `setLocation` method is invoked in order to move the component to its new location. This move not only affects the location of the selected component, but the lock also informs all the arcs connected to it, all the labels attached to it and any other object that needs to know about change of location. Each of these objects update its location aspect and broadcasts the change to its peer view objects. In this way, just by moving the one selected object, all the objects that needed to be moved in order to keep the model drawing consistent are also moved. If the user clicks the left mouse button in the new location, the `QNBModel` instance resets the selected component to null (i.e. no selected component) and the lock remains in its new location.

4.6.4 Solving A Model

The `QNBModel` class is responsible for writing the model description file, invoking the separate analysis tool, and reading the results file. The `QNBModel` singleton first walks the model, starting with the classes and following all visit arcs to the locks and servers. As the nodes and arcs are traversed, the model parameters are printed to the model description file. Once the file is complete, the `QNBModel` singleton forks a child process which embodies a UNIX shell command that performs the bounds computation. When the child process terminates, the `QNBModel` singleton reads the results and posts them to the appropriate class nodes in the model.

4.7 CLP(BNR) Bounds Analyzer

The Bounds Analyzer software consists of a single BNR Prolog context which consists of approximately 4000 lines. These lines are separated into files where each file contains code that implements a distinct part of the analyzer. The following is a description of the different files and the functions they implement.

qnbmain.clp This is the main program body. The entry point to the program, `qnb/1`, is implemented by the code in this file.

qnbmodel.clp This code implements the tabular data structure that represents the queueing model which is used during the analysis.

qnbclass.clp This code implements a data structure to represent a class in the model. The class structure is used in the table implemented by `qnbmodel.clp`.

qnbsvc.clp This implements a data structure to contain the information needed to represent the service of a particular resource by a particular class.

qnbncub.clp This file contains the code that implements the *no-contention* upper bound inequalities.

qnbutub.clp This file contains the code that implements the *utilization based* upper bound inequalities.

qnbgenlb.clp This code implements the high level functionality that computes the lower bounds. The details are handled in several separate files.

qnbwait.clp This file contains the waiting-time expressions that are used by the code in the module `qnbgenlb.clp`.

qnbpri.clp This file contains the mathematical expressions to handle calculations related to primary resource delays.

qnbsec.clp This file contains the mathematical expressions to handle calculations related to secondary resource delays.

qnbsmath.clp This file contains the code that handles symbolic mathematics.

qnbutil.clp This file implements several miscellaneous utilities that are used by the code in the other files.

qsort.clp This file implements the *quicksort* sorting algorithm that is used in the lower bound calculations for lower priority contenders of a non-preemptive priority resource. Recall from Chapter 3 that in these calculations, the contenders are sorted in non-increasing order of service demand at the NPP resource.

The next two sections cover some of this functionality in slightly more detail focusing first on the data structures used to represent the model and then on the general form of the algorithms used to solve the inequalities.

4.7.1 The Prolog Data Structures

Understanding the data structures used in the analyzer is key in understanding how the analyzer works. The data structures consist of two abstract data types, one to represent the class data and one to represent the data concerning a service of a class by a resource, and tables to map the class and service information to the individual classes and resources.

High Level Data Structures

Before the particulars of the data structures used in the implementation of the bounds solver are discussed, a brief explanation of Prolog data structures in general is given. Prolog consists of *terms* and *operators* that are assembled into *clauses* that represent logical declarative statements. These terms may be simple, consisting of only a single member, as in the following examples: `father`, `121`, `3.1415927`, and `a`. However, Prolog also permits compound terms that are composed of a *functor* and a list of arguments in the following cases: `fraction(1,2)`, `mother(mary, john)`. The number of arguments in the term is

called its *arity*. In Prolog, functors may be overloaded because any terms having the same functor but different arities can never be unified. The notation

functor/arity

is often used to name a particular kind of term. Using the above examples, we may write: *father/0*, *mother/2*, and *fraction/2*. For a more complete discussion of Prolog, including the topics of operators and clauses, consult the *BNR Prolog User's Guide* [9] or any Prolog textbook.

The tabular form of the data structures is shown in Figure 4.8. In the figure, three

Figure 4.8: The Prolog tabular data structures used within the bounds analyzer software.

tables can be seen. On the left is a single column table that is a vector of classes. Each row in the table contains a single cell which holds information representing a class in the model. In the figure, the cells in this left-most table are marked c_c where the subscript indicates the class c . At the bottom of the figure, it is indicated that the cell indicated by c_c

actually refers to a Prolog structure `class/5`. Next to the class vector is a matrix of primary resource services. Each row in the matrix corresponds to a class, the same class as in the same row of the class vector. Each column of the matrix corresponds to a primary resource. Each cell in the matrix has a label p_{cr} which contains information related to class c and primary resource r . This information is contained in a `service/7` Prolog data structure as indicated at the bottom of the figure. The rightmost table in the figure is another matrix of service information. This matrix relates the classes and secondary resources. Again, the rows correspond to the classes and the columns correspond to the resources. The cells are labeled s_{ck} for class c and secondary resource k . The data is contained in the Prolog data structure `service/7` as was the case for the primary resources.

Abstract Data Types

The `class/5` and `service/7` Prolog data structures are treated in the analyzer code as abstract data types. That is, the internals of these structures are inspected and modified only by routines defined in the `qnbclass.clp` and `qnbsvc.clp` files, respectively. This allows for major changes to the number and type of fields stored in these structures without necessitating major changes throughout the remainder of the code.

The Class/5 Data Structure

The `class/5` data structure is shown in Figure 4.9. The data contained in the structure as shown in the figure consists of the name of the class (in the input file to the analyzer, the classes and resources are referred to by textual identifiers, not numeric indexes as in the mathematical model), the index or row number (c) of the class, an interval representing the class frequency, f_c , a real valued scalar indicating the average think time, Z_c , and the integer number of customers in the class, N_c .

The Service/7 Data Structure

The `service/7` data structure is shown in Figure 4.10. The data contained in the service

class/5

Name	Index	f_c	Z_c	N_c
------	-------	-------	-------	-------

Name	Class name
Index	Integer valued class index (row number)
f_c	Class frequency, a real valued interval variable
Z_c	Class think time, a real valued constant
N_c	Number of customers in class, an integer constant

Figure 4.9: The **class/5** record of the bounds analyzer data structure.

structure is a bit more complicated than that of the class structure. Note first of all, that the position of the **service/7** within the tabular data structure determines which class (c) and which resource (k) are being described. The **service/7** structure consists of the name of the resource, an integer index that is either zero if the resource is a secondary resource or equal to the index of the primary resource r as used in the mathematical model, and a list of visit information records.

Each cell of the visit information list, marked v_r , contains an instance of a **vdata/2** structure. As indicated at the bottom of Figure 4.10, the **vdata/2** structure contains the visit count $V_k(c_r)$ and the service demand $S_k(c_r)$ for a particular resource and a particular class. The first two-tuple in the list is for direct visits of the class c to the resource k ($V_k(c_0), S_k(c_0)$). If the resource k is a primary resource, then this is the only entry in the list of **vdata/2** structures. If the resource k is a secondary resource, then the remaining two-tuples in the list refer to the primary resources r , $r = 1 \dots R$, in the system. If the class does not visit the resource k through a particular primary resource r , or if there are no direct visits by the class c to the resource k , then the corresponding entry in the visit list is **nil**.

Figure 4.10: The **service/7** record of the bounds analyzer data structure and the **vdata/2** record it contains.

The next two data elements in the `service/7` structure are the scheduling policy of the resource k and the priority that the class c has at this resource. The data structures permit classes to have different priorities at different resources.

If the resource k is a primary resource, the next data element is a list of intervals representing the quantity $Q_r(c_0, m_0)$. The list contains an entry for each $m = 1 \dots C$. If the resource is a secondary resource, this data element is `nil`. The last data element in the structure is the number of instances. This is an integer that indicates the number of exact duplicates of this resource. Multiservers are not handled in the model: each of the exact duplicates has its own queue.

4.7.2 Analysis Algorithms

The following sections briefly describe how the inequalities are computed. First, some general principles are discussed.

Each inequality is formed symbolically. A Prolog data structure consisting of all the variables and operators is formed in memory. This mathematical expression is then simplified symbolically, taking advantage of multiplication by zero or one, addition of zero, division by identity, etc. to reduce the expression. Once this is completed, the simplified inequality is asserted into the CLP(BNR) constraint network maintained by the CLP(BNR) engine. As each inequality is asserted, the CLP(BNR) engine performs narrowing operations on all the interval values in the constraint network until they are all as narrow as possible while still fulfilling the truth requirements of the inequalities. When all the inequalities have been so asserted, the upper and lower bounds of the resulting constrained throughput intervals are printed as the results.

Because the inequalities are generated symbolically, an option exists to print them before they are solved. This printing is done in `LATEX` code since it is easy to automatically generate, format, and print mathematical expressions in `LATEX`. After formatting, it is straightforward to check the inequalities term by term against the model theory.

No Contention Bound

The no-contention bound is formed for a particular class by taking the row that corresponds to that class and summing across all resources in that row, adding the contention according to the expressions in Chapter 3.

Utilization Bound

The inequalities for the utilization bound are obtained for a particular resource by taking both the corresponding column of the appropriate service matrix, and the class vector, and summing over the rows adding the utilization according to the expressions in Chapter 3.

Lower Bound

The inequalities for the lower bound are obtained for a particular class by summing over the class vector and various rows and columns of the two service matrices according to the inequalities presented in Chapter 3. The algorithm to compute the lower bound is quite complicated as a result of the differences between the various type of resource scheduling. For example, it is necessary to sort collections of services for the non-preemptive priority scheduling cases.

4.8 Supporting Software

In addition to the Java user interface and the CLP(BNR) analyzer there is a C program called `qnb` that is a “wrapper” for the solver. It handles command line options, file arguments, and any operations that are better handled in C than in Prolog. Also, the C program executes shell and `awk` scripts to invoke BNR Prolog and to parse the results. The parsed results make it easier for the Java interface to obtain the results of the analysis for display to the user.

4.9 Discussion

The design and implementation of applications with graphical user interfaces is simplified by the MVC framework. The application specific code is straightforward to design and understand.

The use of the aspect/occurrence delivery system means that the model and view abstractions can be highly decoupled. It also allows for the decentralization of control to maximize polymorphism among model components and view components. Aspects and features are inherited allowing the Java inheritance systems to be used to maximize code reuse.

The HTK and MVC packages amount to approximately eleven thousand lines of Java code. The application specific code for the QNB application is about thirty-four hundred lines. All the Java code with the exception of the `ErrorDialog` and `MultiLineLabel` classes were written by Alex Hubbard, these latter two classes were written by Marc Courtois. At the time that the graphical interface used in this thesis was written, GUI builders and generic frameworks such as MVC were not widely available at low cost for Java. So development of the GUI was done using only the standard Java graphics interface.

Chapter 5

Validation

5.1 Introduction

This chapter demonstrates that the analysis software correctly implements the mathematical inequalities of the queueing network model given in Chapter 3 and that the model itself correctly bounds the class throughputs.

In Section 5.2, the technique that was used to determine whether the analyzer software matched the bounds inequalities is discussed. The analyzer software is then used in Section 5.3 in two case studies to show that the model does bound the class throughputs and to demonstrate some of the features of the modelling technique.

Section 5.4, discusses some of the difficulties encountered in using the bounds analysis technique including some problems with the CLP(BNR) system that was used to implement the bounds computations.

5.2 Validation of The Analyzer Software

5.2.1 Checking The Inequalities Symbolically

In Section 4.7.2, it was indicated that the bounds analyzer performed the bounds computations symbolically and that it was only at the final stage of the computation that the

the inequalities were added to the CLP(BNR) constraint network to actually bound the class throughput intervals. An option has been added to the bounds analyzer to cause the symbolic inequalities to be printed in $\text{\LaTeX}$ format rather than to be solved numerically. The $\text{\LaTeX}$ formatted inequalities can then be printed and inspected by hand to determine if they correctly match the expressions required by the bounds model theory discussed in Chapter 3.

This comparison was done for various queueing network systems ranging from the most simple to more complex networks. During the preliminary testing of the software, this comparison technique proved invaluable in revealing programming errors in the software. In each case where a discrepancy was observed, the programming error responsible was corrected. In the final testing of the analyzer, the automatically generated expressions were found to be the same as those derived by hand. Figure 5.1 shows an abbreviated sample of the automatically generated $\text{\LaTeX}$ and its formatted counter part.

```

\noindent{\bf No-contention bound for }$f_{c2}$
\begin{equation}
\begin{array}{lcl}
1 \ \& \geq \ f_{c2}(Z_{c2} \ \& \\
& + \ V_{k1}(c2_{0})S_{k1}(c2_{0}) \ \& \\
& + \ V_{k2}(c2_{0})S_{k2}(c2_{0}) \ \& \\
& + \ V_{k3}(c2_{0})S_{k3}(c2_{0}) \ \& \\
& + \ V_{k4}(c2_{0})S_{k4}(c2_{0}) \ \&
\end{array}
\end{equation}

```

Figure 5.1: A sample of an automatically generated $\text{\LaTeX}$ symbolic inequality and its formatted counter part.

5.2.2 Regression Tests

As indicated in Chapter 2, research on bounds analysis for queueing networks with non-FIFO scheduling but without shared resources has already been published [36]. The bounds analyzer was used on each of the example systems of [36] as a regression test to verify that the analyzer correctly handled these older systems. In every case, the results produced by the new analyzer exactly matched the results of the previous analyzer.

5.3 Validation of The Bounds Computation

Two case studies were done to validate the queueing network bounds technique. In the first case study, a model of a client-server system used by a telemarketing company based in part on the example in Chapter 7 of the book “Capacity Planning and Performance Modeling” by Menasce, *et.al.* [37] is studied. The model for the system is derived and the work-load determined. The model is solved and the results are analyzed. A discussion of the “feasible throughput region” which is guaranteed to contain the class throughputs is presented. Next, the client-server system enhanced by the addition of a second server is described. The effect this has on the performance of the system is determined. A final modification to the client-server system is made by changing the scheduling at the server to a priority based scheduling policy.

The second case study consists of a more complex queueing network model that features a larger number of classes and resources and a more varied selection of scheduling policies.

5.3.1 A Client-Server System

In this section, a simple two class client-server system with one shared resource and two non-shared resources where all resources have FIFO scheduling policies is considered.

Description of The System

Consider the operations of a telemarketing company like the one described in Chapter 7 of [37]. Customers call a one-eight-hundred number to place orders from a catalog. A second one-eight-hundred number is available to check on their orders. There is a pool of telephone operators ready to take calls on the two numbers. A number of operators, determined by the proportion α , are dedicated for taking orders. The rest of the operators only check on orders.

The company intends to use a client-server architecture to handle its customers’ calls. Each telephone operator has a PC (client) with which he/she places orders or checks on orders stored in the server. In addition to customer/order information, the operator can also

call up a graphical display of any product in the catalog. The workstations are connected by an Ethernet to the server running on a workstation equipped with a CPU and two disks. One disk stores the telemarketing orders database while the other stores the product images in a compressed format.

Based on the company's previous experience and on a client-server prototype that they had developed, the following information was obtained (see [37] for details). When a call to place an order comes in, the operator spends an average of 18 seconds interacting with the client. Each order results in, on average, four SQL requests for the database and two image requests for the image repository. Each SQL request generates an average response of 1000 bytes. The average size of the compressed image file generated by a response is 100 Kbytes. Each SQL transaction requires 0.12 seconds of CPU time and 0.054 seconds of time at the database disk. Each image transaction requires 0.04 seconds of CPU time and 0.3 seconds of time at the image disk. When a call to check the status of an order is made the operator spends ten seconds interacting with the customer and one SQL request is made to the database. One in ten such calls requires that an image be retrieved and displayed on the operator's workstation. The workstations that are used by the operators take 1.5 seconds to uncompress and display an image. The time spent by the workstations processing the SQL requests and replies is small and can be ignored. The company is considering using either a single CPU server or a multi-processor sever with an option of two or three CPU's.

Consider a situation in which the company is expecting a growth in workload which is handled by increasing the number of operators. An analysis of the degradation in performance is important in such a situation. On the other hand, if the system is underutilized the company may be interested in adding a new application: the payroll computation that is currently processed by a mainframe could be transfered to the client-server system. Whether or not the payroll processing workload can be handled by the system without seriously deteriorating response time for operator queries requires analyzing system performance. Also, consider the designer of the server subsystem who is interested in improving the performance of the server by incorporating multiple processors. The improvement in performance

in response to the addition of each processor needs to be investigated for understanding the price-performance benefit for the new system.

Although approximate average performance measures may be acceptable in certain situations a performance guarantee is often required in client-server systems. A manager in the telemarketing company is interested in making sure that the mean response time is not larger than a specified worst case value (see [37]). Predicting performance through exact analysis is difficult for many real systems because the underlying assumptions required are not satisfied. Approximate analysis is unusable because it does not provide a performance guarantee. Using simulation is possible but is expensive both in terms of simulator development as well as execution time. The bounds analysis presented in this thesis is capable of computing a worst case value for system performance and is a cost effective solution in such a situation. Moreover the upper and lower bounds in system throughput can be used to delineate a feasible region that is guaranteed to contain the system throughput.

Deriving A Model with Workload Parameters

The telemarketing system described above is conveniently represented by a queueing network model (see Figure 5.2). There are two classes of clients: one class ($c = 1$) to represent the workstations which are used by the operators who take orders and one class ($c = 2$) to represent the workstations used by the operators who handle queries. There are three secondary resources in this system: the network (LAN), and the two disks (DK1 and DK2). There is one primary resource in this system: the server process and the CPU on which it runs. Since the CPU runs only the server process, both of them can be modelled by a single primary resource. The server process must be modelled as a primary resource since any request received at the server is executed until it is completed before another request can be handled. This means that the clients access and then “hold” the server process while the disks are used for servicing the request.

The workload parameters of the queueing network model are to be derived from the description of the system. The derivation of the parameters for the order class from the

Figure 5.2: The queueing network model for the Telemarketing System.

given information is presented. Taking the proportion of “order” operators, α , to be 0.8, the total number of clients in the order class is $0.8N$ where N is the total number of operators¹. The *think time*, Z_1 , of the order class is the time the operator spends interacting with the customer plus the processing time at the operator’s workstation, 21 seconds². For each order, there are four SQL requests to the server each needing 0.12 seconds on the CPU and two image requests to the server each needing 0.04 seconds on the CPU. These six visits in total, each of an average of 0.0933333 CPU seconds³, are indicated in Figure 5.2 by the dashed arc connecting the order class to the server. Each SQL request requires one access to the database disk, DK1, and each image request requires one visit to the image disk, DK2. Therefore, of the six visits from the order class to the server, two thirds of them require a visit to DK1 and one third require a visit to DK2. Each SQL access to DK1 takes 0.054 seconds and each image access to DK2 takes 0.3 seconds. These visit counts and service demands are marked on the dashed arcs connecting the arc from the order class to DK1 and DK2 respectively.

The rate at which an Ethernet can transmit data depends on the physical dimensions of the network, the number of workstations connected to it as well as the packet and header sizes. An approximate expression for the number of packets per second that an Ethernet can transmit is given in [24]:

$$\mu_p(n) = \frac{1}{\overline{L_p}/B + SC} \quad (5.1)$$

where n is the number of stations connected to the network, $\overline{L_p}$ is the average packet length in bits, B is the bandwidth of the network in megabits per second, S is the slot duration of the network (the time required to detect a collision) in microseconds and C is the average number of collisions given by $(1 - 1/n)^{-(n-1)} - 1$. Table 5.1 gives the values of these parameters used in the analysis of the telemarketing example as specified in [37].

¹In the experiments described here, N was always taken to be a multiple of 10 so that $0.8N$ would not involve any fractional component.

²The think time of 21 seconds is given by 18 seconds to interact with the customer + 3 seconds to uncompress 2 images.

³The average CPU demand, 0.0933333, is given by $(4 \times 0.12 + 2 \times 0.04)/6$.

Table 5.1: The Ethernet parameters used in the Telemarketing client-server system.

<i>Name</i>	<i>Symbol</i>	<i>Value</i>
Bandwidth	B	10 Mbps
Slot duration	S	51.2 μ s
Average packet length	$\overline{L_p}$	1518 bits
Packet data length	L_d	1492 bits

We assume that each SQL and image *request* requires one Ethernet packet. To determine the number of packets required for an average SQL response, we divide the number of bits in the response by the number of data bits in a packet (given in Table 5.1) and take the ceiling: $N_p = \lceil (8 \times 1000) / L_d \rceil = 6$. The number of packets for an image, 537, is determined in the same way. Adding up all the packets required for the four SQL requests, two image requests, and their responses, we get 1104 packets in total. This will be the number of visits per cycle that the order class makes to the LAN resource. The service demand for a single packet is given by $1/\mu_p(n)$ where $\mu_p(n)$ is given by Equation 5.1 with $n = N + 1$ (N client workstations and 1 server). This information is indicated in Figure 5.2 as a label on the solid arc connecting the order class to the LAN resource.

The think time, visit counts and service demands related to the query class, all of which are given in Figure 5.2, are also obtained from the description above.

Analysis of the Results

Figure 5.3 shows the throughputs for both classes of clients of the Telemarketing System. Included on each plot are curves for upper and lower bounds. The mean value for the system throughput as computed by a simulation program is also shown. An Erlang-2 distribution for service times and a geometric distribution for number of visits were used in the simulation. Such distributions are consistent with the assumptions underlying the bounds as discussed in Chapter 3. All simulations reported in this paper were run sufficiently long such that the throughput intervals are within $\pm 5\%$ at a 95% confidence level.

The upper bound on the throughput of the order class X_1^+ is observed to increase with

Figure 5.3: The queueing network bounds and simulation results for class throughputs of the telemarketing client-server system.

N , the total number of clients and to level off around $N = 20$. This is due to the saturation of the primary resource. As more and more clients are added more work is done per second and the class 1 throughput increases initially. However, at $N = 20$, the primary resource saturates upper bounding X_1 at approximately 0.7. It is interesting to observe a slight drop in X_1^+ as N increases beyond 20. A similar effect is observed in the simulation result as well. This drop in throughput is due to the increase in service time at the network as more and more client workstations are added. The lower bound X_1^- increases with N and tends to level off at higher values of N . However, the knee of the lower bound curve is less sharp than that for the upper bound.

The upper bound in the query class X_2^+ does not exhibit the saturation effect observed with the order class. The upper bound on the utilization of a device is an increasing function of both X_1^+ and X_2^+ . Since X_1^+ decreases with N , for $N > 20$, the contribution of the order class to the utilization of the primary resource, the bottleneck device, decreases. This allows X_2^+ , the throughput bound for the query class, which is a “light” user of system resources, to continue to increase with an increase in N .

It is interesting to observe that the simulation value for the system throughput is closer to the upper bound for the order class and to the lower bound for the query class. Moreover, this set of simulation results holds only for the distributions described in the beginning of the subsection. Other distributions for service times and visit counts for resources can, in general, produce different values of X_1 and X_2 . Thus, in the absence of detailed knowledge of the stochastic behaviour of the system, lower bounds X_1^- and X_2^- provide a guarantee on system performance. This is often required in conservative design of systems.

The bounding technique can be used in engineering the performance of client-server systems. For example, one may decide on the number of operators required to achieve a specific guaranteed throughput for a specific class. The appropriate value of N can be read directly from the X_c^- ($c = 1$ or 2) versus N curve. Upper and lower bounds on response-time

for class c can be obtained from X_c^- and X_c^+ by applying Little's Law:

$$T_c^+ = \frac{N_c}{X_c^-} \quad \text{and} \quad T_c^- = \frac{N_c}{X_c^+}$$

Then response-time graphs can be used for answering such questions as: what is the maximum number of workstations that can be connected to the network such that the response-time for each class does not exceed a specified limit. The effect of using higher speed networks, CPU's, and disks on the performance bounds can be determined in a few seconds by using the bounds tool.

Tool Performance

An important attribute of a tool is its computational complexity. Since the tool described here uses CLP(BNR), the internals of which are not available for examination, the complexity of the tool was determined through comparisons of its solution times against simulation times. Table 5.2 lists the measurements of run-times for the bounds analyses and the simulations for ten different values of the number of clients. The simulations were executed on various workstations concurrently to reduce the overall simulation time. By comparing the runtimes obtained on the same machine, it is observed that for a number of experiments the bounds solver is at least three orders of magnitude faster than the corresponding simulation.

Feasible Throughput Region

When a system contains two or three classes of clients, a plot of the class throughputs can provide a better understanding of system performance. A plot of the class 2 throughput against the class 1 throughput is shown in Figure 5.4. Figure 5.4(a) shows all the various constraints on the throughputs: no-contention bounds for both classes, utilization bounds for all devices and general lower bounds for both classes. Figure 5.4(b) shows a closeup of the feasible throughput region. The box circumscribed around the feasible throughput region corresponds to the bounds computed by the tool. Due to the shape of this box, the bounds computed by the tool are referred to as *box bounds*. A more general discussion of

Table 5.2: The bound solution time versus simulation time for telemarketing system. Times marked with (†) are for Sun Ultra 1 workstations and those with (‡) are for HP 735 workstations.

<i>Number of Clients</i>	<i>Bounds (Seconds)</i>	<i>Simulation (mm:ss.ss)</i>
10	0.69‡	10:50.00†
20	0.71‡	13:44.43†
30	0.63‡	15:36.61‡
40	0.64‡	17:40.61‡
50	0.63‡	12:12.44‡
60	0.63‡	14:24.09†
70	0.64‡	19:24.47†
80	0.64‡	17:58.60‡
90	0.64‡	19:27.61‡
100	0.69‡	18:02.61‡

the feasible throughput region is presented in [36].

5.3.2 The Client-Server System with Two CPUs

An extension of the telemarketing problem with a dual CPU server is presented next.

The telemarketing company wants to consider using a multiprocessor server with two CPU's for its client-server system. What effect would this have on the response times experienced by the operators? Figure 5.5 shows the model for the dual CPU configuration. The same technique can be applied for three or more processors. In the dual CPU system, the workload formerly handled by the single CPU has now been divided evenly between the two CPU's by dividing the number of visits to each in half. This corresponds to a system in which each CPU runs its own copy of the server process. Figure 5.6 compares the dual CPU configuration ($m = 2$) with the single CPU configuration ($m = 1$).

The X_1^+ curve for the two CPU case increases with N until it saturates ($N = 40$) at a value twice that of X_1^+ for the single CPU case. This is expected since the best case speedup for doubling the number of processors is a factor of two. The major part of the X_2^+ graph which corresponds to the no-contention bound is unaltered by the incorporation

Figure 5.4: The throughput-throughput constraints curves showing the feasible throughput region for the Telemarketing client-server system with 20 clients.

Figure 5.5: The queueing network model for the Telemarketing System with a two-CPU server configuration.

Figure 5.6: The bounds and simulation results for the telemarketing client-server system with a single CPU server ($m = 1$) and a dual CPU server ($m = 2$).

of an extra CPU. The effect of server saturation is visible only at very high values of N and incorporating an extra processor produces little improvement in X_2^+ over the given range of N . Changing the number of processors does not alter X_1^- or X_2^- . The lack of sensitivity of both class 1 and class 2 lower bounds to the addition of an extra processor is due to the worst case behaviour assumed for the derivation of the lower bounds: whenever a tagged client visits a particular processor, all the other contenders choose to visit the same processor. This contributes to the spread between the upper and lower bound for a particular class. Knowledge of system behaviour could potentially be used for improving the lower bound and forms an important direction for further research.

5.3.3 Adding Non-FIFO Scheduling to the System

Suppose that the two servers are configured to treat the query class of client (class 2) as having a higher priority than the order class (class 1). This means that when a query request arrives at a server, it advances in any queue that might be present until it is ahead of any order requests already in the queue. The servers do not implement preemption, however, so that any request already in service is completed before the next can begin even if the request in service is of lower priority than the newly arrived request. Figure 5.7 shows the results for the dual CPU configuration with priority scheduling at the servers and with the query class having higher priority. In the figure, the bounds results are compared to results obtained from a layered queueing network approximation (LQN). The *lqns* [15] tool was used to generate these results.

Looking at Figure 5.7a, it can be seen that LQN results for X_1 are bounded by X_1^+ and X_1^- for both the FIFO and NPP cases. The bounds analysis indicates that the throughput for X_1 for the NPP case decreases with N more rapidly than in the FIFO case and the LQN results agree. Looking at the class 2 throughput, X_2 , in Figure 5.7b, it can be seen that FIFO and NPP upper bounds are the same. The reason for this is that, over the range of N shown, the class 2 throughput upper bounds do not saturate and hence the scheduling policy does not have much of an affect. Note however that the lower bounds *are* affected

Figure 5.7: The bounds results (indicated by X^- and X^+ with scheduling type FIFO or NPP) and LQN results (indicated by $X(lqn)$) for the telemarketing client-server system with a dual CPU server with non-preemptive priority scheduling at the primary resources. The query class (class 2) has higher priority than the order class (class 1).

by changes in the scheduling policy and that as expected the NPP lower bound is higher than the FIFO lower bound. In both the FIFO and NPP cases, the LQN X_2 curves are bounded above by X_2^+ and below by X_2^- .

Suppose that the priorities of the order and query classes are reversed. What effect would this have on the throughputs of the system? Figure 5.8 shows the results for the dual CPU configuration with priority scheduling at the servers and with the order class having the higher priority. As in Figure 5.7, LQN results generated by *lqns* are compared to the bounds results.

Looking at Figure 5.8a, the curves are similar to those of Figure 5.7a except that, as expected, the bounds in the NPP and FIFO curves are reversed so that the class 1 NPP throughput is slightly higher than the class 1 FIFO throughput. This is due to the fact that class 1 is now favoured with a higher priority in the NPP case. In Figure 5.8b, the NPP and FIFO throughput upper bounds are again identical for the same reason as in Figure 5.7b. Notice however that the NPP lower bound quickly decreases to zero where it remains over the rest of the range of N . The NPP LQN result similarly decreases with N . The reason for this decrease is that the lower priority query class is being starved by the higher priority, heavier workload order class. Comparing Figure 5.7b and Figure 5.8b, it is apparent that putting the query class (class 2) at a lower priority than the order class (class 1) is detrimental to the customers who require the query service and provides only a marginal increase in performance for the customers placing orders.

5.3.4 A More Complex Example

In the last sections, sample two-class client-server systems were analyzed. Next, a slightly more complicated system involving an increased number of classes and resources with various other scheduling policies is investigated. Figure 5.9 shows the queueing network model representing the system. The visit counts and service demands are not shown on the figure in order to reduce the clutter; they are given instead in Table 5.3. The model consists of four classes three instances of the same type of primary resource and four secondary

Figure 5.8: The bounds results (indicated by X^- and X^+ with scheduling type FIFO or NPP) and LQN results (indicated by $X(lqn)$) the telemarketing client-server system with a dual CPU server with non-preemptive priority scheduling at the primary resources. The order class (class 1) has higher priority than the query class (class 2).

Figure 5.9: A more complex queueing network model. The three parameters to the right of each class are from top to bottom: The number of instances N_c , the think time Z_c , and the class's priority.

Table 5.3: Visit count and service demand parameters. Rows in which “*direct visit*” appears indicate direct visits from the class defined by the given row to the secondary resources indicated by the columns. The other rows are for indirect visits through the indicated primary resource, p1. For the case where V is zero, S can have no value and is simply marked with a “-”.

Class	Type of Visit	Primary		Secondary							
		p1		k1		k2		k3		k4	
		V	S	V	S	V	S	V	S	V	S
c1	direct	0	-	1.0	0.25	0	-	0	-	0	-
	indirect	-	-	0	-	0	-	0	-	0	-
c2	direct	0.5	1.5	0	-	0.5	1.0	0.5	1.0	2.0	1.0
	indirect	-	-	0	-	1.0	1.0	1.0	1.0	4.0	1.0
c3	direct	2.0	1.5	0.01	1.0	0.5	3.0	0.5	3.0	2.0	3.0
	indirect	-	-	0	-	0	-	0	-	0	-
c4	direct	0.167	1.4	0	-	0	-	0	-	0	-
	indirect	-	-	0.333	10.0	0.333	10.0	0.333	10.0	2.0	10.0

resources. Although not used in this example, it is also possible to have several identical instances of a type of secondary resource. The three identical primary resources have non-preemptive priority while the four secondary resources each has a different one of the four types of secondary resource scheduling: preemptive priority, FIFO, non-preemptive priority, and processor sharing. The case where the primary resources have FIFO scheduling instead of priority scheduling is also considered. The throughput bounds for the four classes are given in Table 5.4 for both cases. Looking at the throughput bounds of Table 5.4 it can

Table 5.4: The throughput bounds for NPP and FIFO scheduling at the primary resource.

Class	NPP Scheduling		FIFO Scheduling	
	At Primary		At Primary	
	X^-	X^+	X^-	X^+
c1	0.02210	0.02210	0.02210	0.02210
c2	0.00277	0.12442	0.00284	0.12500
c3	0.00078	0.16297	0	0.16289
c4	0	0.02801	0	0.02801

be seen that the bounds for class 1 are very tight: the upper and lower bounds on the throughput are identical for both the NPP and FIFO cases. This is because class 1 visits only server k1 and because it has the highest priority at k1, a server that has a preemptive priority queueing discipline. For these reasons, its throughput is completely deterministic with the upper and lower bounds inequalities reducing to a common simple expression.

As expected, changing the scheduling at the primary resources from NPP to FIFO has little effect on the bounds. The reason for this is that there are three instances of primary resource and only three of the customers use the primary resources. Since there is little contention at the primary resources, changing the scheduling policy there should not have a great effect on the bounds.

Some of the bounds are wide in this example. For example, in the case the customer class c4 the lower bounds could not be narrowed at all.

5.4 Problems Encountered

In this section, some of the more significant problems with the bounds analysis technique and software are discussed.

5.4.1 Bounds Looseness

A limitation of the bounds analysis technique results from an inherent property of constraint logic programming with intervals. When multiple intervals are combined in a mathematical expression, the more distinct intervals there are the more loose the resulting bounds tend to be. This problem can be difficult to diagnose as it often occurs unexpectedly. For example, the simple CLP(BNR) program listed in Figure 5.10 demonstrates this property. This tiny program computes the “same” inequality two different ways. In the first case, there are five identical intervals $[0, 10]$ which are summed and constrained to be less than 5 but greater than three. In the second case, there is only one interval and its product with 5 is constrained to be less than 5 and greater than 3. One might expect that, since $X_1 = X_2 = X_3 = X_4 = X_5 = X$, the expressions $X_1 + X_2 + X_3 + X_4 + X_5$ and $5X$ might

```

test :-
    % Declarations
    [X, X1, X2, X3, X4, X5]:real(0,10),

    % First case: summing
    {X1 + X2 + X3 + X4 + X5 < 5},
    {X1 + X2 + X3 + X4 + X5 > 3},

    % Second case: multiplicative factor
    {5 * X < 5},
    {5 * X > 3},

    % Printing the results
    print_interval_variables(    [['X1', X1],
                                ['X2', X2],
                                ['X3', X3],
                                ['X4', X4],
                                ['X5', X5],
                                ['X', X]] ).

    % Print the given labels and intervals.
    print_interval_variables( [] ) :- nl, !.
    print_interval_variables( [[N, X]|Ivars] ) :-
        range( X, [Xl, Xu] ),
        write( N, ': ', Xl, ', ', Xu, '\n' ),
        print_interval_variables( Ivars ).

```

Figure 5.10: A test program demonstrating how multiple identical intervals produces loose bounds while using a multiplication factor keeps bounds tight.

be equivalent. However, in interval arithmetic, this is not the case. Figure 5.11 shows the results of running this simple program. Notice that X_1 through X_5 have not been

```
?- test.
X1: 0.0, 5.0
X2: 0.0, 5.0
X3: 0.0, 5.0
X4: 0.0, 5.0
X5: 0.0, 5.0

X: 0.6, 1.0

?- test().
YES
```

Figure 5.11: The results generated by the simple test program. Note that the five identical bounds, X_1 through X_5 , are much looser than the single bound X .

constrained as tightly as X , as would have been desired. Now, take the queueing network used for the telemarketing client-server system of Section 5.3.2. The results of using a single primary resource type with two instances (replicated) to model the two CPUs of the system (as was done in Section 5.3.2) are compared to the results of using two distinct but identical servers (non-replicated). The model description file for each case is given in Figures 5.12 and 5.13. Table 5.5 lists the throughput results for the two cases. It can

Table 5.5: A Comparison of throughput results for replicated and non-replicated primary resources.

Class	Replicated		Non-Replicated	
	X^-	X^+	X^-	X^+
order	0.44381	1.36942	0.01070	1.45129
query	0.55616	1.92814	0.01456	1.92815

be seen from this comparison that although the two systems are identical, the results for the non-replicated case are looser than those for the replicated case. This effect is more pronounced for a similar comparison featuring three CPUs. The reason for this looseness

```

classes( [order, query] ).

instances( order, 80 ).
instances( query, 20 ).

think( order, 21 ).
think( query, 10.15 ).

primary_resources( [server] ).
instances( server, 2 ).

servers( [lan, dk1, dk2] ).

visit( order, lan, 1104, X ) :- lan_rate( X ).
visit( query, lan, 60.7, X ) :- lan_rate( X ).

lan_rate( X ) :-
    instances( order, No ),
    instances( query, Nq ),
    N is No + Nq,
    X is 0.0001548 + 0.0000512*(1.0/(1.0-1.0/(N+1)))**N - 1).

visit( order, server, 6.0/M, 0.093333333 ) :- instances( server, M ).
visit( query, server, 1.1/M, 0.112727273 ) :- instances( server, M ).

visit( order, server, dk1, 2/3, 0.054 ).
visit( order, server, dk2, 1/3, 0.3 ).

visit( query, server, dk1, 10/11, 0.054 ).
visit( query, server, dk2, 1/11, 0.3 ).

```

Figure 5.12: The model description file for the telemarketing client server system with two CPUs modeled by a resource type with two instances.

```

classes( [order, query] ).

instances( order, 80 ).
instances( query, 20 ).

think( order, 21 ).
think( query, 10.15 ).

primary_resources( [server1, server2] ).

servers( [lan, dk1, dk2] ).

visit( order, lan, 1104, X ) :- lan_rate( X ).
visit( query, lan, 60.7, X ) :- lan_rate( X ).

lan_rate( X ) :-
    instances( order, No ),
    instances( query, Nq ),
    N is No + Nq,
    X is 0.0001548 + 0.0000512*(1.0/(1.0-1.0/(N+1)))**N - 1).

visit( order, server1, 6.0/2, 0.093333333 ).
visit( order, server2, 6.0/2, 0.093333333 ).

visit( query, server1, 1.1/2, 0.112727273 ).
visit( query, server2, 1.1/2, 0.112727273 ).

visit( order, server1, dk1, 2/3, 0.054 ).
visit( order, server1, dk2, 1/3, 0.3 ).

visit( query, server1, dk1, 10/11, 0.054 ).
visit( query, server1, dk2, 1/11, 0.3 ).

visit( order, server2, dk1, 2/3, 0.054 ).
visit( order, server2, dk2, 1/3, 0.3 ).

visit( query, server2, dk1, 10/11, 0.054 ).
visit( query, server2, dk2, 1/11, 0.3 ).

```

Figure 5.13: The model description file for the telemarketing client-server system with two CPUs modeled by two distinct but identical resources (no replication).

is that the non-replicated cases have more interval variables in the upper and lower bound inequalities than the replicated cases where a multiplication factor is used instead. As can be seen in the inequalities given in Chapter 3, when replication is used, interval variables are replaced by fixed scalar multiplication factors (M_r for primary resources and M_k for secondary resources) which has the effect of tightening the bounds.

5.4.2 CLP(BNR) Bugs

Some queueing systems could not be solved using the bounds analyzer because of internal errors in the CLP(BNR) system. For example, the following test, which was distilled from a fairly large queueing network, causes the CLP(BNR) system to crash:

```
bug1 :-
    F:real(0,_),
    {F =< 1 / (4 + 0.089)}.
```

When this tiny program is run, the following behaviour is exhibited by the CLP(BNR) system:

```
?- bug1.
Run Time Error: Arithmetic type error
Enter 'y' if you want a traceback:y
Traceback:
  Executing -> error
  Called by -> $ria_relation
  Called by -> $braces
  Called by -> {}
  Called by -> $listener_evaluate
  Called by -> $$mult_soln
  Called by -> $mult_soln
  Called by -> $eval_query
  Called by -> $listener_execute
  Called by -> $listener
  Called by -> recovery_unit
  Called by -> capsule
  Called by -> $listener_loop
  Called by -> listener
  Called by -> listener
```

It is possible to “play” with some of the constants to get this test program to work. For example, the following test program is successful.

```
bug1a :-
    F:real(0,_),
    {F =< 1 / (4.089)}.
```

In each case where the CLP(BNR) inexplicably fails as in the first case above, the error always manifests itself as an error in the `$ria_relation` routine causing a stack trace as above. This exact behaviour was encountered from time to time in the attempt to solve queueing systems to validate the model.

5.5 Discussion

The technique of generating symbolic expressions in `LATEX` format and then printing these for debugging purposes proved to be invaluable in the initial debugging and later verification of the bounds analysis software. It should be noted, however, that the technique can only be reasonably used on very small and simple models. Larger, more complex models generate too many expressions containing too many terms for any manual analysis to be practical.

The model description file syntax for the bounds analyzer is backwards compatible with previous queueing network bounds analyzers making it trivial to use old models for regression testing.

The tool has been used in case studies showing how it can be applied to performance engineering problems where decisions are to be made on the basis of capacity and performance. The various features of the model were demonstrated.

The use of customer classes and server types which allows multiple identical modelling components to be represented by single interval instances goes a long way to keeping the bounds results tight. However, more research could be done in this area to try to tighten the bounds more in those cases where many distinct customer classes and server types are required to model a particular system.

This work was done using BNR Prolog version 4.1.3 Alpha. A newer version of BNR Prolog has been created at Nortel but could not be acquired for this work. Future work with CLP(BNR) should be done using a newer version in order to overcome the `$ria_relation` problems.

Chapter 6

Conclusions

6.1 Discussion

The queueing network bounds technique is useful for analyzing the performance of various types of systems. As shown in the the first case study of Chapter 5, performance modelling is an important tool for engineering a system to meet certain performance criteria, such as a maximum acceptable response time. In particular, the queueing network bounds technique presented in this thesis is well suited for this type of performance constraint since the lower bound produced by the modelling technique can be used as a performance guarantee.

The modelling technique can be used in a wider range of applications. The ability to handle various non-FIFO queueing disciplines and simultaneous resource possession make the technique applicable to modelling systems that many other queueing network based techniques cannot handle.

As a result of being an analytical technique, the bounds computation method presented in the thesis does not suffer from the same expenses that simulation techniques suffer. The bounds analysis tool computes bounds for moderately large systems in reasonable times. Compare, for example, the bounds solution times versus the simulation times for the telemarketing system discussed in Section 5.3.1. For a system of 100 clients, the bounds solution needed less than one second to complete whereas the simulation took over 18

minutes.

CLP(BNR) is an excellent platform for handling mathematical models particularly when bounds analysis is involved. It permits the model to be expressed in a natural, relational, way that is considerably simpler to program and debug.

The tool, QNBApp, provides a simple interface to the modelling technique. The user need not understand the inequalities involved in the computation of the bounds results. Instead the user of the tool need only understand the concepts of classes, resources, and visits. The tool guides the user in entering the parameters of the model and displays the resulting bounds directly on the graphical representation of the model.

6.2 Summary of Contributions

6.2.1 Formal Model

A complete formal mathematical derivation of the bounds for the extended queueing network model with simultaneous resource possession with FIFO as well as non-FIFO scheduling at queueing centres has been presented. Some preliminary work on FIFO shared resources used in this analysis was done by Majumdar [29]. Whereas non-FIFO scheduling at ordinary (non-shared) resources had been presented in [36], the work on priority scheduling at shared resources is new. In addition, the combination of shared resources and non-FIFO scheduling requires significant change in the structure of the model in terms of the inequalities used to constrain the throughput bounds of the system.

6.2.2 Bounds Analysis Tool

Approximately 4000 lines of CLP(BNR) were written for implementing the bounds analysis technique proposed in this thesis. The work starts from older software written by Hubbard for supporting the research described in [35] and [36]. To support simultaneous resource possession with non-FIFO scheduling policies, the older bounds analyzer software had to be nearly completely rewritten.

To provide an easy-to-use interface to the model, a graphical user interface written. The interface was written in Java and consists of approximately 15000 lines of code in total. The bulk of the interface code consists of a variant of the Model View Control application framework, a generic graphical tool building framework.

6.2.3 Validation of the Model and Tool

Two case studies were performed to test and validate the modelling technique in general and the bounds analysis tool that was built. Various hardware and software architecture design options are explored in the context of a telemarketing client-server system demonstrating many of the features and strengths of the modelling approach. A second case study further demonstrates the ability of the extended model to handle systems with shared resources and various scheduling disciplines.

The work on bounds analysis of systems with simultaneous resource possession but with FIFO scheduling only has been published [35]. Included in the paper are portions of the telemarketing case study.

6.3 Future Research

The following are some potentially fruitful areas for future research.

6.3.1 Tightening the Bounds

As indicated previously, the usefulness of any bounds model is intimately related with the tightness of the bounds it produces as results. Therefore, research could be done into techniques for tightening the bounds. This research could include, among other things, the following ideas.

Majumdar, in [29], derives “flow-based” inequalities that can further tighten the throughput bounds for systems with FIFO scheduled resources. The flow-based inequalities result from considering the balance between the input and output flow of traffic at a resource

and impose an upper bound on the expected number of visits of a resource made per second by a contending customer that can contribute to the mean queueing delay experienced by a particular customer at the resource. These flow-based expressions cannot be generalized to non-FIFO scheduled resources and were not included in the work presented here.

The scenario on which the lower bound of the model is based is *very* pessimistic: at any resource, each customer visiting that resource waits for all other customers before gaining access to the resource. Designing a system for achieving the performance guarantees may lead to low resource utilization in certain situations. Can this assumption be relaxed if knowledge of the routing in the system is known? If yes, it may be possible to take advantage of this information to improve the lower bound, thus tightening the bounds.

6.3.2 Model Extensions

Currently, only a single layer of shared resources is supported. It would be worthwhile adding the capability of handling multiple layers. The first extension in this area might be to handle multiple layers of shared resources with FIFO scheduling at all resources. A further extension would then be to handle the NPP scheduling at shared resources and the various other policies at non-shared resources.

There are other modelling components that could be included in the bounds analysis modelling technique, for example, *multi-servers* should be handled. The round-robin scheduling technique is currently approximated by the processor sharing scheduling policy. Extension to the model to handle round-robin explicitly could be added.

Bibliography

- [1] Applied Logic Systems, Inc., PO Box 180, Newton Centre, MA 02159 USA. *CLP(BNR) User Guide & Reference*, 1996.
- [2] G. Balbo and G. Serazzi. Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Performance Evaluation*, 30(2):115–152, July 1997.
- [3] R. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing*. Holt, Reinhart and Winston Inc., 1975.
- [4] R. M. Bryant, A. E. Krzesinski, M. K. Lakshmi, and K. M. Chandy. The mva priority approximation. *ACM Transactions On Computer Systems*, 2(4):335–359, November 1984.
- [5] M. Campione and K. Walrath. *The Java Tutorial, Object-Oriented Programming for the Internet*. Addison-Wesley Publishing Company, 1996.
- [6] J. G. Cleary. Logical arithmetic. Res. Rep. 86/235/9. Department Comput. Sci., University of Calgary, Calgary, 1986.
- [7] A. Colmerauer. An introduction to Prolog-III. *Communications of the ACM*, pages 70–90, July 1990.
- [8] Computer Research Laboratory, Bell-Northern Research Ltd., Ottawa, Ontario, Canada. *BNR Prolog Reference Manual*, 1988.
- [9] Computer Research Laboratory, Bell-Northern Research Ltd., Ottawa, Ontario, Canada. *BNR Prolog User Guide*, 1988.
- [10] B. J. Cox and A. J. Novobilski. *Object-Oriented Programming: an Evolutionary Approach*. Addison-Wesley Publishing Company, second edition, 1991.
- [11] M. Dincbas, H. Simonis, and P. van Hentenryck. Solving large combinatorial problems in logic programming. *Journal of Logic Programming*, 8(1&2):72–93, 1990.
- [12] D. Eager and K. Sevcik. Performance bound hierarchies for queueing networks. *ACM Trans. on Computer Systems*, 1:99–115, May 1983.

- [13] D. Eager and K. Sevcik. Bound hierarchies for multiple-class queueing networks. *Journal of the ACM*, 33(2), 1986.
- [14] G. Franks, A. Hubbard, S. Majumdar, J. Neilson, D. Petriu, J. Rolia, and M. Woodside. A toolset for performance engineering and software design of client-server systems. *Performance Evaluation*, 24:117–136, 1995.
- [15] R. G. Franks. *Performance Analysis of Distributed Server Systems*. PhD thesis, Carleton University, Department of Systems and Computer Engineering, Ottawa, Canada, Apr. 1998. In Progress.
- [16] W. C. Graham and S. Majumdar. The performance of multithreading and scheduling on client-server systems. In *Proc. IPCCC '97 Conference*, Tempe, Arizona, February 1997.
- [17] T. J. Hickey. CLP(F) and constrained ODEs. In Jourdan, Lim, and Yap, editors. *Proceedings of the Workshop on Constraint Languages and their use in Problem Modelling*. European Computer-Industry Research Centre (ECRC), Nov 1994. Technical Report ECRC-94-38.
- [18] E. Hyvonen. Constraint reasoning based on interval arithmetic. In *Proceedings of IJCAI*, pages 193–199, 1989.
- [19] P. Jacobson and E. Lazowska. Analyzing queueing networks with simultaneous resource possession. *Communications of the ACM*, 25(2):142–151, February 1982.
- [20] J. Jaffar and S. Michaylov. Methodology and implementation of a CLP system. In J. L. Lassez, editor. *Proc. 4th Int. Conf. on Logic Programming*. MIT Press, 1987.
- [21] B. Jenq, W. Kohler, and D. Towsley. A queueing network model for distributed database testbed system. *IEEE Transactions on Software Engineering*, 14:908–921, July 1988.
- [22] T. Kerola. The composite bound method for computing throughput bounds in multiple class environments. *Performance Evaluation*, 6:1–9, January 1986.
- [23] J. Kriz. Throughput bounds for closed queueing networks. *Performance Evaluation*, 4:1–10, January 1984.
- [24] E. Lazowska, J. Zahorjan, G. Graham, and K. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1984.
- [25] J. D. C. Little. A proof of the queueing formula $l = \lambda w$. *Operations Research*, 9:383–387, 1961.

- [26] J. Luethi, S. Majumdar, and G. Haring. Mean value analysis for computer systems with variabilities in workload. In *Proc. the 2nd Annual IEEE International Computer Performance & Dependability Symp. (IPDS)*, Urbana-Champaign (IL), September 1996.
- [27] J. Luethi, S. Majumdar, G. Kotsis, and G. Haring. Bounds-based performance analysis for distributed systems with variabilities and uncertainties in workload. In *Proc. Austrian/Hungarian Workshop on Distributed and Parallel Systems (DAPSYS'96)*, September 1996. [preliminary version of journal paper].
- [28] J. Luethi, S. Majumdar, G. Kotsis, and G. Haring. Performance bounds for distributed systems with workload variabilities and uncertainties. *Parallel Computing (Special Issue: Distributed and Parallel Systems: Environments and Tools)*, 22(13), March 1997.
- [29] S. Majumdar. Throughput bounds and performance guarantees for closed multiclass queueing networks with simultaneous resource possession. Department of Systems and Computer Engineering, Carleton University, 1992.
- [30] S. Majumdar. Application of relational interval arithmetic to computer performance analysis: a survey. *Constraints (Special Issue: Interval Constraints)*, 2(2), October 1997.
- [31] S. Majumdar and A. Hubbard. Performance bounds for client-server systems using a queueing network model. Technical report, Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa Ontario, Canada, K1S 5B6, 1997.
- [32] S. Majumdar and A. Hubbard. Computation of bounds and performance guarantees for client-server systems using a queueing network model. In *Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences: Advanced Technology Track*, Maui, Hawaii, January 1998. IEEE Computer Society Press, 10662 Los Vaqueros Circle, P. O. Box 3014, Los Alamitos, CA 90720-1314.
- [33] S. Majumdar and R. Ramadoss. Interval-based performance analysis of computing systems. In *International Workshop on Modelling And Simulation of Computer and Telecommunication Systems (MASCOTS'95)*, pages 345–351, January 1995.
- [34] S. Majumdar, C. Woodside, J. Neilson, and D. Petriu. Performance bounds for concurrent software with rendezvous. *Performance Evaluation*, 13(4):207–236, 1991.
- [35] S. Majumdar, C. Woodside, J. Neilson, and D. Petriu. Robust box bounds: Throughput guarantees for closed multiclass queueing networks with minimal stochastic assumptions. In *Proceedings of IEEE Infocom '92*, pages 8C.1.1–8C.1.10, Florence, Italy, May 1992.
- [36] S. Majumdar and C. M. Woodside. Robust bounds and throughput guarantees for closed multiclass queueing networks. *Performance Evaluation*, 32:101–136, 1998.

- [37] D. A. Menasce, V. A. F. Almeida, and L. W. Dowdy. *Capacity Planning and Performance Modeling*. Prentice Hall PTR, Englewood Cliffs, New Jersey 07632, 1994.
- [38] D. Michandani and P. Biswas. Characterizing and modeling ethernet performance of distributed DECwindows applications. tech. rep., Digital Equipment Corporation. Nashua, NH, U.S.A., September 1989.
- [39] R. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ. 1966.
- [40] R. Muntz and J. Wong. Asymptotic properties of closed queueing network models. *Proc. Eight Conf. on Information Sciences and Systems*, pages 348–352, 1974.
- [41] W. J. Older. The application of relational arithmetic to x-ray diffraction crystallography. Technical report, Bell Northern Research, PO Box 3511, Station C. Ottawa, Ontario, Canada, K1Y 4H7. February 1988.
- [42] W. J. Older. Application of relational interval arithmetic to ordinary differential equations. Technical report, Bell Northern Research, PO Box 3511, Station C. Ottawa, Ontario, Canada, K1Y 4H7. March 1995.
- [43] W. J. Older. Using interval arithmetic for non-linear constrained optimization. Technical report, Bell Northern Research, PO Box 3511, Station C. Ottawa, Ontario, Canada, K1Y 4H7. year unknown.
- [44] W. J. Older and F. Benhamou. Programming in CLP(BNR). In *PPCP'94*, Newport, RI (USA), 1993.
- [45] W. J. Older, F. Benhamou, and A. Vellino. Constraint logic programming on boolean, integer, and real intervals. Technical report, Bell Northern Research, PO Box 3511, Station C, Ottawa, Ontario, Canada, K1Y 4H7, 1994.
- [46] W. J. Older and A. Vellino. Extending prolog with constraint arithmetic on real intervals. In *Proc. Canadian Conference on Electrical and Computer Engineering*, Ottawa, 1990.
- [47] W. J. Older and A. Vellino. Constraint arithmetic on real intervals. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming: Selected Research*, pages 175–195. MIT Press, Cambridge, MA, 1993.
- [48] ParcPlace Systems, Inc., 1150 Plymouth Street, Mountain View, California 94043. *Objectworks/Smalltalk User's Guide*, 1990.
- [49] R. Ramadoss. Interval arithmetic-based analytical performance modelling. Master's thesis, Dept. Systems and Computer Engineering, Carleton University, 1994.
- [50] M. Reiser and S. Lavenberg. Mean-value analysis of closed multichain queueing networks. *Journal of the Association for Computing Machinery*, 27(2):313–322, April 1980.

- [51] Russian Research Institute of Artificial Intelligence. *Solving Problems with UniCalc. Brief User's Guide for UniCalc. Version 2.1*. 1994.
- [52] L. E. Stephens and L. W. Dowdy. Convolutional bound hierarchies. In *Proc. of the 1984 ACM SIGMETRICS Conf.*, pages 120–133. Cambridge, U.S.A. 1984.
- [53] B. Stuck and E. Arthurs. *A Computer and Communications Network Performance Analysis Primer*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632. 1985.
- [54] R. Suri. Generalized quick bounds for performance of queueing networks. *Computer Performance*, 5(2):116–120. 1984.
- [55] N. C. Wilhelm. A general model for the performance of disk systems. *Journal of the ACM*, 24(1):14–31, January 1977.
- [56] C. Woodside, S. Majumdar, and J. Neilson. Interval arithmetic for computing performance guarantees in client-server software. In *Proc. 1991 International Conference on Computing and Information (ICCI'91)*, pages 535–546, Ottawa, Canada, May 1991.
- [57] C. M. Woodside, J. E. Neilson, D. C. Petriu, and S. Majumdar. The stochastic rendezvous network model for performance of synchronous client-server-like distributed software. *IEEE Transactions on Computers*, 44(1):20–34, January 1995.
- [58] J. Zahorjan, J. N. P. Hume, and K. C. Sevcik. A queueing model of a rotational position sensing disk system. *Infor.* 16(3):199–216, October 1978.
- [59] J. Zahorjan, K. C. Sevcik, D. L. Eager, and B. Galler. Balanced job bound analysis of queueing networks. *Communications of the ACM*, 25:134–141, January 1982.

Appendix

Lemma 1 *Let u and v denote two sources of traffic that generate visits to a resource k with a FIFO queueing discipline. If λ_u and λ_v are the expected rates of visits to k from u and v respectively then:*

$$Q_k(u, v) \leq \min(1, \frac{\lambda_v}{\lambda_u})$$

Proof: A formal proof of the lemma is presented in [35]. The lemma states that if the rate of visit to k made by v is larger than the rate of visits made by u then in the worst case every visit of u can be delayed by a single service of v . If the rate of visits made by v is smaller than that of u then the probability that u is delayed by v is upper bounded by the ratio of the rates of the visits: λ_v/λ_u .

□

Lemma 2 *Let u and v denote two sources of traffic that generate visits to a resource k with a FIFO queueing discipline. Assume that service demands at k are NBUE distributed. If λ_u and λ_v are the expected rates of visits to k from u and v respectively and $S_k(v)$ is the expected value of service demand at k per visit of v , then the per visit delay experienced by u at k due to v is upper bounded as follows:*

$$d_k(u, v) \leq \min(1, \frac{\lambda_v}{\lambda_u}) S_k(v)$$

Proof: From Lemma 1, the expected number of times per visit source u can see source v

ahead of it at k is upper bounded as follows:

$$Q_k(u, v) \leq \min(1, \frac{\lambda_v}{\lambda_u})$$

Since the service demand are NBUE distributed, $S_k(v)$ is the upper bound on the expected time remaining to complete a service of v just starting or already in progress at k at the time that u makes its visit to k . Therefore the maximum queueing delay that u can experience per visit at k due to v is $Q_k(u, v)S_k(v)$:

$$d_k(u, v) \leq \min(1, \frac{\lambda_v}{\lambda_u})S_k(v)$$

□

Theorem 1 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have FIFO scheduling, then:*

- (a) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_0) \leq \min(1, \frac{f_m V_k(m_0)}{f_c V_k(c_0)})S_k(m_0)$$

Proof: Consider the direct visits of c to k coming from a message source u and the direct visits of m to k coming from message source v . Since the number of cycles per second the class c customer makes is given by f_c and the number of direct visits it makes to resource k per cycle is given by $V_k(c_0)$, we may write the rate of visits from source u , λ_u , as $\lambda_u = f_c V_k(c_0)$. Similarly, $\lambda_v = f_m V_k(m_0)$. Let $S_k(v)$ represent the service demand at resource k due to the class m customer so that $S_k(v) = S_k(m_r)$. If $S_k(v)$ is NBUE distributed, then the proof follows directly by applying Lemma 2.

- (b) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_{r'}) \leq \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) S_k(m_{r'})$$

Proof: The proof follows by applying Lemma 2 with the direct visits of c to k coming from a source u at a rate of $\lambda_u = f_c V_k(c_0)$ and the indirect visits of m to k holding r' coming from source v with rate $\lambda_v = f_m V_{r'}(m_0) V_k(m_{r'})$ and with the NBUE distributed per visit demand $S_k(v) = S_k(m_{r'})$.

- (c) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_0) \leq \min(1, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(m_0)$$

Proof: The proof follows by applying Lemma 2 with the indirect visits of c to k through primary resource r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) V_k(c_r)$ and the direct visits of m to k coming from source v with rate $\lambda_v = f_m V_k(m_0)$ and with the NBUE distributed per visit demand $S_k(v) = S_k(m_0)$.

- (d) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_{r'}) \leq \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(m_{r'})$$

Proof: The proof follows by applying Lemma 2 with the indirect visits of c to k through primary resource r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) V_k(c_r)$ and the indirect visits of m to k holding r' coming from source v with rate $\lambda_v =$

$f_m V_{r'}(m_0) V_k(m_{r'})$ and with the NBUE distributed per visit demand $S_k(v) = S_k(m_{r'})$.

□

Lemma 3 *Consider a resource r that has a priority scheduling queueing discipline. If u and v are two traffic sources that generate visits to r with rates λ_u and λ_v , respectively, and source v has a higher priority at r than u , then the expected number of times u can be delayed by v at r for a single visit of u to r is upper bounded as follows:*

$$Q_r(u, v) \leq \frac{\lambda_v}{\lambda_u}$$

Proof: Since r has priority scheduling and v has a higher priority than u , v can delay a single visit of u multiple times. The maximum number number of visits per second of v to r than can delay u is the total number of visits per second that v makes to r , λ_v . Since source u makes λ_u visits to r per second, the number of times that u can be delayed by v at r per visit of u is upper bounded by the ratio of these two rates:

$$Q_r(u, v) \leq \frac{\lambda_v}{\lambda_u}$$

□

Lemma 4 *Let u and v denote two sources of traffic that generate visits to a resource k with a priority queueing discipline. Assume that the service demands at k are NBUE distributed. If λ_u and λ_v are the expected rates of visits to k from u and v respectively, $S_k(v)$ is the expected value of the service demand that each visit of traffic source v requires of k , and traffic source v has a higher priority at k than traffic source u , then the delay experienced per visit by source u is upper bounded as follows:*

$$d_k(u, v) \leq \frac{\lambda_v}{\lambda_u} S_k(v)$$

Proof: From Lemma 3, the expected number of times per visit that source u can be delayed by source v at k is upper bounded as follows:

$$Q_k(u, v) \leq \frac{\lambda_v}{\lambda_u}$$

Since the service demands are NBUE distributed, $S_k(v)$ is the upper bound on the expected time remaining to complete a service of v that is yet to start or is already in progress at k at the time when u makes its visit of k . Therefore the maximum queueing delay that u can experience per visit of k due to v is $Q_k(u, v)S_k(v)$:

$$d_k(u, v) \leq \frac{\lambda_v}{\lambda_u} S_k(v)$$

□

Lemma 5 *Let u and v denote two sources of traffic that generate visits to a resource k with a preemptive priority queueing discipline. If λ_u and λ_v are the expected rates of visits to k from u and v respectively and $S_k(v)$ is the expected value of the NBUE distributed service demand that each visit of traffic source v requires of k , then the average delay per visit of u to k due to v is:*

$$d_k(u, v) \leq \begin{cases} \frac{\lambda_v}{\lambda_u} S_k(v) & \text{if } v \text{ has higher priority at } k \\ \min(1, \frac{\lambda_v}{\lambda_u}) S_k(v) & \text{if } v \text{ and } u \text{ have equal priority at } k \\ 0 & \text{if } v \text{ has lower priority at } k \end{cases}$$

Proof: Since the service demands are NBUE distributed, Lemmas 4 and 2 are applicable. There are three cases to consider:

1. If v has a higher priority at k than u , then by application of Lemma 4 we have:

$$d_k(u, v) \leq \frac{\lambda_v}{\lambda_u} S_k(v)$$

2. Now, if v has the same priority at k as u then the tie is broken by using a FIFO rule. In the worst case, v can appear ahead of u a single time in the server queue. Thus, by Lemma 2 we have

$$d_k(u, v) \leq \min(1, \frac{\lambda_v}{\lambda_u}) S_k(m_0)$$

3. Finally, if v has a lower priority at k than u , then u will preempt v if v is in service or overtake v if v is in queue at k . Therefore v can never delay u at k . Thus, we have

$$d_k(u, v) = 0$$

Putting these three cases together, we have our result.

□

Theorem 2 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have PP scheduling, then:*

- (a) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_k(c_0)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) S_k(m_0) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

Proof: The proof follows from Lemma 5 by recognizing the direct visits of c to k as coming from a traffic source u with traffic rate $\lambda_u = f_c V_k(c_0)$, the direct visits of m to k as coming from a traffic source v with traffic rate $\lambda_v = f_m V_k(m_0)$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(m_r)$.

- (b) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

Proof: The proof follows from Lemma 5 by recognizing the direct visits of c to k as coming from a traffic source u with traffic rate $\lambda_u = f_c V_k(c_0)$, the indirect visits of m to k through primary resource r' as coming from a traffic source v with traffic rate $\lambda_v = f_m V_{r'}(m_0) V_k(m_{r'})$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(m_{r'})$.

- (c) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(m_0) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

Proof: The proof follows from Lemma 5 by recognizing the indirect visits of c to k through primary resource r as coming from a traffic source u with traffic rate $\lambda_u = f_c V_r(c_0) V_k(c_r)$, the direct visits of m to k as coming from a traffic source v with traffic rate $\lambda_v = f_m V_k(m_0)$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(m_0)$.

- (d) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary re-*

source r' of a class m customer to k is upper bounded by the following inequality:

$$d_k(c_r, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \\ 0 & m \in \mathcal{L}_k(c) \end{cases}$$

Proof: The proof follows from Lemma 5 by recognizing the indirect visits of c to k through primary resource r as coming from a traffic source u with traffic rate $\lambda_u = f_c v_r(c_0) V_k(c_r)$, the indirect visits of m to k through primary resource r' as coming from a traffic source v with traffic rate $\lambda_v = f_m V_{r'}(m_0) V_k(m_{r'})$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(m_{r'})$.

□

Lemma 6 *This is Lemma 4 of [36], repeated here for completeness. Consider an ordered set of two-tuples $\{(y_i, x_i)\}$ (where $i = 0 \dots J$, $0 \leq x_i \leq x_i^+$, and $y_i \geq 0$) sorted in non-increasing order of y_i and a function $B = \sum_{i=0}^J x_i y_i$ that is to be maximized under the constraint: $\sum_{i=0}^J x_i \leq A$. Then given any set $\{x_i^+\}$ and $A > 0$, B is maximized when:*

$$\begin{aligned} x_0 &= \min(A, x_0^+) \\ x_i &= \min((A - \sum_{j=0}^{i-1} x_j), x_i^+) \quad \text{for } i = 1 \dots J \end{aligned}$$

Proof: If $x_0^+ \geq A$ then since $y_0 \geq y_j$ ($j = 1 \dots J$):

$$B_{max} = A y_0$$

If $x_0^+ \leq A$ then since y_0 is the largest of all y_j ($j = 0 \dots J$):

$$B_{max} = x_0^+ y_0 + B_{max}^{(1)}$$

where $B_{max}^{(1)} = \sum_{i=1}^J x_i y_i$ is a function that is maximized under the constraint $\sum_{i=1}^J x_i \leq$

$A - x_0^+$. The two previous inequalities can be combined as:

$$B_{max} = \min(A, x_0^+)y_0 + B_{max}^{(1)}$$

Substituting $A^{(1)}$ for $A - x_0$ where $x_0 = \min(A, x_0^+)$ and following the development of B_{max} , $B_{max}^{(1)}$ can be expressed in the following manner:

$$B_{max}^{(1)} = \min(A^{(1)}, x_1^+)y_1 + B_{max}^{(2)}$$

where $B_{max}^{(2)} = \sum_{i=2}^J x_i y_i$ is a function that is maximized under the constraint $\sum_{i=2}^J x_i \leq A^{(1)} - x_1$ where $x_1 = \min(A^{(1)}, x_1^+)$. This can be continued indefinitely until y_J is reached. Thus by defining $A^{(0)} = A$, and $B_{max}^{(0)} = B_{max}$ we get a series of $j = 0 \dots J$ equations of the form:

$$B_{max}^{(j)} = x_j y_j + B_{max}^{(j+1)} \quad \text{with} \quad x_j = \min(A^{(j)}, x_j^+)$$

where

$$A^{(j)} = A^{(j-1)} - x_{j-1} \quad \text{for } j > 0 \text{ and } B_{max}^{(J+1)} = 0$$

Adding these equations and cancelling the common terms on the right hand and left hand side yields:

$$B_{max} = \min(A, x_0^+) + \sum_{i=1}^J x_i y_i \quad \text{with} \quad x_i = \min((A - \sum_{j=0}^{i-1} x_j), x_i^+)$$

□

Lemma 7 *This lemma is based on Lemma 5 of [36]. Let u denote a source of traffic that generates visits to a resource k that has a non-preemptive priority queueing discipline. Assume that service demands at k are NBUE distributed. Let $l = 1 \dots L$ denote a set of classes of traffic sources that generate visits to k at a lower priority than u . Let N_l be the number of identical traffic sources in each class l . Let λ_u and λ_l be the expected rates of*

visits to k from u and the classes l . Let $S_k(l)$ be the expected value of the service demands at k for the sources in class l . Let $d_k(u, l)$ be the delay that source u experiences at resource k due to a visit of k by source l . Without loss of generality, let us assume that these classes are sorted such that for $a, b \in \{1 \dots L\}$ and $a < b$ we have $S_k(a) \geq S_k(b)$. Let $D_k(u)$ be the average delay per visit of u to k due to all the lower priority traffic sources. This delay, $D_k(u)$, is upper bounded as follows:

$$D_k(u) = \sum_{l=1}^L N_l d_k(u, l) \leq \sum_{l=1}^L N_l \alpha_k(u, l) S_k(l)$$

where

$$\alpha_k(u, l) = \begin{cases} \min(\frac{1}{N_l}, \frac{\lambda_l}{\lambda_u}) & l = 1 \\ \min(\frac{1}{N_l}(1 - \sum_{j=1}^{l-1} N_j \alpha_k(u, j)), \frac{\lambda_l}{\lambda_u}) & l > 1 \end{cases}$$

Proof: Since $\alpha_k(u, l)$ is the probability that the traffic source u is delayed at resource k by a particular source of class l , then by Lemma 1 we have:

$$\alpha_k(u, l) \leq \alpha_k^+(u, l) = \min(1, \frac{\lambda_l}{\lambda_u}) \quad (1)$$

Let $Q_k(u, l) \leq Q_k^+(u, l)$ be the expected number of class l visits that can delay a single visit of traffic source u at resource k .

$$Q_k(u, l) = N_l \alpha_k(u, l) \quad \text{and} \quad Q_k^+(u, l) = N_l \alpha_k^+(u, l) \quad (2)$$

If $s_k(l)$ is the time remaining needed to complete a service for a visit of a class l source to resource k measured from the time that u makes its visit to k , then the total delay experienced per visit by the source u due to the whole class l , $D_k(u, l) = N_l d_k(u, l)$, is given by:

$$D_k(u, l) = N_l d_k(u, l) = Q_k(u, l) s_k(l) \quad (3)$$

Since only one lower priority source can delay a single visit of u to k , we have:

$$\sum_{l=1}^L Q_k(u, l) \leq 1$$

To compute an upper bound on the queueing delay seen by the source u due to the all the lower priority sources, $\sum_{l=1}^L N_l d_k(u, l)$, we maximize $\sum_{l=1}^L Q_k(u, l) s_k(l)$ under the constraint $\sum_{l=1}^L Q_k(u, l) \leq 1$. By applying Lemma 6 with $A = 1$ and substituting $Q_k(u, l)$ for x , and the $s_k(l)$ for the y_i we have that $\sum_{l=1}^L Q_k(u, l) s_k(l)$ is maximized under the constraint $\sum_{l=1}^L Q_k(u, l) \leq 1$ when:

$$Q_k(u, l) = \begin{cases} \min(1, Q_k^+(u, l)) & l = 1 \\ \min((1 - \sum_{j=1}^{l-1} Q_k(u, j)), Q_k^+(u, l)) & l > 1 \end{cases}$$

Using Equation 2, we may write:

$$N_l \alpha_k(u, l) = \begin{cases} \min(1, N_l \alpha_k^+(u, l)) & l = 1 \\ \min((1 - \sum_{j=1}^{l-1} N_j \alpha_k(u, j)), N_l \alpha_k^+(u, l)) & l > 1 \end{cases}$$

Now, using Equation 1 and simplifying, we have:

$$\alpha_k(u, l) = \begin{cases} \min(\frac{1}{N_l}, \frac{\lambda_l}{\lambda_u}) & l = 1 \\ \min(\frac{1}{n_l}(1 - \sum_{j=0}^{l-1} N_j \alpha_k(u, j)), \frac{\lambda_l}{\lambda_u}) & l > 1 \end{cases} \quad (4)$$

Since the service demands at k are NBUE distributed, the upper bound on the expected time remaining to complete a service of l that is yet to start or is already in progress at k is given by $s_k(l) \leq S_k(l)$. Taking this into consideration and using Equation 2 in Equation 3 with Equation 4 we have:

$$D_k(u) = \sum_{l=1}^L N_l d_k(u, l) \leq \sum_{l=1}^L N_l \alpha_k(u, l) S_k(l)$$

where

$$\alpha_k(u, l) = \begin{cases} \min(\frac{1}{N_l}, \frac{\lambda_l}{\lambda_u}) & l = 1 \\ \min(\frac{1}{n_l}(1 - \sum_{j=0}^{l-1} N_j \alpha_k(u, j)), \frac{\lambda_l}{\lambda_u}) & l > 1 \end{cases}$$

□

Lemma 8 *Let u represent a particular source of traffic that generates visits to a resource k that has a non-preemptive priority queueing discipline. Let v represent a traffic source that contends with the traffic source u . Assume that the service demands at k are NBUE distributed. Let N_v be the number of identical traffic sources in each class v . Let λ_u and λ_v be the expected rates of visits to k from u and v respectively and let $S_k(v)$ be the expected value of the service demand that each visit of traffic source v requires of k . Then, the average delay per visit of u to k due to v , a higher or equal priority source is given by:*

$$d_k(u, v) \leq \begin{cases} \frac{\lambda_v}{\lambda_u} S_k(v) & \text{if } v \text{ has higher priority at } k \\ \min(1, \frac{\lambda_v}{\lambda_u}) S_k(v) & \text{if } v \text{ and } u \text{ have equal priority at } k \end{cases}$$

Suppose that the set of traffic sources $v = 1 \dots L$, is the complete set of sources that can contend with the source u and that have lower priority at k than u . Without loss of generality, assume that this set of lower priority sources is sorted such that for any two lower priority sources a and b , $a < b$, we have $S_k(a) \geq S_k(b)$. Then, we also have:

$$\sum_{v=1}^L N_v d_k(u, v) \leq \sum_{v=1}^L N_v \alpha_k(u, v) S_k(v)$$

where

$$\alpha_k(u, v) = \begin{cases} \min(\frac{1}{N_v}, \frac{\lambda_v}{\lambda_u}) & v = 1 \\ \min(\frac{1}{N_v}(1 - \sum_{j=1}^{v-1} N_j \alpha_k(u, j)), \frac{\lambda_v}{\lambda_u}) & v > 1 \end{cases}$$

Proof: Since the service demands are NBUE distributed, Lemmas 4, 2 and 7 are applicable. There are three cases to consider:

1. If v has a higher priority at k than u , then by application of Lemma 4 we have:

$$d_k(u, v) \leq \frac{\lambda_v}{\lambda_u} S_k(v)$$

2. Now, if v has the same priority at k as u then the tie is broken by using a FIFO rule. In the worst case, v can appear ahead of u a single time in the server queue. Thus, by Lemma 2 we have

$$d_k(u, v) \leq \min(1, \frac{\lambda_v}{\lambda_u}) S_k(m_0)$$

3. Finally, if we have $v = 1 \dots L$, the complete, sorted, set of lower priority contenders of u at k , then by application of Lemma 7 we have:

$$\sum_{v=1}^L N_v d_k(u, v) \leq \sum_{v=1}^L N_v \alpha_k(u, v) S_k(v)$$

where

$$\alpha_k(u, v) = \begin{cases} \min(\frac{1}{N_v}, \frac{\lambda_v}{\lambda_u}) & v = 1 \\ \min(\frac{1}{N_v}(1 - \sum_{j=1}^{v-1} N_j \alpha_k(u, j)), \frac{\lambda_v}{\lambda_u}) & v > 1 \end{cases}$$

Putting these three cases together, we have our result.

□

Theorem 3 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. Assume that the secondary resources indicated below have NPP scheduling. Without loss of generality, assume further that the lower priority contenders $m \in \mathcal{L}_k(c)$ are labeled $m = 1 \dots L$ and are sorted such that for $a, b \in \mathcal{L}_k(c)$, $a < b \Rightarrow S_k(a_0) \geq S_k(b_0)$.*

- (a) *The delay experienced per direct visit of a class c customer to a secondary resource k*

due to the direct visits of a class m customer to k is upper bounded by the following:

$$d_k(c_0, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_k(c_0)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) S_k(m_0) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $m \in \mathcal{L}_k(c)$.

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_0, m_0) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_0, m_0) S_k(m_0)$$

where

$$\alpha_k(c_0, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_0)), \frac{f_m V_k(m_0)}{f_c V_k(c_0)}) & m > 1 \end{cases}$$

Proof: The proof follows from Lemma 8 by recognizing the direct visits of c to k as coming from a traffic source u with traffic rate $\lambda_u = f_c V_k(c_0)$, the direct visits of any contender m to k as coming from a traffic source v with traffic rate $\lambda_v = f_m V_k(m_0)$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(m_0)$.

- (b) *The delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following:*

$$d_k(c_0, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $m \in \mathcal{L}_k(c)$,

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_0, m_{r'}) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_0, m_{r'}) S_k(m_{r'})$$

where

$$\alpha_k(c_0, m_{r'}) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_0, j_{r'})), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)}) & m > 1 \end{cases}$$

Proof: The proof follows from Lemma 8 by recognizing the direct visits of c to k as coming from a traffic source u with traffic rate $\lambda_u = f_c V_k(c_0)$, the indirect visits of any contending source m to k through primary resource r' as coming from a traffic source v with traffic rate $\lambda_v = f_m V_{r'}(m_0) V_k(m_{r'})$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(m_{r'})$.

- (c) *The delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following:*

$$d_k(c_r, m_0) \leq \begin{cases} \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)} S_k(m_0) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) S_k(m_0) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders, $m \in \mathcal{L}_k(c)$,

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_r, m_0) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_r, m_0) S_k(m_0)$$

where

$$\alpha_k(c_r, m_0) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_r, j_0)), \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_0)}) & m > 1 \end{cases}$$

Proof: The proof follows from Lemma 8 by recognizing the indirect visits of c to k through primary resource r as coming from a traffic source u with traffic rate $\lambda_u = f_c V_r(c_0) V_k(c_r)$, the direct visits of any contending source m to k as coming from a traffic source v with traffic rate $\lambda_v = f_m V_k(m_0)$, and the NBUE distributed demand

per visit of v to k as $S_k(v) = S_k(m_0)$.

- (d) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following:*

$$d_k(c_r, m_{r'}) \leq \begin{cases} \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(m_{r'}) & m \in \mathcal{H}_k(c) \\ \min(1, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) S_k(m_{r'}) & m \in \mathcal{E}_k(c) \end{cases}$$

and for all lower priority contenders. $m \in \mathcal{L}_k(c)$,

$$\sum_{m \in \mathcal{L}_k(c)} N_m(c) d_k(c_r, m_{r'}) \leq \sum_{m \in \mathcal{L}_k(c)} N_m(c) \alpha_k(c_r, m_{r'}) S_k(m_{r'})$$

where

$$\alpha_k(c_r, m_{r'}) = \begin{cases} \min(\frac{1}{N_m}, \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) & m = 1 \\ \min(\frac{1}{N_m} (1 - \sum_{j=1}^{m-1} N_j \alpha_k(c_r, j_{r'})), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)}) & m > 1 \end{cases}$$

Proof: The proof follows from Lemma 8 by recognizing the indirect visits of c to k through primary resource r as coming from a traffic source u with traffic rate $\lambda_u = f_c v_r(c_0) V_k(c_r)$, the indirect visits of any contending source m to k through primary resource r' as coming from a traffic source v with traffic rate $\lambda_v = f_m V_{r'}(m_0) V_k(m_{r'})$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(m_{r'})$.

□

Lemma 9 *This lemma is similar to Lemma 2 of [36] except that where Lemma 2 of [36] finds the upper bound on $\sum_v d_k(u, v)$ here we are interested only in a single traffic source v . Let u and v denote two sources of traffic that generate visits to a resource k with a PS queueing discipline. If λ_u and λ_v are the expected rates of visits to k from u and v respectively*

and $S_k(u)$ and $S_k(v)$ are the average demands at k for visits of u and v respectively, then the per visit delay experienced by u at k due to v is upper bounded as follows:

$$d_k(u, v) \leq \min(S_k(u), \frac{\lambda_v}{\lambda_u} S_k(v))$$

Proof: It is assumed that the customers at resource k are served in a Round Robin fashion with each customer receiving a finite service of duration τ seconds at a time. The lemma for Processor Sharing is then proved by computing the delay for Round Robin with $\tau \rightarrow 0$. If n_u and n_v are the expected numbers of quanta of service received from k by u and v in one second, respectively, then the expected per second delay seen by u due to v , $D_k(u, v)$, is upper bounded by:

$$D_k(u, v) \leq \min(n_u \tau, n_v \tau) \quad (5)$$

This is proven by considering the following two cases:

1. $n_u \leq n_v$: For every quantum of service received by u the delay introduced by v can be at most τ . Thus, we have: $D_k(u, v) \leq n_u \tau$.
2. $n_u \geq n_v$: In the worst case, u can be resident at k for every quantum of service received by v so that $D_k(u, v) \leq n_v \tau$.

Combining these two cases, we get the Inequality 5. As τ approaches zero, $n_u \tau$ and $n_v \tau$ approach the service at k per second received by u and v respectively:

$$\lim_{\tau \rightarrow 0} n_u \tau = \lambda_u S_k(u) \quad \text{and} \quad \lim_{\tau \rightarrow 0} n_v \tau = \lambda_v S_k(v)$$

Substituting these limits into Inequality 5 we get:

$$D_k(u, v) \leq \min(\lambda_u S_k(u), \lambda_v S_k(v))$$

Dividing the per second delay seen by u at k due to v by the rate of visits of u to k yields

the per visit delay seen by u at k due to v :

$$d_k(u, v) \leq \min(S_k(u), \frac{\lambda_v}{\lambda_u} S_k(v))$$

□

Theorem 4 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined at the beginning of this section. If the secondary resources have PS scheduling, then:*

- (a) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_0) \leq \min(S_k(c_0), \frac{f_m V_k(m_0)}{f_c V_k(c_0)} S_k(m_0))$$

Proof: The proof follows by applying Lemma 9 with the direct visits of c to k coming from a source u at a rate of $\lambda_u = f_c V_k(c_0)$ and the direct visits of m to k as coming from source v with rate $\lambda_v = f_m V_k(m_0)$ and with the per visit demands $S_k(u) = S_k(c_0)$ and $S_k(v) = S_k(m_{r'})$.

- (b) *The average delay experienced per direct visit of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_0, m_{r'}) \leq \min(S_k(c_0), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_k(c_0)} S_k(m_{r'}))$$

Proof: The proof follows by applying Lemma 9 with the direct visits of c to k coming from a source u at a rate of $\lambda_u = f_c V_k(c_0)$ and the indirect visits of m to k holding r' coming from source v with rate $\lambda_v = f_m V_{r'}(m_0) V_k(m_{r'})$ and with the per visit demands $S_k(u) = S_k(c_0)$ and $S_k(v) = S_k(m_{r'})$.

- (c) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the direct visits of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_0) \leq \min(S_k(c_r), \frac{f_m V_k(m_0)}{f_c V_r(c_0) V_k(c_r)} S_k(m_0))$$

Proof: The proof follows by applying Lemma 9 with the indirect visits of c to k through primary resource r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) V_k(c_r)$ and the direct visits of m to k coming from source v with rate $\lambda_v = f_m V_k(m_0)$ and with the per visit demands $S_k(u) = S_k(c_r)$ and $S_k(v) = S_k(m_0)$.

- (d) *The average delay experienced per indirect visit through primary resource r of a class c customer to a secondary resource k due to the indirect visits through primary resource r' of a class m customer to k is upper bounded by the following inequality:*

$$d_k(c_r, m_{r'}) \leq \min(S_k(c_r), \frac{f_m V_{r'}(m_0) V_k(m_{r'})}{f_c V_r(c_0) V_k(c_r)} S_k(m_{r'}))$$

Proof: The proof follows by applying Lemma 9 with the indirect visits of c to k through primary resource r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) V_k(c_r)$ and the indirect visits of m to k holding r' coming from source v with rate $\lambda_v = f_m V_{r'}(m_0) V_k(m_{r'})$ and with the per visit demands $S_k(u) = S_k(c_r)$ and $S_k(v) = S_k(m_{r'})$.

□

Lemma 10 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. The average number of direct visits of a class m customer to a primary resource r that delay a class c customer at r on each direct visit to r by the class c customer is upper bounded by*

the following inequality:

$$Q_r(c_0, m_0) \leq \min(1, \frac{f_m V_r(m_0)}{f_c V_r(c_0)})$$

Proof: The proof follows by applying Lemma 1 with the direct visits of c to r as coming from a source u at a rate of $\lambda_u = f_c V_r(c_0)$ and the direct visits of m to r coming from source v with rate $\lambda_v = f_m V_r(m_0)$.

□

Theorem 5 Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have FIFO scheduling, then:

- (a) The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by direct visits of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_0) \leq \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0)$$

Proof: Note that each visit of resource r by the class c customer has a probability of $Q_r(c_0, m_0)$ of being delayed by a customer of class m . The proof then follows by applying Lemma 2 with the indirect visits of m to k holding primary resource r that block c at r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the direct visits of n to k coming from source v with rate $\lambda_v = f_n V_k(n_0)$, and with the NBUE distributed per visit demand $S_k(v) = S_k(n_0)$.

- (b) The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_{r'}) \leq \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'})$$

Proof: Note that each visit of resource r by the class c customer has a probability of $Q_r(c_0, m_0)$ of being delayed by a customer of class m . The proof then follows by applying Lemma 2 with the indirect visits of m to k holding primary resource r that block c at r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the indirect visits of n to k holding primary resource r' , $r' \neq r$ coming from source v with rate $\lambda_v = f_n V_{r'}(n_0) V_k(n_{r'})$, and with the NBUE distributed per visit demand $S_k(v) = S_k(n_{r'})$.

□

Theorem 6 Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have PP scheduling, then:

- (a) The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by direct visits of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_0) \leq \begin{cases} \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0) & n \in \mathcal{H}_k(m) \\ \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) & n \in \mathcal{E}_k(m) \\ 0 & n \in \mathcal{L}_k(m) \end{cases}$$

Proof: Note that $Q_r(c_0, m_0)$ is the expected number of times per visit to resource r that the class c customer is delayed at r by a customer of class m . The proof then follows from Lemma 5 by recognizing the indirect visits of m to k through primary resource r that block c at r as coming from a traffic source u with traffic rate $\lambda_u = f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the direct visits of n to k as coming from a traffic source v with traffic rate $\lambda_v = f_n V_k(n_0)$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(n_{r'})$.

- (b) *The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:*

$$d_k(c_r, m_r, n_{r'}) \leq \begin{cases} \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'}) & n \in \mathcal{H}_k(m) \\ \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'}) & n \in \mathcal{E}_k(m) \\ 0 & n \in \mathcal{L}_k(m) \end{cases}$$

Proof: Note that $Q_r(c_0, m_0)$ is the expected number of times per visit to resource r that the class c customer is delayed at r by a customer of class m . The proof then, follows from Lemma 5 by recognizing the indirect visits of m to k through primary resource r that block c at r as coming from a traffic source u with traffic rate $\lambda_u = f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the indirect visits of n to k through primary resource r' as coming from a traffic source v with traffic rate $\lambda_v = f_n V_{r'}(n_0) V_k(n_{r'})$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(n_{r'})$.

□

Theorem 7 *Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. Assume that the secondary resources indicated below have NPP scheduling. Without loss of generality, assume further that the lower priority contenders $n \in \mathcal{L}_k(m)$ are labeled $n = 1 \dots L$ and are sorted such that for $a, b \in \mathcal{L}_k(c)$, $a < b \Rightarrow S_k(a_0) \geq S_k(b_0)$ then:*

- (a) *The delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k*

by direct visits of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_0) \leq \begin{cases} \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0) & n \in \mathcal{H}_k(m) \\ \min(1, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_0) & n \in \mathcal{E}_k(m) \end{cases}$$

and for all lower priority contenders, $n \in \mathcal{L}_k(m)$.

$$\sum_{n \in \mathcal{L}_k(m)} V_n(c, m) d_k(c_r, m_r, n_0) \leq \sum_{n \in \mathcal{L}_k(m)} V_n(c, m) \alpha_k(c_r, m_r, n_0) S_k(n_0)$$

where

$$\alpha_k(c_r, m_r, n_0) = \begin{cases} \min(\frac{1}{V_n}, \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & n = 1 \\ \min(\frac{1}{V_n} (1 - \sum_{j=1}^{n-1} V_j \alpha_k(c_r, m_r, j_0)), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) & n > 1 \end{cases}$$

Proof: Note that $Q_r(c_0, m_0)$ is the expected number of times per visit to resource r that the class c customer is delayed at r by a customer of class m . The then proof follows from Lemma 8 by recognizing the indirect visits of m to k through primary resource r that block c at r as coming from a traffic source u with traffic rate $\lambda_u = f_c v_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the direct visits of any contender n visiting k as coming from a traffic source v with traffic rate $\lambda_v = f_n V_k(n_0)$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(n_{r'})$.

- (b) *The delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:*

$$d_k(c_r, m_r, n_{r'}) \leq \begin{cases} \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'}) & n \in \mathcal{H}_k(m) \\ \min(1, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}) S_k(n_{r'}) & n \in \mathcal{E}_k(m) \end{cases}$$

and for all lower priority contenders, $n \in \mathcal{L}_k(m)$,

$$\sum_{n \in \mathcal{L}_k(m)} N_n(c, m) d_k(c_r, m_r, n_{r'}) \leq \sum_{n \in \mathcal{L}_k(m)} N_n(c, m) \alpha_k(c_r, m_r, n_{r'}) S_k(n_{r'})$$

where

$$\alpha_k(c_r, m_r, n_{r'}) = \begin{cases} \min\left(\frac{1}{N_n}, \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) & n = 1 \\ \min\left(\frac{1}{N_n} \left(1 - \sum_{j=1}^{n-1} N_j \alpha_k(c_r, m_r, j_{r'})\right), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)}\right) & n > 1 \end{cases}$$

Proof: Note that $Q_r(c_0, m_0)$ is the expected number of times per visit to resource r that the class c customer is delayed at r by a customer of class m . The proof then follows from Lemma 8 by recognizing the indirect visits of m to k through primary resource r that block c at r as coming from a traffic source u with traffic rate $\lambda_u = f_c v_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the indirect visits of any contender n visiting k through primary resource r' as coming from a traffic source v with traffic rate $\lambda_v = f_n V_{r'}(n_0) V_k(n_{r'})$, and the NBUE distributed demand per visit of v to k as $S_k(v) = S_k(n_{r'})$.

□

Theorem 8 Consider a multiclass queueing network whose parameters are described in Table 3.1 and which satisfy the minimal stochastic assumptions outlined in Section 3.1.1. If the secondary resources indicated below have PS scheduling, then:

- (a) The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by direct visits of a class n customer at k is upper bounded by the inequality:

$$d_k(c_r, m_r, n_0) \leq \min\left(S_k(m_r), \frac{f_n V_k(n_0)}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_0)\right)$$

Proof: Note that $Q_r(c_0, m_0)$ is the expected number of times per visit to resource r that the class c customer is delayed at r by a customer of class m . The proof then follows by applying Lemma 9 with the indirect visits of m to k holding primary resource r that block c at r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the direct visits of n to k as coming from source v with rate $\lambda_v = f_n V_k(n_0)$, and with the per visit demands $S_k(u) = S_k(m_r)$ and $S_k(v) = S_k(n_0)$.

- (b) *The average delay experienced per visit of a class c customer at primary resource r due to contention from a class m customer holding r and blocked at a secondary resource k by indirect visits through primary resource r' , $r' \neq r$, of a class n customer at k is upper bounded by the inequality:*

$$d_k(c_r, m_r, n_{r'}) \leq \min(S_k(m_r), \frac{f_n V_{r'}(n_0) V_k(n_{r'})}{f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)} S_k(n_{r'}))$$

Proof: Note that $Q_r(c_0, m_0)$ is the expected number of times per visit to resource r that the class c customer is delayed at r by a customer of class m . The proof then follows by applying Lemma 9 with the indirect visits of m to k holding primary resource r that block c at r coming from a source u at a rate of $\lambda_u = f_c V_r(c_0) Q_r(c_0, m_0) V_k(m_r)$, the indirect visits of n to k holding primary resource r' , $r' \neq r$ coming from source v with rate $\lambda_v = f_n V_{r'}(n_0) V_k(n_{r'})$, and with the per visit demands $S_k(u) = S_k(m_r)$ and $S_k(v) = S_k(n_{r'})$.

□

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