Single and Dual Queueing Schemes with Prioritised Traffic Scheduling and Finite Waiting Room

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Anthony Brian Bedford
B.App.Sci.(Hons.)(Mathematics)

Department of Mathematics and Statistics
Faculty of Applied Science
RMIT University
October 2003

Declaration

The candidate hereby declares that:

- except where due acknowledgment has been made, the work is that of the candidate alone;
- the work has not been submitted previously, in whole or in part, to qualify for any other academic award;
- the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program;
- any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Anthony Brian Bedford
October 2003

Acknowledgements

There are many people who have assisted me along my journey through this thesis.

Firstly, I wish to thank my wife Michelle, and my four children, Catherine, Damian, Michael and Emma who have missed out on a husband and a father along the journey. I am so much looking forward to getting our lives back to normal.

To my supervisor and friend Associate Professor Panlop Zeephongsekul who has embraced my work and enhanced my knowledge enormously. My research has always benefitted from your inspiring guidance.

Thanks to my colleague and friend Associate Professor Cliff Da Costa who has assisted me with everything that it takes to be a quality academic, and for continually encouraging me to further excel.

To all the staff in the Mathematics and Statistics department, for providing me with your support, friendship and teaching.

To RMIT University and the many friends made over the years.

RMIT and the APA - without your financial support I could never have made it.

To Professor Stephen Clarke who has always given me a sporting chance.

To the reviewers of papers along the way my thanks.

To my parents Brian and Mary who always encouraged my love for numbers.

Finally, to God, who never abandoned me.

Dedicated to Clarence Edward Bedford

Contents

List of Figures

List of Tables

Summary

Analysis of new schemes aimed at improving congestion in communications systems is vital for todays service providers. Many techniques are used to evaluate such schemes be it precisely via mathematics or approximately using simulation.

This thesis introduces a new scheme, the multi priority dual queue (MPDQ). The MPDQ is the combination of two concepts, the dual queue introduced by [Hayes et. al., 1999] and prioritised traffic. The MPDQ is a system with finite waiting room with two queues where traffic upon arrival if finding the first queue full wait in the second queue if there is room. When a space becomes vacant in the first queue, a customer at the front of the second queue enters the back of the first, which is the queue that has the service centre at the front of it. The traffic can be of two or more classes.

The analysis of such a system is complex, both analytically using queueing theory and approximately using simulation analysis. Both approaches are taken in this thesis.

To begin, the new algorithmic approach used for the MPDQ is applied for the single buffer model. The steady state and waiting time distributions are obtained and later compared to the MPDQ.

Next the performance characteristics are obtained by solving the steady state and waiting time distributions of a two class MPDQ. Preemptive and non-preemptive service disciplines are investigated. Maple is also used to solve the algorithm.

To broaden the application of the MPDQ scheme, computer simulations using Arena are undertaken to extend the application of the scheme (and existing finite queueing models) to situations with more than two priorities, something that is extremely difficult to solve analytically. Using simulation, comparisons are undertaken for the single and dual queue schemes for more than two priorities with a variety of queueing disciplines used including First In First Out (FIFO), Last In First Out (LIFO), High Class First (HCF), and Low Class First (LCF). Network scenarios are also modelled to determine the performance of the MPDQ in this environment.

Chapter 1

Introduction

1.1 Finite Queues and Communication Systems

Communication systems are under continuous strain in a world where it is not only the demand for information which is increasing, but also the demand for speed of delivery. There are many schemes used for the transfer of information in a wide variety of applications, varying from a telephone call through a mobile phone network, to the need for a document to be printed in an office network or streaming data for a video broadcast across the Internet. With the demand for information increasing by the day, efficient and effective queueing methods are needed to analyse new schemes aimed at achieving faster communication.

In recent times the idea of prioritising traffic has received interest, especially in the communication forum. In many cases these schemes have proven effective in providing quality of service (QoS) for users of the networks. The need for differentiating customers into classes has arisen due to concerns with the levels of delay and loss for real-time mediums. In communication systems, this may be a mobile phone call, a video transmission, or a streaming audio application. When two or more classes of customers share the same network, scheduling and service order are critical determinants of class-wise potential waiting times. Typical examples include packet networks, Local Area Networks (LANs), Wireless Local Area Networks (WLANs) and mobile networks. The problem for video traffic (and real-time media) was identified in [Fritz, 1999], "...video is often a real-time medium. Most applications have no way of reconstructing bad or missing information....Delay also can wreak havoc with video traffic". He further adds "Unfortunately, many switches and routers have single queues only. When there's congestion, all the packets begin lining up in the same queue- which isn't good as far as video is concerned. So try to support multi priority queues." The idea of multi priority queueing models forms a major component of this thesis.

Traffic management in communication environments like the Internet is vital in meeting the demands of users. Various dynamic scheduling algorithms have been introduced with the aim of improving QoS to customers. Some new algorithms, such as dual queueing, are designed to give better QoS to most customers at the expense of a few, rather than give poor QoS fairly to all customers.

The ideas proposed in two key papers led to the unique queueing scheme analysed in this thesis, the Multi Priority Dual Queue (MPDQ). The MPDQ was derived from the Dual Queue scheme proposed in [Hayes et. al., 1999] and adapted to include the prioritised customer approach proposed to relieve congestion in [Odlyzko, 1999a]. The MPDQ introduces different classes into this type of scheme with the aim of providing better service to high class customers without completely disadvantaging low class customers. This is possible not only because of the prioritisation of customers, but also due to the MPDQ's queue structure; a partitioned single queue which typically follows a highest class first (HCF) discipline. [Odlyzko, 1999a] introduced the idea of Paris Metro Pricing for the Internet, where networks are partitioned into several channels using priorities. In this paper he discusses the problem with traditional priority schemes whereby higher priority customers completely block lower priority ones. His system is a user based system whereby the Paris Metro Pricing sets the cost of each class and lets the users decide on their QoS. His discussion on classes is worthy of note, concluding that classes (Odlyzko uses the term channels) should be small - "possibly two, but more likely three or four". The later analysis in this thesis explores this idea for the MPDQ.

The Dual Queue scheme in [Hayes et. al., 1999] involved the use of MPEG traces, also used in [Rose, 1995], which consisted of 20 files. The MPDQ scheme has two queues of finite space: the primary queue, which feeds into the service centre, and the secondary queue, which acts as a waiting room when the primary queue is full. Upon arrival, a customer finding the primary queue full waits in the secondary queue if there is room. When a space becomes vacant in the primary queue, the customer at the front of the secondary queue joins the end of the primary queue. [Hayes et. al., 1999] analysed the delay characteristics of the Dual Queue against the pre-existing scheduling regimes FIFO (First in First Out) and a modified Deficit Round Robin technique (DRR) [Shreedhar and Varghese, 1996], and demonstrated distinct advantages using the Dual Queueing scheme. This work was extended to a wireless local area network where minor modifications were made to the Dual Queue (DQ) and it was shown to outperform standard Round Robin scheduling in this application for untethered environments [Ranasinghe et. al., 2001]. These analyses were undertaken using a fixed amount of traffic with no statistical assumptions made on the underlying distributions of the arrival or service processes.

This thesis explores a multi-priority queueing system that is especially relevant in the Internet Engineering Task Force (IETF) Integrated Services Processes or Differentiated Services architectures. The MPDQ scheduling discipline provides a simple and effective mechanism for scheduling in these architectures. Introducing different classes into this scheme adds additional complexity to the DQ with the aim of class-based QoS. Customers within this system of greater importance (or higher class) have precedence over any lower classed customer, with the aim of reduced waiting times. This MPDQ differs from the original DQ in that [Hayes et. al., 1999] used large finite buffers. As [Hayes et. al., 1999] deals with the handling of traffic streams the scheduling here is changed to cope with individual prioritised customers, and instigates an abatement of both queues. The setting of the abatement threshold to be equal to the capacity of the queues significantly changes the operation of the DQ. The MPDQ is especially relevant for the Internet's development, and also has merits for use in certain manufacturing systems where a second holding area is available. In this thesis, loss is determined by available queue size, and a time-discard policy in which customers are ejected from the system after a period of time is reached is not used as in [Hayes et. al., 1999].

This thesis contributes substantial work in the field of priority finite queueing. Firstly, techniques of solving elementary finite queueing models are discussed. Next, the new priority models are introduced. A new technique to solving the steady state probabilities and waiting time distribution for a single queue preemptive priority model is given. An algorithm in Maple is referenced so that researchers can enter any size of queue and arrival/service rates and arrive at a solution.

The problem of solving the steady state probabilities and expected waiting times for the MPDQ is mathematically complex. Both the exponential preemptive and exponential non-preemptive models are analysed in detail, with algorithms providing steady state solutions for two classes of customers.

As the analysis of both conventional and unique queueing models unfolded, the need for simulation became apparent. Traditional queueing techniques here accommodate the dual queue for two classes of customers. It was necessary to simulate this dual queue for a variety of queueing regimes, queue sizes, and arrival/service rates beyond two classes. Simulation also provided an environment in which the analytical results could be compared and validated.

Using simulations, comparisons are undertaken for the single and dual queue schemes for more than two priorities. Furthermore, there are a variety of queueing disciplines investigated, such as First In First Out (FIFO), Last In First Out (LIFO), High Class First (HCF), and Low Class First (LCF), for both the MPDQ and existing finite queueing models. The MPDQ in a network scenario is also investigated via simulation.

1.2 Contributions

The major contributions met by this thesis are

- The construction of an algorithm that obtains the general steady-state solution for a single finite buffer system with two classes of customers dispensing a preemptive priority service.
- The construction of an algorithm that obtains the waiting time for a single finite buffer system with two classes of customers dispensing a preemptive priority service.
- The construction of an algorithm that obtains the general steady-state solution to the queueing system for the MPDQ with preemptive priority customers and finite waiting room.
- The construction of an algorithm that obtains the waiting time for the MPDQ with preemptive priority customers and finite waiting room.
- The construction of an algorithm that obtains a general steady-state solution to the queueing system for the MPDQ with non-preemptive priority customers and finite waiting room.
- The simulation of single-queue and MPDQ systems for both priority and non-priority customers with the evaluation of the performance characteristics of the alternative methods.
- The evaluation of the MPDQ in a network scenario via simulation with networks containing FIFO service centres

This research provides a framework for communications providers in determining the size and type of systems to put in place to cater for their needs. The implementation of a priority scheme for the MPDQ can assist in reducing loss effects, and this research provides a general solution that can be applied to this and other specific congestion problems faced by communications providers and users.

1.3 Key Related Works

The work in this thesis contains four broad areas of analysis. The first is the determination of the steady state probabilities for the preemptive priority single finite buffer queue, the preemptive MPDQ and the non-preemptive MPDQ. Following on from this is the determination of the waiting time distributions, derived for the preemptive

priority single finite buffer queue and the preemptive MPDQ. The third area is the simulation of single finite buffer models and MPDQ systems for prioritised traffic. Finally, the simulation of networks is covered in the fourth area. As this work covers both queueing theory and communication, some references on those fields will be cited now to highlight the development in the field.

Figure 1.1 illustrates the organisation of the thesis in terms of the fields covered.

1.3.1 Congestion Control Schemes

There have been a wide variety of scheduling methods studied which aim to reduce congestion in communication systems. Many differentiate customers through marking and dropping processes [Labrador and Banerjee, 1999], [Feng et. al. , 1999]. Others use time-marking and derivatives of this to allocate a degree of fairness in service, such as Self-Clocked Fair Queueing (SCFQ) and Credit Based Fair Queueing (CBFQ) [Golestani, 1994], [Chan et. al., 1997]. [Bagwell et. al., 1995] uses simulation to determine the merits of Stop-and Go queueing discipline (SGQ) and compares the results with the classic $M/M/1/K$ model. [Jang et. al., 1997] uses a Dual Queue Length Threshold (DQLT) to divide real time and non-real time traffic to separate queues. A simulation approach to Wireless Packet Discard Policies was undertaken in [Siripongwutikorn et. al., 2000], and a simulation approach was undertaken to analyse the scheme in a network scenario. They looked at transition probabilities for the error model and a specific type of loss called Message Loss Ratio. Weighted Round Robin (WRR) is looked at in [Wang and Yonatan, 2000]. Congestion at a network level is also tackled. [Bahk and El Zarki, 1992] switches routing schemes to improve congestion. [Balbo et. al., 1985] analyse complex priority network models by combining petri nets and product form results. An excellent taxonomy of QoS schemes can be found in the introduction of [Hayes, 2001], which also introduces the dual queueing scheme discussed earlier.

1.3.2 Simulation and Applications

The Internet and congestion are inglorious comrades, with the web often referred to as the 'world wide wait' [Odlyzko, 1999b]. Simulation provides a valuable tool in evaluating models that aim to reduce this wait. Some simulation work in communications includes:
[Balbo et. al., 1985] identifies a way of measuring loss; [Boyce and Gaglianello, 1998] determines the effect of loss on MPEG video, and [Clark and Fang, 1998] uses simulation to determine the effectiveness of the allocated capacity framework for the Internet. Further works in simulation are discussed in Part IV.

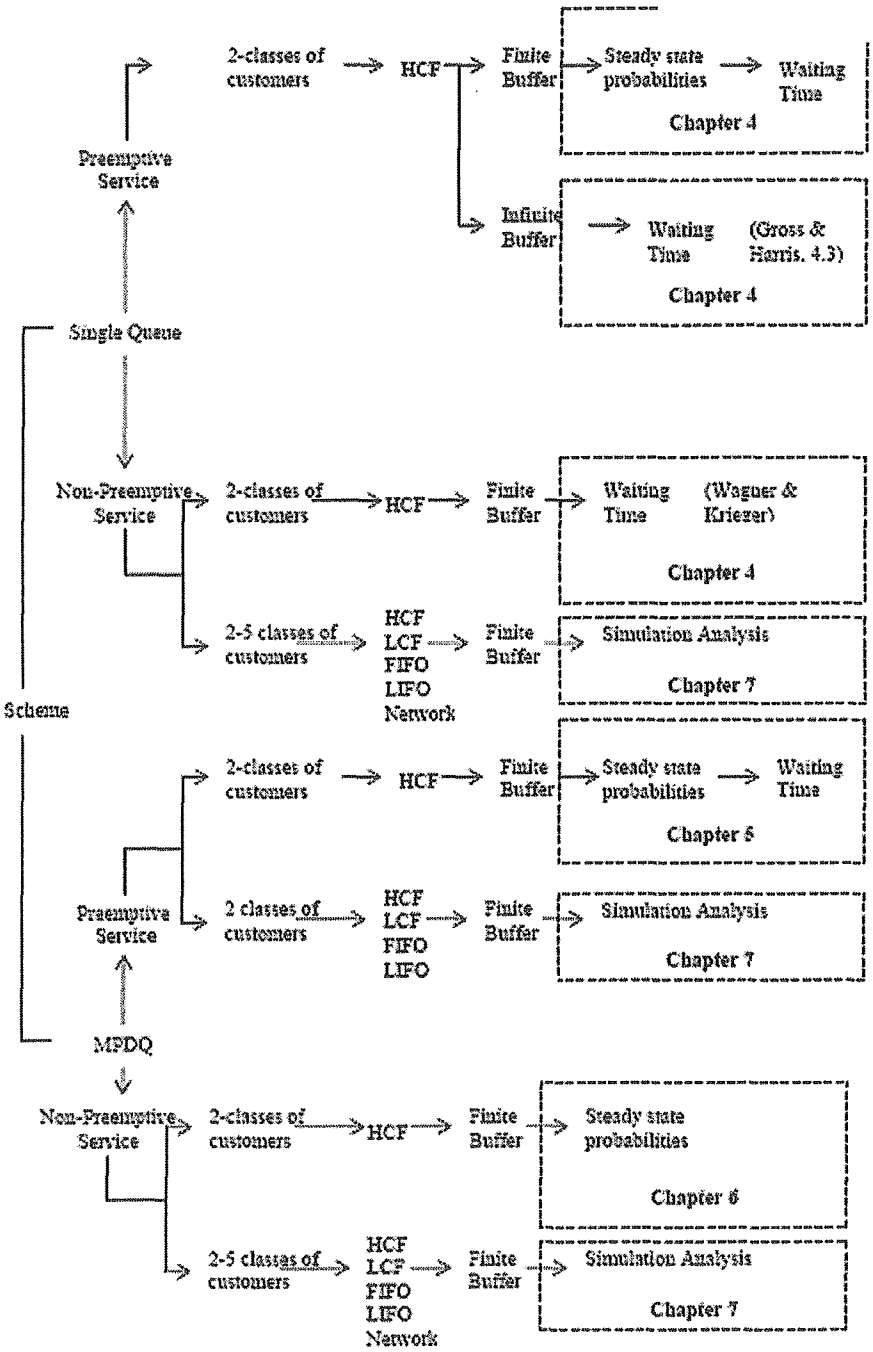

Figure 1.1: Thesis areas of research by chapter

1.3.3 Analytical Solutions of Single Queues

Figure 1.1 highlights the fields of research in this thesis. Other key work related to the topics in the figure are now cited. The bracketed item indicates what is solved and how, and various techniques which were used to achieve the same solution.

Preemptive priority model: [Heathcote, 1960] (probability- Laplace transforms), [Yeo, 1963] (waiting time - Laplace transforms), [Chang, 1965] (probability, waiting times, busy period - Laplace transforms), [Marks, 1973] (probability, balance equation), [Miller, 1981] (probability - Matrix-geometric), [Adiri and Domb, 1982] (probability, waiting times - Laplace transforms), [Rege and Sengupta, 1985] (waiting times - Laplace transforms), [Kramer, 1987] (waiting time - Laplace transforms), [Gail, Hunter and Taylor 1992] (probability, waiting times - generating functions).

Preemptive priority finite models: [Singh, 1977] (probability - matrix calculus), [Adiri and Domb, 1982] (probability, waiting times - Laplace transforms), [Bondi, 1989] (waiting time - Recurrence relations)

Non-preemptive priority models: [Marks, 1973] (probability, balance equation), [Miller, 1981] (probability - Matrix-geometric), [Adiri and Domb, 1982] (probability, waiting times - Laplace transforms), [Jain and Dhyani, 1997] (probability, balance equations), [Daigle and Roughan, 1999] (queue length distribution, Fourier transforms), [Feng et. al., 2000] (waiting times, queue length - generating functions)

Non-preemptive priority finite models: [Kapadia et. al., 1984] (probabilities - balance equations), [Bondi, 1989] (waiting time - Recurrence relations), [Wagner, 1997] and [Wagner and Krieger, 1999] (probability, waiting time - Laplace transforms)

Other schemes : [Cidon and Sidi, 1990] and [Cidon et. al., 1993] also use recursive relations to solve priority and finite queues respectively.

1.4 Organisation of the Thesis

This thesis investigates both single and dual queueing schemes with prioritised traffic scheduling and finite waiting room. This work is divided into four main parts : Part I is an introduction to the models analysed, Part II is an analysis of the single finite buffer model, Part III consists of the dual queueing models, Part IV investigates simulations of both single and dual queueing models. These are complimented by Part V, the conclusion of this work.

Part I - INTRODUCTION TO FINITE QUEUEING MODELS

The part contains two chapters. Chapter 2 exhibits the preliminaries referred to in later chapters. This includes the nomenclature used in the thesis. Chapter 3 looks at the operational characteristics of priority disciplines on single and dual queueing

models.

Part II - SINGLE QUEUE MODELS

In this part the preemptive priority single finite buffer model is analysed. The states of the system are exhibited in matrix form. A new computational algorithm is derived to obtain the steady state probabilities for the two class single buffer model. Further, a recursive algorithm is used to derive the expected waiting times for class one and two customers. Classical results are also given for other single server models.

Part III- DUAL QUEUE MODELS

This part is divided into two chapters. Chapter 5 solves the steady state probabilities using a computational algorithm for the preemptive MPDQ for two classes of customers. A recursive algorithm is used to obtain the waiting times for the two classes of customers. Chapter 6 solves the steady state probabilities for the non-preemptive service discipline.

Part IV - SIMULATION STUDIES

This part contains simulation of single and dual finite buffer models for 2 to 5 classes of customers. Queueing regimes are varied and performance statistics are analysed. Further, some simple networks are analysed to determine the quantity of optimal MPDQ's. This is undertaken to determine the behaviour of the MPDQ which is beyond current analytical analysis.

1.5 Publications

- Chapter 4 : Material has been accepted for publication in the proceedings of *Inaugural Sri Lankan International Statistical Conference.* Further material from this chapter is published in a research report.
- Chapter 5 : Material has been accepted for publication in the refereed journal: *European Journal of Operational Research.* Further material from this chapter has been submitted to the same journal and appears in two separate research reports.
- Chapter 6 : Material is under preparation for submission to a refereed journal.
- Chapter 7 : Material has been published in the following refereed proceedings/books:

Two papers in the text: *Topics in Applied and Theoretical Mathematics and Computer Science,*

Two papers in the conference proceedings: *Recent Advances in Computational Science and Engineering,* and

One paper in the conference proceedings: *Lecture Notes in Computer Science.*

Further, material has been accepted for publication in the refereed journal *International Journal of Computational Science and Engineering.*

Detailed references are given in the bibliography.

Part I

INTRODUCTION TO FINITE QUEUEING MODELS

Chapter 2

Mathematical Preliminaries

Queueing theory in communication began with the development of the telephone. The fact that callers could potentially wait in a queue, receive an engaged signal, or hang-up due to long waits led to the need for analysis of the performance of a variety of queueing systems.

The principles underlying the approaches taken to solve queueing models is introduced here. To begin, terms used throughout this work are defined. The mathematical preliminaries used to obtain the performance characteristics of various queueing models is then given. The techniques of solution are then given, including the methods of solution. [Kleinrock, 1975a], [Kleinrock, 1975b], [Ng, 1996], and [Bunday, 1996] are some of many excellent introductions into a broad range of queueing models.

2.1 Nomenclature

The solution processes of this work have application in both the queueing theory and communications forums. Hence, there are three terms that are used to describe arriving units depending upon the topic of interest. In general queueing theory, and for the majority of this work, *customer* refers to a single unit arrival (not in batches or bulk). An equivalent term is *traffic* which is used more in the communication environment. Finally, the term *packet* may be used. This refers to data which arrives to a system consisting of a quantity of information which may vary in size. In this work the size of packets is considered to be constant. This is in line with the approach in [Wagner and Krieger, 1999] and [Wagner, 1997], with authors including [Kato et. al., 1999], [Bolot, 1993] and [Bondi, 1989] using Markov models to approximate communications systems. The analysis in this thesis provides an approximation to existing systems as the exponential assumption lends to a tractable solution. It is common to find Markov models used as an approximation for communications systems.

The results in this thesis should be viewed as an approximation for communication style systems such as mobile networks, IP networks, the Internet and so on. The size is explicitly related to the queue size, so a queue size of 10 implies 10 packets in the queue. Whilst it is acknowledged that in many systems packet size varies, this is not the primary motivation of this work hence analysis is withheld for later work.

Now for common terms used throughout the thesis. Detailed definitions of some of these will be given whenever they appear for the first time in the thesis.

2.1.1 Arrival Variables

There variables refer to customers arriving to a queueing system or network.

λ_i = mean arrival rate of customers of class i to the queue.

λ_i^{-1} = mean time between arrival for customers of class i to the queue.

λ = mean arrival rate of customers to the queue. In the case of a priority network this is the overall mean arrival, e.g. $\lambda = \lambda_1 + \lambda_2$ for a model with two classes of customers.

$\lambda(t)$ = the arrival rate at time t, i.e. when the arrival process isn't stationary

$\gamma_{n,i}$ = arrival rate to node n for traffic of class i.

2.1.2 Service Variables

There variables refer to customers receiving service at a service center in a queueing system or network.

μ_i = mean service rate of customers of class i .

μ_i^{-1} = mean process time for customers of class i.

μ = mean service rate of customer. In the case of a priority network this is the overall mean service rate, i.e. $\mu = \left(\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}\right)^{-1} \frac{1}{\lambda}$ for a model with two classes of customers.

2.1.3 Traffic Variables

ρ_i = traffic intensity of customers of class i, $\rho_i = \frac{\lambda_i}{\mu_i}$

ρ = total traffic intensity. In the case of a priority network this is the overall traffic intensity, e.g. $\rho = \frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2}$ for a model with two classes of customers.

2.1.4 Performance Variables

W_q = mean waiting time of customers in the queue (not including the customer in service).

W_q^i = mean waiting time in queue i for any customer (not including the customer in service).

W_s = mean waiting time of customers in the system.

W_s^i = mean waiting time in the system for class i customers.

M_s^i = maximum waiting time in the system for class i customers.

L_q = mean number of customers in the queue.

π_{Loss} = probability of loss.

$\overline{Loss}$ = mean loss for all classes.

$\overline{Loss}^i$ = mean loss for class i.

L_s = mean number of customers in the system.

L_s^i = mean number of class i customers in the system.

Ml = Maximum of all loss.

Ml^i = Maximum loss for class i;

π_B = probability of a busy server

$\pi_{\{Q_2=0\}}$ = probability of an empty secondary queue

$N(t)$ is defined as the quantity of traffic waiting in the system at time t

$\Gamma(\alpha, \beta)$ refers to the gamma distribution with location parameter α and scale parameter β.

$\Re^n$ refers to the Euclidean space of dimension n

r^2 = square of the sample correlation coefficient

p = p-value

$P(A)$ =probability of event A occurring

$1_{\{a\}} = \begin{cases} 1 & \text{if } a \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$ This is the indicator of set $\{a\}$.

2.1.5 Scheduling

$FIFO$ = First In First Out

$FCFS$ = First Come First Served

$LIFO$ = Last In First Out

$LCFS$ = Last Come First Served

HCF = Highest Class First

LCF = Lowest Class First

RR = Round Robin

2.1.6 Kendall Lee Notation

The Kendall-Lee notation is typically depicted by six characteristics describing a queueing system. This was introduced by David Kendall in 1953.

It is of the form

$$(a/b/c) : (d/e/f)$$

where

a = the arrival distribution

b = the service distribution

c = the number of parallel servers

d = the queue scheduling (typically takes values such a $FIFO$)

e = the system capacity

f = the size of the population from which the customers come

Some special cases appear in this work (and in general) and are as follows

M = A Poisson arrival process or an exponential inter-arrival process. Also it indicates an exponential service time.

G = general service distributions

M_i = as for M where i =class of customer

2.1.7 Matrix Operations

$e_i(n)$ refers to the i^{th} unit vector of dimension n.

$\overline{v}^t$ refers to the transpose of a column vector $\overline{v}$.

A is a matrix with n rows and m columns with dimension $(m \times n)$, where a_{ij} form the components of the matrix.

I is the identity matrix, 0 is a vector of 0's, 1 is a vector of 1's.

The transpose of a matrix is denoted A^t in reference to the original matrix A.

A diagonal matrix is a square matrix where all components above and below the nain diagonal are zero.

A tridiagonal matrix is a diagonal matrix where $a_{ij} = 0$ if $i > j + 1$ or $j > i + 1$. So the there are only three diagonals where non-zero values are allowed.

The inverse of a square matrix A is denoted by A^{-1} and exists if

$$\mathrm{A}\mathrm{A}^{-1} = \mathrm{A}^{-1}\mathrm{A} = \mathrm{I}$$

2.2 Arrival Assumptions and the Poisson Process

In this thesis, the assumption of constant arrival rates is used (independent of time t). That is, if

1. arrival rates are independent of time

2. arrivals are independent of each other

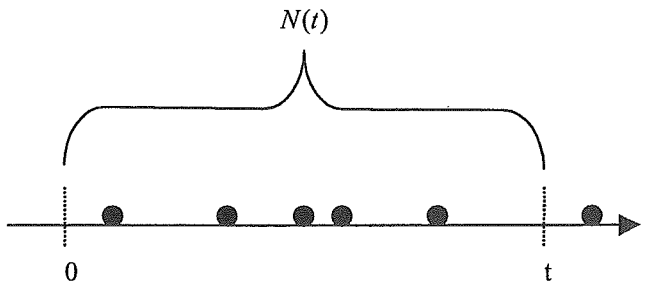

Figure 2.1: Count process, $N(t)$ is Poisson

3. multiple arrivals are impossible

then the inter-arrival times are exponentially distributed. Section 1.8, pp. 23-29 of [Gross and Harris, 1974] links the three points elegantly.

All processes are assumed to be in operation for a long time, that is, $t \to \infty$, and the steady state is reached.

Consider the count process $N(t)$, where $N(t)$ is defined as the quantity of traffic waiting in the queue at time t. This is shown in Figure 2.1.

Now to define the Poisson process:

Definition 1 *A Poisson process of rate $\lambda > 0$ is a counting process $N(t), t \geq 0$ and $N(0) = 0$ for which*

(a) at any time point $0 < t_1 < ... < t_n$ the process increments $N(t_1) - N(t_0)$, $N(t_2) - N(t_1)$, ..., $N(t_n) - N(t_{n-1})$ are independent random variables

(b) For $s \geq 0$ and $t \geq 0$, the random variable $N(t+s) - N(t)$ has the Poisson distribution

$$P(N(t+s) - N(t) = k) = \frac{(\lambda t)^k e^{-\lambda k}}{k!} \tag{2.1}$$

and $P(N(0) = 0) = 1$

The memoryless property of the exponential distribution is an important property in queueing theory. It is as follows.

Let ζ_t =amount of time from t until the first occurrence of a count process after t. If the process has $f(x)$ as the p.d.f. of T , it can be shown that as long as T has finite mean, the limiting distribution of ζ_t is

$$G(y) = \int_0^y \frac{1 - F(x)}{\mu} dx \tag{2.2}$$

where $F(x) = \int_0^x f(t)dt$ and $\mu = \int_0^\infty tf(t)dt$. If $f(x)$ has an exponential distribution with parameter λ, gives $G(y) = 1 - e^{-\lambda y}$. This implies the time to the next

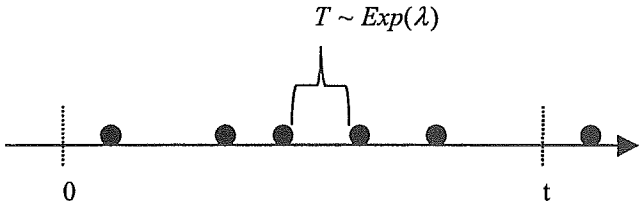

Figure 2.2: Inter-arrival times are distributed exponentially

occurrence from any epoch between consecutive occurrences has the same distribution as the inter-occurence time - the memoryless property.

This is a fundamental result of queueing theory. This is also reflected in the following argument again showing the lack of memory property:

$$\begin{aligned} P(T > t+s|T>t) &= \frac{P(T>t+s, T>t)}{P(T>t)} \\ &= \frac{P(T>t+s)}{P(T>t)} = \frac{e^{-\lambda(t+s)}}{e^{-\lambda(t)}} = e^{-\lambda s} \end{aligned} \tag{2.3}$$

From this note

$$P(T > t+\Delta t|T>t) = e^{-\lambda \Delta t} = 1 - \lambda \Delta t + o(\Delta t) \tag{2.4}$$

and

$$P(T \leq t+\Delta t|T>t) = 1 - e^{-\lambda \Delta t} = \lambda \Delta t + o(\Delta t) \tag{2.5}$$

Note that the probability of at least two events within $(t, t+\Delta t)$ is

$$1 - (1 - \lambda \Delta t + o(\Delta t)) - (\lambda \Delta t + o(\Delta t)) = o(\Delta t) \tag{2.6}$$

So in this small interval the process can increase by at most one occurrence with a non-negligible probability.

2.3 Markov Process

Markov Process is a stochastic process that has no dependence on past history depending only on the present. So a random process $X = \{X(t), t \in \Re\}$ is a Continuous Time Markov Chain (CTMC) if

$$P[X(t+s) = j|X(t) = i, X(s), s \leq t] = P(X(t+s) = j|X(t) = i), \forall i, j \in S \tag{2.7}$$

where S is the state space of the process. In queueing, the CTMC has specific relevance. In CTMC the process makes transitions from one state to another which depend only on the current state. This is after it spends some time on the state it starts from. In CTMC this time is exponentially distributed due to the lack of memory property of the Markov process.

If $\lim_{s \downarrow 0} P\left(X\left(t+s\right)=j|X(t)=i\right)=1_{\{i=j\}}$ then the following relationship holds:

$$P\left[X\left(t+\varepsilon\right)=j|X(t)=i\right]=1_{\{i=j\}}+q\left(i,j\right)\varepsilon+o(\varepsilon) \tag{2.8}$$

$q(i,j)$ are elements of the infinitesimal generator matrix A, which is given below:

$$\mathrm{A}=\begin{bmatrix} -q_0 & q_{0,1} & \cdots \\ q_{1,0} & -q_1 & \\ \vdots & & \ddots \end{bmatrix} \tag{2.9}$$

The diagonal elements of A are equal to the negative sum of the total rate out of the state:

$$q_i=q(i,i)=-\sum_{j\neq i}^{\infty} q(i,j) \tag{2.10}$$

As indicated, the times between consecutive jumps in a CTMC has an exponential distribution. However the jump epochs form a Markov chain (see [Kleinrock, 1975a] pp. 28-30 for more details on Markov chains). Some properties of Markov chains are:

- A Markov chain is irreducible if every state can be reached from every other state, that is, if for all pairs of states $(i,j) \in S^2$there exists an integer n such that $p_{ij}^{(n)}>0$.
- A state is aperiodic if $\gcd\{n : p_{ij}^{(n)}>0\}=1$
- A recurrent state is a state where the chain returns with probability 1.
- A state which is not recurrent is transient
- Ergodic states are recurrent, aperiodic states
- Stationary refers to values (such as probability or processes) that are time-independent

The stationary probability π of the states of an irreducible CTMC in an ergodic set is unique and satisfies

$$\pi \mathrm{A}=0 \tag{2.11}$$

and satisfy the normalisation constraint

$$\pi\bar{\mathrm{e}}= 1 \tag{2.12}$$

For this thesis, all Markov processes are stationary and the CTMC process is in steady state.

An excellent review of CTMC can be found in Chapter 2 of [Kleinrock, 1975a].

2.4 Chapman-Kolmogorov Equations

The transition probability of a stationary CTMC can be defined as

$$p_{i,j}(s) = P(X(s+t) = j | X(t) = i) \tag{2.13}$$

The Chapman-Kolmogorov equations are consequences of the Markov property and they are:

$$p_{i,j}(s+t) = \sum_k p_{i,k}(s) p_{k,j}(t) \tag{2.14}$$

or in matrix notation

$$\mathbf{P}(s+t) = \mathbf{P}(s)\mathbf{P}(t) \tag{2.15}$$

where $\mathbf{P}(t)$ is the transition probability matrix which is defined by

$$\mathbf{P}(t) = \left(p_{i,j}(t)\right), t \geq 0 \tag{2.16}$$

The Kolmogorov differential equations are derived from (2.15) by setting $t = ds$

$$\begin{aligned} \mathbf{P}(s+ds) &= \mathbf{P}(s)\mathbf{P}(ds) \\ &\Rightarrow \mathbf{P}(s+ds) - \mathbf{P}(s) = \mathbf{P}(s)\mathbf{P}(ds) - \mathbf{P}(s) \\ &\Rightarrow \mathbf{P}(s+ds) - \mathbf{P}(s) = \mathbf{P}(s)\left(\mathbf{P}(ds) - \mathbf{I}\right) \\ &\Rightarrow \mathbf{P}'(s) = \mathbf{P}(s)\mathbf{A} \end{aligned} \tag{2.17}$$

where

$$\mathbf{A} = \lim_{\mathbf{ds}\downarrow\mathbf{0}} \left(\frac{P(ds) - \mathbf{I}}{ds} \right) \tag{2.18}$$

which is the Kolmogorov Forward Equation. Similarly from (2.15) by setting $s = dt$

$$\mathbf{P}'(t) = \mathbf{A}\mathbf{P}(t) \tag{2.19}$$

which is the Kolmogorov Backward Equation.

2.5 Birth-death processes in the finite buffer model

A birth-death process is a CTMC which increases or decreases by one unit with each transition, i.e. $p_{ij} = 0$ if $|i - j| > 1$. There are results used throughout this work that require the general steady state distributions of a birth death process. First, let us establish what that is by obtaining the time invariant steady state probabilities.

We can express the state rate transition diagram of a finite model as

$$\pi'(t) = \mathrm{A}\pi(t) \tag{2.20}$$

where $\pi'(t)$ is the rate of change of state probabilities, A is the infinitesimal generator matrix and $\pi(t)$ contains the required probabilities $\pi = [\pi_0, .., \pi_n]^t$. Equation (2.20) follows by the same argument used to obtain the Chapman-Kolmogorov equations.

Let $\pi_i(t) = P(X(t) = j)$ and $\pi(t) = (\pi_0(t), \pi_1(t), \pi_2(t), ..., \pi_n(t))$.

Then because

$$\begin{aligned} \pi_j(t+s) &= \sum_{i \in s} P_{ij}(t)\pi_i(s), \text{ (Markov property)} \\ \therefore \quad \pi(t+s) &= P(t)\pi(s) \end{aligned}$$

and putting $t = ds$ gives

$$\pi(s + ds) = P(ds)\pi(s) \tag{2.21}$$

subtract $\pi(s)$ from both sides gives

$$\begin{aligned} \pi(s+ds) - \pi(s) &= P(ds)\pi(s) - \pi(s) && (2.22) \\ &= (P(ds) - \mathrm{I})\,\pi(s) && (2.23) \end{aligned}$$

Taking the limit:

$$\lim_{ds \downarrow 0} \frac{\pi(s+ds) - \pi(s)}{ds} = \lim_{ds \downarrow 0} \left(\frac{P(ds) - \mathrm{I}}{ds} \right) \pi(s) \tag{2.24}$$

Hence we have the forward equation

$$\pi' = \mathrm{A}\pi. \tag{2.25}$$

Similarly, we have the backward equation

$$\pi' = \pi^t \mathrm{A}. \tag{2.26}$$

Recall for a birth-death process, $p_{ij} = 0$ if $|i-j| > 1$, i.e. $P(t)$ is tridiagonal. In this case

$$\mathbf{A} = \begin{pmatrix} -q_0 & q_{01} & 0 & \cdots & \cdots & \cdots \\ q_{10} & -q_1 & q_{12} & 0 & \cdots & \cdots \\ 0 & q_{21} & -q_2 & q_{23} & 0 & \cdots \\ \vdots & 0 & q_{32} & -q_3 & q_{34} & 0 \\ \vdots & \vdots & 0 & \ddots & \ddots & \ddots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots \end{pmatrix} \tag{2.27}$$

and $\pi' = \pi^t \mathbf{Q}$ becomes

$$\begin{aligned} \pi_0'(t) &= -q_0\pi_0(t) + q_{10}\pi_1(t) \\ \pi_j'(t) &= -q_j\pi_j(t) + q_{j+1,j}\pi_{j+1}(t) + q_{j-1,j}\pi_{j-1}(t),\ j > 0 \end{aligned}$$

If the steady-state solution exists (which it always does in a finite buffer model) then

$$\pi'(t) \to 0 \tag{2.28}$$

and we get

$$\pi\mathbf{A} = 0 \tag{2.29}$$

balance equations, where $\pi = \lim_{t\to\infty} \pi(t)$. The system (2.20) is a linear system of differential equations which can be solved in the standard way. The resultant solution involves constants and exponential terms which approach zero rapidly as $t \to \infty$, leaving the constants as the steady state solution of the system. An example of this is given next.

The $(M/M/2) : (FIFO/2/\infty)$ finite buffer model

To illustrate the steady-state solution process, consider the finite buffer model $(M/M/2) : (FIFO/2/\infty)$ as seen in Figure 2.3. Note that in this example λ_i = arrival rate from state i customers in the system, and μ_i = service rate from state i customers in the system.

The balance equations for this system with the time variable t are as follows:

$$\begin{aligned} \pi_0'(t) &= -\lambda_0\pi_0(t) + \mu_1\pi_1(t) \\ \pi_1'(t) &= \lambda_0\pi_0(t) - (\lambda_1 + \mu_1)\,\pi_1(t) + \mu_2\pi_2(t) \\ \pi_2'(t) &= \lambda_1\pi_1(t) - \mu_2\pi_2(t) \end{aligned} \tag{2.30}$$

where $\pi_i'(t)$ = rate of change for state i.

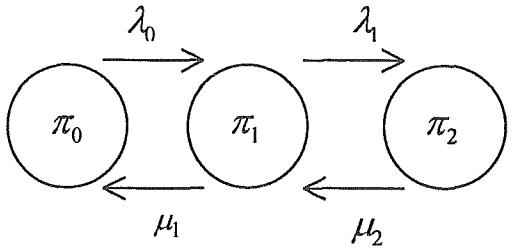

Figure 2.3: State rate transition diagram of the $(M/M/2) : (FIFO/2/\infty)$

Also by total law of probability

$$\sum_{i=0}^{2} \pi_i'(t) = 0$$

The formulae given by (2.30) are a particular case of the Chapman-Kolmogorov equation. This system is simply given in matrix form by (2.20) which in detail is

$$\begin{bmatrix} \pi_0' \\ \pi_1' \\ \pi_2' \end{bmatrix} = \begin{bmatrix} \pi_0 & \pi_1 & \pi_2 \end{bmatrix} \begin{bmatrix} -\lambda_0 & \lambda_0 & 0 \\ \mu_1 & -(\lambda_1 + \mu_1) & \lambda_1 \\ 0 & \mu_2 & -\mu_2 \end{bmatrix}$$

Let us assign $\lambda_0 = \lambda_1 = 4$ and $\mu_1 = 2, \mu_2 = 4$.

So the transition matrix is

$$\mathrm{A} = \begin{bmatrix} -4 & 4 & 0 \\ 2 & -6 & 4 \\ 0 & 4 & 4 \end{bmatrix}$$

Solving for the eigenvalues ($\det(\mathrm{A} - \lambda \mathrm{I}) = 0$) we obtain $0, -4, -10$. This leads to the transient solution:

$$\begin{aligned} \begin{bmatrix} \pi_0 \\ \pi_1 \\ \pi_2 \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & -1 \\ 2 & 3 & 2 \end{bmatrix}^{-1} \begin{bmatrix} k_1 \\ k_2 e^{-4t} \\ k_3 e^{-10t} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{5} & \frac{1}{3} & \frac{1}{15} \\ \frac{2}{5} & 0 & -\frac{1}{5} \\ \frac{2}{5} & \frac{1}{3} & \frac{2}{15} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 e^{-4t} \\ k_3 e^{-10t} \end{bmatrix} \\ &= \begin{bmatrix} \frac{k_1}{5} + \frac{k_2 e^{-4t}}{3} + \frac{k_3 e^{-10t}}{15} \\ \frac{2k_1}{5} - \frac{k_3 e^{-10t}}{5} \\ \frac{2k_1}{5} - \frac{k_2 e^{-4t}}{3} + \frac{2k_3 e^{-10t}}{15} \end{bmatrix} \end{aligned}$$

As $e^{-at} \rightarrow 0$ as $t \rightarrow \infty$ we can remove the time components, so

$$\begin{bmatrix} \pi_0 \\ \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} \frac{k_1}{5} \\ \frac{2k_1}{5} \\ \frac{2k_1}{5} \end{bmatrix}$$

and by the total law of probability,

$$\begin{bmatrix} \pi_0 \\ \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{5} \\ \frac{2}{5} \\ \frac{2}{5} \end{bmatrix}$$

which gives the steady state distribution or long term queue behaviour.

Balance equations

This concept is succinctly summarised in [Ng, 1996] and is key to the solution processes in this work. The use of balance arises when considering the influx/outflux principle. Consider the simple M/M/1 queue in steady state. We have

$$(\lambda + \mu)\pi_k = \lambda\pi_{k-1} + \mu\pi_{k+1}$$

when in state k. This equation considers all influx and outflux into state k. Clearly, $(\lambda + \mu)\pi_k$ is the stochastic flow out (outflux) of state k, and $\lambda\pi_{k-1} + \mu\pi_{k+1}$ represents the stochastic flow into (influx) state k. This gives us the global balance equation for this Markov chain.

If we are to consider the flux across the boundary between states k and $k-1$, we have the equation

$$\lambda\pi_{k-1} = \mu\pi_k$$

which is a local balance equation. Since local balance implies global balance, the solution to the local balance equation is that of the global balance equations.

The $(M/M/1) : (FIFO/c/\infty)$ asymmetric queue

The method of balance is well illustrated further by examining the simple finite queue $(M/M/1) : (FIFO/c/\infty)$. This system is the simplest of all finite queues and one which can be used to compare with more complicated priority based finite queue models.

The finite storage queue grew from the desire to model telephony systems and is one of the oldest in queueing theory. It is a system whereby customers may be unable

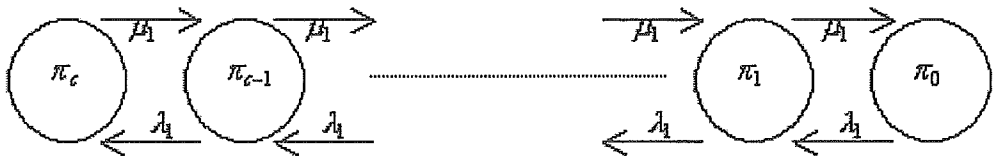

Figure 2.4: State rate diagram for $(M/M/1) : (FIFO/c/\infty)$

to enter the system and are lost. The state-rate transition diagram is given in Figure 2.4 and we derive the balance equations by considering each state:

$$\begin{aligned} \mu_1\pi_1 &= \lambda_1\pi_0 \\ \mu_1\pi_2 + \lambda_1\pi_0 &= (\lambda_1 + \mu_1)\,\pi_1 \\ &\vdots \\ \mu_1\pi_{i+1} + \lambda_1\pi_{i-1} &= (\lambda_1 + \mu_1)\,\pi_i \\ &\vdots \\ \mu_1\pi_c &= \lambda_1\pi_{c-1} \end{aligned} \tag{2.31}$$

and

$$\sum_{i=0}^{c} \pi_i = 1 \tag{2.32}$$

Using the general steady state solution for a birth death process

$$\pi_j = \pi_0 \prod_{i=0}^{j-1} \frac{\lambda_i}{\mu_{i+1}} \tag{2.33}$$

for $j = 1, 2, ...$we obtain

$$\begin{aligned} \pi_j &= \pi_0 \prod_{i=0}^{j-1} \rho_1 \\ &= \pi_0 \rho_1^j \end{aligned} \tag{2.34}$$

and we have

$$\pi_0 + \pi_0\rho_1^1 + \pi_0\rho_1^2 + ... + \pi_0\rho_1^c = 1 \tag{2.35}$$

so

$$\pi_0 \left(1 + \rho_1^1 + ... + \rho_1^c\right) = 1 \tag{2.36}$$

implies

$$\begin{aligned} \pi_0 &= \frac{1}{\left(1 + \rho_1^1 + ... + \rho_1^c\right)} \\ &= \frac{1 - \rho_1}{1 - \rho_1^{c+1}} \end{aligned} \tag{2.37}$$

using the geometric sum result given $0 < \rho_1 < 1$.
So

$$\pi_j = \left(\frac{1-\rho_1}{1-\rho_1^{c+1}}\right)\rho_1^j \tag{2.38}$$

and if $\rho_1 = 1$

$$\pi_j = \left(\frac{1}{c+1}\right) \tag{2.39}$$

Summarising

$$\pi_j = \begin{cases} \frac{1-\rho_1}{1-\rho_1^{c+1}}\rho_1^j & \rho_1 \neq 1 \\ \frac{1}{c+1} & \rho_1 = 1 \end{cases} \tag{2.40}$$

For the expected number in the system, using

$$L_s = \sum_{j=0}^{c} j\pi_j \tag{2.41}$$

we have

$$L_s = \sum_{j=0}^{c} j\left(\frac{1-\rho_1}{1-\rho_1^{c+1}}\rho_1^j\right) \tag{2.42}$$

and after simplifications,

$$L_s = \begin{cases} \frac{\rho_1\left(1-(c+1)\rho_1^c+c\rho_1^{c+1}\right)}{\left(1-\rho_1^{c+1}\right)(1-\rho_1)}\rho_1^j & \rho_1 \neq 1 \\ \frac{c}{2} & \rho_1 = 1 \end{cases} \tag{2.43}$$

This is the simplest finite queueing model and shall be touched on again later.

2.6 Some performance measures

One of the most general of all results in queueing theory is Little's Formula. It is as follows:

Theorem 2 *Suppose that a queue has a limiting distribution and customers arrive and are served in single units (i.e. not in bulk). For any arbitrary time point in the steady state the following relationship holds:*

$$L_q = \lambda W_q.$$

From this it can be shown (see [Kleinrock, 1975a]) that

$$L_s = \lambda W_s.$$

This is used to derive the mean waiting time of customers in the queue (not in service)

$$W_s = W_q + \mu^{-1}$$

Also

$$L_s = L_q + \rho$$

follows from Theorem 2.

2.7 Matrix-Analytic Methods

The matrix-analytic and matrix-geometric approach developed by [Neuts, 1981] combines the beauty of matrix construction with the elegance of transform-free solutions. The Quasi Birth Death process is a generalisation of a simple Birth Death process and, since the latter can be solved algorithmically (using balance equations), it is therefore plausible that the former could as well. This, as shown by Neuts, turns out to be the case. Through lexicographical ordering of the states, the Quasi Birth Death process is displayed and married with the desired probability vector. In this way a vast array of complicated models can be solved using an algorithmic approach which avoids the problems of transform inversion apparent in large queueing systems.

There exists since the conception of this work a huge variety of problems solved by this method. More recent and excellent sources of detailed matrix-analytic theory include [Latouche and Ramaswami, 1999] and [Ramaswami, 1986]. Of interest is the Matrix-geometric method for which many techniques exist to solve the steady-state probabilities. Again, more rigorous analysis can be found in [Neuts, 1981] and [Latouche and Ramaswami, 1999]. Detailed derivation of matrix-analytic schemes can also be found in [Ramaswami, 1986] and in [Latouche and Ramaswami, 1993]. An inspiration for this work is the approach taken by the authors in

[Latouche and Ramaswami, 1993]. By orderly construction of the rate matrix, they show how Quasi-Birth Death processes (QBD) can be solved recursively (pp.130-132). By considering a QBD process that is aperiodic and positive recurrent they partition the probability vector π into submatrices. They then undertake decomposition to find the steady state probabilities. Examples of matrix-analytic applications can be found in work such as [Nunez Queija, 1997]. As matrix-analytic methods are essentially algorithmic this is the approach adopted in this thesis. Further, the algorithms in this thesis translate well into Maple or C based codes.

The fundamental difference between the algorithm here and that in the aforementioned texts is that the dominating structures are used to solve the system in stages,

and it is the unique order in which the system is solved that leads to a tractable solution. Solution of the steady state probabilities using conventional matrix-analytic methods is not possible due to the unique unorthodox structure of the MPDQ. More on the structure will be detailed in Chapter 5 and 6, and Chapter 4 presents the unique process as an introduction for a single preemptive priority finite buffer model.

2.8 Poisson Arrivals See Time Averages (PASTA)

The Poisson process has the unique principle called the PASTA property [Wolff, 1982]. In queueing theory, $\pi_i = P(\text{an arrival sees } i \text{ customers in the system})$, can be estimated by sampling several random instants of time and compute the relative frequency of the instants when there are i customers. So a Poisson arrival acts as a random observer and sees the queue in equilibrium. If we let $N(t)$ = the number of customers in a queue at time t, then let $P(N(t) = k)$ be the probability of there being k customers in the queue. So the steady state probability of having k customers in the queue is given by $\pi_k = \lim_{t\to\infty} P(N(t) = k)$.

To prove the PASTA property, we begin with defining the probability of seeing k customers in the queue just prior to an arrival:

$$a_k = \lim_{t\to\infty} P(N(t^-) = k | \text{an arrival occurs at } t)$$

The PASTA property states that if the arrival process is Poisson then

$$a_k = \pi_k, n = 0, 1, ...$$

Theorem 3 *Define $A(t, t+\Delta t)$ as the event that an arrival occurs in time $[t, t+\Delta t)$. Given that a customer arrives at time t, we have*

$$\begin{aligned} a_k(t) &= P(N(t^-) = k | \textit{arrival at time } t) \\ &= \lim_{\Delta t\to 0} P\left(N(t^-) = k | A(t, t+\Delta t)\right) \\ &= \lim_{\Delta t\to 0} \frac{P\left(N(t^-) = k, A(t, t+\Delta t)\right)}{P(A(t, t+\Delta t))} \\ &= \lim_{\Delta t\to 0} \frac{P\left(N(t^-) = k\right) P\left(A\left(t, t+\Delta t\right)\right)}{P\left(A\left(t, t+\Delta t\right)\right)} \\ &= P\left(N(t^-) = k\right) \end{aligned}$$

Now as $t \to \infty$ we obtain

$$a_k = \lim_{t\to\infty} a_k(t) = \lim_{t\to\infty} P\left(N(t^-) = k\right) = \pi_k$$

2.9 Laplace Transforms

The Laplace transforms technique has been developed as a means of solving differential equations amongst other things. The standard procedure used in solving a linear differential equation with constant coefficients involves obtaining the characteristic equation, searching for a particular solution and finding the coefficients to match the boundary conditions. Laplace transforms provide a convenient alternative, turning many differential equations into an algebraic equation and giving the complete solution in a few steps.

The one-sided Laplace transforms of a function $f(t)$, a real variable function of t, is defined by

$$\mathcal{L}[f(t)] = F(s) = \int_0^\infty e^{-st} f(t) dt,$$

where $F(s)$ is the transform, a function of the complex s which $\mathrm{Re}(s) > 0$ in which $\mathrm{Re}(s)$ = real part of s.

Its usefulness for differential equations comes from recognizing that differentiation just multiplies the Laplace transforms by s, followed by a shift:

$$\begin{aligned}\mathcal{L}\left[\frac{df(t)}{dt}\right] &= \int_0^\infty e^{-st}\frac{df(t)}{dt}dt \\ &= e^{-st}f(t)|_0^\infty + s\int_0^\infty e^{-st}f(t)dt \\ &= sF(s) - f(0)\end{aligned}$$

Furthermore, taking Laplace transform of second derivative gives

$$\begin{aligned}\mathcal{L}\left[\frac{d^2f(t)}{dt^2}\right] &= \int_0^\infty e^{-st}\frac{d^2f(t)}{dt^2}dt \\ &= e^{-st}\frac{df(t)}{dt}|_0^\infty + s\int_0^\infty e^{-st}\frac{df(t)}{dt}dt \\ &= s^2F(s) - sf(0) - \frac{df(0)}{dt}\end{aligned}$$

and by induction for higher derivatives, we obtain

$$\begin{aligned}\mathcal{L}\left[\frac{d^nf(t)}{dt^n}\right] &= \int_0^\infty e^{-st}\frac{d^nf(t)}{dt^n}dt \\ &= s^nF(s) - s^{n-1}f(0) - s^{n-2}\frac{df(0)}{dt} - \cdots - \frac{d^{n-1}f(0)}{dt^{n-1}}\end{aligned}$$

The k^{th} moments of a non-negative random variable X are related to the Laplace transform $F(s)$ by means of the following result:

$$E(X^k) = (-1)^k F^{(k)}(0)$$

where $F^{(k)}(s)$ is the k^{th} derivative of $F(s)$.

Another useful property of Laplace transforms is this:

If

$$f * g(t) = \int_0^\infty f(t-x)g(x)dx$$

is the convolution of two functions $f(x)$ and $g(x)$ with non-negative domains, then

$$\mathcal{L}(f * g) = \mathcal{L}(f)\,\mathcal{L}(g)$$

Chapter 3

Scheduling Regimes in Finite Buffer Models

3.1 Introduction

This chapter introduces the operational characteristics of the single and dual queueing models. In Parts II and III these models are examined analytically for 2 classes of customers. In Part IV, simulation studies are performed for up to five classes of customers. Firstly, the schematics for the single and dual finite models are given. Next, the types of scheduling disciplines are discussed. Then each system's model dynamics under various queueing schedules are given through some examples. This is necessary as the MPDQ behaves differently to the single finite buffer models under different queueing regimes.

3.2 Model Schematics

In this section the design of the models is given. Each design is split into three: the arrival area, the waiting room, and the server.

3.2.1 Prioritised Single Finite Buffer Model

The general algorithm (for i classes) for the prioritised single finite buffer is given in Figure 3.1.

All arrivals are independent Poisson with rate $\lambda_i, i = 1, ..., k$ classes. Service times are independently exponentially distributed with rate $\mu_i, i = 1, ..., k$ classes.

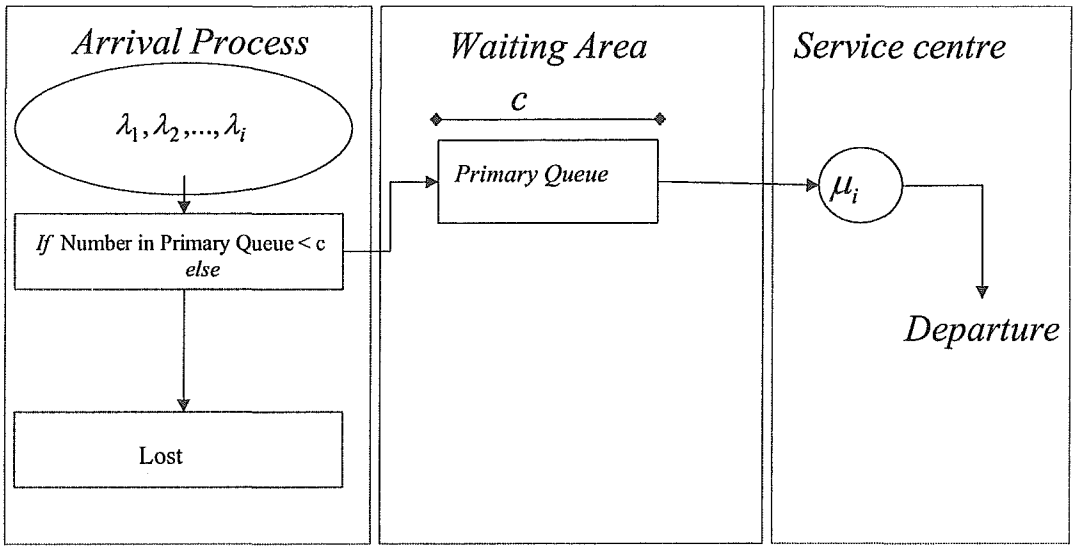

Figure 3.1: Single finite buffer algorithm

3.2.2 Dual Queue Finite Buffer Models

Figure 3.2 illustrates the dual queue. Now c_1 = capacity of the primary queue and c_2 = capacity of the secondary queue. If an arriving customer meets a full primary queue, they must wait in the secondary queue. If the secondary queue is also full then the customer is lost.

The time to jump to the head of line, time to check if the buffer is full or not, and the time in which traffic is transferred to the primary queue are considered to be zero. In reality, this is hardware dependent, with larger buffer sizes possibly delaying the model. If there is no space in the buffer (queue) then the customer is lost. In the steady-state analysis we are using a Highest-Class First (HCF) regime within the dual queue models. This means that higher classed customers will jump to the head of the line before any lower classed customers only within the *same* queue. So a lower classed customer in the primary queue cannot be ejected to the secondary queue even in the presence of higher classed customers within the secondary queue. This differs when considering a single queue in that all high classed customers *in the system* are moved to the head of the queue, whereas in a dual queue, all the high classed *within each queue* are moved to the head. A full primary queue blocks any entry from the secondary queue, hence there is the opportunity for lower classed customers to leave the system first, even in the presence of higher classed ones.

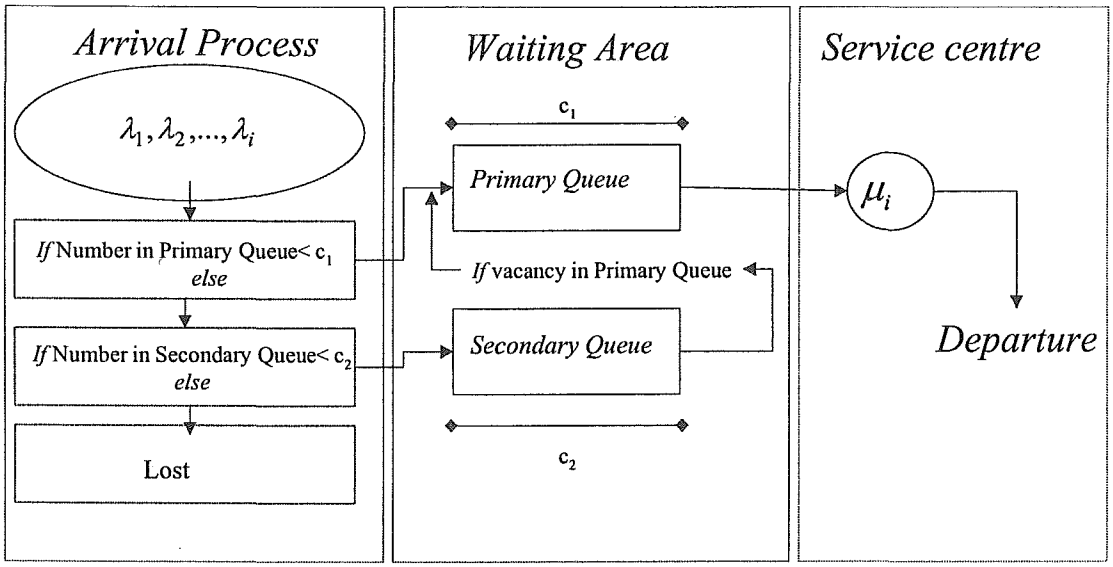

Figure 3.2: MPDQ Algorithm

3.3 Queueing Regimes

Now for a brief summary of the four queueing disciplines analysed here for use in both the single and dual queue simulations. First is the First In First Out (FIFO) queueing discipline. This is the simplest, and is also known as First Come First Serve (FCFS). It is simple to implement as customers are considered sequentially. It is common to many communications systems and networks. Next is the Last In First Out (LIFO), or Last Come First Serve (LCFS) queueing discipline. This discipline is not unusual to communications networks. It involves serving the most recent customers first, while customers already in the queue are kept waiting. Lowest Class First (LCF) is one of the two priority disciplines. In this discipline, lowest class customers jump to the head of the queue behind any already present customers of the same class, and ahead of any higher class of customer. Highest Class First (HCF) is the most important model here as this is the assumed scheduling discipline of the queues in the analysis in Parts II and III. In this discipline, the highest classed customers jump to the head of the queue behind any already present customers of the same or higher class, who, in turn, are in front of any lower class of customer. In reference to the dual queue, by having both queues under the HCF regime implies class differentiation within each queue with the possibility of some lower classed customers still getting through the system in the presence of higher classed ones. This is the motivation for the analysis of the MPDQ. This is a form of congestion control in that the opportunity for some lower classed to exit the system remains due to the structure of the MPDQ system. The partitioning

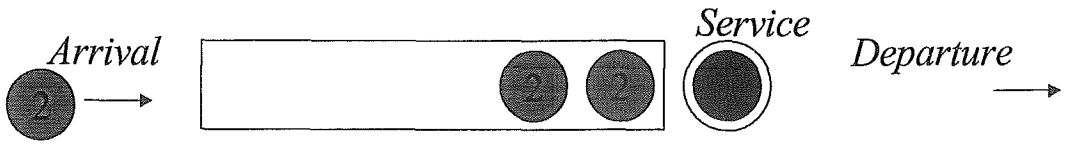

Figure 3.3: Priority Finite Buffer Model Example

of a single queue and restricting entry into the primary when full allows lower classed customers to leave the system, even in the presence of other higher classed customers in the system. This is perceived as 'fairness' to lower classed customers, as they can exit the system in the presence of higher classed customers. In a sense this could be deemed as 'unfair' to higher classed customers, however, as will be shown, this can only occur when the primary queue is full, so they are not disadvantaged all of the time.

An example of a single finite buffer model with a class 1 customer in service, 2 class 2 customers waiting, and an arriving class 2 customer is given in Figure 3.3.

3.4 Service Disciplines

3.4.1 Preemption

The preemptive priority scheme is where a customer of a higher class interrupts and ejects a lower classed customer from service. Three types of preemptive schemes are possible: *preemptive-resume, preemptive repeat-identical* and *preemptive repeat-different.* The resume scheme allows preempted customers to continue service from their initial point of interruption. The other schemes require customers to start over after preemption. The repeat-identical allows the customer to begin again with the full amount of service time required, whereas the repeat-different gives a random service time which does not account for the time lost. Due to the memoryless property of the exponential distribution, there is no need to distinguish between these three types of service. Customers of the same class are served on a FIFO basis. An example is given in Figure 3.4

3.4.2 Non-preemption

The non-preemptive scheme is where a customer of a higher class must wait for a lower classed customer to complete service if the latter was found to be in service upon the arrival of the higher class customer. An example is given in Figure 3.5.

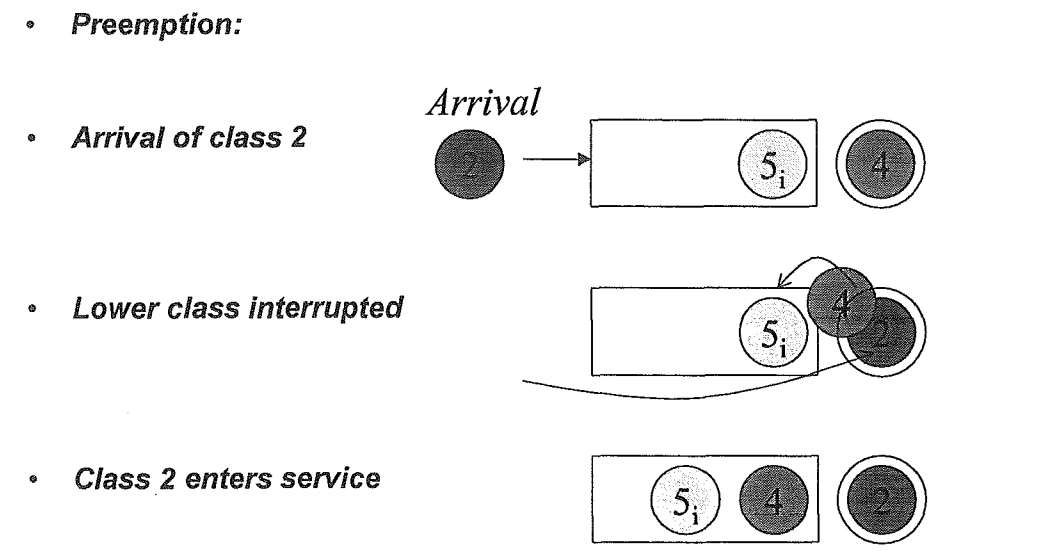

Figure 3.4: Preemption example

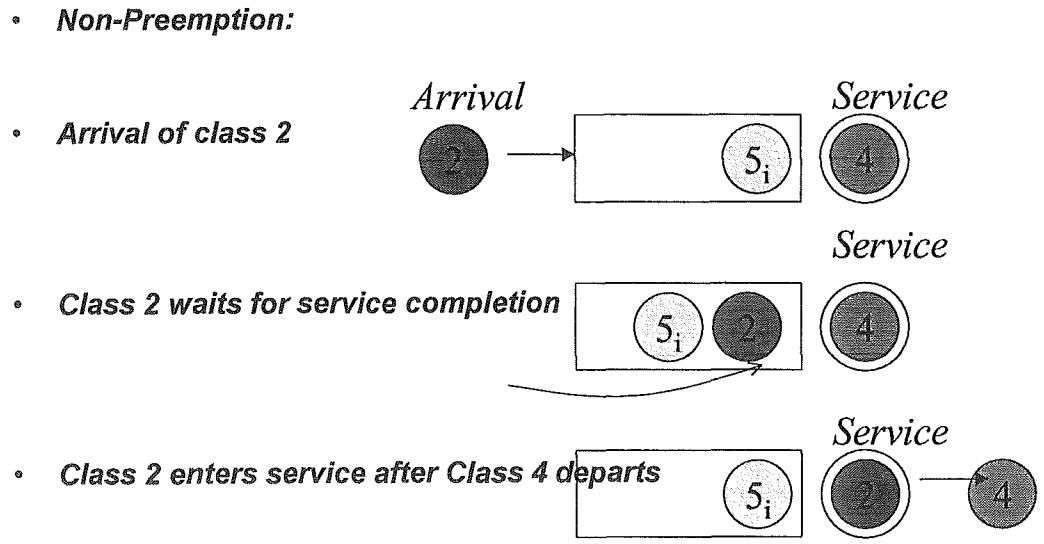

Figure 3.5: Non-preemption example

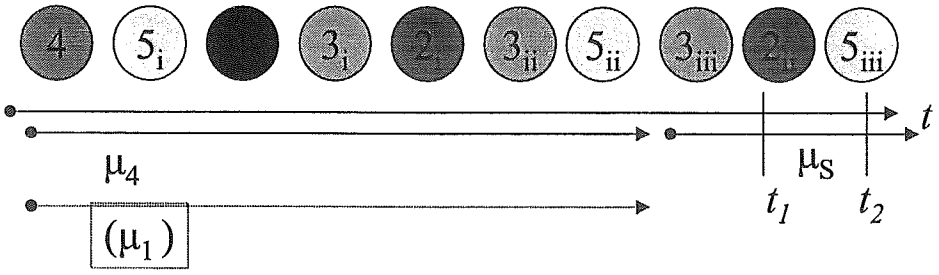

Figure 3.6: Arrival pattern and service rates

3.5 Scheduling Regimes of the Single and Dual Queue

In this section, some examples of the behaviour of single and dual queueing models are given. Consider the arrival pattern and time-line as given in Figure 3.6.

To show how the MPDQ compares with a single queue under various queueing and service regimes, all the following examples are given with the service discipline stated where applicable. From Figure 3.6 the class 4 customer arrives first and immediately enters service. The next customer is class 5, and the subscript is used to identify the order in which the customer arrives. The class 4 customer completes service after the arrival of the second class 5 customer. However if preemption occurs, the class 1 shall enter service, hence the lower line indicating the service time of class 1 customer. The next service rate is μ_S where S =class of next customer. This is dependent upon the queueing regime. What shall be shown is the snapshot of the state of the systems from epoch t_1 to t_2.

All systems have a total waiting room capacity of 10. Figure 3.7 shows the MPDQ under the FIFO regime. What is noticeable is that the lowest class of customer is next in service at t_1, then departs with the highest class entering service (t_2).

Next is the LIFO, shown in Figure 3.8. For LIFO, the customer at the tail of the primary queue is taken into service. At t_1 the second class 3 customer enters service as it is the last to enter the primary queue. Next is the second class 5 customer.

The HCF discipline for non-preemptive is given in Figure 3.9. It can be seen that the class 1 customer enters service immediately after the completion of the first customer's service. All the customers in the primary queue are ordered by arrival time and class. Notice the second class 2 customer reaches the head of the line in the primary queue after a vacancy appears. This customer enters the system as the class 1 is being served. The class 2 customer reaches the H-O-L of the secondary queue, and joins the primary queue after the departure of the class one customer.

The preemptive case for HCF is given in Figure 3.10. Now we see that the notion of 'fairness' is reduced as the higher class will always interrupt the lower class in service.

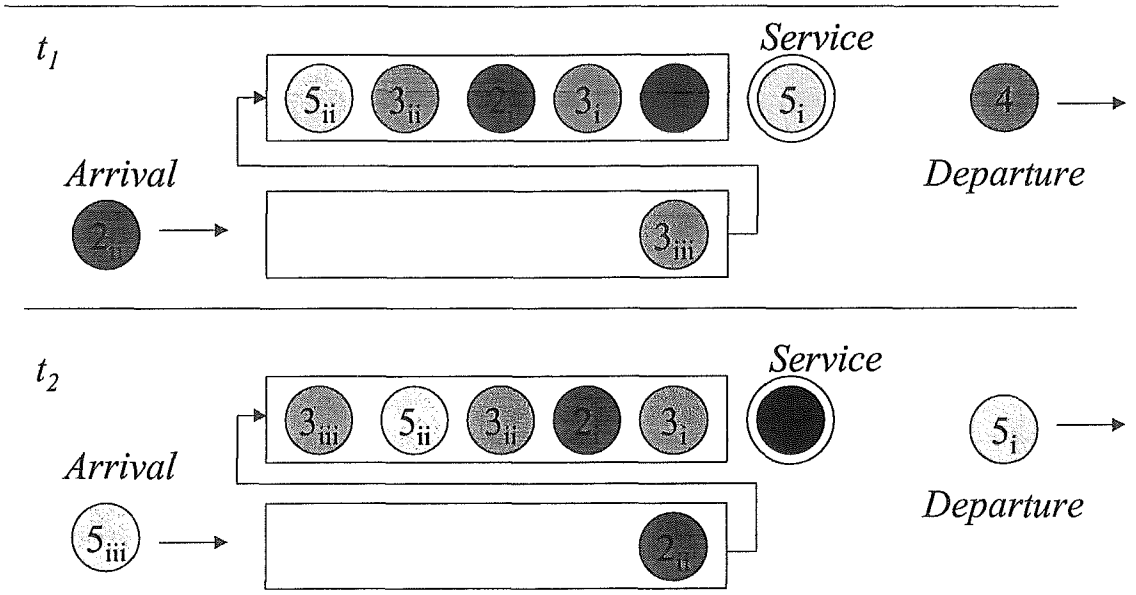

Figure 3.7: FIFO, Dual Queue, Capacity of 10

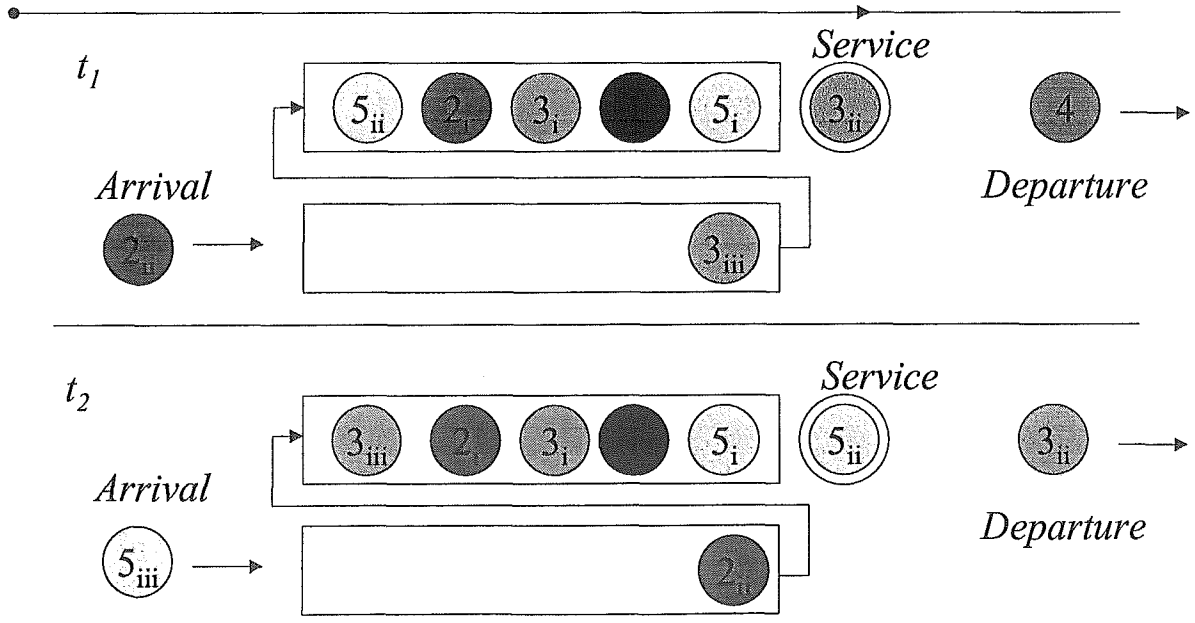

Figure 3.8: LIFO, Dual Queue, Capacity of 10

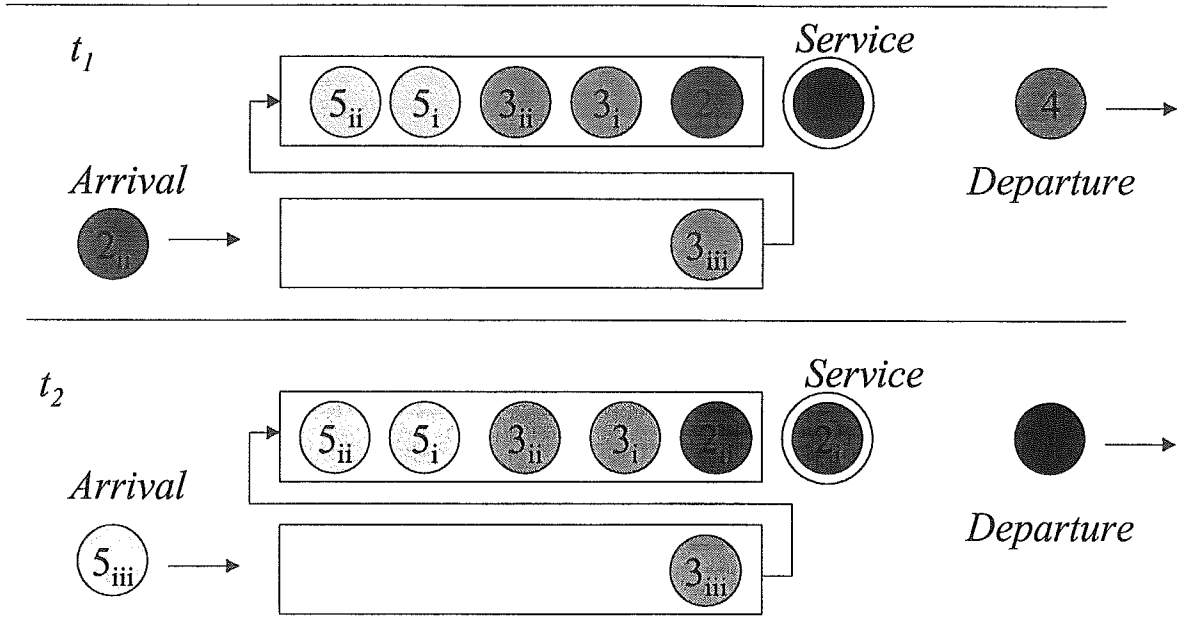

Figure 3.9: Non-preemptive HCF, Dual Queue, Capacity of 10

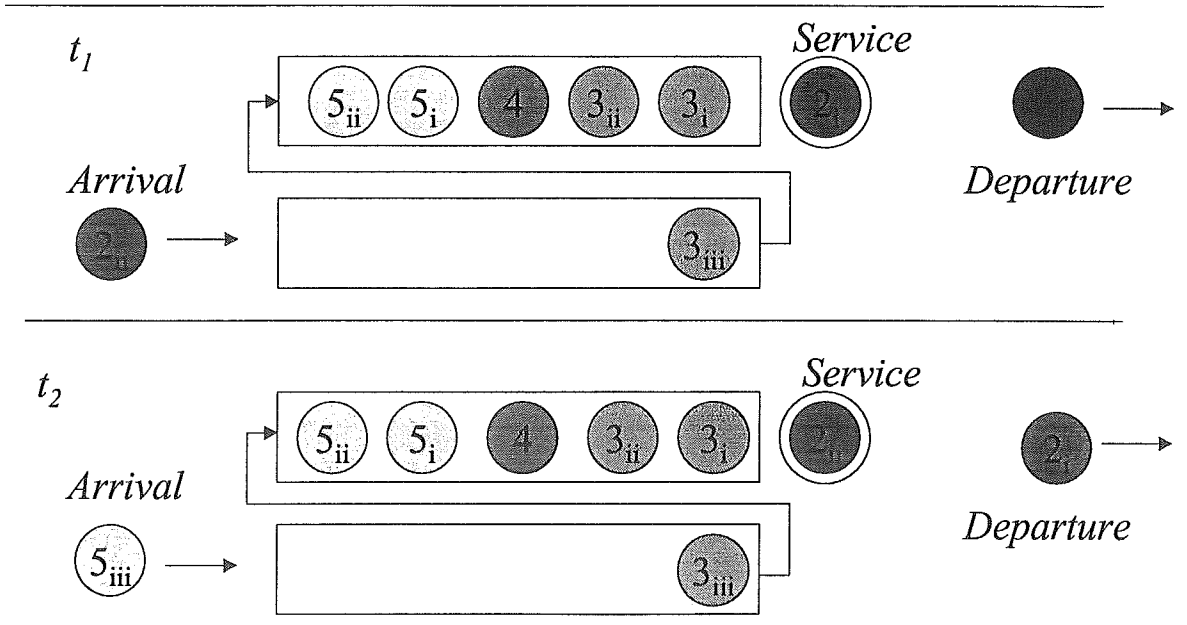

Figure 3.10: Preemptive HCF, Dual Queue, Capacity of 10

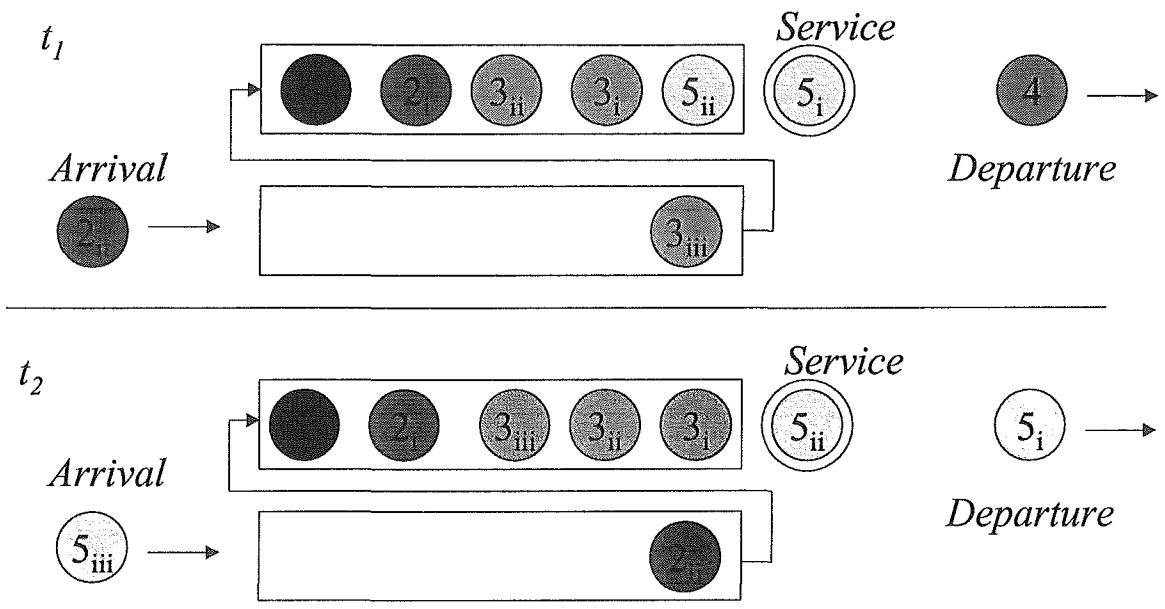

Figure 3.11: Non-preemptive LCF, Dual Queue, Capacity of 10

The last of the disciplines is the non-preemptive LCF, as seen in Figure 3.11. The first class 5 customer is served, followed by the second class 5. Noticeably the newly arrived class 2 now waits in the secondary queue even after the arrival of the third class 5 customer.

The preemptive LCF sees the lowest class through even quicker than the preemptive HCF, as illustrated in Figure 3.12. The preemption means that the class 4 customer is ejected from service and has to wait for the two class 5 customers to be served.

As seen, each queueing regime leads to a different waiting pattern. The FIFO and LIFO offer no priority assistance to customers, whereas HCF and LCF reorder each new arrival if necessary. The comparison of the departure order if no further arrivals occur is given in Table 3.1.

Only the preemptive model allows the first class customer to see its time through service over the already present traffic. This leads to some findings regarding the operational characteristics of the model detailed next.

3.5.1 Service Schemes and Priorities

As mentioned, two types of service schemes are considered here - preemptive and non-preemptive. Some remarks regarding the behaviour of the priority queues are given.

Remark 4 *In the case of preemption in a 2-class MPDQ, the only means whereby a*

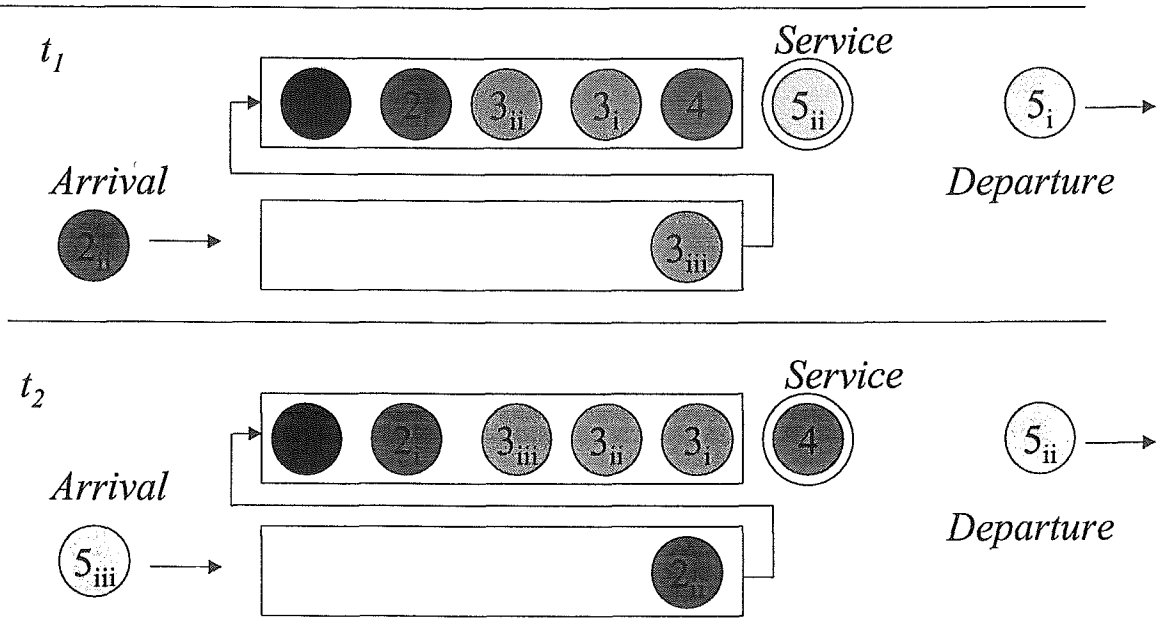

Figure 3.12: Preemptive LCF, Dual Queue, Capacity of 10

	Single queue	Preemptive DQ				Non-Preemptive DQ	
Customer	FIFO	FIFO	LIFO	HCF	LCF	HCF	LCF
4	1	1	1	7	1	1	1
5_i	2	2	10	8	2	8	2
1	3	3	9	1	10	2	10
3_i	4	4	8	4	4	5	4
2_i	5	5	7	2	8	3	8
3_{ii}	6	6	2	5	5	6	5
5_{ii}	7	7	3	9	3	9	3
3_{iii}	8	8	4	6	7	7	7
2_{ii}	9	9	5	3	9	4	9
5_{iii}	10	10	6	10	6	10	6

Table 3.1: Rank of departure for customers by regime

class 2 customer shall not be interrupted whilst in service is if the primary queue is full of class 2 customers.

This occurs due to the operation of the primary queue within the MPDQ. The priority schemes apply to both queues independently (i.e. they are not resorted as a single queue). This is important when considering the construction of the transition of states for the preemptive MPDQ. This Remark can now be extended to k number of classes of customers:

Remark 5 *In the case of preemption in a MPDQ, the only means whereby a customer of a lower class $(j, j \leq k)$ that is currently in service cannot be preempted by a higher class customer $(j-1, ..., 1)$ is if: (i) No higher classed customers are present or arrive to the primary queue prior to the completion of j's service; or (ii) The primary queue is full of the same classed customer as that in service.*

This is important as the MPDQ contains more classes of customers. This leads to difficulties in treating certain customers 'fairly', as they are ejected from service back to the queue. 'Safety' is maintained for the ejected customers in that they return to the head of the customers of the same class, and cannot be ejected from the primary queue. This leads to the next remark:

Remark 6 *No customer is sent back to the secondary queue from the primary queue for any service regime or queueing discipline.*

This assists in the notion of fairness, ensuring that customers who have reached the primary queue should enter service. However, they still must wait for all higher class customers to be served. This is a consequence of the fact that when the primary queue is full, no customer in the secondary queue can join it.

The definition of non-preemptive service is as follows:

Remark 7 *In the case of non-preemption, no customer can be interrupted at any stage of its service by another customer, regardless of class.*

This leads to clear differences in performance characteristics, which are explored later in Part IV where a comparison of preemptive and non-preemptive is detailed. For both the preemptive and non-preemptive models, there is a case which needs to be clearly defined.

Consider the situation where a single class 1 customer is waiting at the head of the secondary as seen in Figure 3.13. A class 1 customer is in service, and the primary queue is full of class 2 customers. In the case of preemptive scheduling there is no

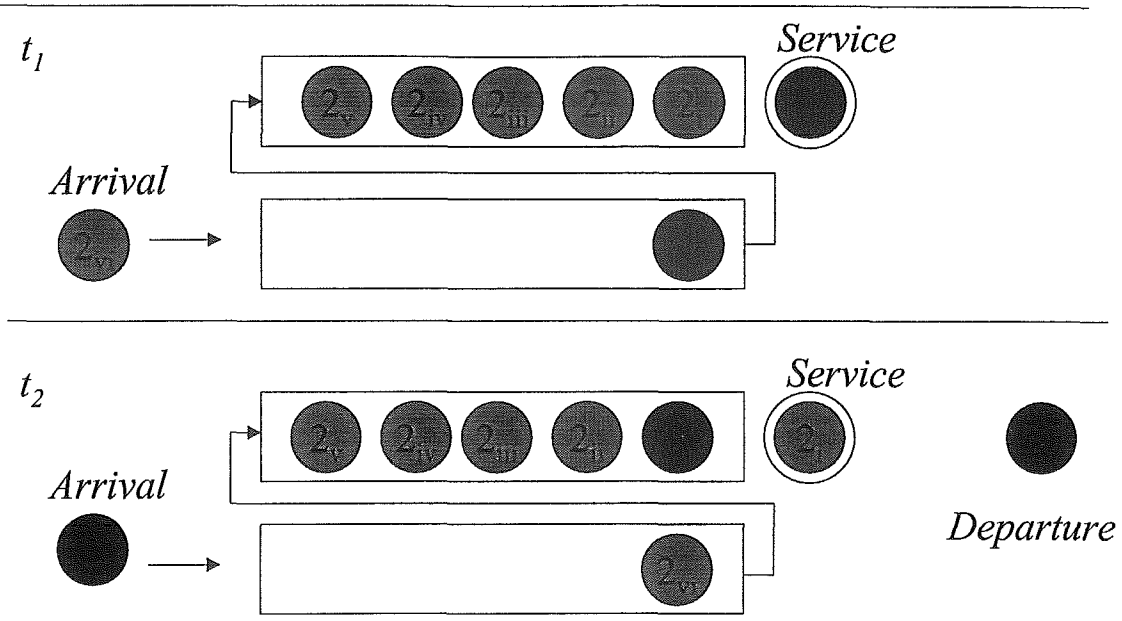

Figure 3.13: Tricky scenario

issue as the class 1 customer shall enter the primary queue and preempt the class 2 customer in service. However, for non-preemptive scheduling we assume that the class 2 customer enters service immediately over the class 1 customer entering from the secondary queue. This is an introduction to the idea of letting some lower class customers through the system to enhance the notion of fairness to lower classed customers.

3.5.2 Other Congestion Management Models

There exists a vast array of congestion management models for which the aim is to increase QoS to customers. [Hayes, 2001] (pp.12-22 Section 1.4) has an excellent taxonomy of buffer management and scheduling models used to enhance QoS. This is an excellent source for the reader who wishes to investigate other congestion management models.

Part II

SINGLE QUEUE MODELS

Chapter 4

Priority Single Buffer Queueing Models

4.1 Introduction

In this chapter, the detailed analysis of priority finite buffer models begins. A new approach to solving both the steady state probabilities and waiting time distribution for a preemptive priority finite buffer queue are given. Some classical work will be looked at first.

4.2 Finite Buffer Models

Queueing models with a preemptive priority discipline have been investigated in various ways under both a theoretical and an applied perspective. Recently this type of model has been shown to have special relevance in communications, especially where there are a limited number of spaces for information packets to be stored. These models are also being used with differentiated traffic in mobile and data transfer systems. A single server preemptive priority finite buffer model is one such model used in these applications that is investigated here.

Pioneering work on preemptive priority queues was undertaken in the 1950-60's, beginning with [White and Christie, 1958] where the preemptive queueing model with an infinite buffer was introduced. Similar work assuming an infinite buffer and incorporating a finite number of different classes of customer includes [Chang, 1965], [Heathcote, 1960] and [Yeo, 1963]. The steady-state solutions obtained in [Heathcote, 1960] utilized recurrence relations involving generating functions, while those in [Chang, 1965] and [Yeo, 1963] utilized Laplace transforms to obtain their results. [Miller, 1981] revisited the infinite buffer size model equipped with the matrix-analytic approach later

summarized in [Neuts, 1981] and [Neuts, 1989]. [Miller, 1981] considered both the preemptive and non preemptive model. [Gail, Hunter and Taylor 1992] also extensively used matrix methods to obtain the solutions of the same model with multiple servers. [Marks, 1973] developed an algorithmic approach to solving both preemptive and non-preemptive $M/M/1$ queues with infinite waiting room. [Singh, 1977] provided an approach similar to that used in this chapter, that is, analysis via global balance. While [Singh, 1977] did not investigate waiting times or allow for class variant service rates, he solved the steady state probabilities (via simple recursion) for a system of parallel servers. This work also involved the use of matrix-calculus to obtain the state rate transitions.

Other papers which involve variations of the preemptive priority queueing scheme have refined and reworked the approach to solutions of earlier models. A quality control perspective involving cost optimization of products with preemptive priority is found in [Venkatesan and Zeephongsekul, 1992]. In [Stephan, 1958], the author looks at two priority classes in a preemptive model evaluating waiting times and steady-state probabilities with infinite storage. Recent work with a communication emphasis has involved the implementation of class partitioning in a single buffer. [Sharma and Virtamo, 2002] considered the number of packets in a buffer to determine loss and steady state probabilities. The recursive approach to solving preemptive priority queueing systems is explored by [Cidon et. al., 1993]. This work provides results based on generating functions to obtain probabilities. Another analysis of a preemptive model was considered in [Kramer, 1987] using an approach based on truncated power series expansion.

More recent work on the non-preemptive priority finite buffer model was undertaken by [Wagner and Krieger, 1999]. They analysed both the steady state distribution and waiting time of this queueing model for two classes of customer. Other work on the non-preemptive model in this field is small in comparison to the preemptive model. The work of [Miller, 1981] solved both preemptive and non-preemptive $M/M/1$ models using recursive computational formulas based on [Neuts, 1981]. Prior to this work [Marks, 1973] tackled the non-preemptive steady state solution by investigation of the balance equations using results from [Morse, 1958]. He too developed an algorithm to solve the steady state probabilities. The interested reader will find these works excellent sources of information regarding different solution processes for non-preemptive queues which are not covered here.

In Section 4.4.1 the state space of the preemptive priority finite buffer model is defined and the infinitesimal generator matrix and all its submatrices are exhibited. The linear system of equations governing the steady state distributions of the model are also stated. In Section 4.4.2, an algorithmic approach to obtaining the solution to the

steady state system is derived. In Section 4.4.3, a waiting time analysis using the steady state probabilities is provided, again employing an algorithmic approach. Finally, in Section 4.4.4, numerical examples illustrating the results obtained in preceding sections are provided.

4.3 Infinite Waiting Room Priority Models

The solution process for the mean waiting time of infinite waiting room models can be derived in the non-preemptive case by avoiding the steady state solutions. By considering the mean residual service time, the mean total service time of same class customers, mean total service time of customers in the system at arrival and Little's result, the expected waiting times and queue lengths can be obtained. For the preemptive model, the use of generating functions is needed to obtain the same performance characteristics (which requires the steady state solution).

As the results are well known they are not stated here, especially as this thesis is concerned with *finite* waiting room models. The interested reader will find treatment of these models in texts such as [Ross, 2001] and [Ng, 1996].

4.4 Preemptive Priority Finite Queue

In this section the approach to solving the steady-state probabilities utilizes a judicious state space labeling accompanied by a matrix-analytic approach, while the approach to obtaining the expected waiting time for different classes of customers exploits the approach established in [Wagner and Krieger, 1999] and [Wagner, 1997] for a non-preemptive queueing finite system. The fundamental difference between the new algorithm here and that in the aforementioned texts is that the dominating tridiagonal structures here are used to solve the system in stages, and it is the unique order in which the system is solved that leads to a tractable solution. Solution of the steady state probabilities using conventional matrix-analytic methods, whilst possible for the preemptive priority finite buffer, is unachievable for the MPDQ models due to the unique and unorthodox structure of the transition matrix. So the aim of this section is to present the machinations of the algorithm on a simple problem, both for the steady-state probabilities and the waiting time distributions.

Two classes of customers arrive to the queueing system as two independent Poisson processes. The class of customer labeled 1 (the high priority class) has a mean arrival rate of λ_1 and the other class, labeled 2 (the low priority class), has a mean arrival rate λ_2. There is a single server dispensing an exponentially distributed service time with rate μ_i for a class i customer, $i = 1, 2$. The operational aspects of this model

were described in detail in Section 3.2.1. The distinct difference between the model in Section 4.3 and this model is that the waiting room is now finite, with capacity c.

The objectives of this section are to:

- provide a new simple algorithmic solution for the steady-state probabilities
- provide a Maple program to solve for the steady-state probabilities
- demonstrate how to obtain the expected waiting times
- investigate some performance statistics

4.4.1 State Space and Linear System of Steady State Distributions

The process describing the state of this queueing system is a homogeneous ergodic Markov chain in continuous time given by $X(t) = (N_1(t), N_2(t))$ where $N_i(t), i = 1, 2$ represents the number of class i customers in the system at time t. $X(t)$ takes values in the state space $\mathcal{S} = \{(i,j) : 0 \leq i + j \leq c\}$. The steady-state distribution of the process will be represented as a probability vector arranged according to the following lexicographical ordering:

$$\mathbf{i} = \{(i,0), (i,1), \ldots, (i, c-i)\}$$

where $i = 0, 1, 2, \ldots, c$.

The steady-state distribution is constructed so that its components are ordered using the above labelling scheme, which gives us

$$\overline{\pi}^t = \left(\overline{\pi}_{\mathbf{0}}^t, \overline{\pi}_{\mathbf{1}}^t, \ldots, \overline{\pi}_{\mathbf{c}}^t\right) \in \Re^{\frac{1}{2}(c+1)(c+2)}$$

where

$$\overline{\pi}_{\mathbf{i}}^t = (\pi_{i,0}, \pi_{i,1,} \ldots \pi_{i,c-i}) \in \Re^{c-i+1}, i \in \{0, \ldots, c\}.$$

The steady-state distribution $\overline{\pi}^t$ exists and is obtained by solving the system of equations

$$\overline{\pi}^t \mathbf{A} = \mathbf{0} \tag{4.1}$$

where $\mathbf{0}$ is the vector of zeroes of an appropriate dimension and $\mathbf{A}$ is the infinitesimal generator matrix of the process. The matrix $\mathbf{A}$ is square, with dimension

$\frac{1}{2}(c+1)(c+2)$ and is partitioned as follows:

$$\mathbf{A} = \begin{pmatrix} \mathbf{D}_0 & \mathbf{N}_0 & 0 & \dots & \dots & 0 \\ \mathbf{P}_1 & \mathbf{D}_1 & \mathbf{N}_1 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \dots & \vdots \\ 0 & \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & & \mathbf{P}_{c-1} & \mathbf{D}_{c-1} & \mathbf{N}_{c-1} \\ 0 & \dots & & 0 & \mathbf{P}_c & \mathbf{D}_c \end{pmatrix} \tag{4.2}$$

State Transitions and Detailed Structures of Submatrices

The following detailed descriptions of the submatrices of **A** are obtained from the arrivals to, and departures from, each feasible state. Figure 4.1 exhibits the rate transition diagram of the system from which the components of the submatrices of **A** are derived, which are in turn used in the global balance equations. To begin with, the submatrices contained in the main diagonal, $\mathbf{D}_i$, will be defined. All submatrices contain the appropriate rate parameters λ_i and μ_i, where $i = 1, 2$, and $\lambda = \lambda_1 + \lambda_2$.

For each i, $\mathbf{D}_i$ is the infinitesimal generator corresponding to transitions between states in **i**. These are all in tridiagonal form and are square matrices:

$$\mathbf{D}_0 = \begin{pmatrix} -\lambda & \lambda_2 & 0 & \dots & \dots & 0 \\ \mu_2 & -(\mu_2+\lambda) & \ddots & \ddots & & \vdots \\ 0 & \mu_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \lambda_2 & 0 \\ \vdots & & \ddots & \ddots & -(\mu_2+\lambda) & \lambda_2 \\ 0 & \dots & \dots & 0 & \mu_2 & -\mu_2 \end{pmatrix}$$

where $\mathbf{D}_0 \in \Re^{(c+1)\times(c+1)}$

$$\mathbf{D}_i = \begin{pmatrix} -(\mu_1+\lambda) & \lambda_2 & 0 & & \dots & 0 \\ 0 & -(\mu_1+\lambda) & \ddots & & & \\ & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \lambda_2 & 0 \\ & & & \ddots & -(\mu_1+\lambda) & \lambda_2 \\ 0 & & \dots & & 0 & -\mu_1 \end{pmatrix}$$

where $\mathbf{D}_i \in \Re^{(c-i+1)\times(c-i+1)}, i = 1, 2, \dots, c-1$ and $\mathbf{D}_c = (-\mu_1)$.

Next consider the two forms of off-diagonal submatrices. The transitions and corresponding instantaneous rates giving rise to each submatrix are summarised in the accompanying tables:

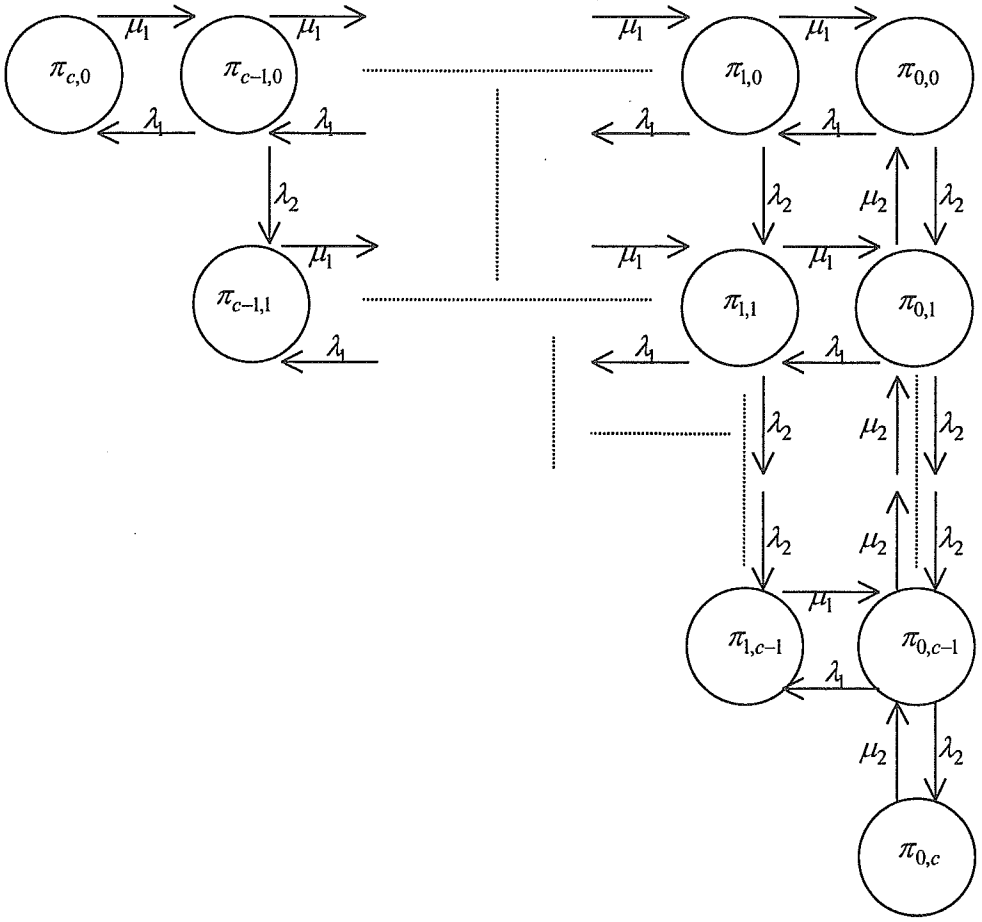

Figure 4.1: The rate transition diagram for preemptive priority finite buffer model for 2-classes of customers

$$\mathbf{N}_i = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ \vdots & \ddots & \lambda_1 \\ 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c-i+1)\times(c-i)} \tag{4.3}$$

From	To	Rate
(i,j)	$(i+1,j)$	λ_1

where $i = 0, 1, \ldots, c-1$;

$$\mathbf{P}_i = \begin{pmatrix} \mu_1 & 0 & & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \mu_1 & \ddots & \\ 0 & \cdots & 0 & \mu_1 & 0 \end{pmatrix} \in \Re^{(c-i+1)\times(c-i+2)} \tag{4.4}$$

From	To	Rate
(i,j)	$(i-1,j)$	μ_1

where $j = 0,\ldots,c-i; i = 1,\ldots,c$.

It is important to note the position of the zero components in the sub-matrices.

Linear System of Steady State Distributions

To establish the general technique used in the computational algorithm, the system of linear equations (4.1) using the matrices $\mathbf{D}_i, \mathbf{P}_i$ and $\mathbf{N}_i$ is now exhibited. The system is described by the following $c+1$ system of linear vector-matrix equations:

$$1_{\{i<c\}}\left(\overline{\pi}_{i+1}^t \mathbf{P}_{i+1}\right) + \overline{\pi}_i^t \mathbf{D}_i + 1_{\{i>0\}}\left(\overline{\pi}_{i-1}^t \mathbf{N}_{i-1}\right) = \mathbf{0} \tag{4.5}$$

where $i = 0,\ldots,c$.
In addition, $\overline{\pi}$ must satisfy the normalization constraint, that is

$$\overline{\pi}^t \overline{\mathbf{e}} = 1 \tag{4.6}$$

where $\overline{\mathbf{e}}$ is a vector of 1's.

4.4.2 Computational Algorithm

The computational algorithm used to solve (4.1) is based on the structures given by (4.5). To illustrate; consider a system with capacity $c = 4$. The system of generic equations (4.5) gives

$$\begin{aligned} \overline{\pi}_1^t \mathbf{P}_1 + \overline{\pi}_0^t \mathbf{D}_0 &= \mathbf{0} \\ \overline{\pi}_2^t \mathbf{P}_2 + \overline{\pi}_1^t \mathbf{D}_1 + \overline{\pi}_0^t \mathbf{N}_0 &= \mathbf{0} \\ \overline{\pi}_3^t \mathbf{P}_3 + \overline{\pi}_2^t \mathbf{D}_2 + \overline{\pi}_1^t \mathbf{N}_1 &= \mathbf{0} \\ \overline{\pi}_4^t P_4 + \overline{\pi}_3^t \mathbf{D}_3 + \overline{\pi}_2^t \mathbf{N}_2 &= \mathbf{0} \\ \overline{\pi}_4^t \mathbf{D}_4 + \overline{\pi}_3^t \mathbf{N}_3 &= \mathbf{0} \end{aligned} \tag{4.7}$$

The probabilities of the various states are solved by first fixing the number of class 2 customers to zero and solving for all possible class 1 customers. Next, simply solve again across states by incrementing the class 2 customers by one, and solving for all possible class 1 customers. The process continues until finally obtaining the probabilities of the whole system using the normalization constant. This algorithm is unique in that it is based on solving the system in a single iteration based upon the order of the states. At each point a direct solution is obtained for each probability which is scaled at the end of the algorithm. This process is similar to a computational algorithm used in [Bedford, 1999] and [Bruell and Balbo, 1980].

Firstly, define some common constants:

$$\alpha_{i,j} = \frac{\mu_i}{\mu_j + \lambda}, \chi_{i,j} = \frac{\lambda_i}{\mu_j + \lambda}, \rho_i = \frac{\lambda_i}{\mu_i}, i, j \in \{1, 2\}.$$

Initialize the algorithm by setting $\pi_{c-1,0} = 1$. Using this value,

$$\overline{\pi}_{\mathbf{c}} = (\pi_{c,0}) = \rho_1 \tag{4.8}$$

The next step in the procedure is the preliminary stage to working back through the **A** matrix down the partitions. Now find $\pi_{i,0}$, where $i = 0, \ldots, c-2$ using the following equation

$$\pi_{i,0} = \chi_{1,1}^{-1}\pi_{i+1,0} - \rho_1^{-1}\pi_{i+2,0} \tag{4.9}$$

working from $i = c - 2$ down to 0.

The next stage of the algorithm solves the majority of the probabilities, and is achieved using a simple loop. i^*, j^* are used to denote the outer loop of this stage. This is as some inside loops of the solution process require reference to them. Initialize by setting $j^* = 1$ and $i^* = c - 1$:

$$\pi_{0,j^*} = 1_{\{j^*>1\}}\left(-\rho_2\pi_{0,j^*-2} + \alpha_{2,2}^{-1}\pi_{0,j^*-1}\right) + 1_{\{j^*=1\}}\frac{\lambda}{\mu_2}\pi_{0,j^*-1} - \frac{\mu_1}{\mu_2}\pi_{1,j^*-1}. \tag{4.10}$$

Now for $i = i^*, i^* - 1, \ldots, 1$

$$\pi_{i,j^*} = \chi_{2,1}\pi_{i,j^*-1} + \chi_{1,1}\pi_{i-1,j^*} + 1_{\{i<i^*\}}\alpha_{1,1}\pi_{i+1,j^*} \tag{4.11}$$

where $\pi_{i,j^*-1}$is solved. Note that π_{0,j^*} is solved using the last equation (4.11). Simply substituting π_{1,j^*} into π_{2,j^*} through to π_{i^*,j^*} , provides the solution for π_{i^*,j^*}. Next, forward substitute to solve for all π_{i,j^*}, $i = i^* - 1, \ldots, 1$. Then return to (4.10), setting $j^* = j^* + 1$, and $i^* = i^* - 1$, unless $j^* = c, i^* = 1$, in which case this step ceases. To complete the iterative process,

$$\pi_{0,c} = \rho_2\pi_{0,c-1}. \tag{4.12}$$

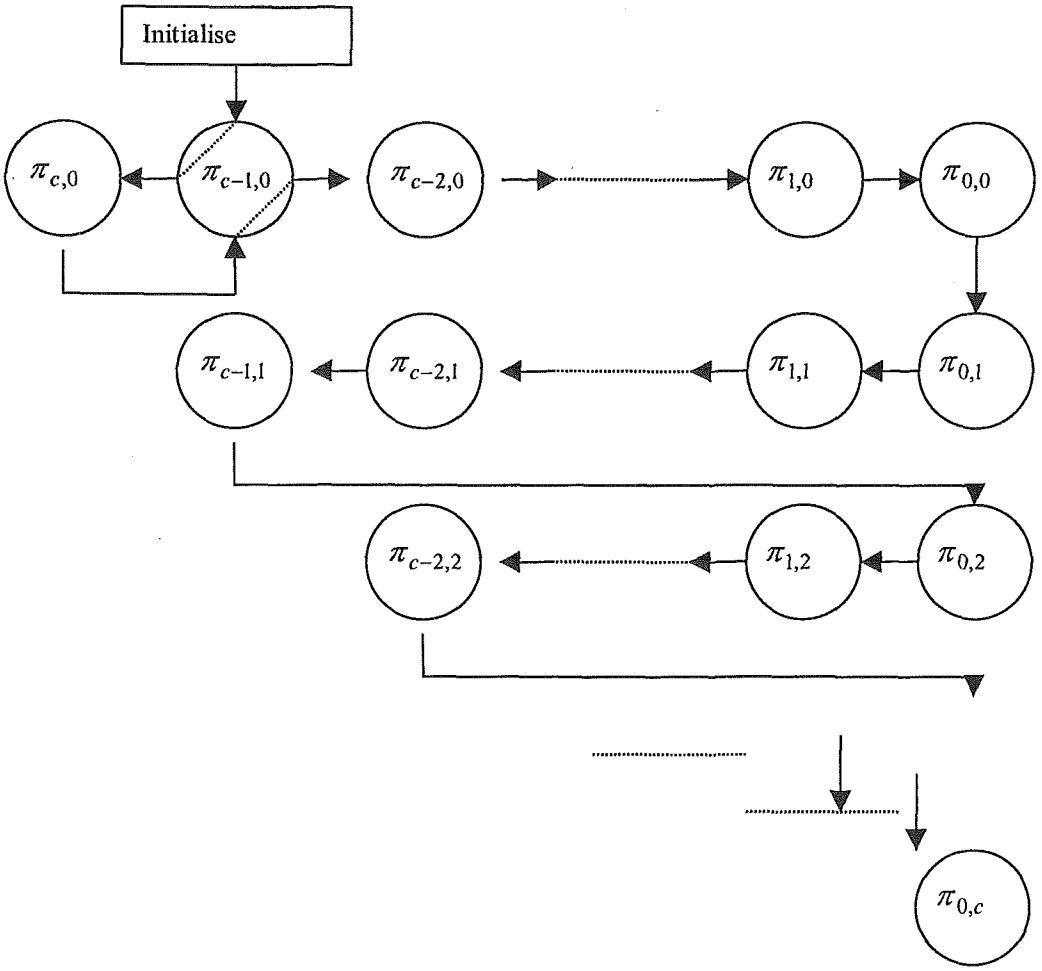

Figure 4.2: Order of solution

The final step of the algorithm involves normalizing the probabilities obtained using (4.6). The order of solution is detailed in Figure 4.2. The arrows indicate the order of solution with the algorithm beginning at $\pi_{c-1,0}$ and finishing at $\pi_{0,c}$.

From Figure 4.2 the Initialize box corresponds to equation (4.8), and the uppermost row corresponds to results obtained using equations (4.9). The start of every new row is detailed in equation (4.10), and across the row in equations (4.11). The last state is obtained in equation (4.12).

4.4.3 Waiting Time Analysis

In this section, an analysis of the waiting time for both classes of customers is given. The analysis is trivial for class 1 customers since their waiting time depends only on the number of same class customers ahead of the customer joining the queue. If

the queueing system contains a positive number of class 2 customers and no class 1 customers, then the customer in service will be ejected from service and their place taken by the new class 1 arrival. Thus, the distribution of the waiting time for an arriving class 1 customer is a weighted sum of Erlang distributions $E(n, \mu_1)$ with n phases and parameter μ_1, i.e. with corresponding density function

$$f(t; n, \mu_1) = \frac{\mu_1^n}{(n-1)!} t^{n-1} e^{-\mu_1 t}$$

when $t > 0$, and a discontinuity at $t = 0$ with a jump value equal to $\sum_{j=0}^{c-1} \pi_{0,j}$ in case of an empty system. To be specific, define

$$I_Q = \{(i, j) \in \mathcal{N}^2 : 0 \leq i + j < c\}$$

where $\mathcal{N}$ is the set of non-negative integers and let

$$\Pi_Q = \sum_{(i,j) \in I_Q} \pi_{i,j}.$$

Using the PASTA property (see Section 2.8), the waiting time density of a class 1 customer is given by

$$W_1(t) = \frac{1}{\Pi_Q} \left\{ \left(\sum_{j=0}^{c-1} \pi_{0,j} \right) \delta(t) + \sum_{j=0}^{c-2} \sum_{i=1}^{c-1-j} \pi_{i,j} f(t; i, \mu_1) \right\} \tag{4.13}$$

where $\delta(t)$ is the Dirac delta function.

For class 2 customers, the k^{th} moment, $k = 1, 2, \ldots$ of the waiting time will be derived through an argument based on first passage time of an appropriate stochastic process. To be specific, consider the waiting time of an arbitrary tagged class 2 customer C_2 from the time he joins the queue at $t = 0$ to the time he is first served. Note that C_2 only joins the queue if it isn't full. His waiting time will depend on the waiting times of customers (of both classes) ahead of him in the queue, and also of class 1 customers that subsequently join the queue (if it isn't full) while he is waiting to be served. Define the following stochastic process

$$\begin{aligned} Y(t) &= (N_1(t), L_2(t), F(t)) \\ \text{where} & \\ N_1(t) &= \text{number of class 1 customers in queue at time } t, \\ L_2(t) &= \text{number of class 2 customers in front and } \textit{including} \text{ tagged customer} \\ &\quad \text{in queue at time } t \\ \text{and} \quad F(t) &= \text{available waiting room in queue at time } t. \end{aligned}$$

Note that $Y(t)$ takes on values in the set

$$\mathcal{R} = \{(i,j,k) \in \mathcal{N}^3 : 0 \leq i \leq c, 0 \leq j \leq c, 0 \leq k \leq c\}.$$

In the sequel, t is suppressed in the above notation whenever it is not ambiguous to do so.

Next, define

$$C = L_2 + N_1 + F \tag{4.14}$$

and call this the *capacity* of the queue. This is introduced to account for the reduction in free spaces due to the arrival of class 2 customers after the arrival of C_2. This does not take place with queues which have an infinite waiting room since the amount of free spaces remains infinite with any number of arrivals into the system. Assume that the capacity of a queue will be reduced by one *only* in the case of an arrival of a class 2 customer into the system. An arrival of either class of customer will decrease F by 1, whereas service completed by any class of customer will increase F by 1. From (4.14), the capacity of a queue will be reduced by one *only* in case of an arrival of a class 2 customer into the system. No customer can enter the system when $F = 0$. We also assume that initially, just prior to the tagged customer joining the queue, $C = c$.

Define the set of states $\mathcal{A} \subset \mathcal{R}$ by

$$\mathcal{A} = \{(0,1,F) : 0 \leq F \leq c-1\}.$$

Let $T(n_1, l_2, f)$ be the first passage time for the process $Y(t)$ to be absorbed into $\mathcal{A}$ starting initially with $Y(0) = (n_1, l_2, f)$ where $l_2 \geq 1$. This is precisely the time it will take for the tagged customer to be served given that immediately prior to him joining the queue, he observes n_1 customers of class 1, $l_2 - 1$ customers of class 2 in the system and f is given by (4.14). Denote the Laplace transform of $T(n_1, l_2, f)$ by $\hat{W}_{n_1,l_2,f}(s)$, i.e.

$$\hat{W}_{n_1,l_2,f}(s) = E(e^{-sT(n_1,l_2,f)}) \tag{4.15}$$

where $Re(s) > 0$.

Using the PASTA property (see Section 2.8), the Laplace transform of the time to absorption of Y, the waiting time of a class 2 customer, given that he can join the queue with initial capacity c, is

$$\hat{W}_2(s) = \frac{1}{\Pi_Q} \left(\sum_{(i,j) \in I_Q} \pi_{i,j} \hat{W}_{i,j+1,c-i-j-1}(s) \right). \tag{4.16}$$

Relationships Between the Moments of Waiting Time for Class 2 Customers

In this section, the development of several relationships between the k^{th} moments of waiting times for class 2 customers for various combinations of N_1, L_2 and F is given. Firstly, consider the following relationships between Laplace transforms (4.15). Using a conditional argument based on the first transition of the process $Y(t)$ from an initial state to a state in $\mathcal{R}$ before absorption into $\mathcal{A}$ and the exponentially distributed time it takes to complete this first transition (cf. [Neuts, 1981], [Wagner, 1997]).

(i) $0 < L_2 \leq c$,

$$\hat{W}_{0,L_2,0}(s) = \frac{\mu_2}{s+\mu_2}\hat{W}_{0,L_2-1,1}(s); \tag{4.17}$$

(ii) $0 < N_1 < c, 0 < L_2 < c$,

$$\hat{W}_{N_1,L_2,0}(s) = \frac{\mu_1}{s+\mu_1}\hat{W}_{N_1-1,L_2,1}(s); \tag{4.18}$$

(iii) $0 < L_2 < c, 0 < F < c$,

$$\begin{aligned}\hat{W}_{0,L_2,F}(s) &= \frac{\mu_2}{s+\mu_2+\lambda}\hat{W}_{0,L_2-1,F+1}(s) + \frac{\lambda_1}{s+\mu_2+\lambda}\hat{W}_{1,L_2,F-1}(s) \\ &+ \frac{\lambda_2}{s+\mu_2+\lambda}\hat{W}_{0,L_2,F-1}(s);\end{aligned} \tag{4.19}$$

(iv) $0 < N_1 < c, 0 < L_2 < c, 0 < F < c$,

$$\begin{aligned}\hat{W}_{N_1,L_2,F}(s) &= \frac{\mu_1}{s+\mu_1+\lambda}\hat{W}_{N_1-1,L_2,F+1}(s) + \frac{\lambda_1}{s+\mu_1+\lambda}\hat{W}_{N_1+1,L_2,F-1}(s) \\ &+ \frac{\lambda_2}{s+\mu_1+\lambda}\hat{W}_{N_1,L_2,F-1}(s).\end{aligned} \tag{4.20}$$

The next set of relationships, between various kth moments of the waiting time $W^{(k)}_{N_1,L_2,F}$, are obtained from the above equations by differentiating the Laplace transforms k times and evaluating at $s = 0$ i.e.

$$W^{(k)}_{N_1,L_2,F} = (-1)^k \frac{d^k \hat{W}_{N_1,L_2,F}(s)}{ds^k}|_{s=0}. \tag{4.21}$$

(i) $1 < L_2 \leq c$,

$$\mu_2 W^{(k)}_{0,L_2,0} - \mu_2 W^{(k)}_{0,L_2-1,1} = kW^{(k-1)}_{0,L_2,0}; \tag{4.22}$$

(ii) $0 < N_1 < c, 0 < L_2 < c$,

$$\mu_1 W^{(k)}_{N_1,L_2,0} - \mu_1 W^{(k)}_{N_1-1,L_2,1} = kW^{(k-1)}_{N_1,L_2,0}; \tag{4.23}$$

(iii) $0 < L_2 < c, 0 < F < c$,

$$(\mu_2 + \lambda)W^{(k)}_{0,L_2,F} - \lambda_1 W^{(k)}_{1,L_2,F-1} = kW^{(k-1)}_{0,L_2,F} + \mu_2 W^{(k)}_{0,L_2-1,F+1} + \lambda_2 W^{(k)}_{0,L_2,F-1}; \tag{4.24}$$

(iv) $0 < N_1 < c, 0 < L_2 < c, 0 < F < c$,

$$(\mu_1 + \lambda)W^{(k)}_{N_1,L_2,F} - \lambda_1 W^{(k)}_{N_1+1,L_2,F-1} = kW^{(k-1)}_{N_1,L_2,F} + \mu_1 W^{(k)}_{N_1-1,L_2,F+1} + \lambda_2 W^{(k)}_{N_1,L_2,F-1}. \tag{4.25}$$

A Recursive Algorithm for Calculating kth Moment of Waiting Time for Class 2 Customers

For fixed $L_2 > 1$ and C, we define the following $C - L_2 + 1$ dimensional vector

$$W^{(k)}_{L_2,C} = \begin{pmatrix} W^{(k)}_{0,L_2,C-L_2} \\ W^{(k)}_{1,L_2,C-L_2-1} \\ \vdots \\ W^{(k)}_{C-L_2,L_2,0} \end{pmatrix}$$

and when $L_2 = 1$, let

$$W^{(k)}_{1,C} = \begin{pmatrix} W^{(k)}_{1,1,C-2} \\ W^{(k)}_{2,1,C-3} \\ \vdots \\ W^{(k)}_{C-1,1,0} \end{pmatrix}.$$

Next, obtain recursive relationships for $W^{(k)}_{L_2,C}$, $L_2 \geq 1$. It is necessary to distinguish between three possible cases: when $L_2 > 2, L_2 = 2$ and when $L_2 = 1$ respectively.

(a) $\mathbf{L_2 > 2}$:

Assume for the moment that $C > L_2$. Using (4.24) in case $N_1 = 0$, (4.25) in cases $N_1 = 1, 2, \ldots, C - L_2 - 1$ and (4.23) in case $N_1 = C - L_2$ (note that $F = C - L_2 - N_1$ from (4.14)), we obtain the following vector-matrix equation:

$$\mathbf{A}_{L_2,C} W^{(k)}_{L_2,C} = kW^{(k-1)}_{L_2,C} + \mathbf{\Gamma}_{2,L_2,C} W^{(k)}_{L_2,C-1} + \mu_2 e_1 (C - L_2 + 1) e_1^t (C - L_2 + 2) W^{(k)}_{L_2-1,C} \tag{4.26}$$

where

$$\mathbf{A}_{L_2,\mathcal{C}} = \begin{pmatrix} (\mu_2+\lambda) & -\lambda_1 & 0 & & \cdots & 0 \\ -\mu_1 & (\mu_1+\lambda) & -\lambda_1 & \ddots & & \\ 0 & -\mu_1 & \mu_1+\lambda & -\lambda_1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 & 0 \\ & & \ddots & -\mu_1 & (\mu_1+\lambda) & -\lambda_1 \\ 0 & & \cdots & & -\mu_1 & \mu_1 \end{pmatrix}$$

$$\in \Re^{(\mathcal{C}-L_2+1)\times(\mathcal{C}-L_2+1)}$$

and

$$\mathbf{\Gamma}_{L_2,\mathcal{C}} = \begin{pmatrix} \lambda_2 & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & 0 & \lambda_2 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \in \Re^{(\mathcal{C}-L_2+1)\times(\mathcal{C}-L_2)}.$$

Using (4.22), observe that the recursive equation (4.26) still holds when $\mathcal{C} = L_2$ provided we interpret $\mathbf{A}_{L_2,L_2} = (\mu_2)$ and make $\mathbf{\Gamma}_{L_2,L_2}$ void for that case.

(b) $L_2 = 2$:

Applying the same steps as in **(a)**, if $\mathcal{C} > 2$, the recursive relationship is

$$\mathbf{A}_{2,\mathcal{C}} W_{2,\mathcal{C}}^{(k)} = kW_{2,\mathcal{C}}^{(k-1)} + \mathbf{\Gamma}_{2,\mathcal{C}} W_{2,\mathcal{C}-1}^{(k)}. \tag{4.27}$$

If $\mathcal{C} = 2$, $W_{2,2}^{(k)} = (W_{0,2,0}^{(k)})$ and (4.22) gives

$$\mu_2 W_{0,2,0}^{(k)} = kW_{0,2,0}^{(k-1)} + \mu_2 W_{0,1,1}^{(k-1)} = kW_{0,2,0}^{(k-1)}. \tag{4.28}$$

(c) $L_2 = 1$:

Again, first assume that $\mathcal{C} > 2$. Applying (4.25) when $N_1 = 1, 2, \ldots, \mathcal{C}-2$ and (4.23) when $N_1 = \mathcal{C}_1 - 1$,

$$\mathbf{A}_{1,\mathcal{C}} W_{1,\mathcal{C}}^{(k)} = kW_{1,\mathcal{C}}^{(k-1)} + \mathbf{\Gamma}_{1,\mathcal{C}-1} W_{1,\mathcal{C}-1}^{(k)} \tag{4.29}$$

where

$$\mathbf{A}_{1,\mathcal{C}} = \begin{pmatrix} (\mu_1+\lambda) & -\lambda_1 & 0 & & \cdots & 0 \\ -\mu_1 & (\mu_1+\lambda) & -\lambda_1 & \ddots & & \\ 0 & -\mu_1 & \mu_1+\lambda & -\lambda_1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 & 0 \\ & & \ddots & -\mu_1 & (\mu_1+\lambda) & -\lambda_1 \\ 0 & & \cdots & & -\mu_1 & \mu_1 \end{pmatrix} \in \Re^{(\mathcal{C}-1)\times(\mathcal{C}-1)}.$$

If $C = 2, W_{1,2}^{(k)} = (W_{1,1,0}^{(k)})$ and (4.23) gives

$$\mu_1 W_{1,1,0}^{(k)} = kW_{1,1,0}^{(k-1)} + \mu_1 W_{0,1,1}^{(k-1)} = kW_{1,1,0}^{(k-1)}.$$

4.4.4 Numerical Examples

This section contains the results of numerical computations which are aimed at illustrating the algorithms obtained earlier. Consider the model with parameters $c = 5, \lambda_1 = 2, \lambda_2 = 3, \mu_1 = 5$ and $\mu_2 = 4$.

Steady State Probabilities

Initiate the algorithm by setting $\pi_{4,0} = 1$. Using this $\overline{\pi}_5 = (\pi_{5,0}) = (0.4)$.

Now solve for $\pi_{i,0}$,where $i = 0, \ldots, 3$:

$$\begin{aligned}
\pi_{3,0} &= \chi_{1,1}^{-1}\pi_{4,0} - \rho_1^{-1}\pi_{5,0} \\
\pi_{2,0} &= \chi_{1,1}^{-1}\pi_{3,0} - \rho_1^{-1}\pi_{4,0} \\
\pi_{1,0} &= \chi_{1,1}^{-1}\pi_{2,0} - \rho_1^{-1}\pi_{3,0} \\
\pi_{0,0} &= \chi_{1,1}^{-1}\pi_{1,0} - \rho_1^{-1}\pi_{2,0}
\end{aligned}$$

Hence $\pi_{3,0} = 4$, $\pi_{2,0} = 17.5$, $\pi_{1,0} = 77.5$ and $\pi_{0,0} = 343.75$ are obtained.

Now to the looping process which is initialized by setting $j^* = 1$ and $i^* = 4$:

$$\pi_{0,1} = \frac{\lambda}{\mu_2}\pi_{0,0} - \frac{\mu_1}{\mu_2}\pi_{1,0}$$

giving $\pi_{0,1} = 332.8125$. Now for $i = 4, \ldots, 1$

$$\begin{aligned}
\pi_{4,1} &= \chi_{2,1}\pi_{4,0} + \chi_{1,1}\pi_{3,1} \\
\pi_{3,1} &= \chi_{2,1}\pi_{3,0} + \chi_{1,1}\pi_{2,1} + \alpha_{1,1}\pi_{4,1} \\
\pi_{2,1} &= \chi_{2,1}\pi_{2,0} + \chi_{1,1}\pi_{1,1} + \alpha_{1,1}\pi_{3,1} \\
\pi_{1,1} &= \chi_{2,1}\pi_{1,0} + \chi_{1,1}\pi_{0,1} + \alpha_{1,1}\pi_{2,1}.
\end{aligned}$$

Through back-substitution $\pi_{1,1} = 105$, $\pi_{2,1} = 30.375$, $\pi_{3,1} = 8.25$ and $\pi_{4,1} = 1.95$. This completes one iteration of the algorithm. Using the same method as above the following solutions are obtained:

$j^* = 2, i^* = 3 : \pi_{0,2} = 359.7656250, \pi_{1,2} = 122.8535156, \pi_{2,2} = 38.80078125,$
$\pi_{3,2} = 10.23515625.$

$j^* = 3, i^* = 2 : \pi_{0,3} = 406.2963868, \pi_{1,3} = 137.7060547, \pi_{2,3} = 39.18144531$

$j^* = 4, i^* = 1 : \pi_{0,4} = 472.2100831, \pi_{1,4} = 135.7538330.$

Completing the iteration process $\pi_{0,5} = 354.1575623$.

$\pi_{i,j}$	j	0	1	2	3	4	5	*Total*
i	0	0.1146	0.1110	0.1199	0.1355	0.1574	0.1181	0.7565
	1	0.0258	0.0351	0.0410	0.0459	0.0453	–	0.1931
	2	0.0058	0.0101	0.0129	0.0131	–	–	0.0419
	3	0.0013	0.0028	0.0034	–	–	–	0.0075
	4	0.0003	0.0007	–	–	–	–	0.0010
	5	0.0001	–	–	–	–	–	0.0001
Total		0.1479	0.1597	0.1772	0.1945	0.2027	0.1181	

Table 4.1: Steady state probabilities

Finally, normalize the probabilities, giving the probabilities $\pi_{i,j}$ displayed in Table 4.1.

What is of interest from Table 4.1 is the probability of finding the absence of any class 1 customers in the system, which is given by $\sum \pi_{0,j} = 0.7565$. This is quite high when considering the closeness of the marginal probabilities for class 2 customers. The highest single probability is $\pi_{0,4}$. Also, for class 1 customers, $L_s^1 = \sum_{i=0}^{c} i\pi_{i\bullet} = 0.30$, and for class 2 customers, $L_s^2 = \sum_{j=0}^{c} j\pi_{\bullet j} = 2.4989$.

Waiting times

Now to obtain the expected waiting time of a class 2 customer. This is equal to $-\hat{W}_2^{(1)}(0)$ where $\hat{W}_2(s)$ is given by (4.15). Here $I_Q = \{(i,j) : 0 \leq i+j < 5\}$ and Table 4.1 gives $\Pi_Q = 0.8194$ (This is the sum of all probabilities where the system is not full). Note that the following vectors are required in the calculation of expected waiting time:

$$W_{1,5}^{(1)}, W_{2,5}^{(1)}, W_{3,5}^{(1)}, W_{4,5}^{(1)} \text{ and } W_{5,5}^{(1)}.$$

Apply (4.29) recursively, first to $W_{1,3}^{(1)}$ and then repeating this process twice to get

$$W_{1,5}^{(1)} = (0.2531, 0.4261, 0.6126, 0.8126)^t.$$

Next, apply (4.28) to get the obvious result that $W_{0,2,0}^{(1)} = \frac{1}{\mu_2}$ and then work recursively forward using (4.27) to get

$$W_{2,5}^{(1)} = (0.3647, 0.6280, 0.8596, 1.0596)^t.$$

For $W_{3,5}^{(1)}$ and $W_{4,5}^{(1)}$, use the values previously calculated and apply (4.26) recursively to obtain

$$W_{3,5}^{(1)} = (1.1254, 1.5296, 1.7296)^t$$
$$\text{and } W_{4,5}^{(1)} = (1.5372, 1.7372)^t.$$

Finally, (4.26) is applied (in the special case $L_2 = C$) to obtain

$$W_{5,5}^{(1)} = W_{0,5,0}^{(1)} = 1.7872.$$

Putting all these together, the expected waiting time of a class 2 customer is 1.0661 units of time.

From (4.13), it can easily be deduced that the expected waiting time of a class 1 customer is

$$E(W_1) = \frac{1}{\Pi_Q} \left\{ \sum_{j=0}^{c-2} \sum_{i=1}^{c-1-j} \frac{i\pi_{i,j}}{\mu_1} \right\}$$

which is 0.534 units of time for this example. Note that this is about 50% of a class 2 customer's mean waiting time.

Table 4.2 lists several performance measures for various arrival and service times, as well as queue capacities. The performance measures are:

(i) The utilization, or probability of finding a busy server

$$\pi_B = 1 - \pi_{0,0}. \tag{4.30}$$

(ii) The probability of finding only class one customer in the system

$$\pi^1 = \sum_{i=1}^{c} \pi_{i,0}. \tag{4.31}$$

(iii) The probability of finding only class two customers in the system

$$\pi^2 = \sum_{j=1}^{c} \pi_{0,j}. \tag{4.32}$$

(iv) The probability that a potential customer is lost to the system

$$\pi_{Loss} = \sum_{\{(i,j):i+j=c\}} \pi_{i,j}. \tag{4.33}$$

Note that from Table 4.2 as the queue size increases $\pi_B \to \rho$, where $\rho < 1$, and $\pi_B \to 1$, where $\rho > 1$. Further, $\pi_{Loss} \to 0$ as $c \to \infty$, which is to be expected when capacity increases for models where $\rho < 1$.

λ_1	λ_2	μ_1	μ_2	ρ	c	π_B	π^1	π^2	π_{Loss}
1	5	1	10	1.5	15	0.9265	0.0126	0.4878	0.0686
1	5	1	10	1.5	40	0.9920	0.0014	0.4936	0.0567
1	2	5	10	0.4	> 10	0.4	0.0950	0.2	0.0003
2	2	5	10	0.6	10	0.5925	0.1429	0.2	0.0035
2	2	5	10	0.6	40	0.6	0.1403	0.2	≈ 0
5	1	10	1	1.5	> 15	1	≈ 0	0.7682	0.3788
2	1	10	5	0.4	10	0.4	0.1302	0.2	≈ 0
2	2	10	5	0.6	10	0.5993	0.0772	0.4	0.0010
2	2	10	5	0.6	40	0.6	0.0770	0.4	≈ 0

Table 4.2: Performance Measures

Model	λ_1	λ_2	μ_1	μ_2	c	ρ	*error*
I	1	2	5	10	10	0.4	0
II	2	2	5	10	10	0.6	0.0003
	2	2	5	10	40	0.6	0
III	2	1	10	5	10	0.4	0
IV	2	2	10	5	10	0.6	0.0005
	2	2	10	5	40	0.6	0
V	3	2	5	10	10	0.8	0.0018
VI	3	1	4	4	10	1	0.0096

Table 4.3: Convergence error

4.4.5 Convergence

As all probabilities require normalising once the algorithm is complete, this, in some models, leads to small errors in convergence. Maple finalises the probabilities using the following commands:

$$\begin{aligned} G &= solve(ak = 1, k) \\ \text{where } a &= \sum_{\forall i,j \in S} \pi_{i,j} \end{aligned} \tag{4.34}$$

The *solve* command here obtains the value k such that all probabilities sum to unity. G is allocated this value (see [Bruell and Balbo, 1980] for a similar treatment) and all $\pi_{i,j}$ are then recalculated. Due to the division of large numbers, for some models the normalised probabilities do not precisely sum to unity. To analyse this briefly, a table is produced comparing the error for various sized systems.

From Table 4.3, for the models in which the traffic intensity approaches 1, the convergence error increases above 0.01%. Noticeable is that increased buffer size decreases the error. Where the traffic intensity is low, convergence presents no problem. In the high traffic intensity models where $\rho_2 > \rho_1$ the error is larger than when the reverse

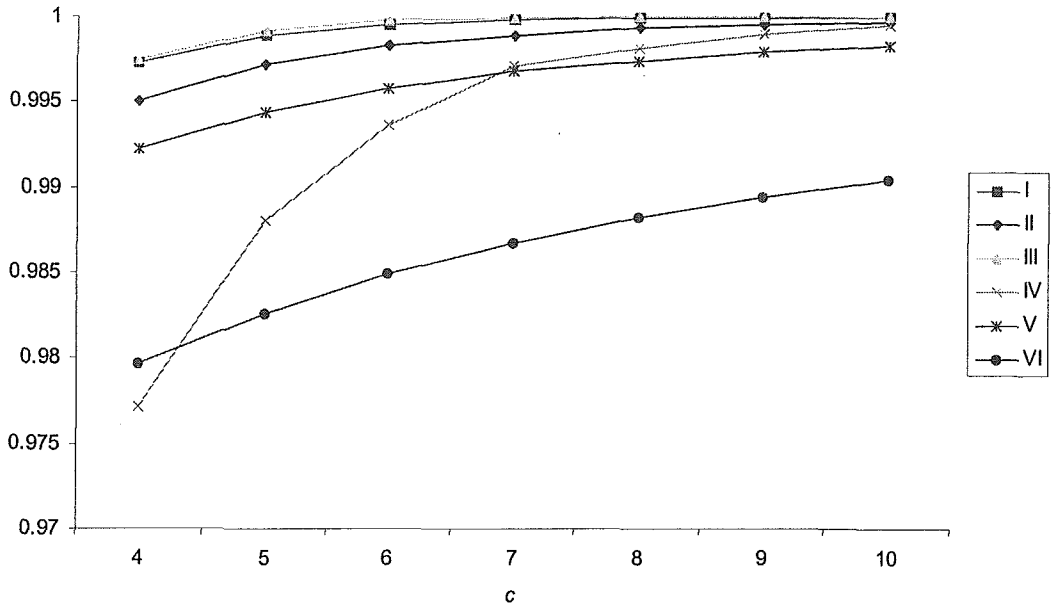

Figure 4.3: Convergence of a

holds. In Figure 4.3, convergence is illustrated for each model as a function of the capacity c. The error is largest in small capacity systems with high traffic intensity and the error decreases as c increases. Note the $y-axis$ is the sum of all probabilities, a as defined in (4.34) after recalculation. A Wilcoxon Signed Rank test on the Models indicated that we can accept the null hypothesis that $a = 1$ when $c > 5$ using a 5% error level.

Whilst this is of some concern in very small systems, the error levels diminish rapidly for systems of capacity over 10, except in the very high traffic intensity models. Practically this presents little problem as realistically models exceed this small size constraint.

4.5 Conclusion

In this section four types of single finite buffer models were introduced: the preemptive and non-preemptive single server two class infinite buffer models, and the two corresponding finite buffer models. A new approach to solving the steady-state probabilities and waiting time for the preemptive finite buffer model was given. The derivative of the probabilities is unique in that the solution algorithm is solved in one step based on the ordering of equations, and this was shown in a few simple iterations. Further

a brief look at convergence was given, highlighting problems in model with small capacities and high traffic intensity. A Maple algorithm is provided in the Appendix, and some performance characteristics were given. This model shall be revisited and used as a comparison with the next two models, the preemptive and non-preemptive MPDQ, covered in Chapters 5 and 6 respectively.

Part III

DUAL QUEUE MODELS

Chapter 5

Preemptive Multi-Priority Dual Queue

5.1 Introduction

In this chapter, the detailed analysis of the preemptive MPDQ begins with a look at the state space. Section 5.3 describes the states of the preemptive MPDQ system and its corresponding infinitesimal generator matrix. Next the system of vector-matrix linear equations which are required to obtain solution of the steady-state distribution are exhibited in Section 5.4. In Section 5.5 the computational algorithm to solve these systems of equations is shown. Section 5.6 begins the waiting time analysis by looking at Class 1 customers. Class 2 are dealt with in Section 5.7. The results are then illustrated through examples in Section 5.8. The work in this chapter leads from the previous. The techniques for solving the steady state probabilities and waiting time distribution used in chapter 4 are enhanced substantially to accommodate the large quantity of states in the preemptive MPDQ model. This in turn is extended later in chapter 6. Now a look at the MPDQ and other priority models.

5.2 Priority Models and the Preemptive MPDQ

The dual queue was designed to improve the performance characteristics for customers in communication networks. Specifically, it was designed not only to improve existing notions of fairness, but also to maximise the number of customers who receive good service. In [Hayes et. al., 1999] the new scheduling discipline called the dual queue was proposed and was shown to be superior in many criteria over existing FIFO and Fair Queueing schemes in a series of packet simulations. This work has recently been extended to include wireless local area networks (WLANs) [Ranasinghe et. al., 2001],

and with some minor adjustments, the dual queue scheme has again proven superior to existing schemes using simulated packet traffic. The model presented in this chapter is a MPDQ with a class-based priority scheme, rather than the time-out criteria used in the dual queue in [Hayes et. al., 1999] and [Ranasinghe et. al., 2001]. Much of the exhaustive analysis in the theoretical arena has been on modifications of the two basic finite queues or buffers. Length threshold [Jang et. al., 1997], cyclic servers [Hartanto et. al., 1995], synchronization queues [Takahashi et. al., 2000], Bernoulli schedules [Lee, 1997], Bernoulli schedules with switching times [Feng et. al., 1998] and dynamic server assignment [Boxma and Down, 1997] are some of the types of two-queue models analysed. None of these include a priority scheme. However, priority schemes have been used throughout many communications models [Morris, 1981], including wireless packet communications [Ogawa et. al., 2000]. A two-queue priority system application to a call centre was analysed by [Brandt and Brandt, 1999] where impatient customers were considered, and [Leemans, 2001] looked at a two class server model used in call centres. [Venkataramani et. al., 1997] looked at an infinite dual non-preemptive system via matrix analytic analysis.

The form of MPDQ analysed in this chapter involves two classes of customers arriving to the dual queueing system as two independent Poisson processes. The class of customer labelled 1 (the high class) has a mean arrival rate λ_1 and the other class, labelled 2 (the low class), has mean arrival rate λ_2. The dual queue is partitioned into the *primary queue* and *secondary queue*. There is a single server at the primary queue dispensing an exponentially distributed service time with rate μ_i to class i customers, $i = 1, 2$. Both queues have *finite* capacity. The primary queue employs a preemptive priority service discipline where class 1 customers have priority over class 2 customers. When an arriving customer finds the primary queue full, he waits in the secondary queue (provided it isn't full) which has no service facility and only serves as a holding queue for the overloaded primary queue. If both queues are full, an arriving customer is lost to the system. In the secondary queue, customers arrange themselves in order of priority, with class 1 customers ahead of class 2 customers. Once a space is vacant in the primary queue, the head of the line in the secondary queue joins the primary queue. A schematic diagram describing the MPDQ was detailed in Section 3.2.2.

The remarkable ability to solve complicated queueing systems using matrix-analytic methods has been explored in detail in [Neuts, 1981], [Neuts, 1989] and [Latouche and Ramaswami, 1999]. Also, many queueing systems are solved using computational recursive algorithms [Bruell and Balbo, 1980]. In this chapter, a combination of the two is used to provide a novel approach for obtaining an exact solution to the dual queue discussed above. Many priority based models have simply solved steady-state probabilities using recursive schemes based on product-form solutions

[Cidon and Sidi, 1990], [Cidon et. al., 1993]. In [Balbo et. al., 1985], the authors use a form of hierarchical decomposition to solve general priority models. This is not the case here. In Hayes' thesis [Hayes, 2001] he discusses the difficulties in solving the dual queue analytically. He uses an approximation of the dual queue model - a simple overflow model using global balance equations (pp. 135-137). The primary queue (referred to as the α queue) holds a finite quantity of packets and the secondary queue (referred to as the β queue) is an overflow without size constraint. Some operational constraints were relaxed to model the queue, and the α queue had the same distribution as the $(M/M/1) : (K/\infty/\infty)$. Then, matrix-analytic techniques were used to address shortcomings of his earlier model, and again the analysis provided an estimation of the actual model.

If the infinitesimal generator can be presented in a block tridiagonal form, then this will, in general, lead to an explicit solution of the steady-state distribution through simple algorithms. While based on this idea, the steady-state distribution obtained here is not derived in the conventional way. This is due to the fact that the states described for the MPDQ do not render the infinitesimal generator into a workable block tridiagonal form. Also, not all submatrices are invertible, specifically the ones representing transitions between the primary and secondary queues.

A computational algorithm will be devised which will circumvent these difficulties and produce an approach to solving dual queues of this type in a systematic way. The solution process for the steady state probabilities is based on the order of the equations. Via direct substitution, the system is solved uniquely in a single iteration in a similar vein to that undertaken in Chapter 4.

Next the expected waiting time for both classes of customers in the primary and secondary queue is derived. This will be accomplished by observing what an arbitrary tagged customer of a particular class experiences when he joins the queue. The exact distribution of the waiting time of a class 1 customer will be derived and an algorithm for computing the k^{th} moment, $k = 1, 2, \ldots$ of waiting time of a class 2 customer will be obtained using transform techniques.

The objectives of this Chapter are to:

- provide the new algorithmic process for solving the steady-state probabilities based on global balance and equation order
- demonstrate a new approach to obtaining the expected waiting times
- investigate some performance statistics
- provide a Maple program to evaluate the MPDQ steady state probabilities

5.3 State Space

Let c_i = capacity of queue $i, i = 1, 2$. The homogeneous ergodic Markov process describing the MPDQ takes values in a state space S. S can be partitioned in two distinct ways, corresponding to the case where the secondary queue is empty and when it is not empty respectively. In the first case, define $S_1 = \{(i,j) : 0 \leq i+j \leq c_1\}$ where the first number of the pair i = number of customers of class 1 and j = number of customers of class 2. In the second case, define $S_2 = \{(i,i',j') : 0 \leq i'+j' \leq c_2, i = 0,1,\ldots,c_1\}$ where i' = number of customers of class 1 and j' = the number customers of class 2 in the secondary queue. As the primary queue is full, there is no need to state the number of class 2 customers as this is simply given by $c_1 - i$. The state space is therefore given by $S = S_1 \cup S_2$.

The steady-state distribution will be represented as a vector dependent upon the size of each of the two queues. The labelling of the members of the vector is done according to the following lexicographical ordering:

$$\mathbf{i} = \{(i,0),(i,1),\ldots,(i,c_1-i)\}$$
$$\mathbf{i0} = \{(i,0,1)(i,0,2),\ldots,(i,0,c_2)\}$$

and for $\mathbf{j} = 1,2,\ldots,\mathbf{c_2}$

$$\mathbf{ij} = \{(i,j,0),(i,j,1),\ldots,(i,j,c_2-j)\}$$

$i = 0,1,2,\ldots,c_1$.

The vector is constructed so that its components are ordered using the above labelling scheme, which gives us

$$\overline{\pi}^t = \left(\overline{\pi}^t_{\mathbf{0}}, \overline{\pi}^t_{\mathbf{0,0}}, \ldots, \overline{\pi}^t_{\mathbf{0,c_2}}, \overline{\pi}^t_{\mathbf{1}}, \overline{\pi}^t_{\mathbf{1,0}}, \ldots, \overline{\pi}^t_{\mathbf{1,c_2}}, \ldots, \overline{\pi}^t_{\mathbf{c_1}}, \overline{\pi}^t_{\mathbf{c_1,0}}, \ldots, \overline{\pi}^t_{\mathbf{c_1,c_2}}\right)$$

where $\overline{\pi}^t \in \Re^{\frac{1}{2}(c_1+1)\left(c_2^2+3c_2+c_1+2\right)}$.

The components of the above steady-state distribution can be described as follows: $\overline{\pi}_{\mathbf{i}}$ is the probability vector that is defined when there are no customers present in the secondary queue, whereas $\overline{\pi}_{\mathbf{ij}}$ is the probability vector that is defined when there are customers present in the secondary queue (so the primary queue is full). Any probability that is a component of the first type of vector has general form $\pi_{i,j}$ and any probability that is a component of the second type of vector has general form $\pi_{i,i',j'}$. Thus, for the primary queue

$$\overline{\pi}_i = (\pi_{i,0}, \pi_{i,1}, \ldots \pi_{i,c_1-i})^t \in \Re^{c_1-i+1}, i \in \{0,\ldots,c_1\}$$

and, for the secondary queue when $j = 0$,

$$\overline{\pi}_{i,0} = (\pi_{i,0,1}, \pi_{i,0,2}, \ldots, \pi_{i,0,c_2})^t \in \Re^{c_2}, i \in \{0,\ldots,c_1\}$$

and when $j > 0$

$$\overline{\pi}_{i,j} = (\pi_{i,j,0}, \pi_{i,j,1}, \ldots \pi_{i,j,c_2-j})^t \in \Re^{c_2-j+1}, i \in \{0, \ldots, c_1\}, j \in \{1, \ldots, c_2\}.$$

5.3.1 Infinitesimal Generator Matrix A

The steady-state distribution $\overline{\pi}^t$ exists and is obtained by solving the system of equations

$$\overline{\pi}^t \mathbf{A} = \mathbf{0} \tag{5.1}$$

where $\mathbf{0}$ is the vector of zeroes of an appropriate dimension and $\mathbf{A}$ is the infinitesimal generator matrix of the process. From (5.1) $\mathbf{A}$ is a square matrix of dimension $\frac{1}{2}(c_1+1)(c_2^2+3c_2+c_1+2)$ which becomes quite large even for a system with small capacity. For example a primary queue of capacity 4 and secondary of capacity 6 will generate a matrix $\mathbf{A}$ of dimension 150. From the description of the dual queueing system given, $\mathbf{A}$ can be partitioned into submatrices whose detailed structure will be described in detail below. The general structure of $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{pmatrix} \Lambda_0 & \Pi_0 & 0 & \ldots & \ldots & 0 \\ \Omega_1 & \Lambda_1 & \Pi_1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \Omega_{c_1-1} & \Lambda_{c_1-1} & \Pi_{c_1-1} \\ 0 & \ldots & & 0 & \Omega_{c_1} & \Lambda_{c_1} \end{pmatrix} \tag{5.2}$$

where

$$\Lambda_0 = \begin{pmatrix} \mathbf{D}_0 & \mathbf{M}_{0,\lambda_2} & \mathbf{M}_{0,\lambda_1} & 0 & & 0 \\ \mathbf{O}_{0,\mu_2} & \mathbf{D}_{0,0} & \mathbf{Q}_{0,\lambda_1} & 0 & & \\ 0 & 0 & \mathbf{D}_{0,1} & \mathbf{R}_{0,1,\lambda_1} & & \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \mathbf{R}_{0,c_2-1,\lambda_1} \\ 0 & & \cdots & & 0 & \mathbf{D}_{0,c_2} \end{pmatrix} \tag{5.3}$$

$$\Pi_0 = \begin{pmatrix} \mathbf{N}_{0,\lambda_1} & 0 & & \cdots & & 0 \\ 0 & 0 & & & & \\ \mathbf{S}_{0,\mu_2} & \mathbf{U}_{0,1} & \ddots & & & \vdots \\ 0 & 0 & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \cdots & & 0 & \mathbf{U}_{0,c_2} & 0 \end{pmatrix} \tag{5.4}$$

and for $i = 1, 2, \ldots, c_1$

$$\Lambda_i = \begin{pmatrix} \mathbf{D}_i & \mathbf{M}_{i,\lambda_2} & \mathbf{M}_{i,\lambda_1} & 0 & & \cdots & 0 \\ 0 & \mathbf{D}_{i,0} & \mathbf{Q}_{i,\lambda_1} & 0 & & & \\ \mathbf{S}_{i,\mu_1} & \mathbf{U}_{i,1} & \mathbf{D}_{i,1} & \mathbf{R}_{i,1,\lambda_1} & \ddots & & \vdots \\ 0 & 0 & \mathbf{U}_{i,2} & \mathbf{D}_{i,1} & \ddots & \ddots & \\ \vdots & & \ddots & \mathbf{U}_{i,2} & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \ddots & \mathbf{R}_{i,c_2-1,\lambda_1} \\ 0 & & \cdots & & 0 & \mathbf{U}_{i,c_2} & \mathbf{D}_{i,c_2} \end{pmatrix} \tag{5.5}$$

$$\Omega_i = \begin{pmatrix} \mathbf{P}_{i,\mu_1} & 0 & \cdots & 0 \\ \mathbf{Y}_{i,\mu_1} & \mathbf{T}_{i,\mu_1} & & \\ 0 & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ & & & \\ 0 & & \cdots & 0 \end{pmatrix} \tag{5.6}$$

and

$$\Pi_i = \begin{pmatrix} \mathbf{N}_{i,\lambda_1} & 0 & \cdots & 0 \\ 0 & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & 0 \end{pmatrix} \tag{5.7}$$

All blank areas in the above submatrices are zeroes. Note that as the dual queues may have a combination of mixed odd and even capacities, some of the submatrices are not square matrices and hence non-invertible. Also, some of the submatrices contain only one non-zero element while others are in the familiar tridiagonal form.

Incorporating all the state transitions generated by the arrival and departure patterns together with the structures of the MPDQ system, the above submatrices of A will be described in full detail, with the entries of each submatrix containing the appropriate rate parameters λ_i and $\mu_i, i = 1, 2$.

State Transitions and Detailed Structures of Submatrices

The following detailed descriptions of the submatrices of A are obtained from the assumptions of the MPDQ and they mirror the arrivals to and departures from and

between the primary and secondary queues. To begin with, the **D** matrices will be defined. These are all in tridiagonal form and they are square matrices:

$$\mathbf{D}_0 = \begin{pmatrix} -\lambda & \lambda_2 & 0 & & \cdots & 0 \\ \mu_2 & -(\mu_2+\lambda) & \ddots & \ddots & & \vdots \\ 0 & \mu_2 & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \lambda_2 & 0 \\ & & & \ddots & -(\mu_2+\lambda) & \lambda_2 \\ 0 & \cdots & & 0 & \mu_2 & -(\mu_2+\lambda) \end{pmatrix} \tag{5.8}$$

where $\mathbf{D}_0 \in \Re^{(c_1+1)\times(c_1+1)}$

$$\mathbf{D}_i = \begin{pmatrix} -(\mu_1+\lambda) & \lambda_2 & 0 & & \cdots & 0 \\ 0 & -(\mu_1+\lambda) & \ddots & & & \\ & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \lambda_2 & 0 \\ & & & \ddots & -(\mu_1+\lambda) & \lambda_2 \\ 0 & & \cdots & & 0 & -(\mu_1+\lambda) \end{pmatrix} \tag{5.9}$$

where $\mathbf{D}_i \in \Re^{(c_1-i+1)\times(c_1-i+1)}, i = 1, 2, \ldots, c_1$.

$$\mathbf{D}_{0,0} = \begin{pmatrix} -(\mu_2+\lambda) & \lambda_2 & 0 & & \cdots & 0 \\ \mu_2 & -(\mu_2+\lambda) & \ddots & & & \\ & \mu_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \lambda_2 & 0 \\ & & & \ddots & -(\mu_2+\lambda) & \lambda_2 \\ 0 & & \cdots & & \mu_2 & -\mu_2 \end{pmatrix} \tag{5.10}$$

where $\mathbf{D}_{0,0} \in \Re^{(c_2)\times(c_2)}$.

$$\mathbf{D}_{i,0} = \begin{pmatrix} -(\mu_1+\lambda) & \lambda_2 & 0 & & \cdots & 0 \\ 0 & -(\mu_1+\lambda) & \ddots & & & \\ & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \lambda_2 & 0 \\ & & & \ddots & -(\mu_1+\lambda) & \lambda_2 \\ 0 & & \cdots & & 0 & -\mu_1 \end{pmatrix} \tag{5.11}$$

where $\mathbf{D}_{i,0} \in \Re^{(c_2)\times(c_2)}$, $i = 1, 2, \ldots, c_1$ and $\mathbf{D}_{c_1,0} = (-\mu_1)$

$$\mathbf{D}_{0,j} = \begin{pmatrix} -(\mu_2+\lambda) & \lambda_2 & 0 & & \cdots & 0 \\ 0 & -(\mu_2+\lambda) & \ddots & & & \\ & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \lambda_2 & 0 \\ & & & \ddots & -(\mu_2+\lambda) & \lambda_2 \\ 0 & & \cdots & & 0 & -\mu_2 \end{pmatrix} \tag{5.12}$$

where $\mathbf{D}_{0,j} \in \Re^{(c_2-j+1)\times(c_2-j+1)}$, $j = 1, 2, \ldots, c_2$ and $\mathbf{D}_{0,c_2} = (-\mu_2)$

$$\mathbf{D}_{i,j} = \begin{pmatrix} -(\mu_1+\lambda) & \lambda_2 & 0 & & \cdots & 0 \\ 0 & -(\mu_1+\lambda) & \ddots & & & \\ & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \lambda_2 & 0 \\ & & & \ddots & -(\mu_1+\lambda) & \lambda_2 \\ 0 & & \cdots & & 0 & -\mu_1 \end{pmatrix} \tag{5.13}$$

where $\mathbf{D}_{i,j} \in \Re^{(c_2-j+1)\times(c_2-j+1)}$, $i = 1, 2, \ldots, c_1, j = 1, 2, \ldots, c_2$ and $\mathbf{D}_{c_1,c_2} = (-\mu_1)$. It is easy to check that the matrix $\mathbf{D}_i$ is the infinitesimal generator corresponding to transitions between states in **i** whereas $\mathbf{D}_{i,j}$ is the infinitesimal generator corresponding to transitions between states in **ij**.

Next consider all the off-diagonal submatrices. In the following, each of these submatrices is given. The transitions and corresponding instantaneous rates giving rise to the submatrix are summarised in the accompanying tables:

$$\mathbf{M}_{i,\lambda_k} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ \lambda_k & 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c_1-i+1)\times(c_2)} \tag{5.14}$$

From	To	Rate
$(0, c_1)$	$(0, 1, 0)$	λ_1
$(0, c_1)$	$(0, 0, 1)$	λ_2
$(i, c_1 - i)$	$(i, 1, 0)$	λ_1
$(i, c_1 - i)$	$(i, 0, 1)$	λ_2

where $i = 1, 2, \ldots, c_1$ and $k = 1, 2$;

$$\mathbf{N}_{i,\lambda_1} = \begin{pmatrix} \lambda_1 & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & 0 & \lambda_1 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \in \Re^{(c_1-i+1)\times(c_1-i)} \tag{5.15}$$

From	To	Rate
(i,j)	$(i+1,j)$	λ_1

where $i = 0, 1, \ldots, c_1 - 1, k = 1, 2$;

$$\mathbf{P}_{i,\mu_1} = \begin{pmatrix} \mu_1 & 0 & & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \mu_1 & \ddots & \\ 0 & \cdots & 0 & \mu_1 & 0 \end{pmatrix} \in \Re^{(c_1-i+1)\times(c_1-i+2)} \tag{5.16}$$

From	To	Rate
(i,j)	$(i-1,j)$	μ_1

where $i = 1, \ldots, c_1, j = 0, \ldots, c_1 - i$;

$$\mathbf{Y}_{i,\mu_1} = \begin{pmatrix} 0 & 0 & \mu_1 \\ \vdots & \ddots & 0 \\ 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c_2)\times(c_1-i+2)} \tag{5.17}$$

From	To	Rate
$(i,0,1)$	$(i-1,c_1-i+1)$	μ_1

where $i = 1, \ldots, c_1$;

$$\mathbf{T}_{i,\mu_1} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \mu_1 & \ddots & & \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mu_1 & 0 \end{pmatrix} \in \Re^{(c_2)\times(c_2)} \tag{5.18}$$

From	To	Rate
$(i,0,1)$	$(i-1,0,j-1)$	μ_1

where $i = 1,\ldots,c_1, j = 2,\ldots,c_2$;

$$\mathbf{O}_{0,\mu_2} = \begin{pmatrix} 0 & \cdots & 0 & \mu_2 \\ \vdots & \ddots & \vdots & 0 \\ 0 & \cdots & 0 & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} \in \Re^{(c_2)\times(c_1+1)} \tag{5.19}$$

From	To	Rate
$(0,0,1)$	$(0,c_1)$	μ_2

$$\mathbf{Q}_i = \begin{pmatrix} 0 & \lambda_1 & 0 & 0 \\ \vdots & \ddots & \ddots & 0 \\ & & & \lambda_1 \\ 0 & \cdots & 0 & 0 \end{pmatrix} \in \Re^{(c_2)\times(c_2)}, \text{where } \mathbf{Q}_0 = \begin{pmatrix} 0 & \lambda_1 \\ 0 & 0 \end{pmatrix}$$

$$\text{for } c_2 = 2 \text{and } \mathbf{Q}_0 = \begin{pmatrix} 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_1 \\ 0 & 0 & 0 \end{pmatrix} \text{for } c_2 = 3 \tag{5.20}$$

From	To	Rate
$(i,0,j)$	$(i,1,j)$	λ_1

where $i = 0,\ldots,c_1$, $j = 0,\ldots,c_2 - 1$.

Similarly,

$$\mathbf{R}_{i,j,\lambda_1} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ \vdots & \ddots & \lambda_1 \\ 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c_2-j+1)\times(c_2-j)}, \text{where } \mathbf{R}_{1,j} = \begin{pmatrix} \lambda_1 \\ 0 \end{pmatrix},$$

$$\text{for } c_2 = 2, \text{and } \mathbf{R}_{1,j} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_1 \\ 0 & 0 \end{pmatrix} \text{ for } c_2 = 3 \tag{5.21}$$

From	To	Rate
(i,j,k)	$(i,j+1,k)$	λ_1

where $i = 0,\ldots,c_1$, $j = 1,\ldots,c_2-1, k = 0,\ldots,c_2-j-1$;

$$\mathbf{S}_{i,\mu_c} = \begin{pmatrix} 0 & \cdots & 0 & \mu_c \\ \vdots & \ddots & \vdots & 0 \\ 0 & \cdots & 0 & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} \in \Re^{(c_2)\times(\min\{c_1,c_1-i+1\})} \tag{5.22}$$

From	To	Rate
$(0,1,0)$	$(1,c_1)$	μ_2
$(i,1,0)$	(i,c_1-i)	μ_1

where $i = 0,\ldots,c_1, c = 1,2$.

Finally,

$$\mathbf{U}_{i,j,\mu_c} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \mu_c & \ddots & & \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mu_c & 0 \end{pmatrix} \in \Re^{(c_2-j+1)\times(\min\{c_2,c_2-j+2\})}, \tag{5.23}$$

$$\text{where } \mathbf{U}_{0,1,\mu_c} = \begin{pmatrix} 0 & 0 \\ \mu_c & 0 \end{pmatrix} \text{ for } c_2 = 2,$$

From	To	Rate
$(0, j, k)$	$(1, j-1, k)$	μ_2
(i, j, k)	$(i, j-1, k)$	μ_1

where $i = 0, \ldots, c_1, j = 1, \ldots, c_2, k = 0, \ldots, c_2 - j, c = 1, 2$.

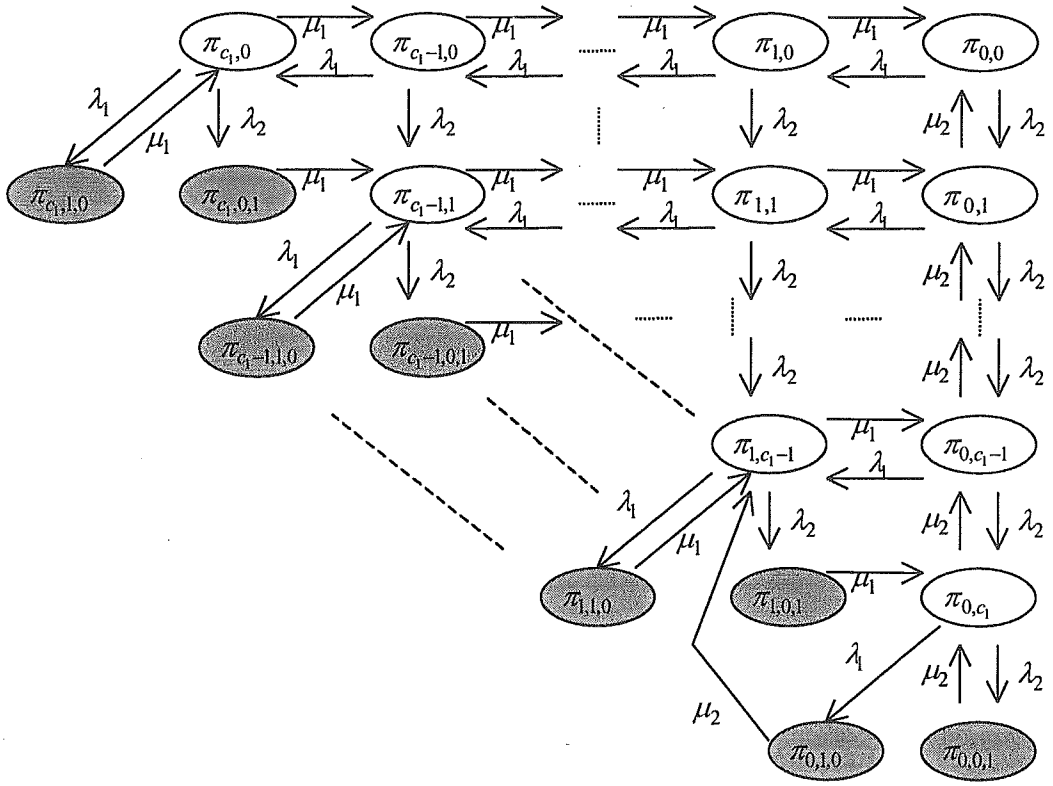

Figure 5.1: State rate transition diagram for the primary queue in the MPDQ

State Rate Diagrams

These diagrams illustrate the complexities of transitions in the MPDQ. Again, the focus is restricted to two classes of customers. Four separate figures are used. The shaded states show transitions in and/or out of this Figure to another one. Figure 5.1 illustrates the transitions to and from the primary queue. The shaded states show transitions in and/or out of this Figure to another one. Figure 5.2 illustrates the transitions to and from the secondary queue for $i = 0, 1$ in $\pi_{i,i',j'}$, whereas Figure 5.3 illustrates the MPDQ for $1 < i < c_1$ and Figure 5.4 illustrates the MPDQ when $i = c_1$.

5.4 Linear System of Steady State Distributions

To establish the general technique used later in the computational algorithm, first establish the system of linear equations using the generic matrices Λ_i, Π_i and Ω_i. Let

$$\alpha_i^t = \left(\overline{\pi}_{\mathbf{i}}^t, \overline{\pi}_{\mathbf{i},0}^t, \overline{\pi}_{\mathbf{i},1}^t, \ldots, \overline{\pi}_{\mathbf{i},\mathbf{c_2}}^t\right) \tag{5.24}$$

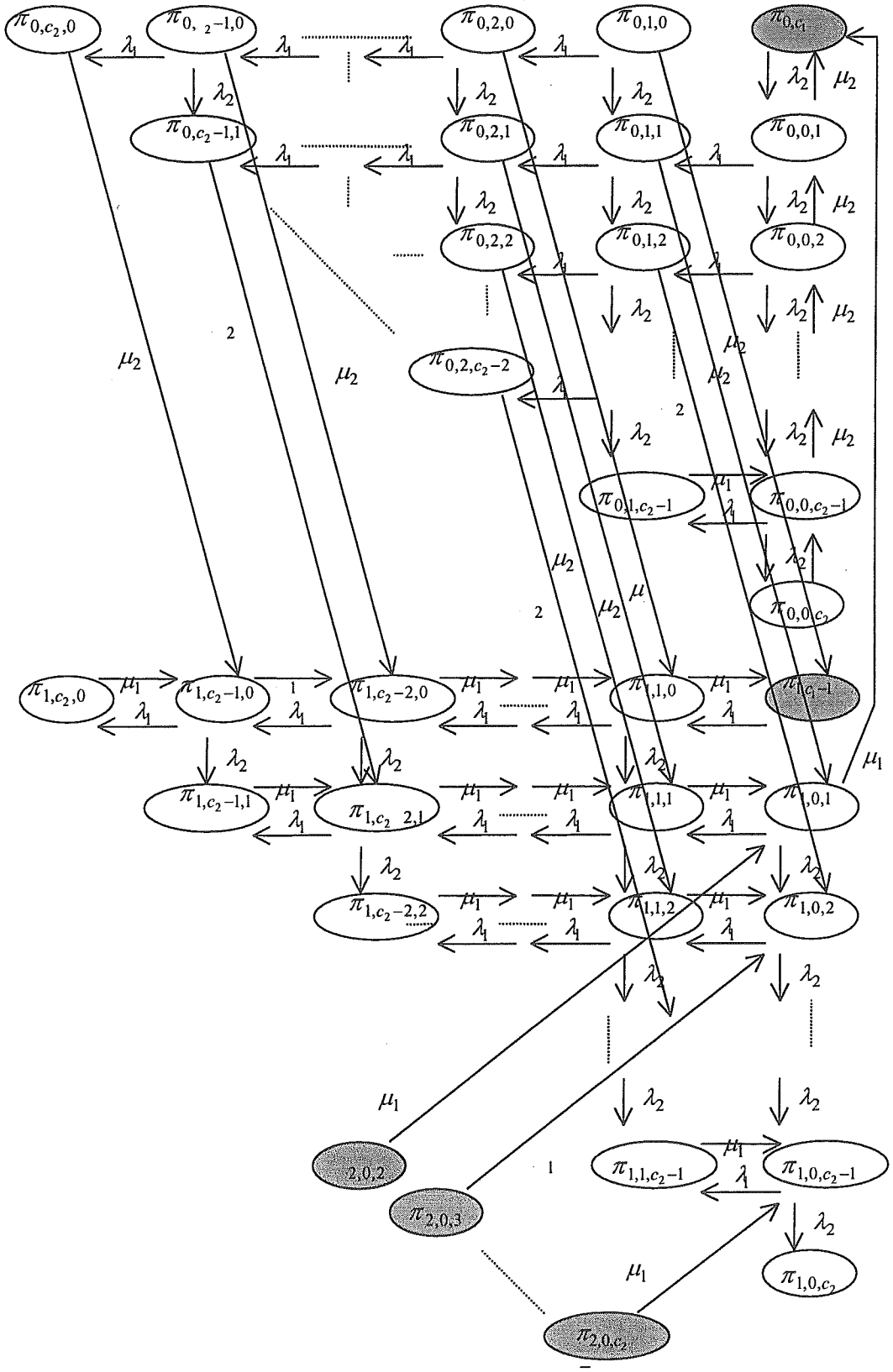

Figure 5.2: Transition diagram for the secondary queue when $i = 0, 1$ in $\pi_{i,i',i'}$

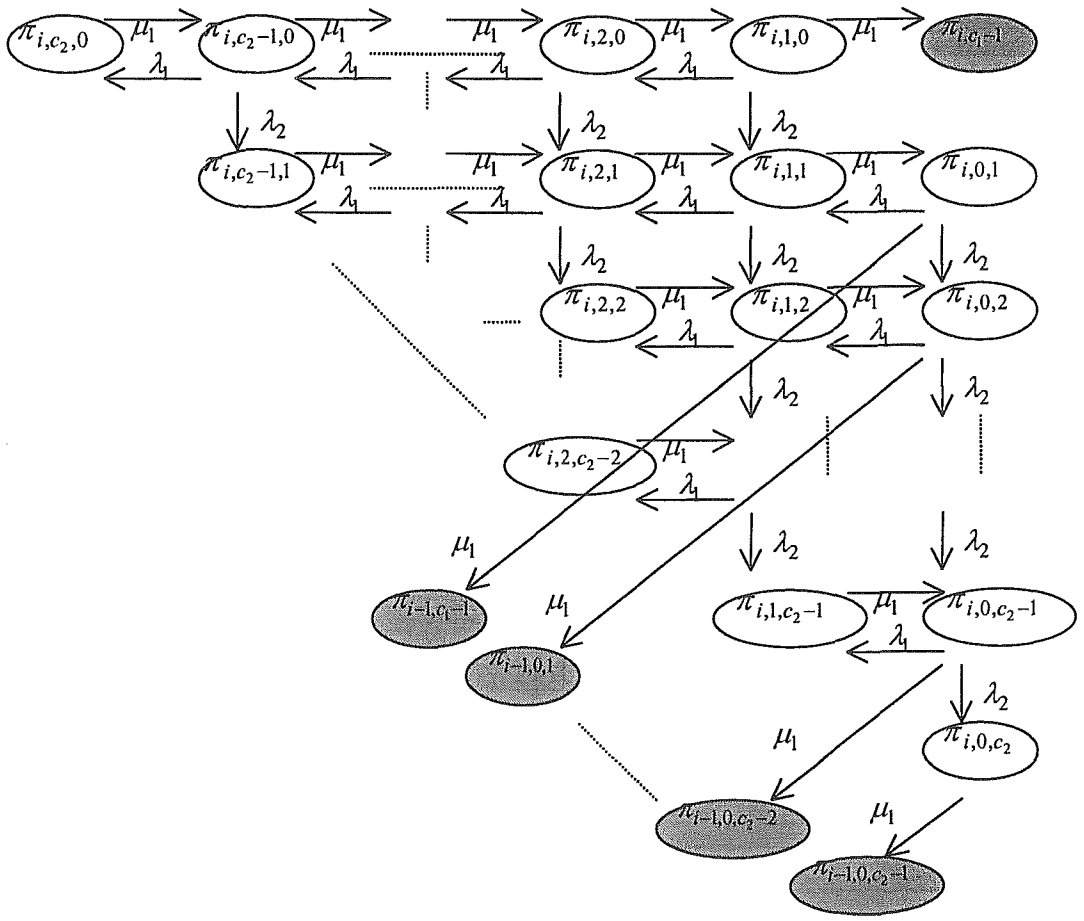

Figure 5.3: State rate transition diagram for the secondary queue when $1 < i < c_1$ in $\pi_{i,i'j'}$ for the MPDQ

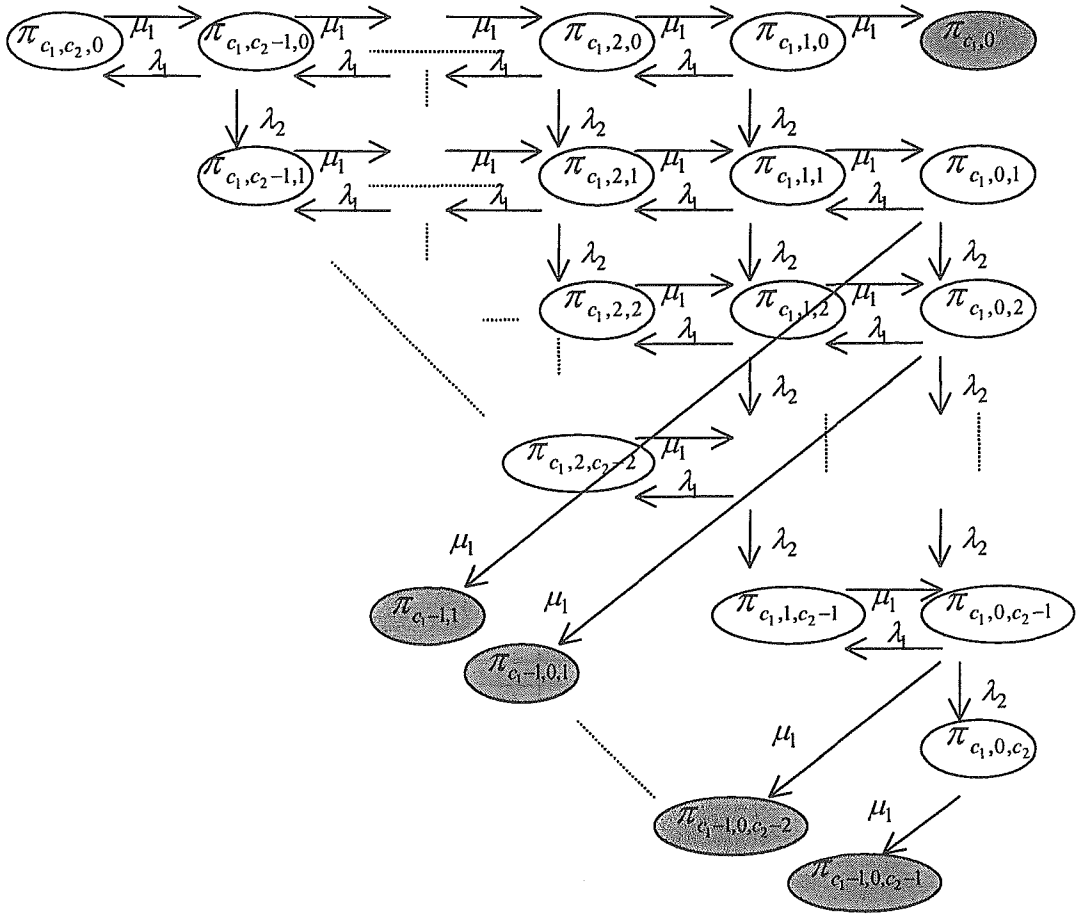

Figure 5.4: State rate transition diagram for the secondary queue when $i = c_1$ in $\pi_{i,i'j'}$ for the MPDQ

where $i = 0, \ldots, c_1$. Therefore $\overline{\pi}^t = \left(\boldsymbol{\alpha}_0^t, \boldsymbol{\alpha}_1^t, \ldots, \boldsymbol{\alpha}_{c_1}^t\right)$ and the system (5.1) can be described by

$$\left(\boldsymbol{\alpha}_0^t, \boldsymbol{\alpha}_1^t, \ldots, \boldsymbol{\alpha}_{c_1}^t\right) \mathbf{A} = \mathbf{0}. \tag{5.25}$$

Expanding the above, the following system of linear equations is derived:

$$1_{\{i \geq 1\}} \left(\boldsymbol{\alpha}_{i-1}^t \boldsymbol{\Pi}_{i-1}\right) + \boldsymbol{\alpha}_i^t \boldsymbol{\Lambda}_i + 1_{\{i < c_1\}} \left(\boldsymbol{\alpha}_{i+1}^t \boldsymbol{\Omega}_{i+1}\right) = \mathbf{0} \tag{5.26}$$

where $i = 0, \ldots, c_1$.

From (5.26) there are $c_1 + 1$ equations in vector-matrix form. Using the detailed structures of the matrices $\boldsymbol{\Pi}_i, \boldsymbol{\Gamma}_i$ and $\boldsymbol{\Omega}_i$ displayed in Section 5.3, the system can be further expanded into several equations in matrix-vector form. For example, for $i = 0$ in (5.26) gives the following $c_2 + 2$ vector-matrix equations involving submatrices of $\boldsymbol{\Gamma}_0$ and $\boldsymbol{\Omega}_1$:

$$\begin{aligned}
\overline{\pi}_0^t \mathbf{D}_0 + \overline{\pi}_{0,0}^t \mathbf{O}_{0,\mu_2} + \overline{\pi}_1^t \mathbf{P}_{1,\mu_1} + \overline{\pi}_{1,0}^t \mathbf{Y}_{1,\mu_1} &= \mathbf{0} \\
\overline{\pi}_0^t \mathbf{M}_{0,\lambda_2} + \overline{\pi}_{0,0}^t \mathbf{D}_{0,0} + \overline{\pi}_{1,0}^t \mathbf{T}_{1,\mu_1} &= \mathbf{0} \\
\overline{\pi}_0^t \mathbf{M}_{0,\lambda_1} + \overline{\pi}_{0,0}^t \mathbf{Q}_{0,\lambda_1} + \overline{\pi}_{0,1}^t \mathbf{D}_{0,1} &= \mathbf{0} \\
\overline{\pi}_{0,\mathrm{i}}^t \mathbf{R}_{0,i,\lambda_1} + \overline{\pi}_{0,\mathrm{i+1}}^t \mathbf{D}_{0,i+1} &= \mathbf{0}
\end{aligned}$$

for $i = 1, 2, \ldots, c_2 - 1$. Finally, the system must satisfy the normalisation constraint, that is

$$\overline{\pi}^t \overline{\mathrm{e}} = 1 \tag{5.27}$$

where $\overline{\mathrm{e}}$ is a vector of 1's.

5.5 Computational Algorithm - Preemptive MPDQ

Now the computational algorithm which will be used to solve the steady-states of the preemptive MPDQ is detailed. While the process of solving the steady state probabilities is quite complicated, the idea is based on the structures given by (5.26). The algorithm follows on from the process used for the preemptive single finite buffer model in Chapter 4. To illustrate this idea, consider a preemptive MPDQ system with primary and secondary queue each with capacity 3. The system of generic equations given by (5.26) gives us

$$\begin{aligned}
\boldsymbol{\alpha}_0 \boldsymbol{\Lambda}_0 + \boldsymbol{\alpha}_1 \boldsymbol{\Omega}_1 &= \mathbf{0} \\
\boldsymbol{\alpha}_0 \boldsymbol{\Pi}_0 + \boldsymbol{\alpha}_1 \boldsymbol{\Lambda}_1 + \boldsymbol{\alpha}_2 \boldsymbol{\Omega}_2 &= \mathbf{0} \\
\boldsymbol{\alpha}_1 \boldsymbol{\Pi}_1 + \boldsymbol{\alpha}_2 \boldsymbol{\Lambda}_2 + \boldsymbol{\alpha}_3 \boldsymbol{\Omega}_3 &= \mathbf{0} \\
\boldsymbol{\alpha}_2 \boldsymbol{\Pi}_2 + \boldsymbol{\alpha}_3 \boldsymbol{\Lambda}_3 &= \mathbf{0}
\end{aligned} \tag{5.28}$$

The algorithm is based on solving all states within $\boldsymbol{\alpha}_{c_1}$ first. Once this is achieved, we can immediately solve $\boldsymbol{\alpha}_{c_1-1}$. So in reference to the above example, we solve $\boldsymbol{\alpha}_3$ and hence can solve $\boldsymbol{\alpha}_2$ using the last equation. Once these two general states are solved we continue back until $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_0$ are finally solved. As shall be shown, the algorithm requires more specific manipulations than this broad outline. To simplify the process, each stage of the process is defined either by the probabilities within $\boldsymbol{\alpha}_i$ that are being solved, or by a sub-stage that may solve probabilities across states. Firstly to define some common constants:

$$\alpha_{i,j} = \frac{\mu_i}{\mu_j + \lambda}, \chi_{i,j} = \frac{\lambda_i}{\mu_j + \lambda} \text{ and } \rho_i = \frac{\lambda_i}{\mu_i} \text{ where } \lambda = \lambda_1 + \lambda_2; i, j \in \{1, 2\}.$$

5.5.1 $\boldsymbol{\alpha}_{c_1}$ stage

In this stage all the components of $\boldsymbol{\alpha}_{c_1}$ are solved. We begin this stage with the initialization

$$\pi_{c_1,c_2-1,0} = 1 \tag{5.29}$$

Within $\boldsymbol{\alpha}_{c_1}$ there is almost no dependence upon states outside of $\boldsymbol{\alpha}_{c_1}$. The transition from $(c_1 - 1, 0)$ to $(c_1, 0)$ with rate λ_1 is the exception. This is the simplest place to begin the solution process. Using (5.29) we immediately obtain

$$\overline{\pi}_{\mathbf{c_1},\mathbf{c_2}} = (\pi_{c_1,c_2,0}) = (\rho_1). \tag{5.30}$$

All $\pi_{c_1,j,0}$ and $\pi_{c_1,0}$ are now found using

$$1_{\{j>0\}}\pi_{c_1,j,0} + 1_{\{j=0\}}\pi_{c_1,0} = \chi_{1,1}^{-1}\pi_{c_1,j+1,0} - \rho_1^{-1}\pi_{c_1,j+2,0}; \tag{5.31}$$

starting with $j = c_2 - 2$ and progressing down to $j = 0$.

With the $\pi_{c_1,j,0}$ states solved in (5.30) and (5.31) they are then used to find all probabilities within $\overline{\pi}_{\mathbf{c_1},\mathbf{i}}$, where $\mathbf{i} = 0, 1, \ldots, c_2$. This is accomplished by solving the following equation recursively:

$$\begin{aligned}\pi_{c_1,i,j} = 1_{\{i<c_2-j\}}[&\alpha_{1,1}\pi_{c_1,i+1,j} + \chi_{2,1}\left(1_{\{j=1\}}\pi_{c_1,0} + 1_{\{j>1\}}\pi_{c_1,i,j-1}\right) \\ &+ 1_{\{i>0\}}\chi_{1,1}\pi_{c_1,i-1,j}] + 1_{\{i=c_2-j\}}\left(\frac{\lambda_2}{\mu_1}\pi_{c_1,i,j-1} + \rho_1\pi_{c_1,i-1,j}\right)\end{aligned} \tag{5.32}$$

In (5.32), we begin by setting $j = 1$. This is incremented by 1 up to $c_2 - 1$ after obtaining $\pi_{c_1,i,j}$ for $i = 0, 1, \ldots, c_2 - j$. Then simply substitute $\pi_{c_1,c_2-j,j}$ into $\pi_{c_1,c_2-j-1,j}$ of the previous equation and simplify. In this way we continue down finally obtaining solution for $\pi_{c_1,0,j}$. After which forward solve for all $\pi_{c_1,i,j}$ for $i = 1, \ldots, c_2 - j$. To finish this stage set

$$\pi_{c_1,0,c_2} = \frac{\lambda_2}{\mu_1}\pi_{c_1,0,c_2-1}. \tag{5.33}$$

5.5.2 $\pi_{i,0}$ stage

The next step in the procedure works up through the A matrix via the partitions. We solve for $\pi_{i,0}$,where $i = 0, \ldots, c_1 - 1$ using the following equation

$$\pi_{i,0} = \chi_{1,1}^{-1}\pi_{i+1,0} - \rho_1^{-1}\left(1_{\{i=c_1-1\}}\pi_{i+1,1,0} + 1_{\{i<c_1-1\}}\pi_{i+2,0}\right) \tag{5.34}$$

working from $i = c_1 - 1$ down to 0.

5.5.3 $\alpha_i, i = 2, \ldots, c_1 - 1$ stage

This stage of the algorithm (which is the largest) solves the majority of the states. i^*, j^* are used to denote the outer loop of this stage. This is as some inside loops of the solution process require reference to them. Initialise this stage by setting $j^* = 1$ and $i^* = c_1 - 1$ to obtain immediately

Step 1

$$\pi_{0,j^*} = 1_{\{j^*>1\}}\left(-\rho_2\pi_{0,j^*-2}\right) + \left(1_{\{j^*=1\}}\frac{\lambda}{\mu_2} + 1_{\{j^*>1\}}\alpha_{2,2}^{-1}\right)\pi_{0,j^*-1} - \frac{\mu_1}{\mu_2}\pi_{1,j^*-1} \tag{5.35}$$

Now for $i = i^*, i^* - 1, \ldots, 1$

$$\pi_{i,j^*} = \chi_{2,1}\pi_{i,j^*-1} + \chi_{1,1}\pi_{i-1,j^*} + \alpha_{1,1}\left(1_{\{i=i^*\}}\left(\pi_{i,1,0} + \pi_{i+1,0,1}\right) + 1_{\{i<i^*\}}\pi_{i+1,j^*}\right) \tag{5.36}$$

where $\pi_{i-1,j^*}, \pi_{i,j^*-1}$are solved. Now simply substitute π_{1,j^*} into π_{2,j^*} and so on to end up with $\pi_{i^*,1}$ in terms of $\pi_{i^*,1,0}$. Label this solution (𝔸).

Next for $j = 1, \ldots, c_2$

$$\begin{aligned}\pi_{i^*,j,0} = {} & \chi_{1,1}\left(1_{\{j=1\}}\pi_{i^*,c_1-i^*} + 1_{\{c_2>j>1\}}\pi_{i^*,j-1,0}\right) + 1_{\{j=c_2\}}\rho_1\pi_{i^*,j-1,0} \\ & + 1_{\{j<c_2\}}\alpha_{1,1}\pi_{i^*,j+1,0}.\end{aligned} \tag{5.37}$$

So back substitute $\pi_{i^*,c_2,0}$ into $\pi_{i^*,c_2-1,0}$ and so on to obtain $\pi_{c_1-1,1}$ in terms of $\pi_{c_1-1,1,0}$. Label this solution (𝔹).

Using the results from (5.36) and (5.37) gives the solution for $\pi_{c_1-1,1,0}$. This is obtained by substituting (𝔹) into (𝔸). Going back with this solution, forward solve for $\pi_{i^*,1,0}, \pi_{i^*,2,0}, \ldots, \pi_{i^*,c_2,0}$ from (5.37) and $\pi_{i^*,j^*}, \pi_{i^*-1,j^*}, \ldots, \pi_{1,j^*}$ from (5.36).

Now to use a nested loop to solve the rest of the states. For the nesting, initialise by setting $k = 1$. Increment after completion through the j stages, then k, then i. First set $j = 0$, $i = i^*$.

Step 2

For $j = 0, 1, \ldots, c_2 - k$;(and if $k = c_2$), $i = i^*$,

$$\begin{aligned}\pi_{i,j,k} &= 1_{\{j<c_2-k\}}\alpha_{1,1}\pi_{i,j+1,k} \\ &+ \chi_{2,1}\left(1_{\{j=0\cap k=1\}}\pi_{i,c_2-i} + 1_{\{(c_2-k>j>0)\cup((c_2>k>1)\cap j=0)\}}\pi_{i,j,k-1}\right) \\ &+ 1_{\{j=c_2-k\cup k=c_2\}}\frac{\lambda_2}{\mu_1}\pi_{i,j,k-1} + 1_{\{j=0\cap k<c_2\}}\alpha_{1,1}\pi_{i+1,j,k+1} \\ &+ \left(1_{\{k<c_2\cap(c_2-k>j>0)\}}\chi_{1,1} + 1_{\{k<c_2\cap j=c_2-k\}}\rho_1\right)\pi_{i,j-1,k}. \end{aligned} \tag{5.38}$$

Now it is simply a case of substituting in (5.38) $\pi_{i,0,k}$ into $\pi_{i,1,k}$ and so on, ending up with the final substitution into $\pi_{i,j,k}$. Then by back substitution, we obtain solutions for $\pi_{i,0,k}$ up to $\pi_{i,j,k}$. This is one cycle of this step. Return to **Step 2** adding one to k, stopping on completion of $k = c_2$.

After this looping process, check if $i^* < c_1 - 1$. If this is the case, calculate across for the rest of the primary queue space for an immediate solution. So for $j = 2, \ldots, c_1 - i^*$,

$$\pi_{i,j} = \chi_{1,1}\pi_{i-1,j} + \alpha_{1,1}\pi_{i+1,j} + \chi_{2,1}\pi_{i,j-1}. \tag{5.39}$$

Now $i^* = i^* - 1$, $j^* = j^* + 1$ returning to **Step 1**, stopping when $i^* = 2$ is complete. In this way, we solve the states $\overline{\pi}_{\mathbf{i}}$, where $\mathbf{i} = 2, \ldots, c_1 - 1$, and $\overline{\pi}_{\mathbf{i,j}}$, where $\mathbf{i} = 2, \ldots, c_1 - 1$, $\mathbf{j} = 0, \ldots, c_2$.

5.5.4 $\alpha_i, i = 0, 1$ stage

The solution process for the final two partitions of the **A** matrix requires some substantial adjustments to those undertaken in the previous stage. The departure of a class 2 customer is now possible outside of the tridiagonal, making the solution process a little more complicated than in prior steps. This can be seen when inspecting the matrices within $\mathbf{\Pi}_0$ where the service rate μ_2 appears. To assist in the solution process the following simplification is required. For any sized MPDQ

$$\pi_{0,j,0} = \begin{cases} \frac{\lambda_1}{\mu_2}\left(\chi_{1,2}\right)^{j-2}\pi_{0,1,0} & j = c_2 \\ \left(\chi_{1,2}\right)^{j-1}\pi_{0,1,0} & j = 2, \ldots, c_2 - 1 \end{cases}, \tag{5.40}$$

To solve within the last 2 partitions some preliminary substitution is required. Within the first partition

$$\pi_{0,c_1} = \chi_{1,2}^{-1}\pi_{0,1,0} \tag{5.41}$$

and

$$\pi_{1,c_1-1} = \alpha_{2,1}^{-1}\pi_{0,c_1-1} - \frac{\mu_2}{\mu_1}\pi_{0,c_1} - \frac{\lambda_2}{\mu_1}\pi_{0,c_1-2} \tag{5.42}$$

In (5.42) it should be noted that π_{0,c_1-1} and π_{0,c_1-2} are already solved. Label this *Result 1*. Next obtain

$$\pi_{0,1,0} = -\frac{\lambda_1}{\mu_2}\pi_{0,c_1-1} + \alpha_{2,1}^{-1}\pi_{1,c_1-1} - \rho_2\pi_{1,c_1-2} - \frac{\mu_1}{\mu_2}\pi_{1,1,0} - \frac{\mu_1}{\mu_2}\pi_{2,0,1} \tag{5.43}$$

Now substitute *Result 1* into (5.43) and simplify, noting that $\pi_{0,c_1-1}, \pi_{0,c_1-2}, \pi_{1,c_1-2}$ and $\pi_{2,0,1}$ are already solved, and label this simplified equation *Result 2*. So *Result 2* has $\pi_{0,1,0}$ is in terms of $\pi_{1,1,0}$.

Next using the probabilities from the state $S = \{(1,1,0)\}$

$$\pi_{1,1,0} = \alpha_{2,1}\chi_{1,2}\pi_{0,1,0} + \chi_{1,1}\pi_{1,c_1-1} + \alpha_{1,1}\pi_{1,2,0}. \tag{5.44}$$

Substitute *Result 1* into (5.44) for π_{1,c_1-1} and simplify, calling this *Result 3*. Now some looping is needed. Begin with j equalling c_2 in

$$\begin{aligned}\pi_{1,j,0} = {} & 1_{\{j<c_2\}}\left(\alpha_{2,1}\pi_{0,j+1,0} + \chi_{1,1}\pi_{1,j-1,0} + \alpha_{1,1}\pi_{1,j+1,0}\right) \\ & + 1_{\{j=c_2\}}\rho_1\pi_{1,c_2-1,0}\end{aligned} \tag{5.45}$$

continuing down to $j = 2$. Notice that the term $\pi_{0,j+1,0}$ can be simplified in each equation using (5.40). From this construction, we substitute $\pi_{1,c_2,0}$ into $\pi_{1,c_2-1,0}$, and so on, down to $\pi_{1,2,0}$. This final substitution will contain $\pi_{1,2,0}$ in terms of $\pi_{1,1,0}$ and $\pi_{0,1,0}$. Label this equation *Result 4*.

Now to bring together all the results. There are 3 equations and 3 unknowns, so, by substituting *Result 4* into *Result 3*, then this into *Result 2*, the solutions for $\pi_{1,2,0}$, $\pi_{1,1,0}$, and $\pi_{0,1,0}$ are obtained. Now by back substitution into (5.45), solve $\pi_{1,j,0}$, where $j = 1, \ldots, c_2$. From (5.42), then obtain π_{1,c_1-1}. Finally obtain π_{0,c_1} as

$$\pi_{0,c_1} = \chi_{1,2}^{-1}\pi_{0,1,0}. \tag{5.46}$$

Next some looping to create a solution with the equations in terms of $\pi_{0,0,1}$. Begin with $j = 1$

$$\pi_{0,j,1} = \chi_{1,2}\pi_{0,j-1,1} + \chi_{2,2}\pi_{0,j,0} \tag{5.47}$$

and generate equations up to $j = c_2 - 1$. Each equation can be simplified for $\pi_{0,j,0}$ using (5.40). Substitute $\pi_{0,1,1}$ into $\pi_{0,2,1}$, and so on, finally obtaining $\pi_{0,c_2-1,1}$ in terms of $\pi_{0,c_2-2,1}$. Then back substituting each equation will have the desired $\pi_{0,j,1}$in terms of $\pi_{0,0,1}$. This simplification is needed to assist in obtaining relationships in the next loop. Begin with $j = c_2 - 1$

$$\begin{aligned}\pi_{1,j,1} = {} & 1_{\{j<c_2-1\}}\left(\alpha_{1,1}\pi_{1,j+1,1} + \alpha_{2,1}\pi_{0,j+1,1}\right) + \pi_{1,j-1,1}\left(1_{\{j<c_2-1\}}\chi_{1,1} + 1_{\{j=c_2-1\}}\rho_1\right) \\ & + \pi_{1,j,0}\left(1_{\{j<c_2-1\}}\chi_{2,1} + 1_{\{j=c_2-1\}}\frac{\lambda_2}{\mu_1}\right)\end{aligned} \tag{5.48}$$

and continue down to $j = 1$. Similarly to (5.47), substitute $\pi_{1,c_2-1,1}$ into $\pi_{1,c_2-2,1}$, and so on, down to $\pi_{1,1,1}$. In each equation, use the simplifications of $\pi_{0,j,1}$ in terms of $\pi_{0,0,1}$ from (5.47), and the solutions already obtained for $\pi_{1,j,0}$ to end up with $\pi_{1,1,1}$ in terms of $\pi_{1,0,1}$ and $\pi_{0,0,1}$. Now to draw together these findings. We have for state $S = \{(1,0,1)\}$

$$\pi_{1,0,1} = \alpha_{1,1}\pi_{1,1,1} + \alpha_{2,1}\pi_{0,1,1} + \chi_{2,1}\pi_{1,c_1-1} + \alpha_{1,1}\pi_{2,0,2} \tag{5.49}$$

By substituting for $\pi_{1,1,1}$ using (5.48), and for $\pi_{0,1,1}$ using (5.47), we obtain $\pi_{1,0,1}$ in terms of $\pi_{0,0,1}$. Label this *Result 5*. Now to complete the simplifications,

$$\pi_{0,0,1} = \alpha_{2,2}^{-1}\pi_{0,c_1} - \rho_2\pi_{0,c_1-1} - \frac{\mu_1}{\mu_2}\pi_{1,0,1} \tag{5.50}$$

and substituting (5.50) in *Result 5* solve $\pi_{1,0,1}$ and then $\pi_{0,0,1}$. Now back substitute to obtain $\pi_{1,j,1}$, where $j = 1, \ldots, c_2 - 1$ from (5.48) and $\pi_{0,j,1}$ where $j = 1, \ldots, c_2 - 1$ from (5.47).

Using the same principles as those used to derive the last equations, apply nested looping to solve the remaining probabilities. Initialise the outer loop setting $k = 2$ and $j = 1$,

$$\pi_{0,j,k} = \pi_{0,j,k-1}\left(1_{\{c_2-k>1\}}\chi_{2,2} + 1_{\{c_2-k=1\}}\rho_2\right) + \pi_{0,j-1,k}\left(1_{\{c_2-k>1\}}\chi_{1,2} + 1_{\{c_2-k=1\}}\frac{\lambda_1}{\mu_2}\right) \tag{5.51}$$

continuing up until $j = c_2 - k$. Similarly, start from $j = c_2 - k$ to obtain

$$\begin{aligned}\pi_{1,j,k} = {} & 1_{\{j<c_2-k\}}\left(\alpha_{1,1}\pi_{1,j+1,k} + \alpha_{2,1}\pi_{0,j+1,k}\right) + \pi_{1,j,k-1}\left(1_{\{j=c_2-k\}}\frac{\lambda_2}{\mu_1} + 1_{\{j<c_2-k\}}\chi_{2,1}\right) \\ & + \pi_{1,j-1,k}\left(1_{\{j=c_2-k\}}\rho_1 + 1_{\{j<c_2-k\}}\chi_{1,1}\right)\end{aligned} \tag{5.52}$$

continuing down to $j = 1$. Next simplify

$$\pi_{1,0,k} = \alpha_{1,1}\pi_{1,1,k} + \alpha_{2,1}\pi_{0,1,k} + \chi_{2,1}\pi_{1,0,k-1} + \alpha_{1,1}\pi_{2,0,k+1} \tag{5.53}$$

Using (5.52) substitute $\pi_{1,c_2-k,k}$ into $\pi_{1,c_2-k-1,k}$, and so on, ending up with $\pi_{1,0,k}$ in terms of $\pi_{0,0,k}$. Label this *Result 6*. Substituting *Result 6* into

$$\pi_{0,0,k} = 1_{\{k=2\}}\left(-\rho_2\right)\pi_{0,c_1} + 1_{\{k>2\}}\left(-\rho_2\right)\pi_{0,0,k-2} + \alpha_{2,2}^{-1}\pi_{0,0,k-1} - \frac{\mu_1}{\mu_2}\pi_{1,0,k} \tag{5.54}$$

to obtain the solution for $\pi_{0,0,k}$. Now back substitute through (5.51), (5.52) and (5.53) to end up with solutions for $\pi_{1,0,k}$, $\pi_{1,1,k}$, ..., $\pi_{1,c_2-k,k}$; $\pi_{0,1,k}$, $\pi_{0,2,k}$, ..., $\pi_{0,c_2-k,k}$. Return to (5.51), adding one to k, and stopping when the $k = c_2 - 1$ loops are complete.

To finish

$$\pi_{0,0,c_2} = \rho_2 \pi_{0,0,c_2-1} \tag{5.55}$$

Hence $\overline{\pi}_{\mathbf{i}}$ is solved where $\mathbf{i} = 0, 1$, and $\overline{\pi}_{\mathbf{i,j}}$ is solved, where $\mathbf{i} = 0, 1,; \mathbf{j} = 0, \ldots, c_2$. Finally, normalising all the probabilities using (5.27) completes the computational algorithm. With the steady-state probabilities obtained, the waiting time analysis can now be undertaken. The process is an extension of the single finite buffer model analysis seen in Section 4.4.3.

5.6 Waiting Times of Class 1 Customers

There is the need to distinguish between two cases of the MPDQ operation: when the primary queue *isn't* full and when it *is*. To start with, the simple case is evaluated first - class 1 customers.

5.6.1 Primary Queue isn't full

If a tagged class 1 customer C_1 joins queue 1 and sees no class 1 customers ahead of him, then C_1 will go to the H-O-L and eject the class 2 customer, if any, who is being served. On the other hand, if C_1 sees $0 < n < c_1$ class 1 customers ahead of him, then the distribution of his waiting time is a weighted sum of Erlang distributions $E(n, \mu_1)$ with n phases, $n \geq 1$, and parameter μ_1, i.e. with corresponding density function

$$f_{n,\mu_1}(t) = \frac{\mu_1^n}{(n-1)!} t^{n-1} e^{-\mu_1 t} \tag{5.56}$$

when $t > 0$. Let

$$\Pi_{Q_1} = \sum_{0 \leq i+j < c_1} \pi_{i,j} \tag{5.57}$$

i.e. the probability that the primary queue isn't full, then, using the PASTA property [Wolff, 1982] the conditional density of the waiting time of C_1 given that queue 1 isn't full is

$$W_1^{(1)}(t) = \frac{1}{\Pi_{Q_1}} \left\{ \left(\sum_{j=0}^{c_1-1} \pi_{0,j} \right) \delta(t) + \sum_{j=0}^{c_1-2} \sum_{i=1}^{c_1-1-j} \pi_{i,j} f_{i,\mu_1}(t) \right\} \tag{5.58}$$

where $\delta(t)$ is the Dirac delta function. Let $\hat{W}_1^{(1)}(s)$ denote the Laplace transform of $W_1^{(1)}(t)$, i.e.

$$\hat{W}_1^{(1)}(s) = \int_0^\infty e^{-st} W_1^{(1)}(t) dt \tag{5.59}$$

where $Re(s) > 0$, then it easily follows that

$$\hat{W}_1^{(1)}(s) = \frac{1}{\Pi_{Q_1}} \left\{ \sum_{j=0}^{c_1-1} \pi_{0,j} + \sum_{j=0}^{c_1-2} \sum_{i=1}^{c_1-1-j} \pi_{i,j} \left(\frac{\mu_1}{s+\mu_1} \right)^i \right\} \tag{5.60}$$

5.6.2 Primary Queue is full

In this case, the tagged customer C_1 joins queue 2 and sees either no class 1 customers or at least one class 1 customer in the primary queue. In the first case, his waiting time is equal to the sum of the service time of the class 2 customer at the head of the queue and all the service times of class 1 customers ahead of him in the secondary queue. In the second case, his waiting time will be equal to the sum of all the service times of all class 1 customers in front of him in the combined queue. Thus the the conditional density of the waiting time of C_1 given that queue 1 is full is

$$W_2^{(1)}(t) = \frac{1}{\Pi_{Q_2}} \left\{ \sum_{0 \le i'+j' < c_2} \pi_{0,i',j'} f_{1,\mu_2} \star f_{i',\mu_1}(t) + \sum_{i=1}^{c_1} \sum_{0 \le i'+j' < c_2} \pi_{i,i',j'} f_{i+i',\mu_1}(t) \right\} \tag{5.61}$$

where $\star$ refers to the convolution operator and

$$\Pi_{Q_2} = \sum_{i=0}^{c_1} \sum_{0 \le i'+j' < c_2} \pi_{i,i',j'}. \tag{5.62}$$

Note here that the corresponding Laplace transform of $W_2^{(1)}(t)$ is given by

$$\begin{aligned} \hat{W}_2^{(1)}(s) &= \frac{1}{\Pi_{Q_2}} \{ \sum_{0 \le i'+j' < c_2} \pi_{0,i',j'} (\frac{\mu_2}{\mu_2 + s})(\frac{\mu_1}{\mu_1 + s})^{i'} \\ &\quad + \sum_{i=1}^{c_1} \sum_{0 \le i'+j' < c_2} \pi_{i,i',j'} (\frac{\mu_1}{\mu_1 + s})^{i+i'} \}. \end{aligned} \tag{5.63}$$

Finally, combining (5.58) and (5.61), the density of the waiting time of C_1, provided he can join the queue, is given by

$$W^{(1)}(t) = \frac{1}{\Pi_{Q_1} + \Pi_{Q_2}} \left\{ \Pi_{Q_1} W_1^{(1)}(t) + \Pi_{Q_2} W_2^{(1)}(t) \right\}. \tag{5.64}$$

5.7 Waiting Times of Class 2 Customers

The waiting time of a class 2 customer is more difficult to obtain than for class 1. This is mainly due to the fact that his waiting time could be delayed by subsequent arrivals of class 1 customers into the system. In addition, any subsequent class 2 arrivals will reduce the amount of waiting rooms thus improving his chance of moving to the front of the line. Thus, any analysis of the waiting time of a class 2 customer will have to consider these two aspects. In this section, an algorithm for obtaining the k^{th} moment, $k = 1, 2, \ldots$ of the waiting time of an arbitrary tagged class 2 customer C_2 will be derived through an analogous approach to that exploited in [Wagner and Krieger, 1999] and [Wagner, 1997]. The approach is based on analysing the first passage time of an

appropriate stochastic process related to queue length. As for class 1 customers, the discussion is broken into two cases, when the primary queue is full and when it isn't.

5.7.1 Primary Queue isn't full

In this case, it is possible for the tagged customer C_2 to enter, at $t = 0$, into the MPDQ through the primary queue. Define the stochastic process $Z(t), t \geq 0$ by

$$\begin{aligned} Z(t) &= (l_{11}(t), L_{21}(t), F_1(t)) \\ \text{where} & \\ l_{11}(t) &= \text{number of class 1 customers in queue 1,} \\ L_{21}(t) &= \text{number of class 2 customers in front and} \\ & \quad \textit{including} \text{ tagged customers in queue 1} \\ F_1(t) &= \text{amount of free space in queue 1} \end{aligned}$$

and let

$$C_1(t) = L_{21}(t) + l_{11}(t) + F_1(t). \tag{5.65}$$

Note that the arrival of a class 2 customer will reduce C_1 by 1 whereas the arrival of either class of customers will reduce F_1 by 1. No arrival can enter the system when $F_1 = 0$. Assume that initially, just prior to C_2 joining queue 1, $C_1 = c_1$. Due to the assumptions underlying the MPDQ, $Z(t)$ is a continuous time Markov chain taking values in the set

$$\begin{aligned} \mathcal{R}_1 &= \{(l_{11}, L_{21}, F_1) \in \mathcal{N}_0^3 : 0 \leq l_{11} \leq c_1, \\ & \qquad 0 \leq L_{21} \leq c_1, 0 \leq F_2 \leq c_1\}. \end{aligned}$$

Define the set of states $\mathcal{A}_1 \subset \mathcal{R}_1$ by

$$\mathcal{A}_1 = \{(0, 1, F_1) : 0 \leq F_1 \leq c_1 - 1\}.$$

Let $T_1(l_{11}, L_{21}, F_1)$ be the first passage time for the process Z to be absorbed into $\mathcal{A}_1$ starting initially in state (l_{11}, L_{21}, F_1) where $L_{21} \geq 1$. This is precisely the time it will take for the tagged customer to be served. Denote the Laplace transform of $T_1(l_{11}, L_{21}, F_1)$ by $\hat{W}_{l_{11}, L_{21}, F_1}(s)$, i.e.

$$\hat{W}_{l_{11}, L_{21}, F_1}(s) = E(e^{-sT_1(l_{11}, L_{21}, F_1)}) \tag{5.66}$$

where $Re(s) > 0$.

Using the PASTA property [Wolff, 1982], the Laplace transform of the time to absorption of $Z(t)$ given that a class 2 customer can join the primary queue is

$$\hat{W}_1^{(2)}(s) = \frac{1}{\Pi_{Q_1}} \left\{ \sum_{0 \le i+j < c_1} \pi_{i,j} \hat{W}_{i,j+1,c_1-i-j-1}(s) \right\} \tag{5.67}$$

Inversion of (5.67) will in principle gives us the distribution of the waiting time of a class 2 customer. However, an equally effective approach is to obtain the k^{th} moment of the waiting time through a recursive algorithm introduced in the following sections.

5.7.2 Relationships between k^{th} Moments of Waiting Time for Class 2 Customers in the Primary Queue

Now to develop several relationships between the k^{th} moments of waiting times for class 2 customers in the primary queue for various combinations of l_{11}, L_{21} and F_1. But first to exhibit the following relationships, which easily follow using a conditional argument based on the first transition of the process $Z(t)$ from an initial state to a state in $\mathcal{R}_1$ before absorption into $\mathcal{A}_1$ and the exponentially distributed time it takes to accomplish this first transition.

(i) $1 < L_{21} \le c_1$,

$$\hat{W}_{0,L_{21},0}(s) = \frac{\mu_2}{s+\mu_2} \hat{W}_{0,L_{22}-1,1}(s); \tag{5.68}$$

(ii) $0 < l_{11} < c_1, 0 < L_{21} \le c_1$,

$$\hat{W}_{l_{11},L_{21},0}(s) = \frac{\mu_1}{s+\mu_1} \hat{W}_{l_{11}-1,L_{21},1}(s); \tag{5.69}$$

(iii) $0 < L_{21} \le c_1, 0 < F_1 < c_1$,

$$\begin{aligned} \hat{W}_{0,L_{21},F_1}(s) &= \frac{\mu_2}{s+\mu_2+\lambda} \hat{W}_{0,L_{21}-1,F_1+1}(s) + \frac{\lambda_1}{s+\mu_2+\lambda} \hat{W}_{1,L_{21},F_1-1}(s) \\ &\quad + \frac{\lambda_2}{s+\mu_2+\lambda} \hat{W}_{0,L_{21},F_1-1}(s); \end{aligned} \tag{5.70}$$

(iv) $0 < l_{11} < c_1, 0 < L_{21} \le c_1, 0 < F_1 < c_1$,

$$\begin{aligned} \hat{W}_{l_{11},L_{21},F_1}(s) &= \frac{\mu_1}{s+\mu_1+\lambda} \hat{W}_{l_{11}-1,L_{21},F_1+1}(s) + \frac{\lambda_1}{s+\mu_1+\lambda} \hat{W}_{l_{11}+1,L_{21},F_1-1}(s) \\ &\quad + \frac{\lambda_2}{s+\mu_1+\lambda} \hat{W}_{l_{11},L_{21},F_1-1}(s). \end{aligned} \tag{5.71}$$

The next set of relationships, between various k^{th} moments of the waiting time $W^{(k)}_{l_{11},L_{21},F_1}$, are obtained from the above equations by differentiating the Laplace transform (5.66) k times and evaluating at $s = 0$ i.e.

$$W^{(k)}_{l_{11},L_{21},F_1} = (-1)^k \frac{d^k \hat{W}_{l_{11},L_{21},F_1}(s)}{ds^k}|_{s=0}. \tag{5.72}$$

(i) $1 < L_{21} \leq c_1$,

$$\mu_2 W^{(k)}_{0,L_{21},0} - \mu_2 W^{(k)}_{0,L_{22}-1,1} = k W^{(k-1)}_{0,L_{21},0}; \tag{5.73}$$

(ii) $0 < l_{11} < c_1, 0 < L_{21} \leq c_1$,

$$\mu_1 W^{(k)}_{l_{11},L_{21},0} - \mu_1 W^{(k)}_{l_{11}-1,L_{21},1} = k W^{(k-1)}_{l_{11},L_{21},0}; \tag{5.74}$$

(iii) $0 < L_{21} \leq c_1, 0 < F_1 < c_1$,

$$\begin{aligned}(\mu_2 + \lambda) W^{(k)}_{0,L_{21},F_1} - \lambda_1 W^{(k)}_{1,L_{21},F_1-1} &= k W^{(k-1)}_{0,L_{21},F_1} + \mu_2 W^{(k)}_{0,L_{21}-1,F_1+1} \\ &+ \lambda_2 W^{(k)}_{0,L_{21},F_1-1}; \end{aligned} \tag{5.75}$$

(iv) $0 < l_{11} < c_1, 0 < L_{21} \leq c_1, 0 < F_1 < c_1$,

$$\begin{aligned}(\mu_1 + \lambda) W^{(k)}_{l_{11},L_{21},F_1} - \lambda_1 W^{(k)}_{l_{11}+1,L_{21},F_1-1} &= k W^{(k-1)}_{l_{11},L_{21},F_1} + \mu_1 W^{(k)}_{l_{11}-1,L_{21},F_1+1} \\ &+ \lambda_2 W^{(k)}_{l_{11},L_{21},F_1-1}. \end{aligned} \tag{5.76}$$

5.7.3 A Recursive Algorithm for Calculating k^{th} Moment of Waiting Time for a Class 2 Customer in the Primary Queue

In this section, utilize (5.73) - (5.76) to obtain a recursive algorithm to calculate the k^{th} moment of waiting time. For fixed $L_{21} > 1$ and C_1, we define the following $C_1 - L_{21} + 1$ dimensional vector

$$W^{(k)}_{L_{21},C_1} = \begin{pmatrix} W^{(k)}_{0,L_{21},C_1-L_{21}} \\ W^{(k)}_{1,L_{21},C_1-L_{21}-1} \\ \vdots \\ W^{(k)}_{C_1-L_{21},L_{21},0} \end{pmatrix} \tag{5.77}$$

and when $L_{21} = 1$, let

$$W^{(k)}_{1,C_1} = \begin{pmatrix} W^{(k)}_{1,1,C_1-2} \\ W^{(k)}_{2,1,C_1-3} \\ \vdots \\ W^{(k)}_{C_1-1,1,0} \end{pmatrix}. \tag{5.78}$$

In the following, obtain a recursive relationship for $W^{(k)}_{L_{21},C_1}, L_{21} \geq 1$. It is necessary to distinguish between 3 possible cases: when $L_{21} > 2, L_{21} = 2$ and $L_{21} = 1$ respectively.

(a) $\mathbf{L_{21} > 2}$:

Assume for the moment that $\mathcal{C}_1 > L_{21}$. Using (5.75) in case $l_{11} = 0$, (5.76) in cases $l_{11} = 1, 2, \ldots, \mathcal{C}_1 - L_{21} - 1$ and (5.74) in case $l_{11} = \mathcal{C}_1 - L_{21}$ (note that $F_1 = \mathcal{C}_1 - L_{21} - l_{11}$ from (5.65)), obtain the following vector-matrix equation:

$$\begin{aligned} A_{L_{21},\mathcal{C}_1} W^{(k)}_{L_{21},\mathcal{C}_1} &= kW^{(k-1)}_{L_{21},\mathcal{C}_1} + \Gamma_{2,L_{21},\mathcal{C}_1} W^{(k)}_{L_{21},\mathcal{C}_1-1} \\ &+ \mu_2 e_1 (\mathcal{C}_1 - L_{21} + 1) e_1^t (\mathcal{C}_1 - L_{21} + 2) W^{(k)}_{L_{21}-1,\mathcal{C}_1} \end{aligned} \quad (5.79)$$

where

$$A_{L_{21},\mathcal{C}_1} = \begin{pmatrix} (\mu_2+\lambda) & -\lambda_1 & 0 & & \cdots & 0 \\ -\mu_1 & (\mu_1+\lambda) & -\lambda_1 & \ddots & & \\ 0 & -\mu_1 & \mu_1+\lambda & -\lambda_1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 & 0 \\ & & \ddots & -\mu_1 & (\mu_1+\lambda) & -\lambda_1 \\ 0 & & \cdots & & -\mu_1 & \mu_1 \end{pmatrix}$$

$$\in \Re^{(\mathcal{C}_1 - L_{21}+1)\times(\mathcal{C}_1-L_{21}+1)}$$

and

$$\Gamma_{2,L_{21},\mathcal{C}_1} = \begin{pmatrix} \lambda_2 & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & 0 & \lambda_2 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \in \Re^{(\mathcal{C}_1 - L_{21}+1)\times(\mathcal{C}_1-L_{21})}.$$

Using (5.73), observe that the recursive equation (5.79) still holds when $\mathcal{C}_1 = L_{21}$ provided $A_{L_{21},L_{21}} = (\mu_2)$ and make $\Gamma_{2,L_{21},L_{21}}$ void for that case.

(b) $\mathbf{L_{21} = 2}$:

Applying the same steps as in the previous case, if $\mathcal{C}_1 > 2$, the recursive relationship is

$$A_{2,\mathcal{C}_1} W^{(k)}_{2,\mathcal{C}_1} = kW^{(k-1)}_{2,\mathcal{C}_1} + \Gamma_{2,2,\mathcal{C}_1} W^{(k)}_{2,\mathcal{C}_1-1}. \quad (5.80)$$

If $\mathcal{C}_1 = 2$, $W^{(k)}_{2,2} = (W^{(k)}_{0,2,0})$ and (5.73) gives

$$\mu_2 W^{(k)}_{0,2,0} = kW^{(k-1)}_{0,2,0} + \mu_2 W^{(k)}_{0,1,1} = kW^{(k-1)}_{0,2,0}. \quad (5.81)$$

(c) $\mathbf{L_{21} = 1}$:

Next obtain a recursive relationship for $W_{1,\mathcal{C}_1}^{(k)}$. Again, first assume that $\mathcal{C}_1 > 2$. Applying (5.76) when $l_{11} = 1, 2, \ldots, \mathcal{C}_1 - 2$ and (5.74) when $l_{11} = \mathcal{C}_1 - 1$, obtain:

$$A_{1,\mathcal{C}_1} W_{1,\mathcal{C}_1}^{(k)} = kW_{1,\mathcal{C}_1}^{(k-1)} + \Gamma_{2,1,\mathcal{C}_1-1} W_{1,\mathcal{C}_1-1}^{(k)} \tag{5.82}$$

where

$$A_{1,\mathcal{C}_1} = \begin{pmatrix} (\mu_1+\lambda) & -\lambda_1 & 0 & & \cdots & 0 \\ -\mu_1 & (\mu_1+\lambda) & -\lambda_1 & \ddots & & \\ 0 & -\mu_1 & \mu_1+\lambda & -\lambda_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ & & \ddots & -\mu_1 & (\mu_1+\lambda) & -\lambda_1 \\ 0 & & \cdots & 0 & -\mu_1 & \mu_1 \end{pmatrix} \in \Re^{(\mathcal{C}_1-1)\times(\mathcal{C}_1-1)}$$

If $\mathcal{C}_1 = 2$, then $W_{1,2}^{(k)} = (W_{1,1,0}^{(k)})$ and (5.74) gives

$$\mu_1 W_{1,1,0}^{(k)} = kW_{1,1,0}^{(k-1)} + \mu_1 W_{0,1,1}^{(k)} = kW_{1,1,0}^{(k-1)}. \tag{5.83}$$

5.7.4 Primary Queue is full

Assume that the primary queue is full but the secondary queue isn't, so it is possible for the tagged customer C_2 to enter into the MPDQ system. Similar to, and with the same motivation as the introduction of $Z(t)$, define

$$Y(t) = (l_{11}(t), l_{12}(t), L_{22}(t), F_2(t))$$

where

$l_{11}(t)$ = number of class 1 customers in Q_1,

$l_{12}(t)$ = number of class 1 customers in Q_2,

$L_{22}(t)$ = number of class 2 customers in front and *including* tagged customers in the system

and $F_2(t)$ = amount of free space in Q_2.

Since the primary queue is full, the number of class 2 customers in it must be equal to $c_1 - l_{11}(t)$.

$Y(t)$ is a continuous time Markov chain which takes values in the set

$$\begin{aligned} \mathcal{R}_2 = \{&(l_{11}, l_{12}, L_{22}, F_2) \in \mathcal{N}_0^4 : 0 \le l_{11} \le c_1, \\ &0 \le l_{12} \le c_2, 0 \le L_{22} \le c_1 + c_2, 0 \le F_2(t) \le c_2\}. \end{aligned}$$

Next define

$$\mathcal{C}_2 = L_{22} + l_{11} + l_{12} + F_2. \tag{5.84}$$

Similar to C_1, C_2 accounts for the reduction in free spaces due to the arrival of class 2 customers after the arrival of the tagged customer. This phenomenon does not occur with queues which have an infinite waiting room since the amount of free spaces remains infinite with any number of arrivals into the system. The arrival of a class 2 customer will reduce the capacity C_2 by 1 whereas an arrival of either class of customer will reduce F_2 by 1. No arrival enters the system when $F_2 = 0$. Assume that initially, just prior to the tagged customer joining the secondary queue, $C_2 = c = c_1 + c_2$.

Define the set of states $\mathcal{A}_2 \subset \mathcal{R}_2$ by

$$\mathcal{A}_2 = \{(0, l_{12}, 1, F_2) : 0 \leq l_{12} < c_2, 0 \leq F_2 \leq c_2\}.$$

Let $T_2(l_{11}, l_{12}, L_{22}, F_2)$ be the first passage time for the process Y to be absorbed into $\mathcal{A}_2$ starting initially in state $(l_{11}, l_{12}, L_{22}, F_2)$ where $L_{22} \geq 1$. This is precisely the waiting time endured by the tagged customer C_2. Denote the Laplace transform of $T_2(l_{11}, l_{12}, L_{22}, F_2)$ by $\hat{W}_{l_{11},l_{12},L_{22},F_2}(s)$.

Appealing to the PASTA property [Wolff, 1982], the Laplace transform of the time to absorption of Y given that a class 2 customer finds the primary queue full and the secondary queue not full is

$$\hat{W}_2^{(2)}(s) = \frac{1}{\Pi_{Q_2}} \left(\sum_{(i,i',j') \in I_{Q_2}} \pi_{i,i',j'} \hat{W}_{i,i',j'+c_1-i+1,c_2-i'-j'-1}(s) \right). \tag{5.85}$$

Combining (5.67) and (5.85), the Laplace transform of the waiting time of C_2 given that the MPDQ system isn't full is

$$\hat{W}^{(2)}(s) = \frac{1}{\Pi_{Q_1} + \Pi_{Q_2}} \left\{ \Pi_{Q_1} \hat{W}_1^{(2)}(s) + \Pi_{Q_2} \hat{W}_2^{(2)}(s) \right\}. \tag{5.86}$$

5.7.5 Relationships between the k^{th} Moments of Waiting Time for Class 2 Customers in the Secondary Queue

Similar to the first case, the following set of relationships between $\hat{W}_{l_{11},l_{12},L_{22},F_2}(s)$ for various combinations of l_{11}, l_{12}, L_{22} and F_2 are given.

(i) $0 < l_{12} \leq c_2, 0 < L_{22} < c$,

$$\hat{W}_{0,l_{12},L_{22},0}(s) = \frac{\mu_2}{s + \mu_2} \hat{W}_{1,l_{12}-1,L_{22}-1,1}(s); \tag{5.87}$$

(ii) $0 < L_{22} \leq c$,

$$\hat{W}_{0,0,L_{22},0}(s) = \frac{\mu_2}{s + \mu_2} \hat{W}_{0,0,L_{22}-1,1}(s); \tag{5.88}$$

(iii) $0 < l_{11} \leq c_1, 0 < L_{22} < c$,

$$\hat{W}_{l_{11},0,L_{22},0}(s) = \frac{\mu_1}{s+\mu_1}\hat{W}_{l_{11}-1,0,L_{22},1}(s); \tag{5.89}$$

(iv) $0 < l_{11} \leq c_1, 0 < l_{12} < c_2, 0 < L_{22} \leq c$,

$$\hat{W}_{l_{11},l_{12},L_{22},0}(s) = \frac{\mu_1}{s+\mu_1}\hat{W}_{l_{11},l_{12}-1,L_{22},1}(s); \tag{5.90}$$

(v) $0 < l_{12} \leq c_2, 0 < L_{22} \leq c, 0 < F_2 < c_2$,

$$\begin{aligned}\hat{W}_{0,l_{12},L_{22},F_2}(s) &= \frac{\mu_2}{s+\mu_2+\lambda}\hat{W}_{1,l_{12}-1,L_{22}-1,F_2+1}(s) \\ &+\frac{\lambda_1}{s+\mu_2+\lambda}\hat{W}_{0,l_{12}+1,L_{22},F_2-1}(s) \\ &+\frac{\lambda_2}{s+\mu_2+\lambda}\hat{W}_{0,l_{12},L_{22},F_2-1}(s);\end{aligned} \tag{5.91}$$

(vi) $0 < L_{22} < c, 0 < F_2 < c_2$,

$$\begin{aligned}\hat{W}_{0,0,L_{22},F_2}(s) &= \frac{\mu_2}{s+\mu_2+\lambda}\hat{W}_{0,0,L_{22}-1,F_2+1}(s) + \frac{\lambda_1}{s+\mu_2+\lambda}\hat{W}_{0,1,L_{22},F_2-1}(s) \\ &+\frac{\lambda_2}{s+\mu_2+\lambda}\hat{W}_{0,0,L_{22},F_2-1}(s);\end{aligned} \tag{5.92}$$

(vii) $0 < l_{11} \leq c_1, 0 < L_{22} < c_2, 0 < F_2 < c_2$,

$$\begin{aligned}\hat{W}_{l_{11},0,L_{22},F_2}(s) &= \frac{\mu_1}{s+\mu_1+\lambda}\hat{W}_{l_{11}-1,0,L_{22},F_2+1}(s) + \frac{\lambda_1}{s+\mu_1+\lambda}\hat{W}_{l_{11},1,L_{22},F_2-1}(s) \\ &+\frac{\lambda_2}{s+\mu_1+\lambda}\hat{W}_{l_{11},0,L_{22},F_2-1}(s);\end{aligned} \tag{5.93}$$

(viii) $0 < l_{11} \leq c_1, 0 < l_{12} < c_2, 0 < L_{22} < c, 0 < F_2 < c_2$,

$$\begin{aligned}\hat{W}_{l_{11},l_{12},L_{22},F_2}(s) &= \frac{\mu_1}{s+\mu_1+\lambda}\hat{W}_{l_{11},l_{12}-1,L_{22},F_2+1}(s) \\ &+\frac{\lambda_1}{s+\mu_1+\lambda}\hat{W}_{l_{11},l_{12}+1,L_{22},F_2-1}(s) \\ &+\frac{\lambda_2}{s+\mu_1+\lambda}\hat{W}_{l_{11},l_{12},L_{22},F_2-1}(s).\end{aligned} \tag{5.94}$$

The next set of relationships, between various k^{th} moments of the waiting time $W^{(k)}_{l_{11},l_{12},L_{22},F_2}$, can then be obtained from the above set of equations by differentiation (5.72):

(i) $0 < l_{12} < c_2, 0 < L_{22} < c$,

$$\mu_2 W^{(k)}_{0,l_{12},L_{22},0} - \mu_2 W^{(k)}_{1,l_{12}-1,L_{22}-1,1} = kW^{(k-1)}_{0,l_{12},L_{22},0}; \tag{5.95}$$

(ii) $0 < L_{22} \leq c$,

$$\mu_2 W^{(k)}_{0,0,L_{22},0} - \mu_2 W^{(k)}_{0,0,L_{22}-1,1} = kW^{(k-1)}_{0,0,L_{22},0}; \tag{5.96}$$

(iii) $0 < l_{11} \leq c_1, 0 < L_{22} \leq c$,

$$\mu_1 W^{(k)}_{l_{11},0,L_{22},0} - \mu_1 W^{(k)}_{l_{11}-1,0,L_{22},1} = kW^{(k-1)}_{l_{11},0,L_{22},0}; \tag{5.97}$$

(iv) $0 < l_{11} \leq c_1, 0 < l_{12} < c_2, 0 < L_{22} < c_2$,

$$\mu_1 W^{(k)}_{l_{11},l_{12},L_{22},0} - \mu_1 W^{(k)}_{l_{11},l_{12}-1,L_{22},1} = kW^{(k-1)}_{l_{11},l_{12},L_{22},0}; \tag{5.98}$$

(v) $0 < l_{12} \leq c_2, 0 < L_{22} < c_2, 0 < F_2 < c_2$,

$$\begin{aligned}(\mu_2+\lambda)W^{(k)}_{0,l_{12},L_{22},F_2} - \lambda_1 W^{(k)}_{0,l_{12}+1,L_{22},F_2-1} &= kW^{(k-1)}_{0,l_{12},L_{22},F_2}(s) \\ +\mu_2 W^{(k)}_{1,l_{12}-1,L_{22}-1,F_2+1} &+ \lambda_2 W^{(k)}_{0,l_{12},L_{22},F_2-1};\end{aligned} \tag{5.99}$$

(vi) $0 < L_{22} < c_2, 0 < F_2 < c_2$,

$$\begin{aligned}(\mu_2+\lambda)W^{(k)}_{0,0,L_{22},F_2} - \lambda_1 W^{(k)}_{0,1,L_{22},F_2-1} &= kW^{(k-1)}_{0,0,L_{22},F_2} + \mu_2 W^{(k)}_{0,0,L_{22}-1,F_2+1} \\ &+ \lambda_2 W^{(k)}_{0,0,L_{22},F_2-1};\end{aligned} \tag{5.100}$$

(vii) $0 < l_{11} \leq c_1, 0 < L_{22} < c_2, 0 < F_2 < c_2$,

$$\begin{aligned}(\mu_1+\lambda)W^{(k)}_{l_{11},0,L_{22},F_2} - \lambda_1 W^{(k)}_{l_{11},1,L_{22},F_2-1} &= kW^{(k-1)}_{l_{11},0,L_{22},F_2} + \mu_1 W^{(k)}_{l_{11}-1,0,L_{22},F_2+1} \\ &+ \lambda_2 W^{(k)}_{l_{11},0,L_{22},F_2-1};\end{aligned} \tag{5.101}$$

(viii) $0 < l_{11} \leq c_1, 0 < l_{12} < c_2, 0 < L_{22} < c_2, 0 < F_2 < c_2$,

$$\begin{aligned}(\mu_1+\lambda)W^{(k)}_{l_{11},l_{12},L_{22},F_2} - \lambda_1 W^{(k)}_{l_{11},l_{12}+1,L_{22},F_2-1} &= kW^{(k-1)}_{l_{11},l_{12},L_{22},F_2} \\ +\mu_1 W^{(k)}_{l_{11},l_{12}-1,L_{22},F_2+1} &+ \lambda_2 W^{(k)}_{l_{11},l_{12},L_{22},F_2-1};\end{aligned} \tag{5.102}$$

5.7.6 A Recursive Algorithm for Calculating k^{th} Moment of Waiting Time for a Class 2 Customer in the Secondary Queue

For fixed l_{11}, L_{22} and C_2 (by necessity, $C_2 \geq L_{22} + l_{11}$ from (5.84)), define the following $C_2 - L_{22} - l_{11} + 1$ dimensional vector

$$W^{(k)}_{l_{11},L_{22},C_2} = \begin{pmatrix} W^{(k)}_{l_{11},0,L_{22},C_2-L_{22}-l_{11}} \\ W^{(k)}_{l_{11},1,L_{22},C_2-L_{22}-l_{11}-1} \\ \vdots \\ W^{(k)}_{l_{11},C_2-L_{22}-l_{11},L_{22},0} \end{pmatrix} \tag{5.103}$$

First obtain a recursive relationship for $W^{(k)}_{l_{11},L_{22},C_2}$ when $l_{11} = 1, 2, \ldots, c_1$. Consider three possible cases:

(**Case 1:** $C_2 - L_{22} - l_{11} > 1$.) Applying (5.101) when $l_{12} = 0$, (5.102) when $l_{12} = 1, 2, \ldots, C_2 - L_{22} - l_{11} - 1$ and (5.98) when $l_{12} = C_2 - L_{22} - l_{11}$, gives the following vector-matrix equation:

$$\begin{aligned} A_{l_{11},L_{22},C_2} W^{(k)}_{l_{11},L_{22},C_2} &= kW^{(k-1)}_{l_{11},L_{22},C_2} \\ &\quad + \Gamma_{l_{11},L_{22},C_2} W^{(k)}_{l_{11},L_{22},C_2-1} \\ &\quad + \mu_1 e_1 (C_2 - L_{22} - l_{11} + 1) e_1^t (C_2 - L_{22} - l_{11} + 2) W^{(k)}_{l_{11}-1,L_{22},C_2} \end{aligned} \tag{5.104}$$

where

$$\begin{aligned} A_{l_{11},L_{22},C_2} &= \begin{pmatrix} (\mu_1+\lambda) & -\lambda_1 & 0 & & \cdots & 0 \\ -\mu_1 & (\mu_1+\lambda) & -\lambda_1 & \ddots & & \\ 0 & -\mu_1 & \mu_1+\lambda & -\lambda_1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -\lambda_1 & 0 \\ & & \ddots & -\mu_1 & (\mu_1+\lambda) & -\lambda_1 \\ 0 & & \cdots & 0 & -\mu_1 & \mu_1 \end{pmatrix} \\ &\in \Re^{(C_2-L_{22}-l_{11}+1)\times(C_2-L_{22}-l_{11}+1)} \end{aligned}$$

and

$$\Gamma_{l_{11},L_{22},C_2} = \begin{pmatrix} \lambda_2 & 0 & \ddots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & 0 & \lambda_2 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \in \Re^{(C_2-L_{22}-l_{11}+1)\times(C_2-L_{22}-l_{11})}.$$

(**Case 2:** $C_2 - L_{22} - l_{11} = 1$.) In this case

$$W^{(k)}_{l_{11},L_{22},L_{22}+l_{11}+1} = \begin{pmatrix} W^{(k)}_{l_{11},0,L_{22},1} \\ W^{(k)}_{l_{11},1,L_{22},0} \end{pmatrix}. \tag{5.105}$$

Using (5.101) with $W^{(k)}_{l_{11},0,L_{22},1}$ and (5.83) with $W^{(k)}_{l_{11},1,L_{22},0}$ gives the same vector-matrix equation (5.104) provided we interpret

$$A_{l_{11},L_{22},L_{22}+l_{11}+1} = \begin{pmatrix} (\mu_1+\lambda) & -\lambda_1 \\ -\mu_1 & \mu_1 \end{pmatrix}$$

and

$$\Gamma_{l_{11},L_{22},L_{22}+l_{11}+1} = \begin{pmatrix} \lambda_2 \\ 0 \end{pmatrix}.$$

(**Case 3:** $C_2 - L_{22} - l_{11} = 0$.) Since

$$W^{(k)}_{l_{11},L_{22},L_{22}+l_{11}} = (W^{(k)}_{l_{11},0,L_{22},0})$$

(5.97) gives

$$\mu_1 W^{(k)}_{l_{11},L_{22},L_{22}+l_{11}} = kW^{(k-1)}_{0,L_{22},L_{22}+l_{11}} + \mu_1 e_1(1)e_1^t(2)W^{(k)}_{l_{11}-1,L_{22},L_{22}+l_{11}}. \tag{5.106}$$

Next consider the vector $W^{(k)}_{0,L_{22},C_2}$ i.e. $l_{11} = 0$. Again, look at these three possible cases:

(**Case 1:** $C_2 - L_{22} > 1$.) Using (5.100) in case $l_{12} = 0$, (5.99) in cases $l_{12} = 1, 2, \ldots, C_2 - L_{22} - 1$ and (5.95) in case $l_{12} = C_2 - L_{22}$, gives rise to the following vector-matrix equation:

$$\begin{aligned} A_{0,L_{22},C_2}W^{(k)}_{0,L_{22},C_2} &= kW^{(k-1)}_{0,L_{22},C_2} + \Gamma_{0,L_{22},C_2}W^{(k)}_{0,L_{22},C_2-1} \\ &+ \mu_2 e_1(C_2 - L_{22} + 1)e_1^t(C_2 - L_{22} + 2)W^{(k)}_{0,L_{22}-1,C_2} \\ &+ M_{0,L_{22},C_2}W^{(k)}_{1,L_{22}-1,C_2} \end{aligned} \tag{5.107}$$

where

$$\begin{aligned} A_{0,L_{22},C_2} &= \begin{pmatrix} (\mu_2+\lambda) & -\lambda_1 & 0 & & \cdots & 0 \\ 0 & (\mu_2+\lambda) & -\lambda_1 & \ddots & & \\ & 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -\lambda_1 & 0 \\ & & & \ddots & (\mu_2+\lambda) & -\lambda_1 \\ 0 & & \cdots & & 0 & \mu_2 \end{pmatrix} \\ &\in \Re^{(C_2-L_{22}+1)\times(C_2-L_{22}+1)}, \end{aligned}$$

and

$$M_{0,L_{22},C_2} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \mu_2 & \ddots & & \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mu_2 & 0 \end{pmatrix} \in \Re^{(C_2-L_{22}+1)\times(C_2-L_{22}+1)}.$$

(**Case 2:** $C_2 - L_{22} = 1$.) Note here that

$$W^{(k)}_{0,L_{22},L_{22}+1} = \begin{pmatrix} W^{(k)}_{0,0,L_{22},1} \\ W^{(k)}_{0,1,L_{22},0} \end{pmatrix}.$$

Using (5.100) with $W^{(k)}_{0,0,L_{22},1}$ and (5.95) with $W^{(k)}_{0,1,L_{22},0}$ gives the same vector-matrix equation (5.107) provided

$$A_{0,L_{22},L_{22}+1} = \begin{pmatrix} (\mu_2+\lambda) & -\lambda_1 \\ 0 & \mu_2 \end{pmatrix},$$

$$\Gamma_{0,L_{22},L_{22}+1} = \begin{pmatrix} \lambda_2 \\ 0 \end{pmatrix}$$

and

$$M_{0,L_{22},L_{22}+1} = \begin{pmatrix} 0 & 0 \\ \mu_2 & 0 \end{pmatrix}.$$

(**Case 3:** $C_2 - L_{22} = 0$.) Note that

$$W^{(k)}_{0,L_{22},L_{22}} = (W^{(k)}_{0,0,L_{22},0})$$

Applying (5.96), the following recursive equation for this case gives:

$$\mu_2 W^{(k)}_{0,L_{22},L_{22}} = kW^{(k-1)}_{0,L_{22},L_{22}} + \mu_2 e_1(1) e_1^t(2) W^{(k)}_{0,L_{22}-1,L_{22}}. \tag{5.108}$$

5.8 Numerical Examples

The steady-state probabilities and waiting time distribution derivations are now detailed through some examples.

The utilisation, or probability of finding a busy server, is given by

$$\pi_B = 1 - \pi_{0,0} \tag{5.109}$$

The probability of finding only class one customers in the system is given by

$$\pi^1 = \sum_{i=1}^{c_1} \pi_{i,0} + \sum_{j=1}^{c_2} \pi_{c_1,j,0} \tag{5.110}$$

The probability of finding only class two customers in the system is given by

$$\pi^2 = \sum_{j=1}^{c_1} \pi_{0,j} + \sum_{k=1}^{c_2} \pi_{0,0,k} \tag{5.111}$$

The probability of a vacant secondary queue is given by

$$\pi_{Q_2=0} = \sum_{i=0}^{c_1} \sum_{j=0}^{c_1} \pi_{i,j} \tag{5.112}$$

5.8.1 Steady State Probabilities

Consider a small dual queueing system with $c_1 = c_2 = 4$. Begin with the α_{c_1} stage and set $\pi_{4,3,0} = 1$, so $\overline{\pi}_{\mathbf{c_1},\mathbf{c_2}} = \rho_1$.

From (5.31) solve directly

$$\begin{aligned}
\pi_{4,2,0} &= \chi_{1,1}^{-1}\pi_{4,3,0} - \rho_1^{-1}\pi_{4,4,0} \\
\pi_{4,1,0} &= \chi_{1,1}^{-1}\pi_{4,2,0} - \rho_1^{-1}\pi_{4,3,0} \\
\pi_{4,0} &= \chi_{1,1}^{-1}\pi_{4,1,0} - \rho_1^{-1}\pi_{4,2,0}
\end{aligned}$$

With the $\pi_{4,j,0}$ states solved they are used to find all $\overline{\pi}_{4,\mathbf{i}}$, where $\mathbf{i} = 0, 1, \ldots, 4$.This is accomplished by solving the following equations recursively from (5.32),

$$\begin{aligned}
\pi_{4,0,1} &= \alpha_{1,1}\pi_{4,1,1} + \chi_{2,1}\pi_{4,0} \\
\pi_{4,1,1} &= \alpha_{1,1}\pi_{4,2,1} + \chi_{2,1}\pi_{4,1,0} + {}_{\{i>0\}}\,\chi_{1,1}\pi_{4,0,1} \\
\pi_{4,2,1} &= \alpha_{1,1}\pi_{4,3,1} + \chi_{2,1}\pi_{4,2,0} + {}_{\{i>0\}}\,\chi_{1,1}\pi_{4,1,1} \\
\pi_{4,3,1} &= \frac{\lambda_2}{\mu_1} + \rho_1\pi_{4,2,1}
\end{aligned}$$

Substitute $\pi_{4,3,1}$ into $\pi_{4,2,1}$ of the previous equation and simplify. In this way continue down finally obtaining solution for $\pi_{4,0,1}$. After which forward solve for all $\pi_{4,i,1}$ for $i = 1, \ldots, 3$.To finish this stage, repeat (5.32) changing k, finishing with $\pi_{4,0,4} = \frac{\lambda_2}{\mu_1}\pi_{4,0,3}$.

Now the $\pi_{i,0}$ stage. Solve for $\pi_{i,0}$,where $i = 0, \ldots, 3$ using the following

$$\begin{aligned}
\pi_{0,0} &= \chi_{1,1}^{-1}\pi_{1,0} - \rho_1^{-1}\pi_{2,0} \\
\pi_{1,0} &= \chi_{1,1}^{-1}\pi_{2,0} - \rho_1^{-1}\pi_{3,0} \\
\pi_{2,0} &= \chi_{1,1}^{-1}\pi_{3,0} - \rho_1^{-1}\pi_{4,0} \\
\pi_{3,0} &= \chi_{1,1}^{-1}\pi_{4,0} - \rho_1^{-1}\pi_{4,1,0}
\end{aligned}$$

Next the $\alpha_i, i = 2, 3$ stage. Initialise by setting $j^* = 1$ and $i^* = 3$:

$$\pi_{0,1} = \frac{\lambda}{\mu_2}\pi_{0,0} - \frac{\mu_1}{\mu_2}\pi_{1,0}$$

Step 1

Now for $i = 3, 2, 1$

$$\begin{aligned}
\pi_{3,1} &= \chi_{2,1}\pi_{3,0} + \chi_{1,1}\pi_{2,1} + \alpha_{1,1}\pi_{3,1,0} + \alpha_{1,1}\pi_{4,0,1} \\
\pi_{2,1} &= \chi_{2,1}\pi_{2,0} + \chi_{1,1}\pi_{1,1} + \alpha_{1,1}\pi_{3,1} \\
\pi_{1,1} &= \chi_{2,1}\pi_{1,0} + \chi_{1,1}\pi_{0,1} + \alpha_{1,1}\pi_{2,1}
\end{aligned}$$

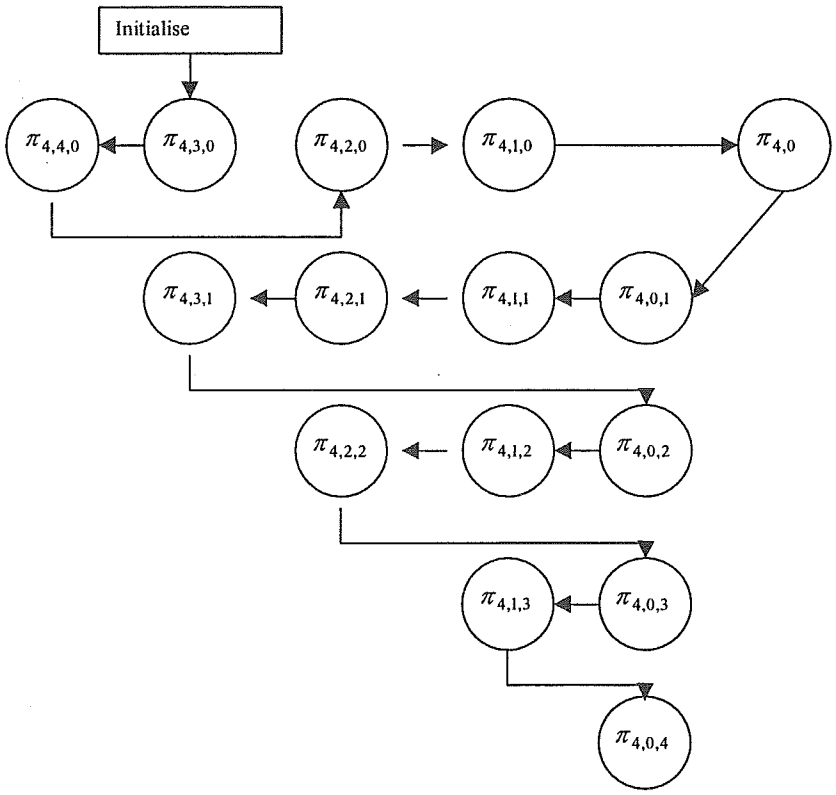

Figure 5.5: Solution order : α_{c_1} stage

where $\pi_{i,0}, \pi_{4,0,1}$are solved. Now simply substitute $\pi_{1,1}$ into $\pi_{2,1}$ and so on, to end up with $\pi_{3,1}$ in terms of $\pi_{3,1,0}$. Label this solution (𝔸).

Next for $j = 1, \ldots, 4$

$$\begin{aligned}
\pi_{3,1,0} &= \chi_{1,1}\pi_{3,1} + \alpha_{1,1}\pi_{3,2,0} \\
\pi_{3,2,0} &= \chi_{1,1}\pi_{3,1,0} + \alpha_{1,1}\pi_{3,3,0} \\
\pi_{3,3,0} &= \chi_{1,1}\pi_{3,2,0} + \alpha_{1,1}\pi_{3,4,0} \\
\pi_{3,4,0} &= \rho_1\pi_{3,3,0}
\end{aligned}$$

Back substitute $\pi_{3,4,0}$ into $\pi_{3,3,0}$ and so on to obtain $\pi_{3,1}$ in terms of $\pi_{3,1,0}$. Label this solution (𝔹). By substituting (𝔹) into (𝔸) solves $\pi_{3,1,0}$. Going back with this solution, forward solve for $\pi_{3,2,0}, \ldots, \pi_{3,4,0}$ and $\pi_{3,1}, \pi_{2,1}, \pi_{1,1}$.

Now use a nested loop to solve the rest of the states: set $k = 1, j = 0, i = 3$

Step 2

$$\pi_{3,0,1} = \alpha_{1,1}\pi_{3,1,1} + \chi_{2,1}\pi_{3,0} + \alpha_{1,1}\pi_{4,0,2} \tag{5.113}$$

Next construct

$$\begin{aligned}
\pi_{3,1,1} &= \alpha_{1,1}\pi_{3,2,1} + \chi_{2,1}\pi_{3,1,0} + \alpha_{1,1}\pi_{3,0,1} \\
\pi_{3,2,1} &= \alpha_{1,1}\pi_{3,3,1} + \chi_{2,1}\pi_{3,2,0} + \alpha_{1,1}\pi_{3,1,1}
\end{aligned} \tag{5.114}$$

To complete these equations obtain

$$\pi_{3,3,1} = \frac{\lambda_2}{\mu_1}\pi_{3,3,0} + \rho_1\pi_{3,2,1}$$

Now substitute $\pi_{3,0,1}$ in (5.113) into $\pi_{3,1,1}$ (5.114) and so on, ending up with the final substitution into $\pi_{3,3,1}$. Then by back substitution, solutions for $\pi_{3,0,1}$ up to $\pi_{3,3,1}$ are obtained. This is one cycle of this step. Now return to **Step 2** adding one to k, stopping on completion of the cycle commencing $k = 4$.

After this looping process, check if $i^* < 3$. If this is the case, calculate across for the rest of the primary queue space for an immediate solution. So for $j = 2, \ldots, c_1 - i^*$,

$$\pi_{i,j} = \chi_{1,1}\pi_{i-1,j} + \alpha_{1,1}\pi_{i+1,j} + \chi_{2,1}\pi_{i,j-1} \tag{5.115}$$

Now $i^* := i^* - 1$, returning to **Step 1**, stopping when $i^* = 2$ is complete. In this way, the states $\overline{\pi}_{\mathbf{i}}$ are solved where $\mathbf{i} = 2, \ldots, c_1 - 1$, and $\overline{\pi}_{\mathbf{i,j}}$,where $\mathbf{i} = 2, \ldots, c_1 - 1$, $\mathbf{j} = 0, \ldots, c_2$.

Now the $\alpha_{0,1}$ stage : To solve within the last 2 partitions, some preliminary substitution is needed. Within the first partition

$$\pi_{1,3} = \frac{\mu_2}{\mu_1}\left(\alpha_{2,2}^{-1}\pi_{0,3} - \rho_2\pi_{0,2} - \chi_{1,2}^{-1}\pi_{0,1,0}\right) \tag{5.116}$$

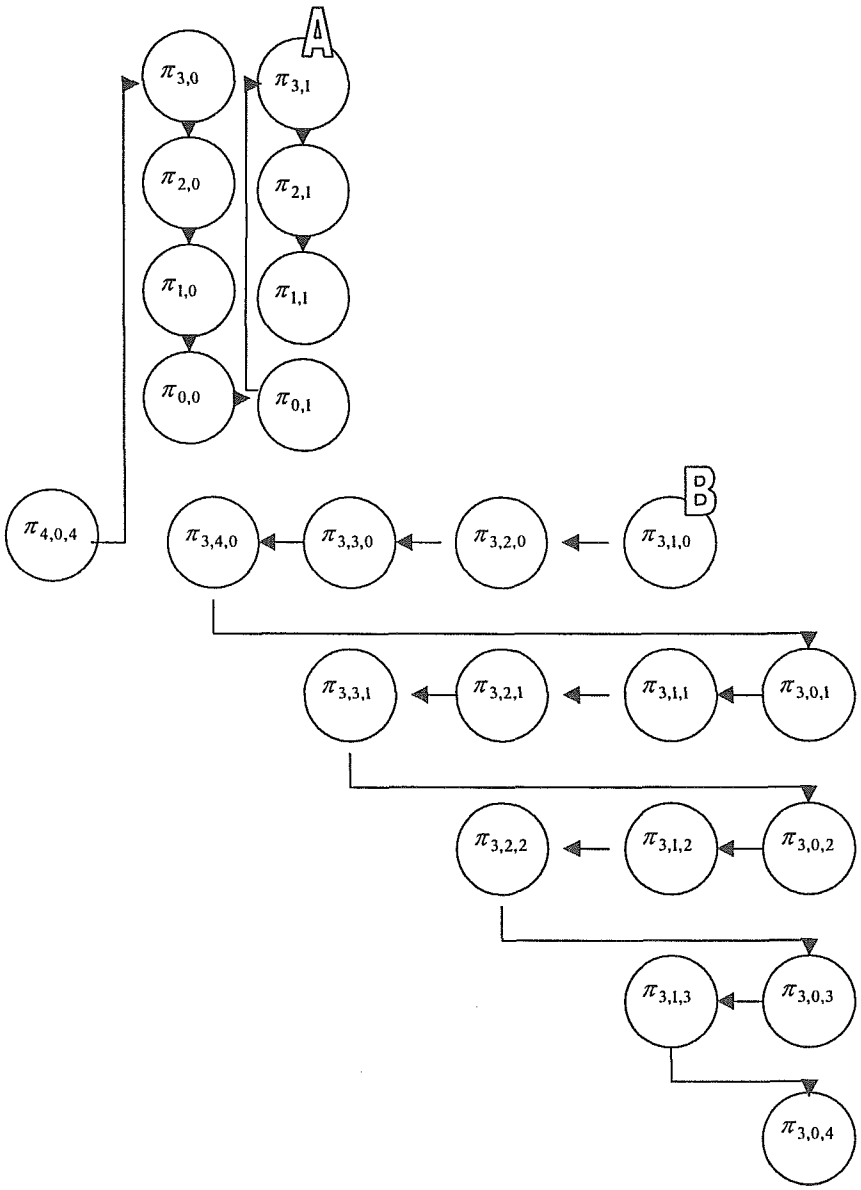

Figure 5.6: $\pi_{i,0}$ and a_3 stage

It should be noted that $\pi_{0,3}$ and $\pi_{0,2}$ are already solved. Label this *Result 2.* Next obtain

$$\pi_{0,1,0} = -\frac{\lambda_1}{\mu_2}\pi_{0,3} - \frac{\lambda_1}{\mu_2}\pi_{0,2} + \alpha_{2,1}^{-1}\pi_{1,3} - \rho_2\pi_{1,2} - \frac{\mu_1}{\mu_2}\pi_{1,1,0} - \frac{\mu_1}{\mu_2}\pi_{2,0,1} \tag{5.117}$$

Now substitute *Result 1* and simplify, noting that $\pi_{0,3}, \pi_{0,2}, \pi_{1,2}$ and $\pi_{2,0,1}$ are already solved, and label this *Result 2.* Now $\pi_{0,1,0}$ is in terms of $\pi_{1,1,0}$. Next obtain using state $S = (1, 1, 0)$

$$\pi_{1,1,0} = \alpha_{2,1}\chi_{1,1}\pi_{0,1,0} + \chi_{1,1}\pi_{1,3} + \alpha_{1,1}\pi_{1,2,0} \tag{5.118}$$

Substitute *Result 1* for $\pi_{1,3}$ and simplify, calling this *Result 3.* Now some looping.

$$\begin{aligned}
\pi_{1,4,0} &= \rho_1\pi_{1,3,0} \\
\pi_{1,3,0} &= \alpha_{2,1}\pi_{0,4,0} + \chi_{1,1}\pi_{1,2,0} + \alpha_{1,1}\pi_{1,4,0} \\
\pi_{1,2,0} &= \alpha_{2,1}\pi_{0,3,0} + \chi_{1,1}\pi_{1,1,0} + \alpha_{1,1}\pi_{1,3,0}
\end{aligned}$$

Notice that the term $\pi_{0,j+1,0}$ can be simplified in each equation.. From this construction, we substitute $\pi_{1,4,0}$ into $\pi_{1,3,0}$, and $\pi_{1,2,0}$. This final substitution will contain $\pi_{1,2,0}$ in terms of $\pi_{1,1,0}$ and $\pi_{0,1,0}$. Label this equation *Result 4.*

Now to bring together the results. Substitute *Result 4* into *Result 3*, then this into *Result 2* ,to obtain the solution for $\pi_{1,2,0}$, $\pi_{1,1,0}$, and $\pi_{0,1,0}$. Now by back substitution solve $\pi_{1,j,0}$, where $j = 1, \ldots, 4$. Then obtain $\pi_{1,3}$. Finally obtain $\pi_{0,4}$ as

$$\pi_{0,4} = \chi_{1,2}^{-1}\pi_{0,1,0}$$

Next, some looping to create a solution using $\pi_{0,0,1}$. So

$$\begin{aligned}
\pi_{0,1,1} &= \chi_{1,2}\pi_{0,0,1} + \chi_{1,2}\pi_{0,1,0} \\
\pi_{0,2,1} &= \chi_{1,2}\pi_{0,1,1} + \chi_{1,2}\pi_{0,2,0} \\
\pi_{0,3,1} &= \chi_{1,2}\pi_{0,2,1} + \chi_{1,2}\pi_{0,3,0}
\end{aligned}$$

Each equation can again be simplified for $\pi_{0,j,0}$. Substitute $\pi_{0,1,1}$ into $\pi_{0,2,1}$, and so on, finally obtaining $\pi_{0,3,1}$ in terms of $\pi_{0,2,1}$. Then back substituting each equation will have the desired $\pi_{0,j,1}$in terms of $\pi_{0,0,1}$. This simplification is needed to assist in obtaining relationships in the next loop. Begin with $j = 3$

$$\pi_{1,3,1} = \rho_1\pi_{1,2,1} + \frac{\lambda_2}{\mu_1}\pi_{1,3,0} \tag{5.119}$$

$$\begin{aligned}
\pi_{1,2,1} &= \alpha_{1,1}\pi_{1,3,1} + \alpha_{1,2}\pi_{0,3,1} + \pi_{1,1,1}\chi_{1,1} + \pi_{1,2,0}\chi_{1,2} \\
\pi_{1,1,1} &= \alpha_{1,1}\pi_{1,2,1} + \alpha_{1,2}\pi_{0,2,1} + \pi_{1,0,1}\chi_{1,1} + \pi_{1,1,0}\chi_{1,2}
\end{aligned}$$

Substitute $\pi_{1,3,1}$ into $\pi_{1,2,1}$, and so on, down to $\pi_{1,1,1}$. In each equation, we can use the simplifications of $\pi_{0,j,1}$in terms of $\pi_{0,0,1}$, and the solutions already obtained for $\pi_{1,j,0}$ to end up with $\pi_{1,1,1}$ in terms of $\pi_{1,0,1}$ and $\pi_{0,0,1}$. Now to draw together these findings. From $S = (1, 0, 1)$

$$\pi_{1,0,1} = \alpha_{1,1}\pi_{1,1,1} + \alpha_{2,1}\pi_{0,1,1} + \chi_{2,1}\pi_{1,3} + \alpha_{1,1}\pi_{2,0,2} \tag{5.120}$$

By substituting for $\pi_{1,1,1}$ using (5.119), and for $\pi_{0,1,1}$ using (5.47), $\pi_{1,0,1}$ is in terms of $\pi_{0,0,1}$. Label this *Result 5*. Now to complete the simplifications,

$$\pi_{0,0,1} = \alpha_{2,2}^{-1}\pi_{0,4} - \rho_2\pi_{0,3} - \frac{\mu_1}{\mu_2}\pi_{1,0,1} \tag{5.121}$$

and substituting in *Result 5* solve $\pi_{1,0,1}$ and then $\pi_{0,0,1}$. Now back substitute to obtain $\pi_{1,j,1}$, where $j = 1, \ldots, 3$ from (5.48) and $\pi_{0,j,1}$ where $j = 1, \ldots, c_2 - 1$ from (5.47).

Using these same principals nested looping is used to solve the remaining vectors. Initialise the outer loop setting $k = 2$ and $j = 1$,

$$\pi_{0,1,2} = \chi_{2,2}\pi_{0,1,1} + \chi_{1,2}\pi_{0,0,2} \tag{5.122}$$

$$\pi_{0,2,2} = \chi_{2,2}\pi_{0,2,1} + \chi_{1,2}\pi_{0,1,2} \tag{5.123}$$

Similarly, start from $j = c_2 - k$ to obtain

$$\pi_{1,2,2} = \pi_{1,2,1}\frac{\lambda_2}{\mu_1} + \pi_{1,1,2}\rho_1 \tag{5.124}$$

$$\pi_{1,1,2} = \alpha_{1,2}\pi_{1,2,2} + \alpha_{2,2}\pi_{0,2,2} + \pi_{1,1,1}\chi_{2,2} \tag{5.125}$$

Next simplify

$$\pi_{1,0,2} = \alpha_{1,2}\pi_{1,1,2} + \alpha_{2,2}\pi_{0,1,2} + \chi_{2,2}\pi_{1,0,1} + \alpha_{1,2}\pi_{2,0,3} \tag{5.126}$$

Using (5.124) substitute $\pi_{1,c_2-k,k}$ into $\pi_{1,c_2-k-1,k}$, and so on, ending up with $\pi_{1,0,k}$ in terms of $\pi_{0,0,k}$. Label this *Result 6*. Substituting *Result 6* into

$$\pi_{0,0,2} = -\rho_2\pi_{0,4} + \alpha_{2,2}^{-1}\pi_{0,0,1} - \frac{\mu_1}{\mu_2}\pi_{1,0,2} \tag{5.127}$$

we obtain $\pi_{0,0,2}$. Consequently, back substitute through (5.122), (5.124) and (5.126) to end up with solutions for $\pi_{1,0,k}, \pi_{1,1,k}, \ldots, \pi_{1,c_2-k,k}$; $\pi_{0,1,k}, \pi_{0,2,k}, \ldots, \pi_{0,c_2-k,k}$. Return to (5.122), add one to k, and stopping when the $k = c_2 - 1$ loop is complete.

Then solve

$$\pi_{0,0,4} = \rho_2\pi_{0,0,3} \tag{5.128}$$

Normalise the probabilities as stated earlier to complete the computational algorithm. The final stage of order of the solutions is given in Figure 5.7.

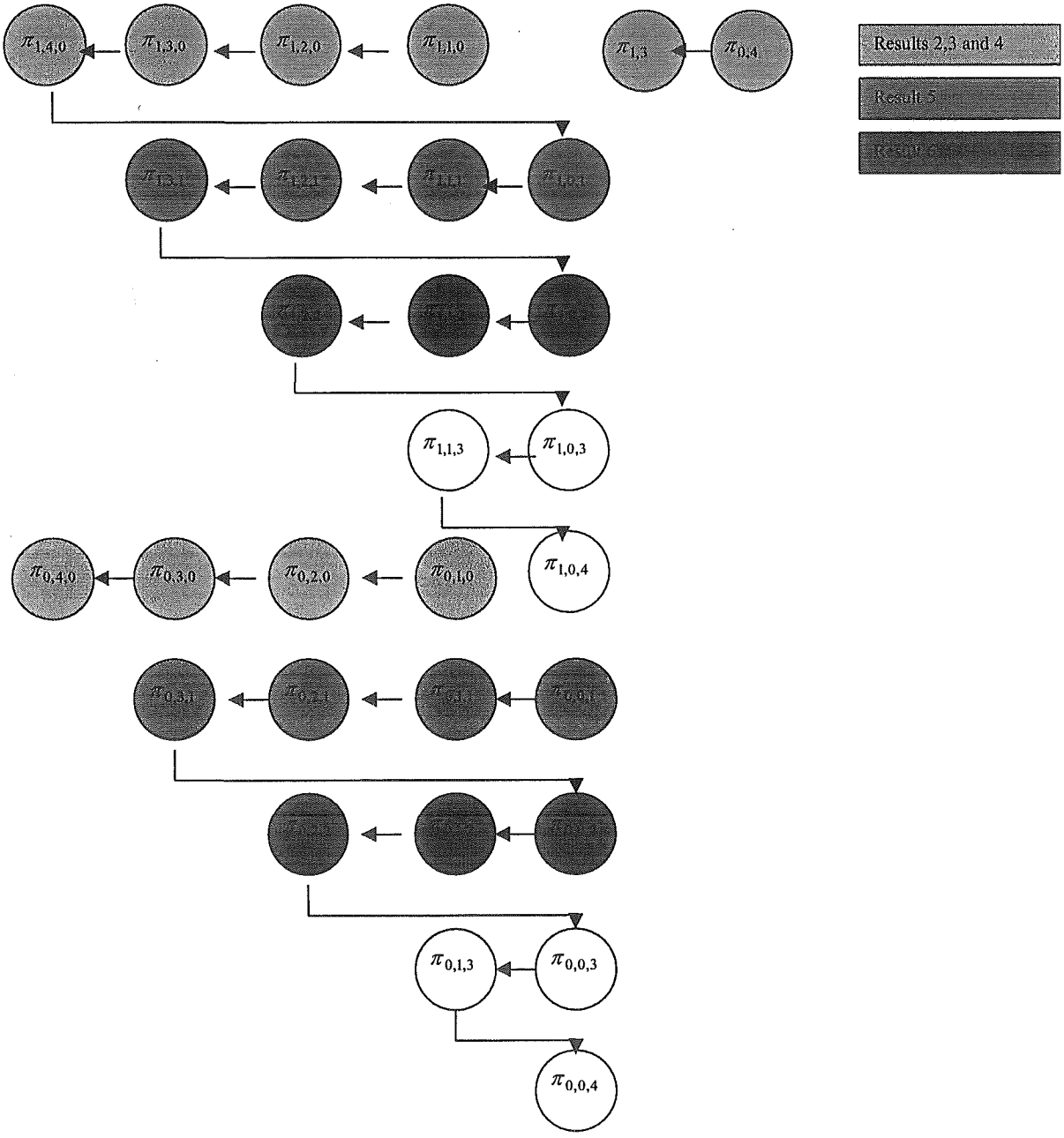

Figure 5.7: Final stage solution order

$\mu_1 = 5, \mu_2 = 4, c_1 = 2; c_2 = 3.$											
	λ_1	λ_2	ρ	π_B	π^1	π^2	$\pi_{Q_2=0}$	π_{loss}	L_s^1	L_s^2	L_s
preMPDQ	.1	.2	.07	.1458	.0167	.1238	.9434	.0020	.17	.47	.41
preSQ				.0700	.0182	.0500		.0000	.04	.10	.15
preMPDQ	1	1.5	.575	.7510	.0433	.4899	.5550	.1001	.51	2.12	2.74
preSQ				.5589	.0766	.3712		.0202	.43	1.25	1.67
preMPDQ	1	2	.7	.8248	.0277	.5606	.4618	.1431	.51	2.59	3.24
preSQ				.6597	.0539	.4829		.0444	.41	1.78	2.19
preMPDQ	2	1.2	.7	.7998	.0855	.3444	.5078	.1217	1.02	1.89	3.03
preSQ				.6604	.1450	.3100		.0380	0.90	1.27	2.17
preMPDQ	4	.5	.925	.8608	.2411	.1366	.4312	.1426	2.17	1.47	3.41
preSQ				.8013	.3442	.1966		.0961	2.06	1.06	3.12
preMPDQ	1	3	.95	.9105	.0121	.6527	.3196	.2334	.49	3.39	4.07
preSQ				.8108	.0256	.6636		.1238	.37	2.88	3.25
preMPDQ	3	1.5	.975	.9019	.0669	.3024	.3456	.2045	1.49	2.24	3.89
preSQ				.8235	.1202	.4000		.1048	1.25	2.03	3.29

Table 5.1: Preemptive MPDQ - Preemptive Single Queue Comparison

Now a look at some specific examples. In Table 5.1 the corresponding statistics for the *single* preemptive queues with the same parameters and capacity are given. Noticeably, the expected number in the system is always greater using the preemptive MPDQ model for this capacity. Loss is also higher. What is reduced is the probability of finding a system with only class 1 customers or class 2 customers. For class 1 customers, we see that there is a lesser chance of finding the system consisting of only class 1 customers under the preemptive MPDQ. This reduction increases markedly as traffic intensity increases.

In Table 5.2 the expected number of customers in the system are given again for the *single* preemptive queues and preemptive MPDQ, this time with a larger system capacity. At low traffic intensity levels, the expected number in the system are almost identical. At higher loads, the expected number in the system for class 2 customers is reduced substantially in comparison to the preemptive single queue model. Further, the overall expected number in the system is also reduced, further reinforcing the conclusion that the MPDQ improves QoS as traffic intensity increases. This was confirmed by a paired t-test, with $p - value = 0.003$ on the hypothesis of equal L_s for the two systems. The trade-off is that loss is again higher for the MPDQ model. We recall that the MPDQ was designed to let through some class 2 customers even in the presence of class 1 customers. The comparison of expected number in the system replicates this design feature.

The Maple code for the performance characteristics are given in Appendix 9.1.2. The solution process in Maple is simpler than by hand as there is no need for back

$\mu_1 = 10, \mu_2 = 10, c_1 = c_2 = 4$											
Model	λ_1	λ_2	ρ	π^1	π^2	π_B	$\pi_{Q_2=0}$	π_{loss}	L_s^1	L_s^2	L_s
preMPDQ	1	2	.3	.0539	.2895	.3979	.9042	.0127	.22	.85	.89
preSQ				.0626	.2000	.3000		.0000	.21	.52	.73
preMPDQ	2	1	.3	.1446	.1290	.3337	.9654	.0034	.46	.41	.51
preSQ				.1519	.1000	.3000		.0000	.45	.28	.73
preMPDQ	2	2	.4	.0993	.2709	.4843	.8935	.0148	.47	.95	1.06
preSQ				.2000	.1156	.4000		.0003	.45	.61	1.06
preMPDQ	2	5	.7	.0323	.5455	.7786	.6785	.0774	.48	2.33	2.74
preSQ				.0457	.4951	.6861		.0140	.43	2.20	2.63
preMPDQ	5	2	.7	.1680	.2115	.7912	.7242	.0328	1.44	1.41	1.96
preSQ				.1917	.2126	.6853		.0114	1.34	1.27	2.61
preMPDQ	5	5	1	.0414	.3799	.9002	.5146	.1195	1.34	2.99	4.12
preSQ				.0480	.4962	.8842		.0738	1.09	3.74	4.83
preMPDQ	7	3	1	.0853	.2226	.8971	.5261	.1094	2.11	2.54	3.96
preSQ				.0968	.3490	.8833		.0661	1.72	3.08	4.80
preMPDQ	3	7	1	.0196	.5595	.8994	.5173	.1427	.74	3.17	4.14
preSQ				.0223	.6511	.8857		.0856	.60	4.29	4.89
preMPDQ	1	9	1	.0057	.7686	.8940	.5372	.1888	.24	3.22	4.01
preSQ				.0061	.8096	.8877		.1017	.18	4.77	4.96
preMPDQ	9	1	1	.2411	.0792	.8916	.5460	.1013	3.41	1.41	3.76
preSQ				.2597	.1994	.8832		.0658	2.73	2.07	4.80

Table 5.2: Preemptive MPDQ - Preemptive Single queue comparison

substitution at each step. Probabilities are assigned (like variables) and the value is immediately updated and stored when calculated.

5.8.2 Convergence

Recall from the single queue model in Section 4.4.5 that the probabilities summed to unity as the size of the system increased, specifically when $c > 5$. The same outcome occurs for the MPDQ model. As seen in Table 5.3, the error level is small in the model where $c_1 = c_2 = 4$ as opposed to $c_1 = 2, c_2 = 3$. Furthermore, as discussed in Section 4.4.5, the error is smaller for models where the classwise traffic intensity is lower for class 1 than class 2. This poses no practical problem in that the design of the MPDQ was for class 2 traffic to arrive with greater frequency than class 1 traffic.

A Wilcoxon Signed Rank test on the tested Models indicated that we can accept the null hypothesis that $a = 1$ (c.f. (4.34)) when $c = (c_1 + c_2) > 4$ at a 5% error level.

$\mu_1 = 10, \mu_2 = 10, c_1 = c_2 = 4$				$\mu_1 = 5, \mu_2 = 4, c_1 = 2; c_2 = 3$			
λ_1	λ_2	ρ	*error*	λ_1	λ_2	ρ	*error*
1	2	.3	.0001	.1	.2	.07	.0000
2	1	.3	.0002	1	1.5	.575	.0038
2	2	.4	.0006	1	2	.7	.0035
2	5	.7	.0011	2	1.2	.7	.0138
5	2	.7	.0054	4	.5	.925	.0597
5	5	1	.0073	1	3	.95	.0025
7	3	1	.0174	3	1.5	.975	.0375
3	7	1	.0023				
1	9	1	.0002				
9	1	1	.0267				

Table 5.3: Convergence error

$\pi_{i,j}$	j	0	1	2
i	0	0.056279	0.054488	0.092901
	1	0.012688	0.028132	–
	2	0.002865	–	–

Table 5.4: Primary Queue probabilities

5.8.3 Waiting Time Distribution

In this section the expected waiting time of each class of customer is obtained for a MPDQ with $c_1 = 2, c_2 = 3, \lambda_1 = 2, \lambda_2 = 3, \mu_1 = 5$ and $\mu_2 = 4$. The following tables (5.4-5.6) contain the stationary distribution over the set of states $S_1 \cup S_2$ for which the performance statistics were displayed in Table 5.1.

First consider the expected waiting time of a class 2 customer. Consider the case when he is able to enter the primary queue. Applying (5.67), his expected waiting

$\pi_{0,i',j'}$	j'	0	1	2	3
i'	0	–	0.127492	0.134123	0.100593
	1	0.020645	0.035213	0.093471	–
	2	0.000000751106	0.009354	–	–
	3	0.000000375553	–	–	–

Table 5.5: Secondary queue with primary queue full of Class 2 customers

$\pi_{1,i',j'}$	j'	0	1	2	3
i'	0	–	0.032536	0.066445	0.039867
	1	0.009183	0.019365	0.038197	–
	2	0.003443	0.009811	–	–
	3	0.000000225454	–	–	–

Table 5.6: Secondary queue with primary queue containing one Class 1 and one Class 2 customer

$\pi_{2,i',j'}$	j'	0	1	2	3
i'	0	–	0.001158	0.000658	0.000395
	1	0.000655	0.000596	0.000621	–
	2	0.000164	0.000337	–	–
	3	0.000065	–	–	–

Table 5.7: Secondary queue with primary queue full of Class 1 customers

time is

$$\begin{aligned} E(W_1^{(2)}) &= \frac{1}{\Pi_{Q_1}}\{\sum_{0\le i+j<2} \pi_{i,j} W^{(1)}_{i,j+1,2-i-j-1}\} \\ &= \frac{1}{\Pi_{Q_1}}\{\pi_{0,0}W^{(1)}_{0,1,1} + \pi_{0,1}W^{(1)}_{0,2,0} + \pi_{1,0}W^{(1)}_{1,1,0}\} \\ &= 0.131\text{unit.} \end{aligned}$$

since $W^{(1)}_{0,1,1} = 0$, and (5.81) and (5.83) give $W^{(1)}_{0,2,0} = \frac{1}{\mu_2}$ and $W^{(1)}_{1,1,0} = \frac{1}{\mu_1}$ respectively.

When the primary queue is full and the customer is able to enter the secondary queue, (5.85) specifies that the following vectors are required in calculating his expected waiting time:

$$W^{(1)}_{0,3,5}, W^{(1)}_{1,2,5}, W^{(1)}_{2,1,5}, W^{(1)}_{0,4,5}, W^{(1)}_{1,3,5}, W^{(1)}_{2,2,5}, W^{(1)}_{0,5,5}, W^{(1)}_{1,4,5} \text{ and } W^{(1)}_{2,3,5}.$$

	Queue 1	Queue 2	Single Queue
Class 1	0.021 (1)	0.201 (9.6)	0.534 (25.4)
Class 2	0.131 (1)	0.735 (5.6)	1.066 (8.1)

Table 5.8: Expected waiting times and scaled values

Applying (5.104) - (5.108) enable the following explicit values for the above vectors to be obtained:

$$\begin{aligned}
W^{(1)}_{0,3,5} &= (0.674817, 0.903093, 1.09927)^t \\
W^{(1)}_{1,2,5} &= (0.614862, 0.84927, 1.09927)^t \\
W^{(1)}_{2,1,5} &= (0.523871, 0.76741, 0.96741)^t \\
W^{(1)}_{0,4,5} &= (0.960723, 1.15713)^t \\
W^{(1)}_{1,3,5} &= (0.907132, 1.10713)^t \\
W^{(1)}_{2,2,5} &= (0.84927, 1.04927)^t \\
W^{(1)}_{0,5,5} &= (1.21072) \\
W^{(1)}_{1,4,5} &= (1.160723) \\
\text{and } W^{(1)}_{2,3,5} &= (1.10713).
\end{aligned}$$

The above gives an expected waiting time of $E(W_2^{(2)}) = 0.735$ units. On average, the expected waiting time in the MPDQ is (refer to (5.86)) 0.652 units.

Next consider the expected waiting time of a class 1 customer. Applying (5.72) when $k = 1$ to (5.60) and (5.63) results in $E(W_1^{(1)}) = 0.021$ units and $E(W_2^{(1)}) = 0.201$ units respectively after some routine calculations. A class 1 customer's expected waiting time in the MPDQ is therefore 0.176 units.

Comparison of the Preemptive Single Finite Buffer Model with the Preemptive MPDQ

Recall from Section 4.4.4 that the expected waiting time for a class 2 customer was 1.066 units. Using a similar calculation to the dual queue case, the expected waiting time of a class 1 customer in the single queue is 0.534 units. Table 5.8 displays all the expected waiting times and scaled values (within brackets) relative to the smallest expected waiting time for each class. It is clear from the figures that the dual queue is superior to the single queue for both classes of customers.

5.9 Conclusion

The analysis of the MPDQ was explicitly detailed. Firstly, the state rate transitions were detailed through matrices and figures, then the system was described in vector-matrix form. A generic solution algorithm was detailed describing the approach to the explicit algorithmic solution undertaken via global balance equations. Four general stages of the algorithm were detailed. The expected waiting times by class are then solved via an algorithm. The analysis was broken down into specific recursive relations based upon the state of the system. It was found that the preemptive MPDQ was superior to the preemptive single queue when traffic intensity was high. Results showed that the discussed notion of fairness was passed on to the lower classed customers. This was apparent when considering the reduced expected number in the system and waiting times in the cited examples. Reference to the Maple code used to solve the algorithm, and convergence of the algorithm was also covered.

Chapter 6

Non-Preemptive Multi-Priority Dual Queue

6.1 Introduction

The non-preemptive dual queue poses the problem of being larger to solve. The customer in service is of importance as they cannot be ejected from service. This leads to more state rate transitions and more complicated matrix structures than those presented in the preemptive case.

6.2 Non-preemptive MPDQ

In Chapter 5 the MPDQ for the preemptive service discipline was analysed. The non-preemptive MPDQ requires a more extensive algorithm as the class of customer in service now needs to be considered. Recall that non-preemptive implies the customer in service cannot be ejected. This can be seen as a 'fairer' service discipline to lower classed traffic as they are not ejected from service in the presence of higher classed customers. The premise of the MPDQ was to allow some lower classed traffic through the system even in the presence of higher classed traffic. This is more achievable in the non-preemptive case due to the no-interrupt to service criteria. In this chapter, the same novel approach used in Chapter 5 will be used to solve for the steady state probabilities.

6.3 State Space

Let c_i = capacity of queue $i, i = 1, 2$. The homogeneous ergodic Markov process describing the non-preemptive MPDQ takes values in a state space S. S can be

partitioned in two distinct ways, corresponding to the case where the secondary queue is empty and when it is not empty respectively. In the first case, define $S_1 = \{0\} \cup \{(i,j,s) : 0 \leq i+j \leq c_1\}$ where the first number i = number of customers of class 1, the second j = number of customers of class 2, and s = class of customer in service. In the second case, define $S_2 = \{(i,i',j',s) : 1 \leq i'+j' \leq c_2, i = 0,1,\ldots,c_1\}$ where i' = number of customers of class 1 and j' = the number customers of class 2 in the secondary queue. As the primary queue is full, there is no need to state the number of class 2 customers as this is simply given by $c_1 - i$. The state space is therefore given by $S = S_1 \cup S_2$.

The steady-state distribution will be represented as a vector dependent upon the size of each of the two queues. The labelling of the members of the vector is undertaken according to the following lexicographical ordering:

$$\mathbf{i,s} = \{(i,0,s), (i,1,s), \ldots, (i,c_1-i,s)\}$$
$$\mathbf{i,0,s} = \{(i,0,1,s)\,(i,0,2,s), \ldots, (i,0,c_2,s)\}$$

for $i = 0,1,2,\ldots,c_1$ and

$$\mathbf{i,j,s} = \{(i,i',0,s), (i,i',1,s), \ldots, (i,i',c_2-i',s)\}$$

for $i' = 0,1,2,\ldots,c_2$; $s = 1,2$

The vector is constructed so that its components are ordered using the above labelling scheme, which gives us

$$\overline{\pi}^t = (\overline{\pi}^t_{\mathbf{0,1}}, \overline{\pi}^t_{\mathbf{0,0,1}}, \ldots, \overline{\pi}^t_{\mathbf{0,c_2,1}}, \overline{\pi}^t_{\mathbf{1,1}}, \overline{\pi}^t_{\mathbf{1,0,1}}, \ldots, \overline{\pi}^t_{\mathbf{1,c_2,1}}, \ldots, \overline{\pi}^t_{\mathbf{c_1,1}}, \overline{\pi}^t_{\mathbf{c_1,0,1}}, \ldots, \overline{\pi}^t_{\mathbf{c_1,c_2,1}},$$
$$\overline{\pi}^t_{\mathbf{0,2}}, \overline{\pi}^t_{\mathbf{0,0,2}}, \ldots, \overline{\pi}^t_{\mathbf{0,c_2,2}}, \overline{\pi}^t_{\mathbf{1,2}}, \overline{\pi}^t_{\mathbf{1,0,2}}, \ldots, \overline{\pi}^t_{\mathbf{1,c_2,2}}, \ldots, \overline{\pi}^t_{\mathbf{c_1,2}}, \overline{\pi}^t_{\mathbf{c_1,0,2}}, \ldots, \overline{\pi}^t_{\mathbf{c_1,c_2,2}})$$

where $\overline{\pi}^t \in \Re^{(c_1+1)(c_2^2+3c_2+c_1+2)}$.

The components of the above steady-state distribution can be described as follows: $\overline{\pi}_{\mathbf{i,s}}$ is the probability vector that is defined when there are no customers present in the secondary queue, whereas $\overline{\pi}_{\mathbf{i,j,s}}$ is the probability vector that is defined when there are customers present in the secondary queue (so the primary queue is full). Any probability that is a component of the first type of vector has general form $\pi_{i,j,s}$ and any probability that is a component of the second type of vector has general form $\pi_{i,i',j',s}$. Thus, for the primary queue

$$\overline{\pi}_{\mathbf{i,s}} = (\pi_{i,0,s}, \pi_{i,1,s}, \ldots \pi_{i,c_1-i,s})^t \in \Re^{c_1-i+1}, i \in \{0,\ldots,c_1\}, s \in \{1,2\}$$

with special case

$$\overline{\pi}_{\mathbf{0,1}} = (\pi_{0,0,0}, \pi_{0,0,1}, \ldots, \pi_{0,c_1,s})^t \in \Re^{c_1+2}$$

and, for the secondary queue when $j = 0$,

$$\overline{\pi}_{\mathbf{i,0,s}} = (\pi_{i,0,1,s}, \pi_{i,0,2,s}, \ldots, \pi_{i,0,c_2,s})^t \in \Re^{c_2}, i \in \{0, \ldots, c_1\}, s \in \{1,2\}$$

and when $j > 0$

$$\overline{\pi}_{\mathbf{i,j,s}} = (\pi_{i,j,0,s}, \pi_{i,j,1,s}, \ldots \pi_{i,j,c_2-j,s})^t \in \Re^{c_2-j+1}, i \in \{0, \ldots, c_1\}, j \in \{1, \ldots, c_2\}, s \in \{1,2\}.$$

6.3.1 Infinitesimal Generator Matrix A

The steady-state distribution $\overline{\pi}^t$ exists and is obtained by solving the Chapman-Kolmogorov system of equations

$$\overline{\pi}^t \mathbf{A} = \mathbf{0} \tag{6.1}$$

where 0 is the vector of zeroes of an appropriate dimension and A is the infinitesimal generator matrix of the process.

A is a square matrix of dimension

$$\left((c_1+1)\left(c_2^2+3c_2+c_1+2\right)+1\right) \times \left((c_1+1)\left(c_2^2+3c_2+c_1+2\right)+1\right) \tag{6.2}$$

A primary queue of capacity 4 and secondary of capacity 6 will generate the matrix A with dimension 301×301. From the description of the dual queueing system given, A can be partitioned into submatrices whose detailed structure is described in detail below.

$$\mathbf{A} = \begin{pmatrix}
\Lambda_{0,1} & \Pi_{0,1} & & & & & & & & \\
\Omega_{1,1} & \Lambda_{1,1} & \Pi_{1,1} & & & & & & & \\
 & \ddots & \ddots & \ddots & & & & & & \\
 & & \Omega_{c_1-1,1} & \Lambda_{c_1-1,1} & \Pi_{c_1-1,1} & & & & & \\
 & & & \Omega_{c_1,1} & \Lambda_{c_1,1} & & & & & \\
\Gamma_0 & & & & & \Lambda_{0,2} & \Sigma_{0,2} & & & \\
\Omega_{1,2} & \Sigma_{1,2} & & & & & \ddots & \Pi_{1,2} & & \\
 & \ddots & \ddots & & & & & \ddots & & \\
 & & \Omega_{c_1-1,2} & \Sigma_{c_1-1,2} & & & & & \Lambda_{c_1-1,2} & \Pi_{c_1-1,2} \\
 & & & \Omega_{c_1,2} & \Sigma_{c_1,2} & & & & & \Lambda_{c_1,2}
\end{pmatrix} \tag{6.3}$$

where

$$\Lambda_{0,s} = \begin{pmatrix} \mathbf{D}_{0,s} & \mathbf{M}_{0,\lambda_2,s} & \mathbf{M}_{0,\lambda_1,s} & 0 & \cdots & 0 \\ 1_{\{s=2\}}\mathbf{O}_{0,\mu_2} & \mathbf{D}_{0,0,s} & \mathbf{Q}_{0,\lambda_1,s} & 0 & & \vdots \\ 0 & 0 & \mathbf{D}_{0,1,s} & \mathbf{R}_{0,1,\lambda_1,s} & \ddots & \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & \mathbf{R}_{0,c_2-1,\lambda_1,s} \\ 0 & & \cdots & & 0 & \mathbf{D}_{0,c_2,s} \end{pmatrix}$$

for $s = 1, 2$.

$$\Pi_{i,s} = \begin{pmatrix} \mathbf{N}_{i,s} & 0 & \cdots & 0 \\ 0 & 0 & & \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & 0 \end{pmatrix}$$

for $i = 1, 2, \ldots, c_1, s = 1, 2$.

$$\Phi_{0,1} = \begin{pmatrix} \mathbf{J}_0 & 0 & \cdots & \cdots & 0 \\ \mathbf{O}_{0,\mu_s} & \mathbf{K}_0 & 0 & & \\ 0 & \cdots & 0 & & \vdots \\ \vdots & & \vdots & & \\ 0 & \cdots & 0 & \cdots & 0 \end{pmatrix}$$

$$\Sigma_{i,s} = \begin{pmatrix} 0 & & \cdots & & & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & & & & \vdots \\ \mathbf{S}_{i,\mu_s,s} & \mathbf{U}_{i,j,s,\mu_s} & \ddots & & & \vdots \\ 0 & 0 & \mathbf{U}_{i,j,s,\mu_s} & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & \mathbf{U}_{i,j,s,\mu_s} & 0 \end{pmatrix}$$

where $s = 1$; $i = 1, ..., c_2$ with special case

$$\Sigma_{0,1} = \begin{pmatrix} 0 & & \cdots & & & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & & & & \vdots \\ \mathbf{S}_{0,\mu_1,1} & \mathbf{U}_{0,j,1,\mu_1} & \ddots & & & \vdots \\ 0 & 0 & \mathbf{U}_{i,j,1,\mu_1} & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & \mathbf{U}_{i,j,1,\mu_1} & 0 \end{pmatrix}$$

$$\Omega_{i,s} = \begin{pmatrix} \mathbf{P}_{i,\mu_s,s} & 0 & \cdots & 0 \\ \mathbf{Y}_{i,\mu_s,s} & \mathbf{T}_{i,\mu_s,s} & 0 & \\ 0 & 0 & 0 & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & 0 \end{pmatrix}$$

where $i = 1, ..., c_1$; $s = 1, 2$.

$$\Lambda_{i,s} = \begin{pmatrix} \mathbf{D}_{i,s} & \mathbf{M}_{i,\lambda_2,s} & \mathbf{M}_{i,\lambda_1,s} & 0 & \cdots & 0 \\ 0 & \mathbf{D}_{i,0,s} & \mathbf{Q}_{i,\lambda_1,s} & 0 & & 0 \\ \mathbf{S}_{i,\mu_1,s} & \mathbf{U}_{i,1,s} & \mathbf{D}_{i,1,s} & \mathbf{R}_{i,1,\lambda_1,s} & \ddots & 0 \\ 0 & 0 & \mathbf{U}_{i,2,s} & \mathbf{D}_{i,1,s} & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \mathbf{R}_{i,c_2-1,\lambda_1,s} \\ 0 & \cdots & & 0 & \mathbf{U}_{i,c_2,s} & \mathbf{D}_{i,c_2,s} \end{pmatrix}$$

where $s = 1, i = 1, \ldots, c_1$

$$\Lambda_{i,s} = \begin{pmatrix} \mathbf{D}_{i,s} & \mathbf{M}_{i,\lambda_2,s} & \mathbf{M}_{i,\lambda_1,s} & 0 & \cdots & 0 \\ 0 & \mathbf{D}_{i,0,s} & \mathbf{Q}_{i,\lambda_1,s} & 0 & & \vdots \\ & & \mathbf{D}_{i,1,s} & \mathbf{R}_{i,1,\lambda_1,s} & \ddots & \\ \vdots & & \ddots & \mathbf{D}_{i,1,s} & \ddots & 0 \\ & & & & \ddots & \mathbf{R}_{i,c_2-1,\lambda_1,s} \\ 0 & & \cdots & & 0 & \mathbf{D}_{i,c_2,s} \end{pmatrix}$$

where $s = 2, i = 1, \ldots, c_1$.

All blank areas in the above submatrices are zeroes. Note that as the dual queues may have a combination of mixed odd and even capacities, some of the submatrices are not square matrices and hence non-invertible. Also, some of the submatrices contain only one non-zero element while others are in the familiar tridiagonal form.

Incorporating all the state transitions generated by the arrival and departure patterns together with the structures of the MPDQ system, we will describe the above submatrices of **A** in full detail and fill in the entries of each submatrix with the appropriate rate parameters λ_i and $\mu_i, i = 1, 2$.

State Transitions and Detailed Structures of Submatrices

The following detailed descriptions of the submatrices of **A** are obtained from the assumptions of the MPDQ and they reflect the arrivals to and departures from (and within) the primary and secondary queues. To begin with, the **D** matrices will be defined. To distinguish between the two queues, the matrices, where required, shall be subscripted corresponding to the customer in service in the same manner as the probability vectors, from $S = S_1 \cup S_2$. For readability, the customer in service subscript shall be *emphasized.*

$$\mathbf{D}_{0,1} = \begin{pmatrix} -\lambda & \lambda_1 & 0 & \cdots & 0 \\ \mu_1 & -(\mu_1+\lambda) & \lambda_2 & & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \lambda_2 \\ 0 & 0 & \cdots & 0 & -(\mu_1+\lambda) \end{pmatrix}$$

where $\mathbf{D}_{0,1} \in \Re^{(c_1+2)\times(c_1+2)}$

$$\mathbf{D}_{0,2} = \begin{pmatrix} -(\mu_2+\lambda) & \lambda_2 & 0 & \cdots & 0 \\ \mu_2 & -(\mu_2+\lambda) & \lambda_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \lambda_2 \\ 0 & \cdots & 0 & \mu_2 & -(\mu_2+\lambda) \end{pmatrix}$$

where $\mathbf{D}_{0,2} \in \Re^{(c_1+1)\times(c_1+1)}$

$$\mathbf{D}_{i,s} = \begin{pmatrix} -(\mu_s+\lambda) & \lambda_2 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ & & \ddots & \ddots & \lambda_2 \\ 0 & \cdots & & 0 & -(\mu_s+\lambda) \end{pmatrix}$$

where $\mathbf{D}_{i,s} \in \Re^{(c_1-i+1)\times(c_1-i+1)}, i = 1,2,\ldots,c_1, s = 1,2.$

$$\mathbf{D}_{0,0,s} = \begin{pmatrix} -(\mu_1+\lambda) & \lambda_2 & 0 & & 0 \\ 1_{\{s=2\}}\mu_2 & \ddots & \ddots & & \\ 0 & \ddots & \ddots & \lambda_2 & 0 \\ & & \ddots & -(\mu_1+\lambda) & \lambda_2 \\ 0 & & 0 & 1_{\{s=2\}}\mu_2 & -\mu_1 \end{pmatrix}$$

where $\mathbf{D}_{0,0,s} \in \Re^{(c_2)\times(c_2)}, s = 1,2.$

$$\mathbf{D}_{i,0,s} = \begin{pmatrix} -(\mu_s+\lambda) & \lambda_2 & 0 & & 0 \\ 0 & \ddots & \ddots & & \\ & & & & 0 \\ & & & -(\mu_s+\lambda) & \lambda_2 \\ 0 & & & & -\mu_s \end{pmatrix}$$

where $\mathbf{D}_{i,0,s} \in \Re^{(c_2)\times(c_2)}$, $i = 1,2,\ldots,c_1, s = 1,2$ and $\mathbf{D}_{c_1,0,s} = (-\mu_s)$

$$\mathbf{D}_{0,j,s} = \begin{pmatrix} -(\mu_s+\lambda) & \lambda_2 & & \\ & \ddots & \ddots & \\ & & -(\mu_s+\lambda) & \lambda_2 \\ & & & -\mu_s \end{pmatrix}$$

where $\mathbf{D}_{0,j,s} \in \Re^{(c_2-j+1)\times(c_2-j+1)}$, $j = 1,2,\ldots,c_2, s = 1,2$ and $\mathbf{D}_{0,c_2,s} = (-\mu_s)$

$$\mathbf{D}_{i,j,s} = \begin{pmatrix} -(\mu_s+\lambda) & \lambda_2 & & \\ & \ddots & \ddots & \\ & & -(\mu_s+\lambda) & \lambda_2 \\ & & & -\mu_s \end{pmatrix}$$

where $\mathbf{D}_{i,j,s} \in \Re^{(c_2-j+1)\times(c_2-j+1)}$, $i = 1,2,\ldots,c_1, j = 1,2,\ldots,c_2$ and $\mathbf{D}_{c_1,c_2,s} = (-\mu_s)$. It is easy to check that the matrix $\mathbf{D}_{i,s}$ is the infinitesimal generator corresponding to transitions between states in **is** whereas $\mathbf{D}_{i,j,s}$ is the infinitesimal generator corresponding to transitions between states in **ijs**.

Next, consider all the off-diagonal submatrices. In the following, each of these submatrices are displayed. The transitions and corresponding instantaneous rates giving rise to the submatrix are summarised in the accompanying tables:

$$\mathbf{M}_{i,\lambda_k,s} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ \lambda_k & 0 & \cdots & 0 \end{pmatrix} \in \Re^{\left(c_1-i+1+1_{\{i=0\cap s=1\}}\right)\times(c_2)}$$

From	To	Rate
$(0, c_1, s)$	$(0, 1, 0, s)$	λ_1
$(0, c_1, s)$	$(0, 0, 1, s)$	λ_2
$(i, c_1 - i, s)$	$(i, 1, 0, s)$	λ_1
$(i, c_1 - i, s)$	$(i, 0, 1, s)$	λ_2

where $i = 1, 2, \ldots, c_1$, $s = 1, 2$ and $k = 1, 2$;

$$\mathbf{N}_{i,s} = \begin{pmatrix} \lambda_1 & & \\ 0 & \ddots & \\ & \ddots & \lambda_1 \\ & & 0 \end{pmatrix} \in \Re^{(c_1 - i + 1) \times (c_1 - i)}$$

From	To	Rate
(i, j, s)	$(i + 1, j, s)$	λ_1

where $i = 1, \ldots, c_1 - 1, j = 0, \ldots, c_1 - i - 1$, $s = 1, 2$; with special case

$$\mathbf{N}_{0,1} = \begin{pmatrix} 0 & \cdots & 0 \\ \lambda_1 & \ddots & \vdots \\ 0 & \ddots & 0 \\ \vdots & \ddots & \lambda_1 \\ 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c_1 + 2) \times (c_1)}$$

From	To	Rate
$(i, j, 1)$	$(i + 1, j, 1)$	λ_1

where $i = 0, j = 0, \ldots, c_1 - 1; s = 1$

$$\mathbf{P}_{i,\mu_s,s} = \begin{pmatrix} \mu_s & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mu_s & 0 \end{pmatrix} \in \Re^{(c_1 - i + 1) \times (c_1 - i + 2)}$$

From	To	Rate
(i,j,s)	$(i-1,j,s)$	μ_s

where $i=1,\ldots,c_1, j=0,\ldots,c_1-i$ and $s=1,2$. Consider the special cases

$$\mathbf{P}_{i,\mu_s,s}=\begin{pmatrix} 0 & \mu_s & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu_s & 0 \end{pmatrix} \in \Re^{(c_1)\times(c_1+2)}$$

From	To	Rate
(i,j,s)	$(i-1,j,s-1_{\{s=2\}})$	μ_s

where $i=1, j=0,\ldots,c_1-i$ and $s=1,2$, and

$$\mathbf{P}_{0,\mu_2,2}=\begin{pmatrix} \mu_2 & 0 & 0 \\ 0 & 0 & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c_1+1)\times(c_1+2)}$$

From	To	Rate
$(0,0,2)$	$(0,0,0)$	μ_2

$$\mathbf{Y}_{i,\mu_s,s}=\begin{pmatrix} 0 & 0 & \mu_s \\ \vdots & \ddots & 0 \\ 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c_2)\times\left(c_1-i+2+1_{\{i=0\}}\right)}$$

From	To	Rate
$(i,0,j',s)$	$(i-1,c_1-i+1,s)$	μ_s

where $i=1,\ldots,c_1, j'=1,\ldots,c_2$ and $s=1,2$.

$$\mathbf{T}_{i,\mu_s,s}=\begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \mu_s & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mu_s & 0 \end{pmatrix} \in \Re^{(c_2)\times(c_2)}$$

From	To	Rate
$(i,0,j',s)$	$(i-1,0,j'-1,s)$	μ_s

where $i=1,\ldots,c_1, j'=1,\ldots,c_2$ and $s=1,2$.

$$\mathbf{S}_{i,\mu_c,s}=\begin{pmatrix}0 & 0 & \mu_s\\ \vdots & 0 & 0\\ 0 & \cdots & 0\end{pmatrix}\in\Re^{(c_2)\times(c_1-i+1-1_{\{i=0\}})}$$

From	To	Rate
$(i,1,0,s)$	$(i,c_1-i,s-1_{\{s=2\}})$	μ_s
$(0,1,0,s)$	$(1,c_1-1,s+1_{\{s=1\}})$	μ_s

where $i=1,\ldots,c_1, s=1,2$.

$$\mathbf{U}_{i,j,s,\mu_s}=\begin{pmatrix}0 & \cdots & \cdots & 0\\ \mu_s & \ddots & & \vdots\\ 0 & \ddots & \ddots & \vdots\\ 0 & 0 & \mu_s & 0\end{pmatrix}\in\Re^{(c_2-j+1)\times(\min\{c_2,c_2-j+2\})}$$

From	To	Rate
$(0,i',j',s)$	$(1,i'-1,j'+1,s)$	μ_s
(i,i',j',s)	$(i,i'-1,j',s)$	μ_s

where $i=0,\ldots,c_1, i'=1,\ldots,c_2, j'=0,\ldots,c_2-j, s=1,2$.

$$\mathbf{O}_{0,\mu_s}=\begin{pmatrix}0 & 0 & \mu_s\\ \vdots & 0 & 0\\ 0 & \cdots & 0\end{pmatrix}\in\Re^{(c_2)\times(c_1+1)}$$

From	To	Rate
$(0,0,1,s)$	$(0,c_1,s)$	μ_s

$$\mathbf{Q}_i = \begin{pmatrix} 0 & \lambda_1 & 0 & 0 \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \lambda_1 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \in \Re^{(c_2)\times(c_2)}, \text{where } \mathbf{Q}_0 = \begin{pmatrix} 0 & \lambda_1 \\ 0 & 0 \end{pmatrix}$$

$$\text{for } c_2 = 2 \text{ and } \mathbf{Q}_0 = \begin{pmatrix} 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_1 \\ 0 & 0 & 0 \end{pmatrix} \text{ for } c_2 = 3$$

From	To	Rate
$(i, 0, j', s)$	$(i, 1, j', s)$	λ_1

where $i = 0, \ldots, c_1$, $j' = 0, \ldots, c_2 - 1$.

Similarly,

$$\mathbf{R}_{i,j,\lambda_1} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ \vdots & \ddots & \lambda_1 \\ 0 & \cdots & 0 \end{pmatrix} \in \Re^{(c_2-j+1)\times(c_2-j)}, \text{where } \mathbf{R}_{1,j} = \begin{pmatrix} \lambda_1 \\ 0 \end{pmatrix},$$

$$\text{for } c_2 = 2, \text{and } \mathbf{R}_{1,j} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_1 \\ 0 & 0 \end{pmatrix} \text{ for } c_2 = 3$$

From	To	Rate
(i, i', j', s)	$(i, i'+1, j', s)$	λ_1

where $i = 0, \ldots, c_1$, $i' = 1, \ldots, c_2 - 1, j' = 0, \ldots, c_2 - j - 1$.

Two new matrices are required which were not required in the preemptive design.

$$J_0 = \begin{pmatrix} \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & & \vdots \\ \mu_1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mu_1 & 0 \end{pmatrix} \in \Re^{(c_1+2)\times(c_1+1)}$$

From	To	Rate
$(0,0,0)$	$(0,0,2)$	λ_2
$(0,j,1)$	$(0,j-1,2)$	μ_1

where $j = 0, \ldots, c_1$ and finally

$$K_0 = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \mu_1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \mu_1 & 0 \end{pmatrix} \in \Re^{(c_2)\times(c_2)}$$

From	To	Rate
$(0,0,j',1)$	$(0,0,j'-1,2)$	μ_1

6.4 Linear System of Steady State Distributions

As is apparent from the previous section, the structure of the infinitesimal generator for the non-preemptive case is more complicated than the preemptive case. Let

$$\boldsymbol{\alpha}_{i,s}^t = \left(\overline{\pi}_{\mathbf{i,s}}^t, \overline{\pi}_{\mathbf{i,0,s}}^t, \overline{\pi}_{\mathbf{i,1,s}}^t, \ldots, \overline{\pi}_{\mathbf{i,c_2,s}}^t\right)$$

where $i = 0, \ldots, c_1, s = 1, 2$. Therefore $\overline{\pi}^t = \left(\boldsymbol{\alpha}_{0,1}^t, \boldsymbol{\alpha}_{1,1}^t, \ldots, \boldsymbol{\alpha}_{c_1,1}^t, \boldsymbol{\alpha}_{0,2}^t, \boldsymbol{\alpha}_{1,2}^t, \ldots, \boldsymbol{\alpha}_{c_1,2}^t\right)$ and the Chapman-Kolmogorov system of equations can be described by

$$\left(\boldsymbol{\alpha}_{0,1}^t, \boldsymbol{\alpha}_{1,1}^t, \ldots, \boldsymbol{\alpha}_{c_1,1}^t, \boldsymbol{\alpha}_{0,2}^t, \boldsymbol{\alpha}_{1,2}^t, \ldots, \boldsymbol{\alpha}_{c_1,2}^t\right) \mathbf{A} = \mathbf{0}.$$

Expanding the above, the following system of linear equations is derived:

For $s = 1$,

$$\begin{aligned} &\boldsymbol{\alpha}_{i,1}^t \Lambda_{i,1} + 1_{\{i<c_1\}} \left(\boldsymbol{\alpha}_{i+1,1}^t \Omega_{i+1,1} + \boldsymbol{\alpha}_{i+1,2}^t \Omega_{i+1,2}\right) + \\ &1_{\{i=0\}} \boldsymbol{\alpha}_{0,2}^t \Gamma_0 + 1_{\{i\geq 1\}} \left(\boldsymbol{\alpha}_{i-1,1}^t \Pi_{i-1,1} + \boldsymbol{\alpha}_{i,2}^t \Sigma_{i,2}\right) = 0 \end{aligned}$$

and for $s = 2$,

$$1_{\{i=0\}} \boldsymbol{\alpha}_{0,1}^t \Phi_{0,1} + \boldsymbol{\alpha}_{i,2}^t \Lambda_{i,2} + 1_{\{i=1\}} \left(\boldsymbol{\alpha}_{0,1}^t \Sigma_{0,1} + \boldsymbol{\alpha}_{0,2}^t \Sigma_{0,2}\right) + 1_{\{i\geq 2\}} \boldsymbol{\alpha}_{i,2}^t \Pi_{i-1,2} = 0$$

The complexity of the system is evident when considering some examples of the system. For $i = 0, s = 1$

$$\boldsymbol{\alpha}_{0,1}^t \Lambda_{i,1} + \boldsymbol{\alpha}_{1,1}^t \Omega_{1,1} + \boldsymbol{\alpha}_{1,2}^t \Omega_{1,2} + \boldsymbol{\alpha}_{0,2}^t \Gamma_0 = 0$$

so in terms of the submatrices

$$\begin{aligned}
\overline{\pi}^t_{0,1}\mathrm{D}_{0,1} + \overline{\pi}^t_{1,1}\mathrm{P}_{1,\mu_1,1} + \overline{\pi}^t_{1,0,1}\mathrm{Y}_{1,\mu_1,1} + \overline{\pi}^t_{0,2}\mathrm{P}_{0,\mu_2,2} + \overline{\pi}^t_{1,2}\mathrm{P}_{1,\mu_2,2} + \overline{\pi}^t_{1,0,2}\mathrm{Y}_{1,\mu_2,2} &= 0 \\
\overline{\pi}^t_{0,0,1}\mathrm{M}_{0,\lambda_2,2} + \overline{\pi}^t_{0,0,1}\mathrm{D}_{0,0,1} + \overline{\pi}^t_{1,0,1}\mathrm{T}_{1,\mu_1,1} &= 0 \\
\overline{\pi}^t_{0,1}\mathrm{M}_{0,\lambda_1,1} + \overline{\pi}^t_{0,0,1}\mathrm{Q}_0 + \overline{\pi}^t_{0,1,1}\mathrm{D}_{0,1,1} &= 0 \\
\overline{\pi}^t_{0,\mathrm{i},1}\mathrm{R}_{0,i,\lambda_1} + \overline{\pi}^t_{0,\mathrm{i}+1,1}\mathrm{D}_{0,i+1,1} &= 0
\end{aligned}$$

for $i = 1, 2, \ldots, c_2 - 1$. Finally, the system must satisfy the normalisation constraint, that is

$$\overline{\pi}^t\overline{\mathrm{e}} = 1$$

6.5 Computational Algorithm - Non-preemptive MPDQ

The algorithm is larger for the non-preemptive case due to the increased quantity in states of the system. The process of solution follows a similar procedure to that for the preemptive. To reiterate the discussion of the computational process as in Chapter 5, the order of solutions is important as the algorithm is dependent upon the dominant tridiagonal-like structure of system. The process of obtaining the steady state probabilities goes through several stages based upon this structure. There are specific manipulations within each stage that lead to a tractable solution. Much of the algorithm utilises simplifications which, when using a program such as Maple, lead to immediate solutions when solving (as opposed to tedious back-substitution). This section is structured so that each subheading refers to the section of probabilities that are currently being solved for.

6.5.1 $\alpha_{c_1,s}; \alpha_{c_2,s}$ stage

This first stage solves all of the boundary probabilities, that is, those within α_{c_1,c_2}. To commence we define

$$\alpha_{i,j} = \frac{\mu_i}{\mu_j + \lambda}, \chi_{i,j} = \frac{\lambda_i}{\mu_j + \lambda}, \rho_i = \frac{\lambda_i}{\mu_i}, i, j \in \{1, 2\}.$$

The algorithm is initialised by

$$\pi_{c_1,c_2-1,0,2} = 1$$

so we obtain immediately

$$\pi_{c_1,c_2,0,2} = \frac{\lambda_1}{\mu_2}.$$

In this stage, the order of solution begins from probabilities of a full system with a class 2 customer in service. Now we iterate to solve immediately

$$\pi_{c_1,i',0,2} = \chi_{1,2}\pi_{c_1,i'+1,0,2}$$

for $i' = c_2 - 2, \ldots, 1$ and the boundary primary queue probability

$$\pi_{c_1,0,2} = \chi_{1,2}\pi_{c_1,1,0,2}.$$

Next to solve the remaining states within α_{c_1,c_2}. We commence by setting $j' = 1$. This is our outer loop. For the inner loop, we iterate $i' = 0, ..., c_2 - j'$ in

$$\begin{aligned}\pi_{c_1,i',j',2} \quad = \quad & 1_{\{i'=0\}}\left[\chi_{2,2}\left(1_{\{j'=1\}}\pi_{c_1,0,2} + 1_{\{j'>1\}}\pi_{c_1,0,2,2}\right)\right] + \\ & 1_{\{0<i'<c_2-j'\}}\left[\chi_{2,2}\pi_{c_1,i',j'-1,2} + \chi_{1,2}\pi_{c_1,i'-1,j',2}\right] + \\ & 1_{\{i'=c_2-j'\}}\left[\rho_2\pi_{c_1,i',j'-1,2} + \frac{\lambda_1}{\mu_2}\pi_{c_1,i'-1,j',2}\right]. \end{aligned} \tag{6.5}$$

At the completion of the i' loop, we add one to j', and return to (6.5), stopping after the completion of the outer loop $j' = c_2 - 1$. This loop is finished by solving

$$\pi_{c_1,0,c_2,2} = \rho_2\pi_{c_1,0,c_2-1,2}.$$

We solve the probabilities $\pi_{i,0,2}; i \in \{0, ..., c_2 - 1\}$ immediately by iterating $i = c_2 - 1, ..., 0$ in the following:

$$\pi_{i,0,2} = \chi_{1,2}^{-1}\pi_{i+1,0,2}. \tag{6.6}$$

The next order of solutions requires the use of states involving no class 2 customers within the primary or secondary queues with a class 1 customer in service. The solution process starts with the simplification of probabilities within the secondary queue. This is a two-stage process, consisting of two separate simplifications which, when brought together, give the solution of all probabilities in (6.7) to (6.12). First set

$$\pi_{c_1,c_2,0,1} = \rho_1\pi_{c_1,c_2-1,0,1}.$$

We begin simplifying with the aim of obtaining $\pi_{c_1,0,1}$ in terms of $\pi_{c_1,1,0,1}$. So for $i' = c_2 - 1, ..., 2$ obtain

$$\pi_{c_1,i',0,1} = \alpha_{1,1}\pi_{c_1,i'+1,0,1} + \chi_{1,1}\pi_{c_1,i'-1,0,1} + \alpha_{2,1}\pi_{c_1,i'-1,0,2} \tag{6.7}$$

and

$$\pi_{c_1,1,0,1} = \alpha_{1,1}\pi_{c_1,2,0,1} + \chi_{1,1}\pi_{c_1,0,1} + \alpha_{2,1}\pi_{c_1,2,0,2}. \tag{6.8}$$

Notice that the last terms $\pi_{c_1,i',0,2}; i \in \{0, ..., c_2\}$ are already solved. So $\pi_{c_1,1,0,1}$ is left in terms of $\pi_{c_1,0,1}$ - label this result $(\mathbb{A})$. Some simple manipulations using the idle state are now required. From the transitions from state $(0, 0, 0)$ we simplify

$$\pi_{0,0,0} = \frac{\mu_1}{\lambda}\pi_{0,0,1} - \frac{\mu_2}{\lambda}\pi_{0,0,2} \tag{6.9}$$

noting that $\pi_{0,0,2}$ has been solved in (6.6). Next, substitute (6.9) into the balance equation for state $(0,0,1)$

$$\pi_{0,0,1} = \chi_{1,1}\pi_{0,0,0} + \alpha_{1,1}\pi_{1,0,1} + \alpha_{2,1}\pi_{1,0,2} \tag{6.10}$$

where $\pi_{1,0,2}$ has been solved. Therefore (6.10) substituted in (6.9) eliminates $\pi_{0,0,0}$. Now we continue the process, with the aim of obtaining $\pi_{c_1,1,0,1}$ in terms of $\pi_{c_1,0,1}$. So for $i = 1, ..., c_1 - 1$ we simplify

$$\pi_{i,0,1} = \alpha_{1,1}\pi_{i+1,0,1} + \chi_{1,1}\pi_{i-1,0,1} + \alpha_{2,1}\pi_{i+1,0,2} \tag{6.11}$$

by substituting for $\pi_{i-1,0,1}$ at each iteration, noting that $\pi_{i+1,0,2}$ has been solved. Finally

$$\pi_{c_1,0,1} = \alpha_{1,1}\pi_{c_1,1,0,1} + \chi_{1,1}\pi_{c_1-1,0,1} + \alpha_{2,1}\pi_{c_1,1,0,2}. \tag{6.12}$$

We label (6.12) result (𝔹), and by substituting (𝔹) into (𝔸) we obtain the solution of $\pi_{c_1,0,1}$ and $\pi_{c_1,1,0,1}$. Via back substitution from (6.7) to (6.11) we obtain the solutions for $\pi_{i,0,1}, i = 0, ..., c_1; \pi_{c_1,i',0,1}, i' = 1, ..., c_2$; and $\pi_{0,0,0}$.

Now the rest of this stage is simple to solve directly via looping. For the outer loop, set $j' = 1$. For $i' = c_2 - j'$, we solve

$$\pi_{c_1,i',j',1} = \rho_1\pi_{c_1,i'-1,j',1} + \frac{\lambda_2}{\mu_1}\pi_{c_1,i',j'-1,1} \tag{6.13}$$

and for $i' = c_2 - j' - 1, ..., 1$ we solve

$$\pi_{c_1,i',j',1} = \alpha_{1,1}\pi_{c_1,i'+1,j',1} + \chi_{1,1}\pi_{c_1,i'-1,j',1} + \chi_{2,1}\pi_{c_1,i',j'-1,1} + \alpha_{2,1}\pi_{c_1,i'+1,j',2} \tag{6.14}$$

and

$$\pi_{c_1,0,j',1} = \alpha_{1,1}\pi_{c_1,1,j',1} + \chi_{2,1}\left(1_{\{j'=1\}}\pi_{c_1,0,1} + 1_{\{j'>1\}}\pi_{c_1,0,j'-1,1}\right) + \alpha_{2,1}\pi_{c_1,i'+1,j',2}. \tag{6.15}$$

Return to equation (6.13), adding one to j', stopping the looping when the $j' = c_2 - 1$ loop is complete. Finally we solve

$$\pi_{c_1,0,c_2,1} = \frac{\lambda_2}{\mu_1}\pi_{c_1,0,c_2-1,1}. \tag{6.16}$$

This solves the probability vector $\overline{\pi}_{c_1,i',1}$ where $i' = 0, ..., c_2$.

6.5.2 Simplification Stage 1

As the states for which class 2 customers are in service are only interrelated to states also solely with a class 2 customer in service, they can be solved simply. However, some modifications are needed before these solutions can be obtained. All probabilities

within $S \in \{(i^*, c_2, 0, 2), ..., (i^*, 0, 0, 2)\}$ can be simplified in terms of a single probability. At various points later, reference to the outer loop i^* is required so that the order of solution is maintained. i^* is a large outer loop variable which we use to solve the majority of probabilities in the next subsection.

We initialise the outer loop by setting $i^* = c_1 - 1$, and simplify as follows:

Simplification A

$$\pi_{i^*,i',0,2} = (\chi_{1,2})^{i'} \pi_{i^*,c_1-i^*,2} \tag{6.17}$$

where $i' = 1, ..., c_2 - 1$ and

$$\pi_{i^*,c_2,0,2} = \frac{\lambda_2}{\mu_1} (\chi_{1,2})^{c_2-1} \pi_{i^*,c_1-i^*,2}. \tag{6.18}$$

Simplification B

$$\pi_{i^*,0,j',2} = (\chi_{2,2})^{j'} \pi_{i^*,c_1-i^*,2} \tag{6.19}$$

where $j' = 1, ..., c_2 - 1$ and

$$\pi_{i^*,0,c_2,2} = \rho_2 (\chi_{2,2})^{c_2-1} \pi_{i^*,c_1-i^*,2}. \tag{6.20}$$

Using Simplifications A and B, we recalculate the remaining states, simplifying in terms of $\pi_{i^*,c_1-i^*,2}$ at each point. So set $i' = 1$. Modify for $j' = 1, ..., c_2 - i'$

$$\pi_{i^*,i',j',2} = \frac{1}{\mu_2 + 1_{\{i'+j'<c_2\}}\lambda} \left(\lambda_1 \pi_{i^*,i'-1,j',2} + \lambda_2 \pi_{i^*,i',j'-1,2}\right). \tag{6.21}$$

Return to equation (6.21), adding one to i', stopping after $i' = c_2 - 1$ is completed. Finish by establishing

$$\pi_{i^*,i',0,2} = \frac{\lambda_1}{\mu_2} \pi_{i^*,i'-1,0,2}. \tag{6.22}$$

Simplification C

This simplification is used to simplify $\pi_{i^*,c_1-i^*,2}$ down to $\pi_{1,c_1-i^*,2}$ in terms of $\pi_{0,c_1-i^*,2}$. Using the transitions from states $\{(1, i^*, 2), ..., (i^*, i^*, 2)\}$, simplify

$$\pi_{i,c_1-i^*,2} = \chi_{1,2} \pi_{i-1,c_1-i^*,2} + \chi_{2,2} \pi_{i,c_1-i^*-1,2} \tag{6.23}$$

where $i = 1, ..., i^*$ noting that $\pi_{i,c_1-i^*-1,2}$ have been solved. At each stage, substitute $\pi_{i-1,c_1-i^*,2}$ into the previous equation and the desired simplification leaves $\pi_{i^*,c_1-i^*,2}$ down to $\pi_{1,c_1-i^*,2}$ in terms of $\pi_{0,c_1-i^*,2}$.

Simplification D

For later use we simplify

$$\pi_{i^*,c_2,0,1} = \rho_1 \pi_{i^*,c_2-1,0,1}. \tag{6.24}$$

Further, for $i' = c_2 - 1, ...1$,

$$\begin{aligned} \pi_{i^*,i',0,1} &= \alpha_{1,1}\pi_{i^*,i'+1,0,1} + \chi_{1,1}\left(1_{\{i>1\}}\pi_{i^*,i'-1,0,1} + 1_{\{i=1\}}\pi_{i^*,c_1-i^*,1}\right) \\ &\quad +\alpha_{2,1}\pi_{i^*,i'+1,0,2} \end{aligned} \tag{6.25}$$

simplifying where $\pi_{i^*,i'+1,0,2}$ is left in terms of $\pi_{i^*,1,2}$ from Simplification A. At each point back-substitute $\pi_{i^*,i',0,1}$ into the prior equation to obtain $\pi_{i^*,1,0,1}$ in terms of $\pi_{i^*,1,2}$ and $\pi_{i^*,1,1}$.

6.5.3 Solving Stage

With the simplifications made for all probabilities where $i' = c_1 - 1, s = 2$, now all probabilities can be solved where $s = 2$ and $j = 1$ in the primary queue. This is achieved by considering the transitions from states $\{(0,1,2),...,(i^*,1,2)\}$. We commence with the boundary state $(i^*,1,1)$ and simplify probabilities within, as this state has no transitions to any states in the dual queue. This solving stage requires two stages of substitution with the final determination of probability $\pi_{0,1,1}$ the key to the solution. We decompose the system once again by using the strict order of equations. To begin, we consider the transition from state $(i^*, c_1 - i^*, 1)$:

$$\begin{aligned} \pi_{i^*,c_1-i^*,1} &= \chi_{1,1}\pi_{i^*-1,c_1-i^*,1} + \chi_{2,1}\pi_{i^*,c_1-i^*-1,1} + \alpha_{1,1}\left(\pi_{i^*,1,0,1} + \pi_{i^*+1,0,1,1}\right) \\ &\quad +\alpha_{2,1}\left(\pi_{i^*,1,0,2} + \pi_{i^*+1,0,1,2}\right). \end{aligned} \tag{6.26}$$

Note that $\pi_{i^*,c_1-i^*-1,1}, \pi_{i^*+1,0,1,1}$ and $\pi_{i^*,1,0,2}$ are already solved. Some substitution from earlier results is needed. Substituting for $\pi_{i^*,1,0,1}$ leads to $\pi_{i^*,c_1-i^*,1}$ and $\pi_{i^*,c_1-i^*,2}$. For $\pi_{i^*,1,0,2}$ this leads to $\pi_{i^*,1,2}$. These substitutions result in leaving $\pi_{i^*,1,1}$ in terms of $\pi_{i^*-1,1,1}$, and $\pi_{i^*,1,2}$. Using states $\{(i^*-1,1,1),...,(0,1,1)\}$ and simplifying at each point we use the following for $i = i^* - 1, ..., 0$

$$\pi_{i,c_1-i^*,1} = 1_{\{i>0\}}\chi_{1,1}\pi_{i-1,c_1-i^*,1} + \alpha_{1,1}\pi_{i+1,c_1-i^*,1} + \chi_{2,1}\pi_{i,c_1-i^*-1,1} + \alpha_{2,1}\pi_{i+1,c_1-i^*,2} \tag{6.27}$$

where $\pi_{i,c_1-i^*-1,1}$ is solved. At each iteration, substitute $\pi_{i+1,c_1-i^*,1}$ from the last equation and using the simplification for $\pi_{i+1,c_1-i^*,2}$ substitute to obtain (6.27) in terms of $\pi_{0,1,2}$. The final result leaves $\pi_{0,c_1-i^*,1}$ in terms of $\pi_{0,c_1-i^*,2}$. Label this result (𝔸). Now, using state $(0, c_1 - i^* - 1, 2)$ we obtain

$$\pi_{0,c_1-i^*,2} = -\rho_2\left(1_{\{i^*=c_1-1\}}\pi_{0,0,0} + 1_{\{i^*<c_1-1\}}\pi_{0,c_1-i^*-2,2}\right) - \frac{\mu_1}{\mu_2}\pi_{0,c_1-i^*,1} + \alpha_{2,2}\pi_{0,c_1-i^*-1,2} \tag{6.28}$$

where $\pi_{0,c_1-i^*-1,2}$ and $\pi_{0,0,0}$; $\pi_{0,c_1-i^*-2,2}$ are solved. Label this result (𝔹). Substituting (𝔹) into (𝔸) we have the solution for $\pi_{0,c_1-i^*-1,1}$. Then obtain $\pi_{0,c_1-i^*,2}$ from (𝔹). Via back substitution we obtain the solution for $\pi_{i,1,1}$ where $i = 1, ..., i^*$ from (6.27) and (6.26) . Going back through the simplification stage the solution is also obtained for and $\pi_{i,1,2}$ where $i = 1, ..., c_1 - 1$ from (6.23)(Simplification C); $\pi_{c_1-1,i',0,1}$ where $i' = 1, ..., c_2$ from (6.24) and (6.25)(Simplification D); $\pi_{c_1-1,i',0,2}$ where $i' = 1, ..., c_2$ from (6.17) and (6.18)(Simplification A); and $\pi_{c_1-1,i',j',2}$ where $i' = 1, ..., c_2; j = 1, ..., c_2 - i'$ from (6.19) and (6.20) and (6.21)(Simplification B).

Further states can now also be solved using this simple loop: Let $j' = 1$, and looping from $i' = c_2 - j'$ we have

$$\pi_{i^*,i',j',1} = \rho_1 \pi_{i^*,i'-1,j',1} + \frac{\lambda_2}{\mu_1} \pi_{i^*,i',j'-1,1} \tag{6.29}$$

where $\pi_{i^*,i',j'-1,1}$ is solved. Now for $i' = c_2 - 2, ..., 1$

$$\pi_{i^*,i',j',1} = \chi_{1,1}\pi_{i^*,i'-1,j',1} + \chi_{2,1}\pi_{i^*,i',j'-1,1} + \alpha_{1,1}\pi_{i^*,i'+1,j',1} + \alpha_{2,1}\pi_{i^*,i'+1,j',2} \tag{6.30}$$

where $\pi_{i^*,i',j'-1,1}$ and $\pi_{i^*,i'+1,j',2}$ are solved, and for $i' = 0$

$$\begin{aligned} \pi_{i^*,0,j',1} \;=\; & \chi_{2,1}\left(1_{\{j'>1\}}\pi_{i^*,i',j'-1,1} + 1_{\{j'=1\}}\pi_{i^*,c_1-i^*,1}\right) + \alpha_{1,1}\left(\pi_{i^*,i'+1,j',1} + \pi_{i^*+1,i',j'+1,1}\right) \\ & + \alpha_{2,1}\left(\pi_{i^*,i'+1,j',2} + \pi_{i^*+1,i',j'+1,2}\right) \end{aligned} \tag{6.31}$$

where all probabilities are solved with the exception of $\pi_{i^*,i',j'-1,1}$. By back substitution, all $\pi_{i^*,i',j'-1,1}$, where $i' = 0, ..., c_2 - 1$ are solved. Return to (6.29) adding one to j', stopping after the completion of $j' = c_2 - 1$. To finish for $i' = 0, j' = c_2$

$$\pi_{i^*,i',j',1} = \frac{\lambda_2}{\mu_1} \pi_{i^*,i',j'-1,1}. \tag{6.32}$$

This process solves the majority of probabilities. Add one to i^* and return to Section 6.5.2, stopping after $i^* = 2$ is complete. All that remains to be solved are the probabilities within $\boldsymbol{\alpha}_{0,s}$ and $\boldsymbol{\alpha}_{1,s}$.

6.5.4 Simplification Stage $\boldsymbol{\alpha}_{0,s}; \boldsymbol{\alpha}_{1,s}$

The remaining solution process requires substantially more simplifications than in the prior section. As there are more interdependencies between the four remaining transitions associated with the probability vectors $\boldsymbol{\alpha}_{1,1}, \boldsymbol{\alpha}_{0,1}$, $\boldsymbol{\alpha}_{0,2}$ and $\boldsymbol{\alpha}_{1,2}$, than for the preemptive model, more simplifications are required before arriving at a solution. For the preemptive model, four equations and four unknowns were required to obtain an initial solution. The process here also ends in four equations with four unknowns, however getting there requires far more work.

The first simplifications involve probabilities with a class 1 customer in service, followed by a class 2 customer in service. The order is again important as early simplifications further simplify later relations.

Simplification E

The first simplification leaves $\pi_{0,i',0,1}$ in terms of $\pi_{0,c_1,1}$.For $i' = 1, .., c_2 - 1$, simplify

$$\pi_{0,i',0,1} = (\chi_{1,1})^{i'} \pi_{0,c_1,1} \tag{6.33}$$

and

$$\pi_{0,c_2,0,1} = \rho_1 (\chi_{1,1})^{c_2-1} \pi_{0,c_1,1}. \tag{6.34}$$

Next, some looping. Set $j' = 1$. This is the outer loop. The inner loop begins at $i' = c_2 - j'$ so simplify

$$\pi_{0,i',j',1} = \rho_1 \pi_{0,i'-1,j',1} + \frac{\lambda_2}{\mu_1} \pi_{0,i',j'-1,1}. \tag{6.35}$$

Continue to iterate, with $i' = c_2 - j' - 1, ..., 1$ in

$$\pi_{0,i',j',1} = \chi_{1,1} \pi_{0,i'-1,j',1} + \chi_{2,1} \pi_{0,i',j'-1,1}. \tag{6.36}$$

At the completion of all i' equations we back substitute, ending up with $\pi_{0,i',j',1}$ in terms of $\pi_{0,0,j',1}, ..., \pi_{0,0,1,1}$ and $\pi_{0,c_1,1}$. Then, add one to j', returning to (6.35) stopping when the $j' = c_2 - 1$ loop is complete. In similar fashion, we now iterate for probabilities with a class 2 customer in service. For $i' = 1, .., c_1 - 1$ simplify

$$\pi_{0,i',0,2} = (\chi_{1,2})^{i'} \pi_{0,c_1,2} \tag{6.37}$$

and

$$\pi_{0,c_2,0,2} = \frac{\lambda_1}{\mu_2} (\chi_{1,2})^{c_2-1} \pi_{0,c_1,2}. \tag{6.38}$$

Again looping, beginning with $j' = 1$. Now for $i' = 1, ..., c_2 - j'$

$$\pi_{0,i',j',2} = \frac{1}{\mu_2 + 1_{\{i'+j'<c_2\}}\lambda} \left(\lambda_1 \pi_{0,i'-1,j',2} + \lambda_2 \pi_{0,i',j'-1,2}\right). \tag{6.39}$$

The term $\pi_{0,i',j'-1,2}$ can be simplified from (6.37) and (6.38), and results from previous simplifications. Hence we have $\pi_{0,i',j',2}$ in terms of $\pi_{0,c_1,2}$ and $\pi_{0,0,j',2}..., \pi_{0,0,1,2}$. Return to (6.39), adding one to j', stopping when the loop for $j' = c_2 - 1$ is complete.

Simplification F

For $i' = c_2$,

$$\pi_{1,c_2,0,2} = \rho_2 \pi_{1,c_2-1,0,2} \tag{6.40}$$

and for $i' = c_2 - 1, ..., 1$

$$\pi_{1,i',0,2} = \chi_{1,2} \left(1_{\{i'>1\}} \pi_{1,i'-1,0,2} + 1_{\{i'=1\}} \pi_{1,c_1-1,2}\right) + \alpha_{2,2} \pi_{0,i'+1,0,2} + \alpha_{1,2} \pi_{0,i'+1,0,1}. \tag{6.41}$$

Back substitute to obtain $\pi_{1,i',0,2}$ in terms of $\pi_{1,c_1-1,2}$, $\pi_{0,c_1,2}$ and $\pi_{0,c_1,1}$, using (6.37) and (??) for $\pi_{0,i'+1,0,2}$, and (6.33) and (6.34) for $\pi_{0,i'+1,0,1}$. Similarly, for $i' = c_2$,

$$\pi_{1,c_2,0,1} = \rho_1 \pi_{1,c_2-1,0,1} \tag{6.42}$$

and for $i' = c_2 - 1, ..., 1$

$$\pi_{1,i',0,1} = \chi_{1,1} \left(1_{\{i'>1\}} \pi_{1,i'-1,0,1} + 1_{\{i'=1\}} \pi_{1,c_1-1,1}\right) + \alpha_{2,1} \pi_{1,i'+1,0,2} + \alpha_{1,1} \pi_{1,i'+1,0,1}. \tag{6.43}$$

Back substitute to obtain $\pi_{1,i',0,1}$ in terms of $\pi_{1,c_1-1,2}, \pi_{1,c_1-1,1}$, $\pi_{0,c_1,2}$ and $\pi_{0,c_1,1}$ using Simplifications E.

Simplification G

Now to "fill-in" the remaining states. We begin with $j' = 1$. Now for $i' = 1, ..., c_2 - j'$

$$\begin{aligned} \pi_{1,i',j',2} &= \frac{1}{\mu_2 + 1_{\{i'+j'<c_2\}}\lambda} \left(\lambda_1 \pi_{1,i'-1,j',2} + \lambda_2 \pi_{1,i',j'-1,2}\right) + \\ &\quad 1_{\{i'+j'<c_2\}} \left(\alpha_{2,2} \pi_{0,i'+1,j',2} + \alpha_{1,2} \pi_{0,i'+1,j',1}\right). \end{aligned} \tag{6.44}$$

From these generated equations we can again simplify using results from Simplifications E and F. By back-substitution this (6.44) yields $\pi_{1,i',j',2}$ in terms of $\pi_{1,0,1,2}$, $\pi_{1,c_1-1,2}$, $\pi_{1,c_1-1,1}$, $\pi_{0,c_1,2}$, $\pi_{0,c_1,1}$ and $\pi_{0,0,j',2}$. Continue iterating, adding one to j', stopping when $j' = c_2 - 1$ stage is complete.

We proceed in similar vein, now for a class 1 customers in service. Begin with $j' = 1$. Now for $i' = 1, ..., c_2 - j'$,

$$\begin{aligned} \pi_{1,i',j',1} &= \frac{1}{\mu_1 + 1_{\{i'+j'<c_2\}}\lambda} \left(\lambda_1 \pi_{1,i'-1,j',1} + \lambda_2 \pi_{1,i',j'-1,1}\right) + \\ &\quad 1_{\{i'+j'<c_2\}} \left(\alpha_{2,1} \pi_{0,i'+1,j',2} + \alpha_{1,1} \pi_{0,i'+1,j',1}\right). \end{aligned} \tag{6.45}$$

From these equations we can again simplify using results from Simplifications E and F. Again, by back-substitution (6.45) gives $\pi_{1,i',j',1}$ in terms of $\pi_{1,0,1,1}, \pi_{1,c_1-1,2}, \pi_{1,c_1-1,1}$, $\pi_{0,c_1,2}$, $\pi_{0,c_1,1}$, $\pi_{0,0,j',1}$ and $\pi_{0,0,j',2}$.Continue iterating, adding one to j', stopping when $j' = c_2 - 1$ stage is complete.

6.5.5 Solving Stage $\alpha_{0,s}, \alpha_{1,s}$

The difficulty in solving this system lies in the large number of interdependencies between states. From the above simplifications we are left with the remaining probabilities in the system in terms of $\pi_{1,0,1,1}, \pi_{1,c_1-1,2}, \pi_{1,c_1-1,1}$, $\pi_{0,c_1,2}$, $\pi_{0,c_1,1}$, $\pi_{0,0,j',1}$ and $\pi_{0,0,j',2}$. The solution to the remaining probabilities comes from a series of further simplifications.

First from the state $(0, c_1 - 1, 1)$ we simplify the equation

$$\pi_{0,c_1-1,1} = \chi_{2,1}\pi_{0,c_1-2,1} + \alpha_{1,1}\pi_{1,c_1-1,1} + \alpha_{2,1}\pi_{1,c_1-1,2} \tag{6.46}$$

where $\pi_{0,c_1-2,1}$ is solved. Next from the state $(0, c_1 - 1, 2)$,

$$\pi_{0,c_1-1,2} = \chi_{2,2}\pi_{0,c_1-2,2} + \alpha_{1,2}\pi_{0,c_1,1} + \alpha_{2,1}\pi_{0,c_1,2} \tag{6.47}$$

giving $\pi_{0,c_1-1,2}$ in terms of $\pi_{0,c_1,1}$ and $\pi_{0,c_1,2}$. From the state $(1, c_1 - 1, 1)$

$$\pi_{1,c_1-1,1} = \chi_{1,1}\pi_{0,c_1-1,1} + \chi_{2,1}\pi_{1,c_1-2,1} + \alpha_{1,1}\left(\pi_{1,1,0,1} + \pi_{2,0,1,1}\right) + \alpha_{1,2}\left(\pi_{1,1,0,1} + \pi_{2,0,1,1}\right). \tag{6.48}$$

Substituting from earlier simplifications leaves $\pi_{1,c_1-1,1}$ in terms of $\pi_{1,c_1-1,2}$, $\pi_{0,c_1,2}$, and $\pi_{0,c_1,1}$. Similarly from the state $(1, c_1 - 1, 2)$,

$$\pi_{1,c_1-1,2} = \chi_{2,2}\pi_{1,c_1-2,2} + \chi_{1,2}\pi_{0,c_1-1,2} + \alpha_{1,2}\pi_{0,1,0,1} + \alpha_{2,2}\pi_{0,1,0,2}. \tag{6.49}$$

This gives $\pi_{1,c_1-1,2}$ in terms of $\pi_{0,c_1,2}$ and $\pi_{0,c_1,1}$. Also,

$$\pi_{0,c_1-2,2} = \chi_{2,2}\pi_{0,c_1-3,2} + \alpha_{2,2}\pi_{0,c_1-1,2} + \alpha_{1,2}\pi_{0,c_1-1,1}. \tag{6.50}$$

As $\pi_{0,c_1-2,2}$ is solved, rearrange (6.50) to obtain $\pi_{0,c_1,2}$ in terms of $\pi_{0,c_1,1}$, using substitutions from (6.46) and (6.47). Now from this we have $\pi_{1,c_1-1,1}, \pi_{1,c_1-1,2}$ also in terms of $\pi_{0,c_1,1}$. Next some more simplifications. The solution is obtained through $\pi_{0,0,j',1}$.

$$\pi_{0,c_1,1} = \chi_{2,1}\pi_{0,c_1-1,1} + \alpha_{1,1}\pi_{1,0,1,1} + \alpha_{2,1}\pi_{1,0,1,2} \tag{6.51}$$

Simplifying gives $\pi_{0,c_1,1}$ in terms of $\pi_{1,0,1,1}$ and $\pi_{1,0,1,2}$. Next modify

$$\pi_{0,c_1,2} = \chi_{2,2}\pi_{0,c_1-1,2} + \alpha_{2,2}\pi_{0,0,1,2} + \alpha_{1,2}\pi_{0,0,1,1} \tag{6.52}$$

by substituting $\pi_{0,c_1,2}$ from (6.50) and $\pi_{0,c_1-1,2}$ from (6.47). This leaves $\pi_{0,0,1,2}$, $\pi_{0,0,1,1}$, $\pi_{1,0,1,2}$, and $\pi_{1,0,1,1}$. From state $(1, 0, 1, 2)$,

$$\pi_{1,0,1,2} = \chi_{2,2}\pi_{1,c_1-1,2} + \alpha_{2,2}\pi_{0,1,1,2} + \alpha_{1,2}\pi_{0,1,1,1}. \tag{6.53}$$

Substitute for $\pi_{1,c_1-1,2}$ from (6.49) into the above. This leaves $\pi_{1,0,1,2}$ in terms of $\pi_{0,0,1,2}, \pi_{0,0,1,1}$, and $\pi_{1,0,1,1}$. Now from state $(1,0,1,1)$,

$$\pi_{1,0,1,1} = \chi_{2,1}\pi_{1,c_1-1,1} + \alpha_{2,1}\left(\pi_{1,1,1,2} + \pi_{2,0,2,2}\right) + \alpha_{1,1}\left(\pi_{1,1,1,1} + \pi_{2,0,2,1}\right). \quad (6.54)$$

This leaves $\pi_{1,0,1,1}$ in terms of $\pi_{1,0,1,2}$, $\pi_{0,0,1,2}$,and $\pi_{0,0,1,1}$.Two more equations. From state $(0,0,1,1)$,

$$\pi_{0,0,1,1} = \chi_{2,1}\pi_{0,c_1,1} + \alpha_{2,1}\pi_{1,0,2,2} + \alpha_{1,1}\pi_{1,0,2,1}. \quad (6.55)$$

This leaves $\pi_{0,0,1,1}$ in terms of $\pi_{1,0,1,1}, \pi_{1,0,1,2}, \pi_{1,0,2,2}$,and $\pi_{1,0,2,1}$. From state $(0,0,1,2)$,

$$\pi_{0,0,1,2} = \chi_{2,2}\pi_{0,c_1,2} + \alpha_{1,2}\pi_{0,0,2,1} + \alpha_{2,2}\pi_{0,0,2,2}. \quad (6.56)$$

This leaves $\pi_{0,0,1,2}$ in terms of $\pi_{1,0,1,1}, \pi_{1,0,1,2}, \pi_{0,0,2,2}$,and $\pi_{1,0,2,1}$. From (6.52) to (6.56) we are left with unsolved probabilities $\pi_{1,0,1,1}, \pi_{1,0,1,2}, \pi_{0,0,1,2}, \pi_{0,0,1,1}, \pi_{1,0,2,1}$ and $\pi_{1,0,2,2}$. These are solved through the following looped simplifications.

For $j' = 2, .., c_2 - 1$ simplify

$$\pi_{0,0,j',2} = \chi_{2,2}\pi_{0,0,j'-1,2} + \alpha_{1,2}\pi_{0,0,j'+1,1} + \alpha_{2,2}\pi_{0,0,j'+1,2} \quad (6.57)$$

and

$$\pi_{0,0,c_2,2} = \rho_2\pi_{0,0,c_2-1,2}. \quad (6.58)$$

This will leave $\pi_{0,0,j',2}$ in terms of $\pi_{0,0,j',1}, \pi_{1,0,2,1}$ and $\pi_{1,0,2,2}$ after back substitution. Next, simplify for $j' = 2, .., c_2 - 1$

$$\pi_{1,0,j',1} = \chi_{2,1}\pi_{1,0,j'-1,1} + \alpha_{1,1}\left(\pi_{1,1,j',1} + \pi_{2,0,j'+1,1}\right) + \alpha_{2,1}\left(\pi_{1,1,j',2} + \pi_{2,0,j'+1,2}\right) \quad (6.59)$$

and

$$\pi_{1,0,c_2,1} = \frac{\lambda_2}{\mu_1}\pi_{1,0,c_2-1,1}. \quad (6.60)$$

By back substitution, obtain $\pi_{1,0,j',1}$ in terms of $\pi_{0,0,2,1}$, $\pi_{1,0,2,2}$ and $\pi_{1,0,3,2}$. Next, for $j' = 2, .., c_2 - 1$

$$\pi_{1,0,j',2} = \chi_{2,2}\pi_{1,0,j'-1,2} + \alpha_{1,2}\pi_{0,1,j',1} + \alpha_{2,2}\pi_{0,1,j',2} \quad (6.61)$$

and

$$\pi_{1,0,c_2,2} = \chi_{2,2}\pi_{1,0,c_2-1,2}. \quad (6.62)$$

By back substitution, we obtain $\pi_{1,0,j',2}$ in terms of $\pi_{0,0,j',1}$. Next for $j' = 2, .., c_2 - 1$

$$\pi_{0,0,j',1} = \chi_{2,1}\pi_{0,0,j'-1,1} + \alpha_{1,1}\pi_{1,0,j'+1,1} + \alpha_{2,1}\pi_{1,0,j'+1,2} \quad (6.63)$$

and

$$\pi_{0,0,c_2,1} = \frac{\lambda_2}{\mu_1}\pi_{0,0,c_2-1,1} \quad (6.64)$$

Now all the remaining probabilities are solved, as $\pi_{1,0,j',1}$ and $\pi_{1,0,j',2}$ are in terms of $\pi_{0,0,j',1}$. Hence all probabilities are now solved via back-substitution from $\pi_{0,0,j',1}$. Finally we normalise the probabilities.

6.6 Numerical Example

Consider a system with $c_1 = c_2 = 4$. This is the smallest system for which the algorithm should be used so enough states can be generated. The process of calculation is shown here to illustrate the solution process.

The utilisation, or probability of finding a busy server, is given by

$$\pi_B = 1 - \pi_{0,0,0} \tag{6.65}$$

The probability of finding only class one customers in the system is given by

$$\pi^1 = \sum_{i=0}^{c_1} \pi_{i,0,1} + \sum_{j=1}^{c_2} \pi_{c_1,i',0,1} \tag{6.66}$$

The probability of finding only class two customers in the system is given by

$$\pi^2 = \sum_{j=0}^{c_1} \pi_{0,j,2} + \sum_{k=1}^{c_2} \pi_{0,0,k,2} \tag{6.67}$$

The probability of a vacant secondary queue is given by

$$\pi_{Q_2=0} = \pi_{0,0,0} + \sum_{i=0}^{c_1} \sum_{j=0}^{c_1} (\pi_{i,j,1} + \pi_{i,j,2}) \tag{6.68}$$

6.6.1 Steady State Probabilities

We begin at the $a_{4,4}$ stage. Initialise by setting $\pi_{4,3,0,2} = 1$ hence $\pi_{4,4,0,2} = \frac{\lambda_1}{\mu_2}$. Now iterate to solve directly

$$\begin{aligned}
\pi_{4,2,0,2} &= \chi_{1,2}\pi_{4,3,0,2} \\
\pi_{4,1,0,2} &= \chi_{1,2}\pi_{4,2,0,2} \\
\pi_{4,0,2} &= \chi_{1,2}\pi_{4,1,0,2}
\end{aligned}$$

Next to solve the remaining states within $a_{4,4}$. Begin by setting $j' = 1$, for $i' = 0, ..., 3$

$$\begin{aligned}
\pi_{4,0,1,2} &= \chi_{2,2}\pi_{4,0,2} \\
\pi_{4,1,1,2} &= \chi_{2,2}\pi_{4,1,0,2} + \chi_{1,2}\pi_{4,0,1,2} \\
\pi_{4,2,1,2} &= \chi_{2,2}\pi_{4,2,0,2} + \chi_{1,2}\pi_{4,1,1,2} \\
\pi_{4,3,1,2} &= \rho_2\pi_{4,3,0,2} + \frac{\lambda_1}{\mu_2}\pi_{4,2,1,2}
\end{aligned}$$

At the completion of the i' loop, add one to j', stopping after the completion of $j' = 3$ finishing at $\pi_{4,0,4,2} = \rho_2\pi_{4,0,3,2}$. Next, solve the states $\pi_{i,0,2}; i \in \{0, ..., c_2 - 1\}$. This

can be achieved directly :

$$\begin{aligned}
\pi_{3,0,2} &= \chi_{1,2}^{-1}\pi_{4,0,2} \\
\pi_{2,0,2} &= \chi_{1,2}^{-1}\pi_{3,0,2} \\
\pi_{1,0,2} &= \chi_{1,2}^{-1}\pi_{2,0,2} \\
\pi_{0,0,2} &= \chi_{1,2}^{-1}\pi_{1,0,2}
\end{aligned}$$

The next process requires the use of states involving no class 2 customers within the primary or secondary queue with a class 1 customer in service. Starting with a secondary queue full/semi-full of class one customers we obtain:

$$\pi_{4,4,0,1} = \rho_1\pi_{4,3,0,1}.$$

$$\begin{aligned}
\pi_{4,3,0,1} &= \alpha_{1,1}\pi_{4,4,0,1} + \chi_{1,1}\pi_{4,2,0,1} + \alpha_{2,1}\pi_{4,2,0,2} \\
\pi_{4,2,0,1} &= \alpha_{1,1}\pi_{4,3,0,1} + \chi_{1,1}\pi_{4,1,0,1} + \alpha_{2,1}\pi_{4,1,0,2} \\
\pi_{4,1,0,1} &= \alpha_{1,1}\pi_{4,2,0,1} + \chi_{1,1}\pi_{4,0,1} + \alpha_{2,1}\pi_{4,2,0,2}
\end{aligned}$$

The probabilities $\pi_{4,i',0,2}$, $i = 1, ..., c_2$ were solved earlier. So $\pi_{4,1,0,1}$ is left in terms of $\pi_{4,0,1}$, this result $(\mathbb{A})$. From $S = (0,0,0)$, $\pi_{0,0,0} = \frac{\mu_1}{\lambda}\pi_{0,0,1} - \frac{\mu_2}{\lambda}\pi_{0,0,2}$,where $\pi_{0,0,2}$ is solved. Substitute this into

$$\pi_{0,0,1} = \chi_{1,1}\pi_{0,0,0} + \alpha_{1,1}\pi_{1,0,1} + \alpha_{2,1}\pi_{1,0,2}$$

where $\pi_{1,0,2}$ has been solved so (6.10) substituted in (6.9) eliminates $\pi_{0,0,0}$. Next obtain $\pi_{4,1,0,1}$ in terms of $\pi_{4,0,1}$, so

$$\begin{aligned}
\pi_{1,0,1} &= \alpha_{1,1}\pi_{2,0,1} + \chi_{1,1}\pi_{0,0,1} + \alpha_{2,1}\pi_{2,0,2} \\
\pi_{2,0,1} &= \alpha_{1,1}\pi_{3,0,1} + \chi_{1,1}\pi_{1,0,1} + \alpha_{2,1}\pi_{3,0,2} \\
\pi_{3,0,1} &= \alpha_{1,1}\pi_{4,0,1} + \chi_{1,1}\pi_{2,0,1} + \alpha_{2,1}\pi_{4,0,2} \\
\pi_{4,0,1} &= \alpha_{1,1}\pi_{4,1,0,1} + \chi_{1,1}\pi_{3,0,1} + \alpha_{2,1}\pi_{4,1,0,2}
\end{aligned}$$

From the above substituting for $\pi_{i-1,0,1}$ at each iteration, and noting that $\pi_{i+1,0,2}$ is solved, label the final result $(\mathbb{B})$. Hence from substituting $(\mathbb{B})$ into $(\mathbb{A})$ obtain $\pi_{4,0,1}$ and $\pi_{4,1,0,1}$. Via back substitution the solutions for $\pi_{i,0,1}$, $i = 0, ..., 4$; $\pi_{4,i',0,1}$, $i' = 1, ..., 4$; and $\pi_{0,0,0}$.

The rest of this stage is simple to solve via looping using equations (6.13) to (6.16), solving all of $\overline{\pi}_{4,i',1}$. Next the large looping stage. Initialise by setting $i^* = 3$. Now through the simplifications, firstly from Simplification A,

$\pi_{3,i',0,2} = (\chi_{1,2})^{i'}\pi_{3,1,2}$ where $i' = 1, ..., 3$ and $\pi_{3,4,0,2} = \frac{\lambda_2}{\mu_1}(\chi_{1,2})^3\pi_{3,1,2}$. Next from Simplification B, $\pi_{3,0,j',2} = (\chi_{2,2})^{j'}\pi_{3,1,2}$ where $j' = 1, ..., 3$ and $\pi_{3,0,4,2} =$

$\rho_2 (\chi_{2,2})^3 \pi_{3,1,2}$. With the above simplifications established, "fill in" the remaining states and simplify in terms of $\pi_{3,1,2}$ at each point for $\pi_{3,i',j',2}$ using (6.21) and (6.22). Simplification C puts the primary queue states with a class 2 in service in terms of $\pi_{0,1,2}$. From (6.23), $\pi_{i,3,2} = \chi_{1,2}\pi_{i-1,3,2} + \chi_{2,2}\pi_{i,2,2}$ where $i = 1, ..., 3$ and $\pi_{i,2,2}$ has been solved. This leaves $\pi_{3,1,2}$ down to $\pi_{1,1,2}$ in terms of $\pi_{0,3,2}$.

The final simplification at this point is from equations (6.24) and (6.25), where $\pi_{3,1,0,1}$ is reduced in terms of $\pi_{3,1,2}$ and $\pi_{3,1,1}$.

Solving Stage

With the simplifications made for all probabilities where $i' = c_1 - 1, s = 2$, all probabilities can be solved where $s = 2$ and $j = 1$ in the primary queue. This is achieved by using the states $S = \{(0, 1, 2), ..., (i^*, 1, 2)\}$. Commence with the boundary $S = (i^*, 1, 1)$ and simplify as this state contains no relationship with two states of the dual queue. This stage requires two stages of substitution with final elimination-solution of probability $\pi_{0,1,1}$. So for $S = (i^*, c_1 - i^*, 1)$ we have

$$\pi_{3,1,1} = \chi_{1,1}\pi_{2,1,1} + \chi_{2,1}\pi_{3,0,1} + \alpha_{1,1}(\pi_{3,1,0,1} + \pi_{4,0,1,1}) + \alpha_{2,1}(\pi_{3,1,0,2} + \pi_{4,0,1,2})$$

Note that $\pi_{i^*,c_1-i^*-1,1}, \pi_{i^*+1,0,1,1}$ and $\pi_{i^*,1,0,2}$ are already solved. Some substitution from earlier results is needed. Substituting for $\pi_{i^*,1,0,1}$ leads to $\pi_{i^*,c_1-i^*,1}$ and $\pi_{i^*,c_1-i^*,2}$. Also $\pi_{i^*,1,0,2}$ leads to $\pi_{i^*,1,2}$. These substitutions result in $\pi_{i^*,1,1}$ in terms of $\pi_{i^*-1,1,1}$, and $\pi_{i^*,1,2}$. Using states $S = \{(i^* - 1, c_1 - i^*, 1), ..., (0, c_1 - i^*, 1)\}$, and simplifying at each point, we use the following: for $i = i^* - 1, ..., 0$

$$\begin{aligned}
\pi_{2,1,1} &= \chi_{1,1}\pi_{1,1,1} + \alpha_{1,1}\pi_{3,1,1} + \chi_{2,1}\pi_{2,0,1} + \alpha_{2,1}\pi_{3,1,2} \\
\pi_{1,1,1} &= \chi_{1,1}\pi_{0,1,1} + \alpha_{1,1}\pi_{2,1,1} + \chi_{2,1}\pi_{1,0,1} + \alpha_{2,1}\pi_{2,1,2} \\
\pi_{0,1,1} &= \alpha_{1,1}\pi_{1,1,1} + \chi_{2,1}\pi_{0,0,1} + \alpha_{2,1}\pi_{1,1,2}
\end{aligned}$$

where $\pi_{2,0,1}, ..., \pi_{0,0,1}$ are solved. After simplification and substitution the final result has $\pi_{0,1,1}$ in terms of $\pi_{0,1,2}$. Label this result (𝔸). Now from state $S = (0, 0, 2)$ we have

$$\pi_{0,1,2} = -\rho_2\pi_{0,0,0} - \frac{\mu_1}{\mu_2}\pi_{0,1,1} + \alpha_{2,2}\pi_{0,0,2}$$

where $\pi_{0,0,2}$ and $\pi_{0,0,0}$ are solved. Label this result (𝔹). Substituting (𝔹) into (𝔸) we have a solution for $\pi_{0,1,1}$. Then obtain $\pi_{0,1,2}$ from (𝔹). Via back substitution we obtain the solution for $\pi_{i,1,1}$ where $i = 1, ..., i^*$ from (6.27) and (6.26). Going back through the simplification stage the solution is also obtained for and $\pi_{i,1,2}$ where $i = 1, ..., c_1 - 1$ from (6.23)(Simplification C); $\pi_{c_1-1,i',0,1}$ where $i' = 1, ..., c_2$ from

(6.24) and (6.25)(Simplification D); $\pi_{c_1-1,i',0,2}$ where $i' = 1,...,c_2$ from (6.17) and (6.18)(Simplification A); and $\pi_{c_1-1,i',j',2}$ where $i' = 1,...,c_2; j = 1,...,c_2 - i'$ from (6.19) and (6.20) and (6.21)(Simplification B). All $\pi_{3,i',j',1}$are then solved by simple looping using equations (6.29) to (6.31). This process is then repeated for $i^* = 2$, eventually solving for all $\pi_{2,i',j',1}$ and $\pi_{3,i',j',2}$.

Simplification stage α_0, α_1

Begin with Simplification E. The following simplifications are for probabilities $\pi_{0,i',j',1}$. The first simplification has $\pi_{0,i',0,1}$ in terms of $\pi_{0,4,1}$.

$$\begin{aligned}
\pi_{0,1,0,1} &= (\chi_{1,1})\,\pi_{0,4,1} \\
\pi_{0,2,0,1} &= (\chi_{1,1})^2\,\pi_{0,4,1} \\
\pi_{0,3,0,1} &= (\chi_{1,1})^3\,\pi_{0,4,1} \\
\pi_{0,4,0,1} &= \rho_1\,(\chi_{1,1})^3\,\pi_{0,4,1}
\end{aligned}$$

Next some looping. Let $j' = 1$. This is the outer loop. The inner loop begins at $i' = 4 - j'$ so

$$\begin{aligned}
\pi_{0,3,1,1} &= \rho_1\pi_{0,2,1,1} + \frac{\lambda_2}{\mu_1}\pi_{0,3,0,1} \\
\pi_{0,2,1,1} &= \chi_{1,1}\pi_{0,1,1,1} + \chi_{2,1}\pi_{0,2,0,1} \\
\pi_{0,1,1,1} &= \chi_{1,1}\pi_{0,0,1,1} + \chi_{2,1}0,1,0,1
\end{aligned}$$

At the completion of all i' equations we back substitute ending up with $\pi_{0,i',j',1}$ in terms of $\pi_{0,0,j',1},...,\pi_{0,0,1,1}$ and $\pi_{0,c_1,1}$. Then add one to j', returning to (6.35) stopping when $j' = c_2 - 1$ loop is complete in similar fashion to the above. The same process for class 2 customers in service has, for $i' = 1,..,3$,

$$\begin{aligned}
\pi_{0,1,0,2} &= (\chi_{1,2})^1\,\pi_{0,4,2} \\
\pi_{0,2,0,2} &= (\chi_{1,2})^2\,\pi_{0,4,2} \\
\pi_{0,3,0,2} &= (\chi_{1,2})^3\,\pi_{0,4,2} \\
\pi_{0,4,0,2} &= \frac{\lambda_1}{\mu_2}\,(\chi_{1,2})^3\,\pi_{0,4,2}
\end{aligned}$$

Again looping, beginning with $j' = 1$. Now for $i' = 1,...,3$

$$\begin{aligned}
\pi_{0,1,1,2} &= \frac{1}{\mu_2+\lambda}\,(\lambda_1\pi_{0,0,1,2} + \lambda_2\pi_{0,1,0,2}) \\
\pi_{0,2,1,2} &= \frac{1}{\mu_2+\lambda}\,(\lambda_1\pi_{0,1,1,2} + \lambda_2\pi_{0,2,0,2}) \\
\pi_{0,3,1,2} &= \frac{1}{\mu_2}\,(\lambda_1\pi_{0,2,1,2} + \lambda_2\pi_{0,3,0,2})
\end{aligned}$$

The term $\pi_{0,i',j'-1,2}$ can be simplified from (6.37) and (6.38) initially, and results from previous loops. Hence simplifications leave $\pi_{0,i',j',2}$ in terms of $\pi_{0,c_1,2}$ and $\pi_{0,0,j',2}..., \pi_{0,0,1,2}$. Return to (6.39) adding one to j', stopping when the loop for $j' = 3$ is complete.

Next from Simplification F. For $i' = 4, ..., 1$

$$\begin{aligned}
\pi_{1,4,0,2} &= \rho_2 \pi_{1,3,0,2} \\
\pi_{1,3,0,2} &= \chi_{1,2}\pi_{1,2,0,2} + \alpha_{2,2}\pi_{0,4,0,2} + \alpha_{1,2}\pi_{0,4,0,1} \\
\pi_{1,2,0,2} &= \chi_{1,2}\pi_{1,1,0,2} + \alpha_{2,2}\pi_{0,3,0,2} + \alpha_{1,2}\pi_{0,3,0,1} \\
\pi_{1,1,0,2} &= \chi_{1,2}\pi_{1,3,2} + \alpha_{2,2}\pi_{0,i'+1,0,2} + \alpha_{1,2}\pi_{0,i'+1,0,1}
\end{aligned}$$

Once expanded back substitute to obtain $\pi_{1,i',0,2}$ in terms of $\pi_{1,3,2}$, $\pi_{0,4,2}$ and $\pi_{0,4,1}$, using (6.37) and (6.38) for $\pi_{0,i'+1,0,2}$, and (6.33) and (6.34) for $\pi_{0,i'+1,0,1}$. Similarly, for $i' = 4, ..., 1$

$$\begin{aligned}
\pi_{1,4,0,1} &= \rho_1 \pi_{1,3,0,1} \\
\pi_{1,3,0,1} &= \chi_{1,1}\pi_{1,2,0,1} + \alpha_{2,1}\pi_{1,4,0,2} + \alpha_{1,1}\pi_{1,4,0,1} \\
\pi_{1,2,0,1} &= \chi_{1,1}\pi_{1,1,0,1} + \alpha_{2,1}\pi_{1,3,0,2} + \alpha_{1,1}\pi_{1,3,0,1} \\
\pi_{1,1,0,1} &= \chi_{1,1}\pi_{1,3,1} + \alpha_{2,1}\pi_{1,2,0,2} + \alpha_{1,1}\pi_{1,2,0,1}.
\end{aligned}$$

Again back substitute to obtain $\pi_{1,i',0,1}$ in terms of $\pi_{1,3,2}, \pi_{1,3,1}$, $\pi_{0,4,2}$ and $\pi_{0,4,1}$ using Simplifications E.

From Simplification G, "fill-in" the remaining states. This stage simplifies down to the terms needed to solve probabilities later. Begin with $j' = 1$. Now for $i' = 1, 2, 3$

$$\begin{aligned}
\pi_{1,1,1,2} &= \frac{1}{\mu_2 + \lambda}(\lambda_1 \pi_{1,0,1,2} + \lambda_2 \pi_{1,1,0,2}) + \alpha_{2,2}\pi_{0,2,1,2} + \alpha_{1,2}\pi_{0,2,1,1} \\
\pi_{1,2,1,2} &= \frac{1}{\mu_2 + \lambda}(\lambda_1 \pi_{1,1,1,2} + \lambda_2 \pi_{1,2,0,2}) + \alpha_{2,2}\pi_{0,3,1,2} + \alpha_{1,2}\pi_{0,3,1,1} \\
\pi_{1,3,1,2} &= \frac{1}{\mu_2}(\lambda_1 \pi_{1,2,1,2} + \lambda_2 \pi_{1,3,0,2}).
\end{aligned}$$

From these generated equations simplify using results from Simplifications E and F. By back-substitution this simplification gives $\pi_{1,i',j',2}$ in terms of $\pi_{1,0,1,2}, \pi_{1,3,2}, \pi_{1,3,1}$, $\pi_{0,4,2}$, $\pi_{0,4,1}$ and $\pi_{0,0,j',2}$. Continue iterating, adding one to j', stopping when $j' = 3$ stage is complete. In similar vein for class 1 customers in service, begin with $j' = 1$, and for $i' = 1, 2, 3$

$$\begin{aligned}
\pi_{1,1,1,1} &= \frac{1}{\mu_1 + \lambda}(\lambda_1 \pi_{1,0,1,1} + \lambda_2 \pi_{1,1,0,1}) + \alpha_{2,1}\pi_{0,2,1,2} + \alpha_{1,1}\pi_{0,2,1,1} \\
\pi_{1,2,1,1} &= \frac{1}{\mu_1 + \lambda}(\lambda_1 \pi_{1,1,1,1} + \lambda_2 \pi_{1,2,0,1}) + \alpha_{2,1}\pi_{0,3,1,2} + \alpha_{1,1}\pi_{0,3,1,1} \\
\pi_{1,3,1,1} &= \frac{1}{\mu_1}(\lambda_1 \pi_{1,2,1,1} + \lambda_2 \pi_{1,3,0,1}).
\end{aligned}$$

From these generated equations again simplify using results from Simplifications E and F. By back-substitution this simplification gives $\pi_{1,i',j',1}$ in terms of $\pi_{1,0,1,1}, \pi_{1,3,2}, \pi_{1,3,1}$, $\pi_{0,4,2}$, $\pi_{0,4,1}$, $\pi_{0,0,j',1}$ and $\pi_{0,0,j',2}$. Continue iterating, adding one to j', stopping when $j' = 3$ stage is complete. From the above simplifications the remaining probabilities in the system are all in terms of at least one of $\pi_{1,0,1,1}, \pi_{1,3,2}, \pi_{1,3,1}$, $\pi_{0,4,2}$, $\pi_{0,4,1}$, $\pi_{0,0,j',1}$ and $\pi_{0,0,j',2}$. The solution to the remaining probabilities comes from a series of further simplifications. First

$$\pi_{0,3,1} = \chi_{2,1}\pi_{0,2,1} + \alpha_{1,1}\pi_{1,3,1} + \alpha_{2,1}\pi_{1,3,2}$$

where $\pi_{0,2,1}$ is solved. Next

$$\pi_{0,3,2} = \chi_{2,2}\pi_{0,2,2} + \alpha_{1,2}\pi_{0,4,1} + \alpha_{2,1}\pi_{0,4,2}$$

giving $\pi_{0,3,2}$ in terms of $\pi_{0,4,1}$ and $\pi_{0,4,2}$. Also we simplify

$$\pi_{1,3,1} = \chi_{1,1}\pi_{0,3,1} + \chi_{2,1}\pi_{1,2,1} + \alpha_{1,1}\left(\pi_{1,1,0,1} + \pi_{2,0,1,1}\right) + \alpha_{1,2}\left(\pi_{1,1,0,1} + \pi_{2,0,1,1}\right).$$

Substituting from earlier simplifications leaves $\pi_{1,3,1}$ in terms of $\pi_{1,3,2}$, $\pi_{0,4,2}$, and $\pi_{0,4,1}$. Similarly

$$\pi_{1,3,2} = \chi_{2,2}\pi_{1,2,2} + \chi_{1,2}\pi_{0,3,2} + \alpha_{1,2}\pi_{0,1,0,1} + \alpha_{2,2}\pi_{0,1,0,2}.$$

From this $\pi_{1,3,2}$ is in terms of $\pi_{0,4,2}$ and $\pi_{0,4,1}$. Now

$$\pi_{0,2,2} = \chi_{2,2}\pi_{0,1,2} + \alpha_{2,2}\pi_{0,3,2} + \alpha_{1,2}\pi_{0,3,1}.$$

As $\pi_{0,2,2}$ is solved, rearrange (6.50) to obtain $\pi_{0,4,2}$ in terms of $\pi_{0,4,1}$, using substitutions from (6.46) and (6.47) . From this we have $\pi_{1,3,1}, \pi_{1,3,2}$ also in terms of $\pi_{0,4,1}$.Next some more simplifications. The solution is obtained through $\pi_{0,0,j',1}$. From state $(0, 4, 1)$

$$\pi_{0,4,1} = \chi_{2,1}\pi_{0,3,1} + \alpha_{1,1}\pi_{1,0,1,1} + \alpha_{2,1}\pi_{1,0,1,2}.$$

Simplifying gives $\pi_{0,4,1}$ in terms of $\pi_{1,0,1,1}$ and $\pi_{1,0,1,2}$. Next, from state $(0, 4, 2)$ simplify

$$\pi_{0,4,2} = \chi_{2,2}\pi_{0,3,2} + \alpha_{2,2}\pi_{0,0,1,2} + \alpha_{1,2}\pi_{0,0,1,1}$$

substituting $\pi_{0,4,2}$ from (6.50) and $\pi_{0,3,2}$ from (6.47). This leaves $\pi_{0,0,1,2}, \pi_{0,0,1,1}, \pi_{1,0,1,2}$, and $\pi_{1,0,1,1}$.From state $(1, 0, 1, 2)$,

$$\pi_{1,0,1,2} = \chi_{2,2}\pi_{1,3,2} + \alpha_{2,2}\pi_{0,1,1,2} + \alpha_{1,2}\pi_{0,1,1,1}.$$

Substitute for $\pi_{1,c_1-1,2}$ from (6.49). This gives $\pi_{1,0,1,2}$ in terms of $\pi_{0,0,1,2}, \pi_{0,0,1,1}$, and $\pi_{1,0,1,1}$. Now from state $(1, 0, 1, 1)$,

$$\pi_{1,0,1,1} = \chi_{2,1}\pi_{1,3,1} + \alpha_{2,1}\left(\pi_{1,1,1,2} + \pi_{2,0,2,2}\right) + \alpha_{1,1}\left(\pi_{1,1,1,1} + \pi_{2,0,2,1}\right).$$

This leaves $\pi_{1,0,1,1}$ in terms of $\pi_{1,0,1,2}$,$\pi_{0,0,1,2}$,and $\pi_{0,0,1,1}$. Two more equations: From state $(0,0,1,1)$,

$$\pi_{0,0,1,1} = \chi_{2,1}\pi_{0,4,1} + \alpha_{2,1}\pi_{1,0,2,2} + \alpha_{1,1}\pi_{1,0,2,1}.$$

This leaves $\pi_{0,0,1,1}$ in terms of $\pi_{1,0,1,1}$,$\pi_{1,0,1,2}$,$\pi_{1,0,2,2}$,and $\pi_{1,0,2,1}$.Also we simplify

$$\pi_{0,0,1,2} = \chi_{2,2}\pi_{0,c_1,2} + \alpha_{1,2}\pi_{0,0,2,1} + \alpha_{2,2}\pi_{0,0,2,2}.$$

This leaves $\pi_{0,0,1,2}$ in terms of $\pi_{1,0,1,1}$,$\pi_{1,0,1,2}$,$\pi_{0,0,2,2}$,and $\pi_{1,0,2,1}$.From (6.52) to (6.56) we have $\pi_{1,0,1,1}$, $\pi_{1,0,1,2}$, $\pi_{0,0,1,2}$, $\pi_{0,0,1,1}$, $\pi_{1,0,2,1}$ and $\pi_{1,0,2,2}$. These are solved through the following looped simplifications.

For $j' = 2,3,4$;

$$\begin{aligned}
\pi_{0,0,2,2} &= \chi_{2,2}\pi_{0,0,1,2} + \alpha_{1,2}\pi_{0,0,3,1} + \alpha_{2,2}\pi_{0,0,3,2} \\
\pi_{0,0,3,2} &= \chi_{2,2}\pi_{0,0,2,2} + \alpha_{1,2}\pi_{0,0,4,1} + \alpha_{2,2}\pi_{0,0,4,2} \\
\pi_{0,0,4,2} &= \rho_2\pi_{0,0,3,2}
\end{aligned}$$

This will leave $\pi_{0,0,j',2}$ in terms of $\pi_{0,0,j',1}$, $\pi_{1,0,2,1}$ and $\pi_{1,0,2,2}$ after back substitution. Next, for $j' = 2,3$

$$\begin{aligned}
\pi_{1,0,2,1} &= \chi_{2,1}\pi_{1,0,1,1} + \alpha_{1,1}\left(\pi_{1,1,2,1} + \pi_{2,0,3,1}\right) + \alpha_{2,1}\left(\pi_{1,1,2,2} + \pi_{2,0,3,2}\right) \\
\pi_{1,0,3,1} &= \chi_{2,1}\pi_{1,0,2,1} + \alpha_{1,1}\left(\pi_{1,1,3,1} + \pi_{2,0,4,1}\right) + \alpha_{2,1}\left(\pi_{1,1,3,2} + \pi_{2,0,4,2}\right) \\
\pi_{1,0,4,1} &= \frac{\lambda_2}{\mu_1}\pi_{1,0,3,1}
\end{aligned}$$

By back substitution, obtain $\pi_{1,0,j',1}$ in terms of $\pi_{0,0,2,1}$, $\pi_{1,0,2,2}$ and $\pi_{1,0,3,2}$. Next, for $j' = 2,..,c_2 - 1$

$$\begin{aligned}
\pi_{1,0,2,2} &= \chi_{2,2}\pi_{1,0,1,2} + \alpha_{1,2}\pi_{0,1,2,1} + \alpha_{2,2}\pi_{0,1,2,2} \\
\pi_{1,0,3,2} &= \chi_{2,2}\pi_{1,0,2,2} + \alpha_{1,2}\pi_{0,1,3,1} + \alpha_{2,2}\pi_{0,1,3,2} \\
\pi_{1,0,4,2} &= \chi_{2,2}\pi_{1,0,3,2}.
\end{aligned}$$

By back substitution, obtain $\pi_{1,0,j',2}$ in terms of $\pi_{0,0,j',1}$. For $j' = 2,..,c_2 - 1$

$$\begin{aligned}
\pi_{0,0,2,1} &= \chi_{2,1}\pi_{0,0,1,1} + \alpha_{1,1}\pi_{1,0,3,1} + \alpha_{2,1}\pi_{1,0,3,2} \\
\pi_{0,0,3,1} &= \chi_{2,1}\pi_{0,0,2,1} + \alpha_{1,1}\pi_{1,0,4,1} + \alpha_{2,1}\pi_{1,0,4,2} \\
\pi_{0,0,4,1} &= \frac{\lambda_2}{\mu_1}\pi_{0,0,3,1}.
\end{aligned}$$

Now all the remaining probabilities are solved, as $\pi_{1,0,j',1}$ and $\pi_{1,0,j',2}$ are in terms of $\pi_{0,0,j',1}$. Hence all probabilities are now solved via back-substitution from $\pi_{0,0,j',1}$. Finally we normalise the probabilities.

Performance Characteristics

Using the above, some comparative performance statistics are given in Table 6.1 between the single preemptive finite buffer model covered in Chapter 4, the preemptive MPDQ covered in Chapter 5 and the non-preemptive model analysed in this chapter.

Of interest is the performance of the non-preemptive MPDQ (npMPDQ) in high traffic intensity. Recall from Chapter 5 that the preemptive MPDQ (preMPDQ) outperformed the preemptive single queue (preSQ) in periods of high traffic intensity. In the comparisons for $\rho \geq 0.7$ in Table 6.1, the loss probability π_{loss} is lowest for the npMPDQ as $\rho \rightarrow 1$. The arrival and services rates have been selected to show the change in performance characteristics as the traffic intensity increases, so as we move down the table the higher the traffic intensity. There is a noticeable improvement when the traffic intensity is higher for class 2 traffic. The classwise QoS can be seen with the expected class 2 customers in the system a little higher under the npMPDQ than the preSQ, however the expected class 1 customers being substantially lower, again as $\rho_2 \rightarrow 1$.

What we also see from Table 6.1 is that the preemptive MPDQ (preMPDQ) always has a lower probability of an empty secondary queue than the non-preemptive MPDQ. This is confirmed statistically, with a one-tail paired t-test yielding a p-value of 0.001. This is expected as the npMPDQ is not interrupted during service, so a longer queue forms. Paired t-tests were conducted to determine significant differences in the expected number in the system for the three systems using the models conducted in Table 6.1. The results are found in Table 6.2.

Six of the hypothesis test returned statistical significance ($p < 0.05$). For the expected number of class 1 customers in the system $\left(L_s^1\right)$, the preMPDQ was lower than the preemptive single queue (preSQ). For the expected number of class 2 customers in the system $\left(L_s^2\right)$, the preMPDQ was lower than the npMPDQ. Hence there are more class 2 customers in the system under non-preemption than preemption. This is also the case overall. Loss is impressive for the npMPDQ. It must be noted that the tests were conducted on all values of ρ, hence there is no distinction between high and low values of traffic intensity. The Maple code for the above examples is found in Appendix 9.1.3.

Some key findings:

- Under high overall traffic intensity (ρ), the probability of a system full of class 1 customers $\left(\pi^1\right)$ is lowest under the npMPDQ
- The expected number of class 1 customer in the system $\left(L_s^1\right)$ is lower than the preSQ and preMPDQ when $\rho \geq 0.7$.

$\mu_1 = 10, \mu_2 = 10$											
Model	λ_1	λ_2	ρ	π^1	π^2	π_B	$\pi_{Q_2=0}$	π_{loss}	L_s^1	L_s^2	L_s
npMPDQ				.0769	.3016	.5567	.8370	.0056	.54	1.45	1.97
preMPDQ	1	2	.3	.0539	.2895	.3979	.9042	.0127	.22	.85	.89
preSQ				.0626	.2000	.3000		.0000	.21	.52	.73
npMPDQ				.1390	.2259	.6062	.8040	.0053	.72	1.46	2.16
preMPDQ	2	1	.3	.1446	.1290	.3337	.9654	.0034	.46	.41	.51
preSQ				.1519	.1000	.3000		.0000	.45	.28	.73
npMPDQ				.0923	.3251	.7598	.7116	.0108	.82	2.16	2.95
preMPDQ	2	2	.4	.0993	.2709	.4843	.8935	.0148	.47	.95	1.06
preSQ				.2000	.1156	.4000		.0003	.45	.61	1.06
npMPDQ				.0323	.4450	.9115	.5486	.0374	.73	3.16	3.86
preMPDQ	2	5	.7	.0323	.5455	.7786	.6785	.0774	.48	2.33	2.74
preSQ				.0457	.4951	.6861		.0140	.43	2.20	2.63
npMPDQ				.0543	.2476	.9597	.4624	.0222	1.06	2.63	3.64
preMPDQ	5	2	.7	.1680	.2115	.7912	.7242	.0328	1.44	1.41	1.96
preSQ				.1917	.2126	.6853		.0114	1.34	1.27	2.61
npMPDQ				.0177	.2697	.9851	.4624	.0533	.85	3.10	3.95
preMPDQ	5	5	1	.0414	.3799	.9002	.5146	.1195	1.34	2.99	4.12
preSQ				.0480	.4962	.8842		.0738	1.09	3.74	4.83
npMPDQ				.0228	.2900	.9900	.0347	.0555	1.05	3.24	4.28
preMPDQ	7	3	1	.0853	.2226	.8971	.5261	.1094	2.11	2.54	3.96
preSQ				.0968	.3490	.8833		.0661	1.72	3.08	4.80
npMPDQ				.0130	.3875	.9760	.3525	.0711	.65	3.56	4.21
preMPDQ	3	7	1	.0196	.5595	.8994	.5173	.1427	.74	3.17	4.14
preSQ				.0223	.6511	.8857		.0856	.60	4.29	4.89
npMPDQ				.0070	.7469	.9497	.4378	.0929	.33	4.16	4.49
preMPDQ	1	9	1	.0057	.7686	.8940	.5372	.1888	.24	3.22	4.01
preSQ				.0061	.8096	.8877		.1017	.18	4.77	4.96
npMPDQ				.0283	.1970	.9940	.2860	.0398	1.26	2.06	3.32
preMPDQ	9	1	1	.2411	.0792	.8916	.5460	.1013	3.41	1.41	3.76
preSQ				.2597	.1994	.8832		.0658	2.73	2.07	4.80

Table 6.1: Non-preemptive MPDQ - comparitive examples

Test	p
$H_0 : L^1_{s,npMPDQ} = L^1_{s,preMPDQ}$ vs $H_1 : L^1_{s,npMPDQ} < L^1_{s,preMPDQ}$	0.139
$H_0 : L^1_{s,npMPDQ} = L^1_{s,preSQ}$ vs $H_1 : L^1_{s,npMPDQ} < L^1_{s,preSQ}$	0.267
$H_0 : L^1_{s,preMPDQ} = L^1_{s,preSQ}$ vs $H_1 : L^1_{s,preMPDQ} < L^1_{s,preSQ}$	0.017
$H_0 : L^2_{s,preMPDQ} = L^2_{s,npMPDQ}$ vs $H_1 : L^2_{s,preMPDQ} < L^2_{s,npMPDQ}$	0.000
$H_0 : L^2_{s,preSQ} = L^2_{s,npMPDQ}$ vs $H_1 : L^2_{s,preMPDQ} < L^2_{s,npMPDQ}$	0.086
$H_0 : L^2_{s,preMPDQ} = L^2_{s,preSQ}$ vs $H_1 : L^2_{s,preMPDQ} < L^2_{s,preSQ}$	0.062
$H_0 : L_{s,preMPDQ} = L_{s,npMPDQ}$ vs $H_1 : L_{s,preMPDQ} < L_{s,npMPDQ}$	0.008
$H_0 : L_{s,preSQ} = L_{s,npMPDQ}$ vs $H_1 : L_{s,preSQ} < L_{s,npMPDQ}$	0.240
$H_0 : L_{s,preMPDQ} = L_{s,preSQ}$ vs $H_1 : L_{s,preMPDQ} < L_{s,preSQ}$	0.004
$H_0 : \pi_{loss,npMPDQ} = \pi_{loss,preMPDQ}$ vs $H_1 : \pi_{loss,npMPDQ} < \pi_{loss,preMPDQ}$	0.002
$H_0 : \pi_{loss,npMPDQ} = \pi_{loss,preSQ}$ vs $H_1 : \pi_{loss,npMPDQ} < \pi_{loss,preSQ}$	0.317
$H_0 : \pi_{loss,preSQ} = \pi_{loss,preMPDQ}$ vs $H_1 : \pi_{loss,preSQ} < \pi_{loss,preMPDQ}$	0.001

Table 6.2: Hypothesis results comparing the three systems

- The npMPDQ performs best when $\rho \geq 0.7$, both in terms of loss and expected customers in the system.
- The npMPDQ system improves the class 1 performance statistics without huge compromises to class 2 performance under high traffic intensity.

6.7 Conclusion

The non-preemptive algorithm was introduced and the analysis detailed. In similar fashion to the preemptive case, the state rate transitions were detailed through matrices and figures, then the system was described in vector-matrix form. A generic solution algorithm was detailed describing the approach to the explicit algorithmic solution undertaken via global balance equations. An example of the solution process were given, including reference to Maple code used to solve the algorithm. Significant key results included the non-preemptive MPDQ model reduced the expected number of class 1 customers waiting for high traffic intensity, and loss levels were lower than the other two systems analysed.

Part IV

SIMULATION STUDIES

Chapter 7

Simulation of the MPDQ and other Finite Buffer Models

7.1 Introduction

The simulation work undertaken on the MPDQ allows analysis and comparison of multiple classes of customers unattainable by conventional algorithmic methods. Whilst the steady state solution for two classes of customers has been shown achievable via an algorithm, this is not the case for three or more classes. The use of simulation enhances our ability to analyse the MPDQ for as many classes as desired. In this chapter, many facets of the MPDQ model using the Arena simulation package are investigated.

The overall aims of this chapter are as follows:

1. Evaluate and compare the MPDQ with FIFO under various scheduling regimes with multiple classes of customers
2. Evaluate the performance statistics of all FIFO and MPDQ models
3. Determine the best scheduling regime for the MPDQ in terms of QoS
4. Determine on the optimal amount of classes of customers in terms of QoS
5. Analyse the performance of the MPDQ in a network scenario

The progression of this chapter is as follows. To begin with, the simulation package Arena is discussed, along with the reasons for simulation. A comparison between simulation and analytical results for the preemptive MPDQ shall be detailed, with some discussion of simulation stopping time. Then a look at the performance characteristics of the preemptive MPDQ for a variety of queueing disciplines, with aim of confirming that priority schemes are superior to classic schemes. Next a more detailed look at

the non-preemptive MPDQ, with comparisons to the single finite buffer counterpart detailed. As shall be shown the MPDQ outperforms the single buffer models in terms of waiting times. A comparison of preemptive and non-preemptive service disciplines is undertaken for the MPDQ, with non-preemptive showing superior performance criteria. Finally, the MPDQ in a network is looked at.

7.2 Simulation in Communication

Much of the analysis on communications models is tackled via simulation. Most analysts compare their schemes with classic models as a baseline, or with other like schemes. [Bagwell et. al., 1995] investigates the Stop-and Go queueing discipline (SGQ), comparing the results with the classic $M/M/1/K$ model, finding that loss decreased logarithmically as buffer size increased. Further low loss (10^{-9}) was difficult to achieve without large buffer sizes even with low traffic intensity (ρ).

Direct simulation techniques in [Bolot, 1993] investigate loss in the Internet using actual UDP packets correcting for service rates across Internet hops. The obvious need for investigation of 'loss prone' applications such as video has also been tackled [Boyce and Gaglianello, 1998], and [Kato et. al., 1999] determines a distribution for traffic on the Internet ($M/M/S(m)$). [Rose, 1995] looks at video traffic simulation, also used in [Hayes et. al., 1999].

Network problems have also been tackled using routing schemes.
[Bahk and El Zarki, 1992] switches from shortest path algorithm when ρ is low to multiple paths in high ρ to distribute the congestion, using a 5-node network.

7.3 Simulation Set Up

All simulations were conducted using the Arena Simulation Software package [Kelton et. al., 2002], a flexible windows based program suitable for a vast area of statistical analysis with dedicated queueing subroutines. Arena has provided effective results of queueing disciplines in prior work [Bedford, 1999] and was again chosen here as the preferred tool of analysis. Arena allows users to define effective class based simulation due to its ability to generate arriving data and service data from any statistical distribution. Further, it can preempt customers from service if needed, and Arena gathers vast performance characteristics, such as waiting times, loss, utilisation, and queue quantities. These characteristics can also be gathered on a class by class basis. In constructing the Arena model, two independent algorithms (for the preemptive and non-preemptive cases) as described in Chapter 3 were constructed for the MPDQ. The single queue schemes were also designed so that various regimes could be compared.

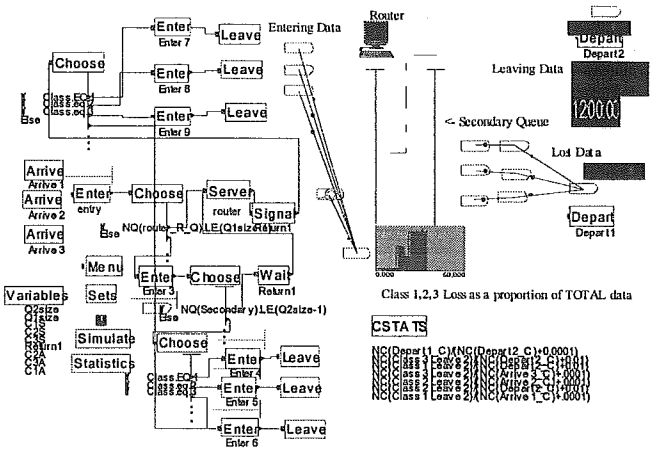

Figure 7.1: Arena-Model MPDQ Non-Preemptive Overview (3 classes)

The preemption process required a separate procedure to check for the quantity of first class packets in the system. The preempt subroutine was designed to expel a lower classed customer from service if a higher classed customer was present in the primary queue. All algorithms are simple to extend to multiple classes of customers. The simulation model developed for the non-preemptive simulation was substantially simpler than the preemptive model. Analytically the reverse holds, with the non-preemptive solution being more difficult to obtain than the preemptive solution using the matrix-analytic computational algorithm solution process (see Chapters 5 and 6). The Arena simulations required low-level programming as the preempt and routing of traffic for the MPDQ is non-conventional to the package. The MPDQ process needed the use of special functions such as *signal* and *wait* to release the appropriate quantity of customers from the secondary to the primary queue. An example of the differences in the two models is given in Figures 7.1 and 7.2.

Figure 7.1 contains the non-preemptive model and Figure 7.2 contains the pre-emptive model. As can be seen, the preemptive requires more commands than the non-preemptive model. This is as Arena does not preempt by default so a separate logic circuit was created to scan the primary queue and preempt the server if a higher classed customer was detected in that queue. Figure 7.3 shows the MPDQ for 5 classes of customers in *Run* mode. A Class 3 customer is in service, with the primary queue full. Two customers are waiting in the secondary queue.

In Arena simulations, there are three possible ways to cease the model running. The first is to set a finite simulation time, the second to define a statistic-dependent conditional statement, and the third is user interruption. Otherwise the simulation will run indefinitely. To determine a suitable number of simulation runs, the method

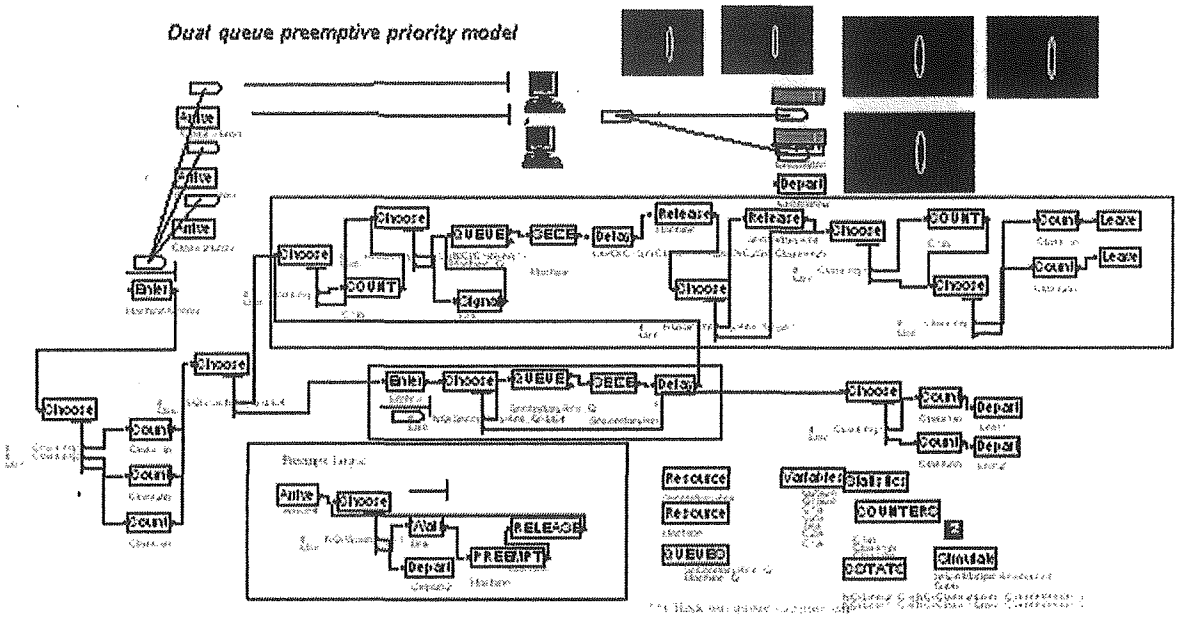

Figure 7.2: Arena-Model MPDQ Preemptive Overview (3 classes)

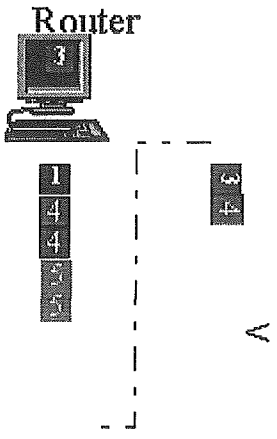

Figure 7.3: Run Mode for the MPDQ HCF Class 5 model

developed in [Law and Kelton, 1982] - sections 9.3 and 9.5, were adopted. Further details are to follow.

For both the arrival of and service to customers, we are assuming that these processes are independent for the classes. Customers here are considered of uniform length, that is, there are no differing sizes of customers and their occupancy within the queue is the same. The buffer size/s for arriving customers was fixed. For a single queue $c = 10$, and for a dual queue $c_1 = c_2 = 5$. Next, to define the performance characteristics used in the tables in this section: $\overline{Loss}$= Average loss for all classes; $\overline{Loss}^i$ = Average loss for class i; Ml = Maximum loss over the simulation period; Ml^i= Maximum loss for class i over the simulation period; and U = Utilisation.

For all models the first class customer is defined as the rarest arrival and longest in service. The rationale behind this is that the 'rarer' traffic will in many cases be the more demanding on system resources and can be seen as either the most or least valuable, depending upon the type of queueing discipline. Examples of high class high demand traffic include videoconference links, streaming audio or streaming video. As more classes are introduced, the performance differences between them may not be as efficient as fewer classes.

All statistics are given here without any added computation, mechanical or code-invoked times that can be found in some communications policies. Examples of these times include policies like time-out thresholds, loss thresholds and throughput constraints. Time-out thresholds are used to discard waiting customers after a predefined period of waiting (an example of this is the dual queue described in [Hayes et. al., 1999]). These is useful especially in LIFO scheduling where some customers are stuck in the queue for long periods. Its application in communications is useful in systems such as TCP/IP, where data can be resent if not received by the source. This threshold 'cleans out' old data which would simply clog the system by reducing waiting space. The simulation work here is designed simply to investigate the properties of identical systems at a fundamental level. Further analysis is simple through the adjustment of the simulation models if such policies are set by communications or other network providers, and remains a task for further analysis.

7.3.1 The Need for Simulation

The problem of solving the steady state probabilities (then obtaining further statistics) for the MPDQ analytically is mathematically complex (as seen in earlier chapters). Consider the lowest level of a single and dual queue with prioritised traffic of only two classes of customers. Recalling the state generation of the MPDQ with two classes of customers, as in Chapters 5 and 6, the dimensions of the irreducible infinitesimal

Buffer Size			Single queue two class finite buffer		MPDQ for two classes	
c_1[1]	c_2	C	Preemptive	Non-preemptive	Preemptive	Non-preemptive
2	2	4	15	21	21	43
3	2	5	21	31	30	61
2	3	5	21	31	33	67
3	3	6	28	43	46	93
5	5	10	66	111	141	283
10	10	20	231	421	781	1563
15	5	20	231	421	456	913
5	15	20	231	421	831	1663
20	20	40	861	1641	5061	10123
50	50	100	5151	10101	68901	137803
100	100	200	20301	40201	525301	1050603
500	500	1000	501501	1001001	631226501	126253003

Table 7.1: A matrix dimensions by Model and Buffer Sizes

generator of the system, A, is given by

$$\frac{1}{2}(c_1+1)\left(c_2^2+3c_2+c_1+2\right)\times\frac{1}{2}(c_1+1)\left(c_2^2+3c_2+c_1+2\right) \tag{7.1}$$

for a preemptive service model, and for a non-preemptive service model it is

$$\left((c_1+1)\left(c_2^2+3c_2+c_1+2\right)+1\right)\times\left((c_1+1)\left(c_2^2+3c_2+c_1+2\right)+1\right) \tag{7.2}$$

where c_1 = primary queue size and c_2 = secondary queue size.

For a single preemptive finite buffer with two classes of customer, as discussed in Chapter 4, the dimensions of the irreducible infinitesimal generator of the system, A is given by

$$\frac{1}{2}(c+1)(c+2)\times\frac{1}{2}(c+1)(c+2) \tag{7.3}$$

and for a non-preemptive service model, also discussed in Chapter 4, it is

$$\left(c^2+c+1\right)\times\left(c^2+c+1\right) \tag{7.4}$$

where c = single queue size.

Consider Table 7.1 comparing the size of A for the respective systems where C = capacity of the system.

Clearly the MPDQ will require more exhaustive demands on computational resources than the single queues as computational time is related to the quantity of states in each model, and the processing power and vector machines of the computer used. The algorithms used to solve such systems directly through a program such as Maple is well beyond simplicity for large models. For the single queue, this method

is simple for most small to medium sized systems ($C < 100$), however the rapidly increasing size of the system increases the systems CPU time and memory requirements when solving for the steady state probabilities.

Exact analysis cannot be undertaken if we consider a MPDQ with more than two classes. Further research is needed to attempt to solve a mixed network of MPDQ and FIFO nodes. Again these systems are complicated when more than two classes are considered. In turn the waiting time is not possible to derive using exact analysis. Whilst a recursive algorithm to determine the moments of the waiting time distribution for each class of customer in a two-class system was shown in Section 5.6, placing a MPDQ within a network complicates these analyses, so the solutions are no longer achievable through these techniques. Fortunately, simulations can be used and are undertaken here. Furthermore, using simulation, comparisons can be undertaken for the single and dual queue schemes for more than two priorities with varying scheduling disciplines and for a variety of performance characteristics.

The simulation work begins by looking at the non-preemptive MPDQ.

7.3.2 Determining a Simulation in Steady State

In Part III, it was shown using non-trivial algorithms, that a steady-state solution for the MPDQ is possible. However, as discussed in the previous section, the computation becomes exhaustive in large systems. As the waiting space increases, the states in the system increase rapidly. Nonetheless, the analytical results are a precise solution, based on the long-term steady-state Markovian birth and death processes.

There are procedures used to determine if variables in a non-terminating simulation have reached steady state. The questions in simulation are firstly, how long should the warm-up period be, and secondly, when to terminate. In this Chapter, two approaches are used based on Welch's procedures ([Welch, 1981], [Welch, 1982]) to determine the answers to these questions. For discussion on the warm-up period, a treatment is given in [Law and Kelton, 1982] in section 9.5. The authors discuss the simplest procedure devised by Welch - the graphical technique - where its specific goal is to determine the time when the expected value of a simulation variable is approximately equal to the estimated steady state mean of that variable. This determined when the transient mean curve 'flattens' out. Specific details of this procedure are detailed on pp. 546-565 of [Law and Kelton, 1982]. As suggested in [Law and Kelton, 1982], Welch's four step procedure was undertaken in this thesis. The process here involved five initial replications of length 50,000, well above what was anticipated, in-line with Welch's methodology. The variable used for calculating the warm up goal was the average loss for class 1 customers. Figure 7.4 details the mean class 1 loss by simulation length.

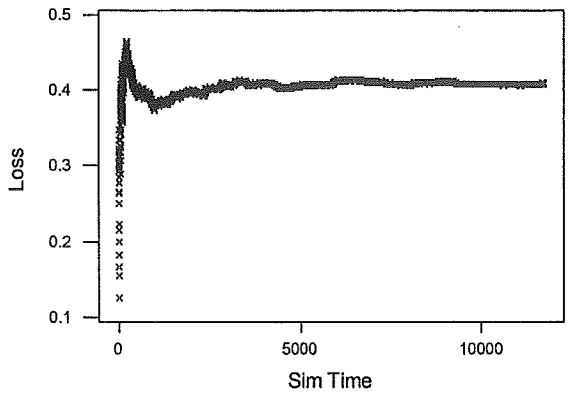

Figure 7.4: Loss estimator over simulation time for pilot simulation

The output is restricted to a little over 10,000 simulation seconds as the variable does not deviate beyond that illustrated. The suitable warm up period was determined to be a little below 10,000. The second process is the quantity of replications. [Law and Kelton, 1982] discusses this process (pp. 536-540) with the comparison of measured and estimated error for our variable, in this case, loss. The calculations are as follows, as described in [Law and Kelton, 1982]. To determine the number of replications n assuming that the estimate of variation $S^2(n)$ does not change appreciably as n increases, an approximate number of total replications, $n^*_\alpha(\beta)$ is given by

$$n^*_\alpha(\beta) = \min\left\{ i \geq n : t_{i-1,1-\frac{\alpha}{2}} \sqrt{\frac{S^2(n)}{i}} \leq \beta \right\}.$$

This is determined iteratively. For the simulations undertaken in this chapter, $n^*_\alpha(\beta)$ was determined after first running a pilot first to determine a suitable warm up period (using Welch's method), then the above was calculated. For this thesis, $\beta = 0.01$ and $\alpha = 0.05$. The results are given in Table 7.2, and all simulations were run using these quantities. So for the 3-class npMPDQ, 10 replications were required, and the 5-node network, 10 replications were also required.

7.4 Performance Characteristics of Different Queueing Regimes for the Preemptive MPDQ

The investigation in this section focuses solely on determining the differences in performance characteristics in the queueing regimes for the preemptive MPDQ. The per-

Classes	npMPDQ	preMPDQ	SQ	Nodes	Network
2	7	6	8		
3	10	5	27	3	13
4	11	5	25	4	11
5	5	5	5	5	10

Table 7.2: Replications, n, required for specific systems

Model	Arrivals		Service		Load
	λ_1	λ_2	μ_1	μ_2	ρ
I	1	2	1	2	2
II	0.8	1.6	1	2	1.6
III	0.3	0.6	1	2	0.6
IV	0.1	0.2	1	2	0.2

Table 7.3: Arrival and Service Rates

formance criteria used to evaluate the differences in regimes are waiting time in the system, loss, and the maximum loss. This model is discussed first to align with the progression of the steady-state analysis in Part III.

7.4.1 Outline

The arrival and service rates used in the models analysed in this section are given in Table 7.3. The load of the system (ρ) decreases down the table. The arrival rates are decreased down the table, with the class 1 customer allocated the lower arrival rate for each model. The queue size was set at 4.

7.4.2 Performance Characteristics

The following four tables are a summary of the classwise characteristics by queueing regime. First, the waiting time in the system by class is analysed, then its maximum, followed by average and maximum loss.

From Table 7.4, for all of the models the waiting time in the system for Class 1 customers is lowest under the HCF regime. For Class 2 customers, the LCF regime is the best of the regimes, with no overlap of the confidence intervals. The margin between the times decreases as the traffic intensity increases. To further solidify the differences, 2-sample t-tests were conducted between each regime of the waiting times for each class of customer. Each test was 1-tailed to conclude if the average waiting time for the lower of the two schemes was statistically significant. It was determined that:

Regime	W_s^{class}	Model I	Model II	Model III	Model IV
HCF	1	2.28 (2.25,2.31)	2.20 (2.17,2.24)	1.57 (1.53, 1.61)	1.17 (1.12, 1.23)
	2	3.04 (3.01,3.07)	2.77 (2.74,2.80)	1.39 (1.35,1.44)	.74 (.69,.80)
FIFO	1	3.10 (3.06,3.14)	2.96 (2.91,3.00)	1.77 (1.73,1.82)	1.23 (1.17,1.30)
	2	2.59 (2.55,2.63)	2.44 (2.39,2.48)	1.29 (1.25,1.33)	.72 (.66,.78)
LCF	1	4.59 (4.53,4.65)	4.13 (4.07,4.19)	1.94 (1.90,2.00)	1.26 (1.19,1.32)
	2	1.87 (1.81,1.93)	1.77 (1.71,1.83)	1.09 (1.05,1.13)	.69 (.62,.76)
LIFO	1	2.98 (2.89,3.08)	2.84 (2.76,2.93)	1.73 (1.68,1.79)	1.21 (1.15,1.28)
	2	2.60 (2.50,2.70)	2.38 (2.32,2.44)	1.22 (1.16,1.27)	.69 (.63,.75)

Table 7.4: Waiting time in the system by class and queue regime

Regime	M_s^{class}	Model I	Model II	Model III	Model IV
HCF	1	14.52	11.62	12.33	8.29
	2	17.54	17.72	14.54	13.68
FIFO	1	14.98	13.13	12.86	11.89
	2	14.97	13.84	11.16	13.68
LCF	1	19.85	18.05	12.14	7.52
	2	13.11	11.89	8.98	13.68
LIFO	1	103.42	78.08	19.48	12.75
	2	80.74	86.17	25.66	13.68

Table 7.5: Maximum waiting time in the system by class and regime

- HCF Class 1 W_s was lower than all other queueing regimes for Models I, II and III, returning $p-$values of 0.000 for every test.
- For Model IV all the models could not be separated at 5% error level. This indicates that at low traffic intensity, the priority regimes offer no advantage in terms of waiting times.
- At low traffic intensity, class 2 waiting times could not be separated statistically. Hence priority offers no advantage.
- For the high load models, LCF Class 2 W_s was lower than all other queueing regimes for Models I, II and III, returning $p-$values of 0.000 for every test.

The maximum waiting time statistic $\left(M_s^{class}\right)$ is useful in determining if any extremely large waits occur for any customers under different regimes. The LIFO discipline tends to disadvantage customers at the front of the queue, on some occasions leaving customers waiting for enormous amounts of time (this shall also be seen later). LIFO is clearly the poorest of the four queueing regimes. Once again, the HCF regime

Regime	$\overline{Loss}^{class}$	Model I	II	III	IV
HCF	1	.513	.406	.040	.001
	2	.508	.413	.040	.000
FIFO	1	.489	.399	.047	.000
	2	.497	.401	.045	.000
LCF	1	.502	.405	.037	.000
	2	.504	.403	.038	.000
LIFO	1	.507	.403	.040	.000
	2	.501	.407	.038	.000

Table 7.6: Average loss by class and regime

provides good upper waiting time control for Class 1 customers. Class 2 customers maximum waiting time is lowest for the LCF regime. Class 2 customers do not experience great waits in the HCF regime. The advantage to Class 1 customers comes at the expense of Class 2 customers, remembering preemption ejects lower class whenever possible.

The average loss $\left(\overline{Loss}^{class}\right)$ is a useful statistic in determining acceptable QoS to customers. This statistic is the mean loss over the simulation period. The 95% confidence intervals are not given in this table as the standard error for all of the models was extremely small, all being less than 0.0002. As seen in earlier sections, loss values are marginally different for different regimes. From Table 7.6 the difference between loss probabilities for the two classes is small, most being ≤ 0.005. Across regimes, for Class 1 and 2 customers, the FIFO provides the lowest loss in Models I and II. In fact the HCF regime has the highest loss, being a little over 2% more than FIFO in Model I. As undertaken earlier, 2-sample t-tests were conducted between each regime for the loss for each class of customer. Each test was 1-tailed to conclude if the lower loss of the two schemes was statistically significant. It was determined that:

- No statistical difference in the four loss probabilities for Model III and IV for both classes.
- For Model I, FIFO class 1 loss was superior, with the HCF returning the poorest loss against all regimes and classes (all statistically significant at 5% error level)
- For Model II, there was no statistical difference in loss for class 1 for all regimes, however HCF class 2 loss was deemed statistically higher than the other schemes.

The maximum loss $\left(Ml_s^{class}\right)$ is the highest recorded loss over the entire simulation period. This is useful in determining the peak loss for customers experienced in the

Regime	Ml_s^{class}	Model I	II	III	IV
HCF	1	.539	.465	.125	.003
	2	.519	.541	.111	.000
FIFO	1	.545	.467	.110	.002
	2	.512	.441	.064	.000
LCF	1	.526	.420	.125	.000
	2	.538	.422	.064	.001
LIFO	1	.556	.455	.176	.000
	2	.561	.450	.063	.000

Table 7.7: Maximum loss by class and regime

simulations. The results are varied as can be seen in Table 7.7. There exists larger differences in class-wise loss than in Table 7.6. In general, maximum Class 1 loss is higher than maximum Class 2 loss. For Model I, LIFO shows the worst loss levels. For Model II, HCF is the worst. Model III has LIFO Class 1 and HCF Class 2 the worst. Finally, Model IV has HCF Class 1 and LCF Class 2 the worst.

So bringing together the findings from Tables 7.4-7.7, and the t-tests, we can say that the HCF provides sufficient advantages in waiting times over the non-priority schemes to warrant its implementation. This does come at the expense of greater loss. Table 7.4 is the most important for customers as long waiting time is a cause of latency and jitter in many real-time web-based applications. An overall decrease in the difference in the statistics is noticeable as the Model number increases. This should come as no surprise as the traffic intensity, ρ, becomes smaller. So the smaller differences therefore should not be discarded as the traffic volume is high. In Model I, the Class 1 and 2 arrival rates, $\lambda_{1,2}$ are both one-tenth that of Model IV's. Hence the traffic intensity in Model IV is 40 times that of Model I's. So a small difference in loss may still lead to a large quantity of customers lost.

7.4.3 CDF of Class 1 Waiting Times

The CDF of waiting time (system) for Class 1 customers is now investigated for pre-emptive customers. In these models all of CDF show reduced waiting time under the HCF regime. In Figures 7.5 to 7.8, the probability of waiting time in the system by waiting times is given. As ρ decreases, the difference between the curves decreases. The next best waits occur for LIFO, then FIFO and the poorest is LCF.

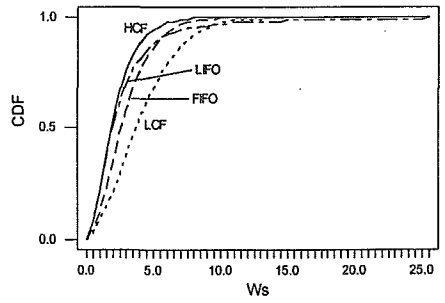

Figure 7.5: CDF of Model I by Regime

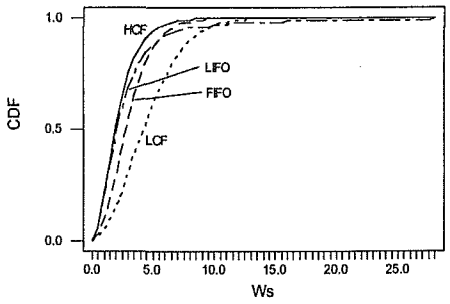

Figure 7.6: CDF of Model II by Regime

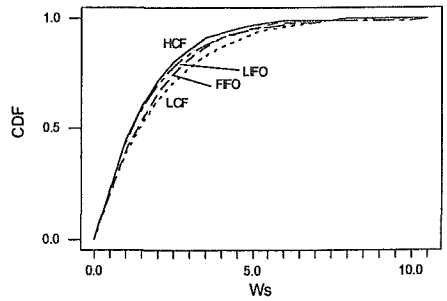

Figure 7.7: CDF of Model III by Regime

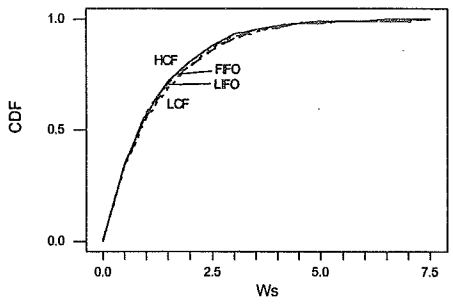

Figure 7.8: CDF of Model IV by Regime

7.4.4 Concluding Remarks

It may be that service providers determine the difference is marginal and the priority system too complicated to introduce such a scheme over traditional FIFO. Nonetheless the preemptive MPDQ under the HCF discipline provides superior delay times over the other three queueing schemes. Further, as the traffic load decreases, so do the improvements, hence the MPDQ performs better when the traffic intensity is high. To summarise, the key findings in this section on the preemptive MPDQ include:

- HCF returns the best waiting times for class 1 customers in periods of high traffic intensity
- At low traffic intensity, the priority regimes offer no advantage in terms of waiting time.
- For the high load models, LCF Class 2 W_s was lower than all other queueing regimes
- The CDF of waiting times in the system showed HCF to be superior over all other regimes

7.5 Performance Characteristics of Different Queueing Regimes for Non-Preemptive MPDQ versus the Single Finite Buffer

In this section the investigation begins with the loss probabilities and utilisation levels for various arrival and service rate combinations for the MPDQ and the single finite buffer model. Up to 5 classes are simulated and comparisons are made across queueing disciplines. The analysis commences with the non-preemptive model which is the simpler of the two service disciplines to simulate. Further, it is also simpler to implement practically in both communications and real life systems as there is no resorting process of the queues or continual checking of who is in the waiting room as in the preemptive case.

7.5.1 Outline

In this section a fixed buffer size with various queueing disciplines for customers of different classes is analysed along with the MPDQ. Recall one of the overall aims of this thesis which is to combine the dual queue idea with that of a priority scheme in the anticipation that prioritised traffic coupled with the dual queue will enhance QoS for customers. This analysis begins the investigation beyond two classes of customers. As described earlier, there are a variety of queueing disciplines that will be investigated here, namely First In First Out (FIFO), Last In First Out (LIFO), High Class First (HCF), and Low Class First (LCF). As the analysis undertaken in Chapter 6 is solely for HCF for two classes solved analytically, these results will provide a useful insight into the behaviour of other schemes with more than to classes of customers. We note for the previous section that HCF was superior in delivering QoS to class 1 customers without enormous expense to class 2 customers, in terms of waiting times. We also recall that non-preemptive MPDQ proved superior to preemptive MPDQ from the analysis in Chapter 6 for two classes of customers. The analysis in this section begins with two-class schemes and extends up to five-class schemes.

7.5.2 Models and Queueing Disciplines

The operational aspects of the models was covered in Chapter 3. As there are multiple classes of customers simulated here, the arrival rates were assigned based on decreasing demand as class increases. The first class customer here is to be considered of the utmost importance when considering time and loss constraints. It was chosen to represent customers that if not given fast service, and indeed first service, will have a detrimental impact on the QoS. This has importance if considering customer appli-

Model	λ_1^{-1}	μ_1^{-1}	λ_2^{-1}	μ_2^{-1}	λ_3^{-1}	μ_3^{-1}	λ_4^{-1}	μ_4^{-1}	λ_5^{-1}	μ_5^{-1}
I	5	1	2	0.2						
II	15	2.5	10	1.5	5	0.5				
III	60	5	30	2.5	15	1.5	5	0.5		
IV	120	10	60	5	30	2.5	15	1.5	5	0.5

Table 7.8: Mean time between arrivals and mean time between service for the models

cations like data transmitted from a live event, typically a streaming video or audio type across the internet or an emergency phone call.

The exponential distribution for both the arrival rate and the service rate of customers. By considering two to five classes, comparisons can be made on how the introduction of more classes changes the performance of the system.

The dual queue model is combined here with prioritised traffic so as to investigate a splitting of what may be viewed as unfairness. By having a dual queue in place, the strong bias towards HCF and LCF models to their respective prioritised customers allows for some traffic of a lower class to move through the queues, unlike a FIFO or LIFO single queue model.

Simulation Set Up

Table 7.8 contains the mean time between arrivals and mean process times for the four models used here. Model I contains 2 classes, Model II, 3 classes and so on. Arena requires service rates in terms of *process time* and arrival rate as *time between arrivals*. To change these Arena requirements to those conventional in queueing theory, the reciprocal of the Arena values gives the appropriate mean arrival and service rate (e.g. Arena *time between arrivals* for Model I class 1 of 5 every second = mean arrival rate of 0.2 per second).

When considering the differing queueing disciplines for each system the FIFO is taken as the baseline. As FIFO by definition avoids all re-organisation of customers, it is the simplest to implement. It would be a poor choice to employ a prioritised method (of a more complicated nature scheduling-wise) if the enhancements were marginal over a FIFO discipline.

7.5.3 Performance Characteristics

For each of the models, the probabilities on the basis of single versus dual queue are investigated, followed by class wise research, and finally a summary of the findings thus far.

	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^1$	$\overline{Loss}^2$	Ml^1	Ml^2	Ml^1	Ml^2
Regime	single		dual		single		dual	
HCF	0.052	0.053	0.062	0.064	0.076	0.060	0.098	0.137
FIFO	0.053	0.057	0.053	0.057	0.072	0.079	0.072	0.079
LCF	0.026	0.031	0.039	0.042	0.038	0.040	0.058	0.089
LIFO	0.062	0.065	0.048	0.047	0.100	0.111	0.091	0.130

Table 7.9: Average and Maximum Class Loss by regime: Model I

	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^1$	$\overline{Loss}^2$
Regime	single		dual	
HCF	0.052,0.0522	0.053,0.0532	0.0623,0.0626	0.0642,0.0644
FIFO	0.0528,0.053	0.0569,0.0571	0.0524,0.0527	0.0564,0.0567
LCF	0.026,0.0263	0.0305,0.0307	0.0391,0.0393	0.0414,0.0416
LIFO	0.062,0.0623	0.0651,00653	0.0475,0.0477	0.0467,0.0469

Table 7.10: Confidence intervals for classwise loss: Model I

Model I - 2 Classes

From Tables 7.9 and 7.11, the performance characteristics for the 2-class model are given. First, to compare the single and dual queue designs. When looking at the average loss by class probabilities and overall average loss, the single queue design is superior for all regimes with the exception of LIFO. The maximum loss by class and overall maximum loss probabilities again show that the single queue design is superior for all regimes.

When considering utilisation, the probabilities are close for both single and dual queue designs. If a low utilisation is desirable, the HCF model is the best, being marginally superior to the LIFO regime. For this Model, the single queue outperforms the dual queue.

When considering the performance of traffic class wise, the results are varied. On average loss by class, Class 1 performs best under a LCF regime, for both single and dual queues. The loss levels are vastly superior under LCF than any other regime, with an average loss of only 0.026 for class 1 customers, half that of the next best loss level. Two-sample t-tests confirmed this, with LCF loss the lowest of all the regimes at the 5% significance level for both classes and types of queue. Tables 7.10 and 7.12 give the 95% confidence intervals for classwise loss and overall loss respectively. The same conclusion can be drawn when considering maximum loss.

For this Model, it is clear that the dual queue offers no advantages over the single queue designs for the arrival and service rates tested in terms of loss and utilisation. The LCF design offers the lowest class wise loss levels and overall loss levels under a

Regime	$\overline{Loss}$	Ml	$\overline{Loss}$	Ml	U	U
	single		dual		single	dual
HCF	0.056	0.064	0.068	0.15	0.937	0.938
FIFO	0.059	0.083	0.059	0.083	0.941	0.941
LCF	0.031	0.041	0.043	0.086	0.954	0.952
LIFO	0.069	0.122	0.05	0.174	0.938	0.944

Table 7.11: Utilisation and Average and Maximum Loss by regime: Model I

Regime	$\overline{Loss}$	$\overline{Loss}$
	single	dual
HCF	0.0559,0.0561	0.0683,0.0685
FIFO	0.0588,0.059	0.0588,0.059
LCF	0.0304,0.0306	0.0427,0.0428
LIFO	0.0689,0.0691	0.0494,0.0497

Table 7.12: Confidence intervals for overall loss: Model I

single queue design. The LCF single queue design is the best choice for this Model, with significance tests justifying this conclusion. With only 2 classes, preliminary analysis revealed the dual queue offers no advantages to traffic in terms of loss and utilisation, with the exception of the LIFO regime, again justified by two-sample t-tests. FIFO behaves identically as it must for validity - there is no difference in the single and dual queues under this regime.

Model II - 3 Classes

With the introduction of a third class of customer, the MPDQ designs exhibit major improvements over the single queueing designs. Tables 7.13, 7.16 and 7.14 now depict a reversal of the losses seen in the 2-class models. The loss probabilities improve dramatically in favour of the dual queueing models. The levels of loss are at their lowest for the priority schemes in comparison to the single models. Tables 7.14, 7.18 and 7.15 exhibit the 95% confidence intervals for classwise mean loss and overall mean loss respectively.

The dual queueing designs are superior over the single scheme for almost all performance characteristics. Of the dual queueing designs, the best queueing regime in terms of average loss by class probabilities and overall average loss is LCF. For maximum loss by class and overall maximum loss probabilities, the dual queue is clearly superior over the single queue designs for all queueing regimes. Noticeably, the priority regimes show the largest reduction in loss from single to dual queue designs.

On a class basis, again the LCF is superior, but only for the dual queue design. If

Regime	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$
		single			dual	
HCF	0.132	0.126	0.145	0.050	0.062	0.062
FIFO	0.116	0.135	0.122	0.116	0.135	0.122
LCF	0.148	0.142	0.156	0.039	0.041	0.047
LIFO	0.165	0.115	0.122	0.052	0.053	0.059

Table 7.13: Average Class Loss by regime: Model II

	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$
Regime		single	
HCF	0.132,0.133	0.126,0.127	0.145,0.146
FIFO	0.115,0.116	0.135,0.135	0.121,0.122
LCF	0.147,0.148	0.141,0.142	0.156,0.156
LIFO	0.165,0.165	0.115,0.116	0.121,0.122

Table 7.14: Confidence intervals for classwise loss: Model II

we were confined to using a single queue design, the LCF is one of the poorest designs in terms of class wise loss. When considering maximum loss by class, the HCF is best for Class 1, LIFO for Class 2, and LCF for Class 3. The FIFO regime is the best for single queue designs, yet poorest in the dual queue designs. The overall loss levels are poor for the single queue schemes.

Most of the dual queue designs, with the exception of maximum loss under FIFO, show considerable improvement over single queue designs. Utilisation is also lower in the dual schemes. This is an important factor when considering the systems ability to run near capacity. The average loss of class points to the LCF model as the best. However it is the HCF which has the lowest maximal loss. It may be difficult to decide which of LCF and HCF is best for 3 classes. In overall terms, the HCF is best, with superior maximum loss rates and utilisation and comparable overall maximum loss. Two-sample t-tests confirm these findings. All the mean overall loss levels were significantly different at 5% error level, with FIFO the best for both single and dual

	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$
Regime		dual	
HCF	0.0501,0.0505	0.0616,0.062	0.0616,0.0619
FIFO	0.115,0.116	0.135,0.135	0.121,0.122
LCF	0.0386,0.0391	0.0408,0.0412	0.0471,0.0475
LIFO	0.0514,0.0518	0.053,0.0534	0.0588,0.0592

Table 7.15: Confidence intervals for classwise loss: Model II

Regime	Ml^1	Ml^2	Ml^3	Ml^1	Ml^2	Ml^3
		single			dual	
HCF	0.181	0.175	0.207	0.064	0.088	0.081
FIFO	0.145	0.160	0.144	0.145	0.160	0.144
LCF	0.174	0.160	0.210	0.097	0.080	0.069
LIFO	0.214	0.138	0.156	0.094	0.076	0.070

Table 7.16: Maximum Class Loss by regime: Model II

	$\overline{Loss}$	Ml	$\overline{Loss}$	Ml	U	U
Regime	single		dual		single	dual
HCF	0.162	0.227	0.064	0.083	0.984	0.927
FIFO	0.144	0.174	0.063	0.144	0.174	0.063
LCF	0.180	0.227	0.047	0.082	0.993	0.940
LIFO	0.150	0.185	0.060	0.077	0.988	0.937

Table 7.17: Utilisation and Average and Maximum Loss by regime: Model II

Regime	$\overline{Loss}$	$\overline{Loss}$
	single	dual
HCF	0.161,0.162	0.064,0.0643
FIFO	0.143,0.144	0.0624,0.0631
LCF	0.179,0.18	0.0464,0.0468
LIFO	0.149,0.15	0.0598,0.0601

Table 7.18: Confidence intervals for overall loss: Model II

Regime	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$
	single				dual			
HCF	0.092	0.166	0.130	0.144	0.020	0.043	0.036	0.028
FIFO	0.111	0.133	0.137	0.137	0.111	0.133	0.137	0.137
LCF	0.108	0.087	0.115	0.091	0.011	0.012	0.019	0.019
LIFO	0.085	0.078	0.064	0.083	0.067	0.043	0.024	0.034

Table 7.19: Average and Maximum Loss by regime and Class: Model III

Regime	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$
	single			
HCF	0.0919,0.0927	0.165,0.167	0.13,0.131	0.144,0.145
FIFO	0.11,0.113	0.133,0.134	0.136,0.137	0.137,0.138
LCF	0.107,0.108	0.0866,0.0876	0.115,0.116	0.0908,0.0916
LIFO	0.0845,0.0856	0.0771,0.0778	0.0637,0.0646	0.0829,0.0836

Table 7.20: Confidence intervals for classwise loss: Model III

queues.

Model III - 4 Classes

The introduction of another class strengths the case for the HCF and LCF models. The loss probabilities (Tables 7.19 - 7.24) are best for the priority disciplines when considering maximum loss. The 95% confidence intervals indicates HCF is best in terms of loss especially in the MPDQ for class 1 customers. Overall, the LCF is particularly good. In both the HCF and LCF models, the middle classes suffer the highest lost. The nature of arrivals for the other schemes does not see such results.

When considering overall loss of the system, the priority schemes are the best. Notice the high levels of loss for the single class schemes, which are far above acceptable levels especially considering the maximum loss levels.

Two sample t-tests indicate for 4 classes, that overall loss is lowest for the HCF MPDQ and highest for the HCF single queue. Further, the MPDQ lowers loss in

Regime	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$
	dual			
HCF	0.02,0.0205	0.0427,0.0436	0.0358,0.0364	0.028,0.0285
FIFO	0.11,0.113	0.133,0.134	0.136,0.137	0.137,0.138
LCF	0.0113,0.0116	0.0122,0.0126	0.0192,0.0196	0.0186,0.0189
LIFO	0.0664,0.0673	0.0424,0.0433	0.0238,0.024	0.034,0.0344

Table 7.21: Confidence intervals for classwise loss: Model III

Regime	Ml_1	Ml_2	Ml_3	Ml_4	Ml_1	Ml_2	Ml_3	Ml_4
	single				dual			
HCF	0.125	0.333	0.227	0.324	0.039	0.063	0.057	0.063
FIFO	0.6	0.5	0.162	0.183	0.6	0.5	0.162	0.183
LCF	0.2	0.117	0.15	0.122	0.031	0.027	0.04	0.030
LIFO	0.125	0.106	0.095	0.17	0.148	0.154	0.046	0.088

Table 7.22: Average and Maximum Loss by regime and Class: Model III

Regime	$\overline{Loss}$	Ml	$\overline{Loss}$	Ml	U	U
	single		dual		single	dual
HCF	0.169	0.75	0.033	0.063	0.974	0.907
FIFO	0.16	0.437	0.16	0.437	0.981	0.981
LCF	0.109	0.142	0.018	0.031	0.988	0.928
LIFO	0.087	0.113	0.037	0.1	0.971	0.914

Table 7.23: Average and Maximum Loss by regime and Class: Model III

Regime	$\overline{Loss}$	$\overline{Loss}$
	single	dual
HCF	0.168,0.169	0.0322,0.0327
FIFO	0.159,0.16	0.159,0.16
LCF	0.109,0.109	0.0182,0.0185
LIFO	0.0862,0.087	0.0364,0.0368

Table 7.24: Confidence intervals for overall loss: Model III

Regime	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$	$\overline{Loss}^5$	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$	$\overline{Loss}^5$
			single					dual		
HCF	0	0.005	0.003	0.002	0.002	0.005	0	0	0	0.002
FIFO	0.001	0.007	0.10	0.009	0.005	0.001	0.007	0.010	0.009	0.005
LCF	0	0	0	0	0	0.005	0.001	0	0.001	0.003
LIFO	0	0	0	0.003	0.014	0	0	0	0.001	0

Table 7.25: Average and Maximum Loss by regime and Class: Model IV

comparison to the single queue for all models across all classes, with the obvious exception of FIFO.

Model IV - 5 Classes

The 5-class model was included as a result of a trend appearing for loss. This is that the middle classes are suffering high levels of loss in respect to the other classes in the 4-class models. In the 5-class models, this trend continued on from the 4-class model. The MPDQ scheme this time improved only for classes 3, 4, and 5 over the single queue. It may seem that the MPDQ scheme may have the system too full of middle class customers to allow high-class customers the chance of arrival. The MPDQ scheme may disadvantage the high class in its two-time wait, that is, its transition from the secondary to primary queue. It is becoming a 'rare arrival event' and the single queue HCF model benefits the high class by letting it jump to the front immediately.

All loss probabilities are small, with the LCF operating at the lowest loss levels. All results are found in Tables 7.25 - 7.30. Whilst the confidence intervals are given, clearly there are no huge levels of loss, or large difference in the confidence intervals. An interesting result is the difference between single and dual queue for the HCF. The MPDQ shows the improvement for the 2^{nd}, 3^{rd} and 4^{th} classed customers. As discussed earlier, it would seem the increase in classes sees the decrease in space for the first class of customer. The LIFO model is ultimately the best, and it is beneficial toward rare arrivals as a rare arrival will (usually) find waiting customers in front of them. The LIFO gives the last customer the advantage of jumping the queue, something that benefits these rare arrivals.

The MPDQ model dampens the severity of LIFO model in terms of loss, with it showing no loss for all but the 4^{th} class customers. It should be noted overall that loss levels are small in the table, and for most applications these levels are not of concern.

The LCF model has the worst overall loss when changing from single to dual queue. The others equal or better improvement in loss levels.

Regime	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$	$\overline{Loss}^5$
			single		
HCF	0	0.00482,0.005	0.00303,0.00313	0.00149,0.00153	0.00194,0.00201
FIFO	0.00125,0.00131	0.00701,0.00718	0.00985,0.01	0.00846,0.00859	0.0044,0.00457
LCF	0	0	0	0	0
LIFO	0	0	0	0.00316,0.00324	0.0135,0.0138

Table 7.26: Confidence intervals for classwise loss: Model IV

Regime	$\overline{Loss}^1$	$\overline{Loss}^2$	$\overline{Loss}^3$	$\overline{Loss}^4$	$\overline{Loss}^5$
			dual		
HCF	0.00538,0.00555	0	0	0	0.00152,0.00158
FIFO	0.00125,0.00131	0.00701,0.00718	0.00985,0.01	0.00846,0.00859	0.0044,0.00457
LCF	0.00518,0.00533	0.00099,0.00103	0	0.00124,0.00127	0.00312,0.00321
LIFO	0	0	0	0.00097,0.000997	0

Table 7.27: Confidence intervals for classwise loss: Model IV

Regime	Ml_1	Ml_2	Ml_3	Ml_4	Ml_5	Ml_1	Ml_2	Ml_3	Ml_4	Ml_5
			single					dual		
HCF	0	0.028	0.018	0.003	0.005	0.012	0	0	0.001	0.004
FIFO	0.004	0.015	0.059	0.013	0.011	0.004	0.015	0.059	0.013	0.011
LCF	0	0	0.001	0.001	0	0.015	0.003	0	0.002	0.009
LIFO	0	0	0.001	0.015	0.043	0	0	0	0.001	0

Table 7.28: Average and Maximum Loss by regime and Class: Model IV

Regime	$\overline{Loss}$	Ml	$\overline{Loss}$	Ml	U	U
	single		dual		single	dual
HCF	0.002	0.008	0.001	0.001	0.603	0.601
FIFO	0.008	0.018	0.008	0.018	0.606	0.606
LCF	0	0.001	0.001	0.003	0.601	0.607
LIFO	0.003	0.012	0.001	0.001	0.603	0.6

Table 7.29: Average and Maximum Loss by regime and Class: Model IV

Regime	$\overline{Loss}$	$\overline{Loss}$
	single	dual
HCF	0.00224,0.00227	0.00069,0.0007
FIFO	0.00768,0.00776	0.00768,0.00776
LCF	0	0.00142,0.00144
LIFO	0.00028,0.00029	0.0005,0.00052

Table 7.30: Confidence intervals for overall loss: Model IV

7.5.4 Concluding Remarks

To this point, the best scheme combining priorities with the dual queue is the HCF discipline, especially for three and four classes. LCF and LIFO showed volatility in certain situations. For service providers, the improvement of loss for the MPDQ under a HCF or FIFO discipline is noteworthy. Next we must consider the other QoS criteria - waiting time. Key conclusions to now are:

- MPDQ offers no huge advantage over the single queue in terms of utilisation and loss for three classes of customers
- The MPDQ HCF and LCF regimes for three classes of customers delivers superior maximum loss and utilisation rates. Overall the dual queue improves overall loss in comparison to the single queue.
- Again, good improvements in loss under the HCF MPDQ for four classes. Middle class loss high, especially for second class customers.
- For five classes, marginal differences in loss amongst all models. Middle class loss again evident.

7.6 Waiting Time Approximations for the Non-Preemptive MPDQ

In this section the analysis focuses solely on waiting times for the Non-preemptive MPDQ. Distribution fits are given for high class customers, and the waiting time for high class customers as a function of various low class arrival rates are investigated.

This section provides further insight into the QoS expectations of the MPDQ. In the previous section it was shown that in terms of loss a priority scheme was beneficial for the MPDQ. Now mean waiting times are estimated, and distribution fits of the waiting times for specific customers are estimated.

Further Simulation Design for waiting time analysis

For the waiting time simulations, the raw data needed to be exported from Arena. The analyses were simplified by using the *Batch/Truncate* option in Arena. This feature 'lumps' all the replications together and from this was obtained an overall picture of a typical system at any point through the simulation period. Again, all maximum values refer to the maximum of all simulation runs, not just a single run, for each respective model.

To be consistent for comparison with the prior section, the buffer size/s (waiting space) for arriving customers was again fixed at size 10 for the single, and for a dual queue the size was 5 for each queue. Table 7.8 contains the arrival and service rates for the four models used here.

7.6.1 Waiting time statistics

The waiting times give further clues as to the best model suitable for each number of classes present. In this section both single and dual queue statistics under the same arrival/service rates are presented. The statistics as given in the following subsections tables are as follows:W_q^i = Average waiting time in queue i for any customer, W_s^i=Average waiting time in the system for class i customers, M_s^i=The maximum waiting time in the system for class i customers.

Model I - 2 Classes

When considering the average waiting time for any customer, as seen by (W_q) in Table 7.31, the single FIFO is best for the single queue (at 5% error). For the MPDQ, W_q^1 are not significantly different, however for W_q^2, LCF is statistically lower than all other regimes, again at a 5% error level. This statistic offers a guide to the efficiency of the system. By comparing the single and dual queue waits for any customer, for all

Regime	W_q	W_q^1	W_q^2	W_s^1	W_s^2	W_s^1	W_s^2	M_s^1	M_s^2	M_s^1	M_s^2
	single	dual		single		dual		single		dual	
HCF	6.81	5.06	4.02	3.28	10.2	3.83	10.2	13.8	34.6	12.5	37.3
FIFO	5.1	5.15	3.71	8.99	8.14	8.99	8.14	25.7	24.7	25.7	24.7
LCF	6.41	5.11	3.37	19.9	3.04	19.7	3.49	70.2	12.3	65.8	13.2
LIFO	6.87	5.25	3.79	8.77	8.11	8.72	8.45	386	493	330	410

Table 7.31: Waiting Time Statistics by Class and regime : Model I

Regime	W_q	W_q^1	W_q^2	W_s^1	W_s^2	W_s^1	W_s^2
	single	dual		single		dual	
HCF	6.67,6.94	4.97,5.15	3.91,4.12	3.21,3.34	10.1,10.4	3.76,3.9	10,10.3
FIFO	5.01,5.19	5.1,5.21	3.65,3.78	8.82,9.16	8.03,8.25	8.82,9.16	8.03,8.25
LCF	6.21,6.61	4.96,5.26	3.24,3.49	19.4,20.4	3,3.07	19.2,20.1	3.45,3.52
LIFO	6.41,7.32	4.81,5.69	3.54,4.04	7.96,9.59	7.56,8.65	7.98,9.47	7.9,9.01

Table 7.32: Waiting Time Statistics Confidence Intervals: Model I

disciplines, the sum of waiting times in the dual queue exceeds the single queue times. It must be noted that the MPDQ scheme forces customers to queue in the secondary queue by design, hence simply adding the two queues (primary and secondary) together and directly comparing with a single queue is not advised.

The average waiting time in the system by class statistics (W_s^i) are close for the single and dual queues for each regime, as seen in Tables 7.31 and 7.32. In terms of the highest class or lowest class spending the shortest time possible in the system or queue, the priority disciplines now show distinct advantages over the non-priority schemes. Caution though is needed as there is no statistical difference in the four regimes for class 1 traffic at the 5% error level. However LCF is the lowest for class 2 traffic. In the single case, the HCF queues are clearly the best for Class 1 customers. Class 2 traffic waits the longest under the HCF regime. The LCF scheme is poor toward Class 1 customers in waiting times. LIFO offers the poorest guarantees for maximum waits. In general, the MPDQ shows small improvements over the single queue models.

Whilst in the previous section LCF was the better of the four disciplines when considering loss, this is not the case here. The time spent in the primary queue by any customer is fairly close, whereas for the secondary queue there are varying average waits. Whilst HCF has the fastest average time in the first queue, it has the poorest in the second – this may lie with the fact that primary queue will contain more Class 1 customers as time continues than any of the other models. It is for similar reasons that the LCF has the least waiting time in the secondary queue, as all Class 2 customers are in the primary queue. HCF has the highest average and maximum loss in both Class 1 and Class 2, with LCF boasting the lowest loss for Class 1 and FIFO for Class

Regime	W_q	W_q^1	W_q^2	W_s^1	W_s^2	W_s^3	W_s^1	W_s^2	W_s^3
	single	dual			single			dual	
HCF	20.4	9.74	7.4	6.97	8.08	37.2	6.73	7.12	23.6
FIFO	21.9	9.56	6.85	16.4	15.4	15.1	16.4	15.4	15.1
LCF	23.3	8.81	6.61	105	12.2	5.53	41.5	11.1	6.1
LIFO	20.6	9.64	7.09	21.7	21.6	25.4	14.4	14.3	16.6

Table 7.33: Waiting Time Statistics: Model II

Regime	W_q	W_q^1	W_q^2
	single	dual	
HCF	19.8,21	9.45,10	7.11,7.69
FIFO	21.6,22.1	9.42,9.71	6.68,7.02
LCF	22.1,24.5	8.36,9.26	6.19,7.04
LIFO	18.2,22.9	8.57,10.7	6.48,7.7

Table 7.34: Waiting Time Statistics Confidence Intervals: Model II

2

The HCF model delivers lowest time in the system to Class 1 customers. Interestingly, LCF gives marginally better times for Class 2, however significantly poorer times for Class 1 in the reverse model. Of the non-priority model, LIFO is the 'fairer' of the two, bringing the average time of the two classes to close levels (statistically significant for single queue and dual queue).

Model II - 3 Classes

Now with a third class, the dual queueing scheme improves in terms of waiting time statistics. All combined time in the queue, time in the system, confidence intervals and maximum time's statistics in Tables 7.33 and 7.36 show the value of the MPDQ. Furthermore, the priority models exhibit major improvements in waiting over their non-priority counterparts. HCF is statistically superior against all other regimes for waiting time in the system for class 1 and 2 customers in both the MPDQ and single queue. The LIFO performs poorly, with high maximum time in the system. However if we wish to have evenness in terms of waiting times, it certainly achieves this - at the risk of waiting, on some occasions, around 60 times longer than the other models. When we consider the importance of first class traffic, HCF delivers. The waiting times are over 3 times less than the next best model, and all lower statistically ($p - value = 0.000$). With this swift service of class 1 customers, a flow-on effect occurs for the 2^{nd} class, with this too having the best waiting time statistic of all the models ($p - value = 0.000$). However the 3^{rd} class is the poorest under HCF.

The MPDQ now outperforms the single queue in terms of both average waiting time

Regime	W_s^1	W_s^2 single	W_s^3	W_s^1	W_s^2 dual	W_s^3
HCF	6.73,7.21	7.76,8.4	36.4,37.9	6.49,6.98	6.88,7.37	23,24.2
FIFO	15.9,17	14.9,15.8	14.8,15.4	15.9,17	14.9,15.8	14.8,15.4
LCF	102,108	11.7,12.6	5.44,5.63	39.4,43.7	10.7,11.5	5.99,6.21
LIFO	16.6,26.7	17.6,25.6	21.9,28.8	12.4,16.4	12.7,16	14.8,18.3

Table 7.35: Waiting Time Statistics Confidence Intervals: Model II

Regime	M_s^1	M_s^2 single	M_s^3	M_s^1	M_s^2 dual	M_s^3
HCF	25.5	45.8	111	24.7	38.6	76.4
FIFO	56.2	55.1	51.6	44.6	40.9	42.4
LCF	224	55.7	25	179	47.6	22.8
LIFO	1480	1190	1320	463	460	1100

Table 7.36: Waiting Time Statistics: Model II

for any customer in the queue, and class wise waits in the system. All HCF customers wait for shorter periods in the MPDQ. In fact, nearly all classes under the disciplines wait shorter periods in the MPDQ. The LIFO experiences problems with extreme maximum waiting times. The problem lies with this system as it sees customers never being served. The FIFO shows evenness for all classes, with improvement in the dual queue. For service providers, the decision of FIFO or one of the priority regimes could be governed by either the maximum threshold or average waits. The priorities here offer better average service, whereas the FIFO offer better maximum thresholds. In the prior section the priority regimes were shown to be superior in terms of loss, and combined with the above results, there use is further justified.

Model III - 4 Classes

The introduction of another class strengthens the case for the HCF model. Whilst the 1^{st}, 2^{nd} and 3^{rd} class customers receive marginally better waiting times under the dual queue, the 4^{th} class benefited from significant improvement. This may be an important factor if considering the value of the 'common' class. The HCF stands alone for waiting times, showing at least 50% improvement in waiting times over its rivals for 1^{st} and 2^{nd} classes.

The confidence intervals revealed no statistical difference in W_q^1 for the four regimes. However, the waiting times again were favorable to the HCF. All hypothesis tests showed the HCF to be highly significant in terms of waiting times less than other regimes for W_s^1, W_s^2 and W_s^3 for both single and dual queues.

The MPDQ again offers superior waiting times for all regimes, making it the best

Regime	W_q	W_q^1	W_q^2	W_s^1	W_s^2	W_s^3	W_s^4	W_s^1	W_s^2	W_s^3	W_s^4
	single	dual			single				dual		
HCF	22.3	10	8.26	9.28	7.6	8.15	36.2	9.07	7.13	7.95	21.1
FIFO	22.9	10.1	8.02	20.7	16.6	16.7	15.5	20.7	16.6	16.7	15.5
LCF	23.4	9.68	6.24	258	44.8	17.8	7.69	75.2	32.6	15.3	6.95
LIFO	20.1	9.67	7.15	32.5	25.4	26.4	21.5	16.8	18.7	15.7	14.4

Table 7.37: Waiting Time Statistics: Model III

Regime	W_q	W_q^1	W_q^2
	single	dual	
HCF	21.7,22.9	9.7,10.3	7.9,8.62
FIFO	22.6,23.2	9.86,10.2	7.76,8.28
LCF	21.4,25.4	9.02,10.3	5.79,6.69
LIFO	17.7,22.4	8.64,10.7	6.39,7.91

Table 7.38: Waiting Time Statistics Confidence Intervals: Model III

Regime	W_s^1	W_s^2	W_s^3	W_s^4	W_s^1	W_s^2	W_s^3	W_s^4
		single				dual		
HCF	8.48,10.1	7.14,8.06	7.76,8.55	35.5,36.9	8.34,9.8	6.66,7.59	7.59,8.32	20.6,21.7
FIFO	19.2,22.2	15.7,17.6	16.1,17.3	15.1,15.9	19.2,22.2	15.7,17.6	16.1,17.3	15.1,15.9
LCF	238,278	41.3,48.4	16.9,18.7	7.53,7.85	66.2,84.2	29.9,35.2	14.6,16.1	6.82,7.08
LIFO	19.8,45.1	16.8,34	20.7,32.1	18.8,24.1	12.7,20.9	14.7,22.7	13.3,18.1	13.2,15.7

Table 7.39: Waiting Time Statistics Confidence Intervals: Model III

Regime	M_s^1	M_s^2	M_s^3	M_s^4	M_s^1	M_s^2	M_s^3	M_s^4
		single				dual		
HCF	39.4	49.2	51.9	103	32.4	40.3	56	99.9
FIFO	88.5	74.8	80.2	89.2	54.8	55.1	53.1	52.2
LCF	738	201	87.9	40.3	330	153	104	31.1
LIFO	1170	1100	1150	1490	403	468	567	669

Table 7.40: Waiting Time Statistics: Model III

Regime	W_q	W_q^1	W_q^2
	single	dual	
HCF	6.01	5.33	10
FIFO	7.45	6.61	12.3
LCF	5	5.63	7.79
LIFO	6.31	5.15	9.32

Table 7.41: Waiting Time Statistics: Model IV

Regime	W_q	W_q^1	W_q^2
	single	dual	
HCF	5.66,6.35	5,5.66	8.69,11.4
FIFO	7.12,7.78	6.34,6.89	11,13.5
LCF	4.69,5.31	5.3,5.96	6.46,9.12
LIFO	5.79,6.82	4.76,5.55	7.14,11.5

Table 7.42: Waiting Time Statistics Confidence Intervals: Model IV

choice for four classes. FIFO consistently gives an evenness across classes, which may be preferential to service providers wanting this quality of service criteria. HCF again outperforms all other regimes, both for the single and dual queues.

Model IV - 5 Classes

The 5-class model was included to further investigate a trend appearing for waiting time (and indeed loss). This is that the middle classes are suffering high levels of loss with respect to the low/high classes, yet are receiving shorter waiting times. From Tables 7.41 to 7.46, for the 5 class models, this trend continued. The 2^{nd} and 3^{rd} classes in the HCF model were the fastest through the system. The dual queueing scheme this time improved only for classes 3, 4, and 5 over the single queue. It may seem that this scheme may have the system too full of middle class customers to allow high-class customers the chance of arrival. The dual scheme may disadvantage the high class in its two-time wait. It is becoming a rare event and the single queue benefits high class by letting it jump to the front immediately. With more classes, the overall system is slowed down, hampering waits for the high class customer.

An interesting result is the marginal difference between single and dual queue for the HCF (all tests comparing the two were not significant). The dual queue shows improvement for the 2^{nd}, 3^{rd} and 4^{th} classes in the dual model. As discussed, it would seem the increase in classes sees the decrease in quality for the first class of customer. The LIFO is beneficial to rare arrivals as a rare arrival will usually find waiting customers in front of them. The LIFO gives the last customer the advantage of jumping the queue, something that benefits the rare arrives here, especially under

Regime	W_s^1	W_s^2	W_s^3	W_s^4	W_s^5	W_s^1	W_s^2	W_s^3	W_s^4	W_s^5
			single					Dual		
HCF	12.5	8.54	8.48	10.5	13.6	14.1	8.79	8.45	10.2	12.6
FIFO	19.5	13.6	11.8	11	9.18	19.5	13.6	11.8	11	9.18
LCF	23	15.8	9.82	6.74	4.48	27.3	17	10.7	7.25	4.75
LIFO	17.2	11.7	10.4	9.91	9.96	16.1	12.2	9.74	8.93	8.08

Table 7.43: Waiting Time Statistics: Model IV

Regime	W_s^1	W_s^2	W_s^3	W_s^4	W_s^5
			single		
HCF	11.4,13.6	7.96,9.13	8.02,8.93	9.99,11	11.8,15.5
FIFO	17.7,21.3	12.6,14.6	11.1,12.4	10.5,11.5	8.18,10.2
LCF	20,26	14.3,17.4	9.25,10.4	6.48,7	3.96,5
LIFO	15.2,19.3	10.5,13	9.45,11.3	9.13,10.7	7.89,12

Table 7.44: Waiting Time Statistics Confidence Intervals: Model IV

Regime	W_s^1	W_s^2	W_s^3	W_s^4	W_s^5
			Dual		
HCF	12.7,15.4	8.09,9.48	7.97,8.93	9.68,10.7	10.6,14.6
FIFO	17.7,21.3	12.6,14.6	11.1,12.4	10.5,11.5	8.18,10.2
LCF	24.2,30.4	15.4,18.5	10.1,11.3	6.99,7.51	4.33,5.16
LIFO	14.2,17.9	10.8,13.6	8.93,10.5	8.35,9.51	6.68,9.48

Table 7.45: Waiting Time Statistics Confidence Intervals: Model IV

Regime	M_s^1	M_s^2	M_s^3	M_s^4	M_s^5	M_s^1	M_s^2	M_s^3	M_s^4	M_s^5
			single					dual		
HCF	53.8	60.4	65	112	198	63.6	74.7	86.9	105	195
FIFO	84.3	79.8	82.1	81.6	82.3	84.3	79.8	82.1	81.6	82.3
LCF	164	172	112	99	87.9	134	118	85.6	59.5	38.1
LIFO	120	176	192	325	318	164	177	137	236	191

Table 7.46: Waiting Time Statistics: Model IV

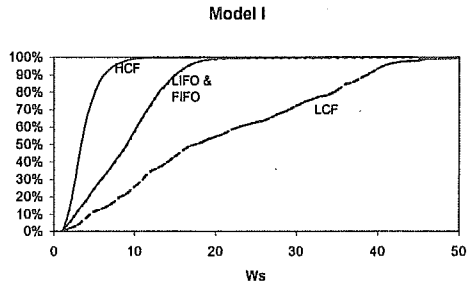

Figure 7.9: CDF of waiting times for Class 1 Model I

the dual queue.

Waiting Time Summary

From the prior section, on the performance characteristics in terms of loss, it was concluded that the dual queue scheme for three or more classes gave better results the single queueing schemes. Furthermore, the HCF regime was the best of the four, reducing waiting times substantially when compared to other models.

7.6.2 Waiting Time Distributions for HCF using the MPDQ

This subsection is arranged into three areas of analysis. Firstly, for each of the four models for the dual queueing scheme, the cumulative distribution function is calculated using a distribution fit for the highest class only. Secondly a probability distribution fit for the dual HCF customers across models is given, and findings compared. Finally, for Models I-IV, the arrival rate of the lowest class was varied and assessed. In this way, the effect of a small to large mean arrival rate of the least important customer can be compared.

CDF of First Class Customers in the Dual Queue

The distribution functions given in Figures 7.9 to 7.12 are constructed using the results from the simulations described earlier. For Models I-IV, the first-class customers CDF is given for each of the queueing disciplines.

As the arrival and service rates are different for the models, we cannot make direct model comparisons. However, from Figure 7.12, the CDF's are closest to each other, and move away as the models decrease in number of classes (i.e. from 4 to 2 classes). This is consistent with the preemptive analysis which showed that as traffic intensity lowered, so did the improvement in the waiting times. The LCF discipline is poorest,

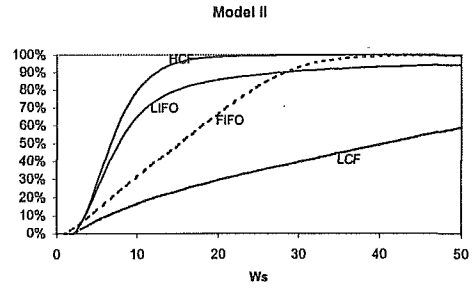

Figure 7.10: CDF of Waiting Times for Class 1 Model II

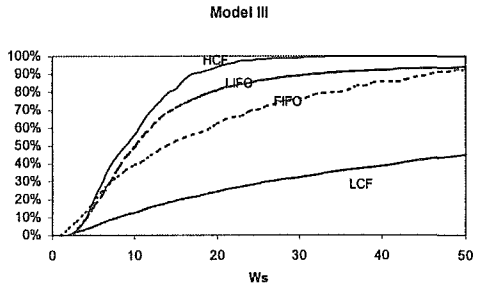

Figure 7.11: CDF of Waiting Times for Class 1 Model III

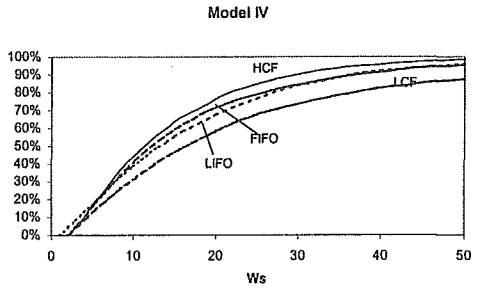

Figure 7.12: CDF of Waiting Times for Class 1 Model IV

Model	Distribution	Square Error	χ^2 p-value	K-S Test p-value
HCF (IV)	$\Gamma(10.1, 1.25)$	0.000345	0.145	> 0.15
HCF (III)	$\Gamma(4.27, 1.89)$	0.004952	0.136	> 0.15
HCF (II)	$\Gamma(2.61, 2.11)$	0.000597	< 0.005	0.00
HCF (I)	$\Gamma(1.37, 2.07)$	0.000671	0.0086	0.07

Table 7.47: Distribution fits for HCF Class 1 by Model

especially in Models II and III (Figures 7.10 and 7.11). The HCF shows superior waiting time probabilities for all models, however the margin closes between disciplines as the classes increase.

When comparing the LIFO and FIFO queues, the LIFO performs well through the middle of the CDF. However the presence of customers at the front of the queue (i.e. the first-in customers), sees the presence of extremely large waiting times in times of congestion, and the LIFO CDF is asymptotic to its upper tail. This has been found to be the case in other queueing models, such as M/G/1 LIFO, where approximations were used [Abate and Whitt, 1997]. In the previous section the LIFO had the lowest levels of loss in many of the four Models. For an increase in QoS, the introduction of a time-out discard limit would increase the loss, but should also have the effect of reducing these 'extreme values' for LIFO waiting times.

Distribution Fit for DQ HCF Class 1

To model the CDF, a distribution fit for each of the HCF curves was undertaken. The curve fitting is undertaken using maximum likelihood estimators. The choice of distribution was made by choosing the distribution with the smallest squared error. The results of Chi-square and Kolmogorov-Smirnov goodness-of-fit tests are also shown in Table 7.47. These results are presented in the form of $p-values$; with the $p-value$ being the largest value of the type-I error probability that allows the distribution to fit the data.

The best fit for each model was the gamma distribution. In Figure 7.13, the plots for the distributions given in Table 7.47 are displayed. In the table, $\Gamma(\alpha, \beta)$ refers to the gamma distribution with location parameter α and scale parameter β. The technique of determining the final values of α and β in the function follow a numerical scheme for $\alpha < 1.5$ (see [Kelton et. al., 2002], [Philips and Beightler, 1972]). For $\alpha \geq 1.5$, three inverted approximations to the gamma distributions based on the Burr family of distributions are used, depending upon the values of α (see [Wheeler, 1975], [Tadikamalla, 1978]). In addition, to calculate the natural logarithm of the $\Gamma(Y)$ function, a polynomial approximation is used [Pike and Hill, 1966].

The results show that as the Models have more classes, the positive skewness

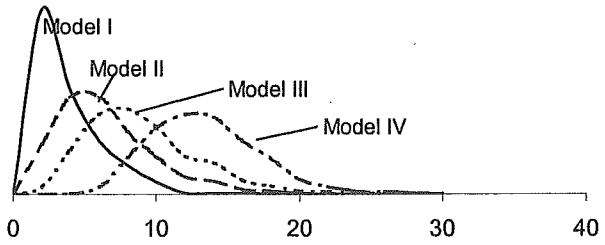

Figure 7.13: Waiting time distribution for HCF Class 1, Models I-IV

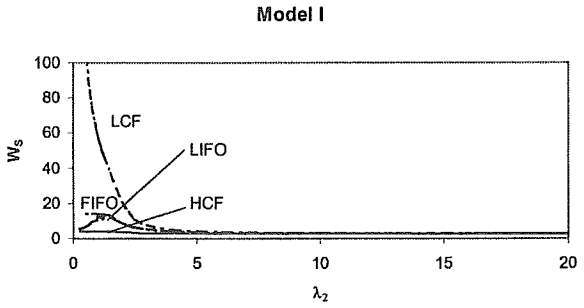

Figure 7.14: Class 1 Waiting Times for Model I by λ_{model}

increases. The location parameter increases as does the scale, leading to a flatter distribution.

Waiting times behaviour with respect to low class customer arrival rates

To analyse the effect of varying low class arrival times, the waiting times of all classes in the four models were examined. For all the disciplines, the waiting times follow similar patterns for each model for the same type of disciplines. Consider specifically the behaviour of the high-class customer when the low class customer's arrival rate is varied. For the FIFO scheme, all four models had the lowest class customer waiting the shortest amount of time, and followed the patterns seen in Figures 7.14 to 7.17.

From all Figures 7.14 to 7.17, the LCF displays asymptotic behaviour, i.e. as $\lambda_{i+1} \rightarrow 0, W_s^{i+1} \rightarrow \infty$, where i is the model number. The LIFO has periods of instability when $\lambda_{i+1} < 5$. This instability increases as the number of classes increase. Once again, the HCF discipline is superior to all the others, with a long flat wait. It is best when the arrival rates are low.

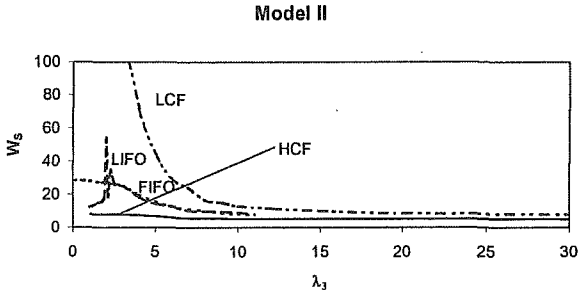

Figure 7.15: Class 1 Waiting Times for Model II by λ_{model}

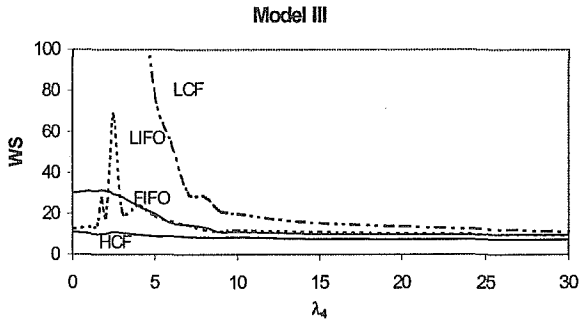

Figure 7.16: Class 1 Waiting Times for Model III by λ_{model}

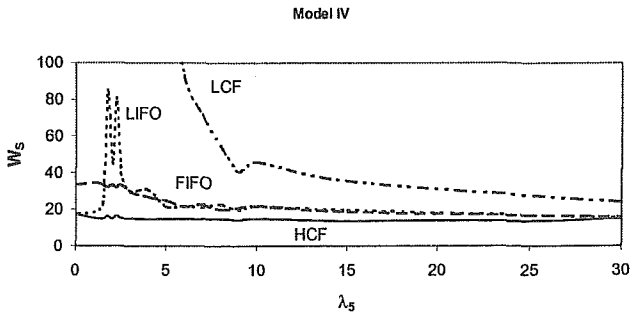

Figure 7.17: Class 1 Waiting Times for Model IV by λ_{model}

7.6.3 Concluding Remarks

The MPDQ's waiting time characteristics under various disciplines have been presented. As a scheme combining priorities with a dual queue, MPDQ HCF discipline for 3 and 4 classes performed well, whereas the LCF and LIFO showed volatility. The key findings of this section are:

- The HCF MPDQ produces superior waiting times for nearly all classes in each of the models.
- The HCF MPDQ clearly has the best waiting time when inspecting the CDF of each of the models
- LIFO cannot guarantee short waits to all customers, as seen when inspecting the maximum time in the system statistics

7.7 Comparison of Preemptive and Non-Preemptive MPDQ

In this section the differences in loss and waiting time between preemptive and non-preemptive service disciplines for the MPDQ are analysed. The best queueing regime for the preemptive MPDQ is then identified. It shall be shown that non-preemptive service dispensing is superior to customers of two classes than the preemptive scheme, and that highest class first (HCF) is the best queueing regime for either of these algorithms.

7.7.1 Outline

The Multi-Priority Dual Queue (MPDQ) was designed to reduce congestion levels in communications models with finite buffers. By splitting a single finite buffer space into two queues to form the MPDQ, and administering a priority scheme to arriving customers, distinct advantages to communications providers in terms of Quality of Service (QoS) requirements is attainable.

The aim of this section is to explore the differences between the non-preemptive and preemptive service disciplines. The results here provide a framework for communications providers in determining how service disciplines contribute to loss and delay, and how modelling can assist in determining how to relieve a network of this type from congestion. In Section 7.7.2, the simulation design is detailed. In Section 7.7.3, the comparison of preemptive and non-preemptive MPDQ is undertaken by looking at the results of loss by queue size, waiting time in the system, and waiting time by queue size. This provides valuable insight into the behaviours of the two types of MPDQ's that can be used by service providers. In a similar analysis to the previous section, investigations are then completed by looking at the preemptive MPDQ under various queueing regimes. Performance statistics and CDF of delay for four models are fitted, and the waiting time distribution is fitted for Class 1 customers under the HCF MPDQ.

7.7.2 MPDQ Design and Simulation

The queueing regime used in this section is, unless otherwise stated, Highest Class First (HCF). The arrivals are independent exponentially distributed arrival processes with mean arrival rate $\lambda_k, k = class$. Once a space is vacant in the primary queue, the head of the line in the secondary queue joins the primary queue. A schematic diagram describing the MPDQ was given in Chapter 3. The non-preemptive case assumes no interruption to the packet in service, whereas the preemptive service discipline will interrupt service of a packet of lower class for one of a higher class. The lower class

packet is returned to the head of the queue and awaits for the server to again become free. They resume requiring full service time

7.7.3 Comparison of Scheduling Disciplines for the MPDQ

The difference between preemption and non-preemption is simple enough to implement for people-type queueing problems - they are simply asked to move back. However its implementation is more substantial in terms of algorithm design and analytical calculation in other environs such as Arena. Recall the case where a customer of class i is always taken over a customer of class j. Consider a customer of class j being in service when a customer of class i arrives. Preemption is when the service of the j^{th} class customer is interrupted and the server starts serving the i^{th} class customer, based on a FCFS approach within classes. In non-preemption, when the service of the j^{th} class customer is completed the server starts serving the i^{th} class customer, based on a FCFS approach within classes. By this definition, we would expect Class 1 customers to be advantaged (in terms of expected waiting time) under the preemptive regime over Class 1 in non-preemptive and Class 2 customer under the non-preemptive regime over preemptive regime.

Loss by Queue Size

Now the effects on loss when queue size is increased is investigated for each service regime. To undertake this, the Models stated in Table ?? are again used. Since the traffic loss policy is impartial to each class of traffic, it was expected that the probability of loss for both traffic classes should be approximately equal, and that was the case here. The loss rates were proportional to the arrival rates of each traffic class, and it was found that for most queue sizes, Class 2 customers loss levels were marginally higher than Class 1 loss levels of the same regime. The load gives a clue as to why Model III and IV loss levels approached zero. From Table ?? Models III and IV have $\rho < 1$. Model IV has the smallest load and no loss. These two models see arriving customers find a system always with at least one vacant waiting space of the waiting room if extended sufficiently.

What is noticeable from all the models is the marginal improvement in loss levels as queue size increases. This is shown in Figures 7.18 to 7.20. There is no difference in the loss levels between service disciplines and classes, with all 95% confidence intervals overlapping (not illustrated for readability). In packet based networks, if there are high volumes of traffic, a small decrease in loss levels would lead to a higher volume of packets receiving some service. For most elementary queueing models where $\rho > 1$, the queue length will grow to infinity. In a finite buffer space, loss shall be experienced, and

$c_1 + c_2$	π_{Loss1}	π_{Loss2}
10	0.4938	0.4986
40	0.4836	0.4970
100	0.4899	0.4858

Table 7.48: Model I Loss by Queue Size

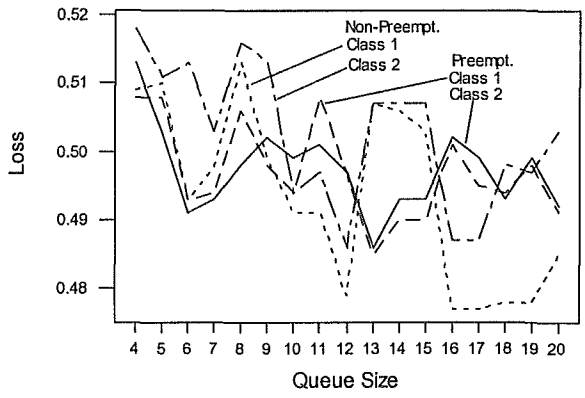

Figure 7.18: Model I Loss by Queue Size

this will only decrease as queue size $\rightarrow \infty$. For customers that follow the distributions stated here, loss levels slowly reduce when addition to the size of the queues is made. As an example, for Model I, in Table 7.48, loss for both classes reduced less than 1% for class 1 customers and less than 1.5% for class 2 customers after a ten-fold increase in queue size.

CDF of Delay

The CDF of delay time, or waiting time in the system, W_s, delivers an overall picture of the QoS to customers for the two types of service regimes. Each model is investigated, with the expectation of shorter waiting times for Class 2 in non-preemptive than preemptive and shorter waiting times for Class 1 in preemptive than non-preemptive. In the four Figures 7.21 -7.24, the CDF's follow the expected exponential curve.

In the single queue with preemptive or non-preemptive infinite priority, it can be shown that the expected waiting time in the queue for a Class 1 packet is less in the preemptive priority case, and it is greater for Class 2 packets in the preemptive case. (This is under the condition that $\rho < 1$). From the CDF's for all models, the non-

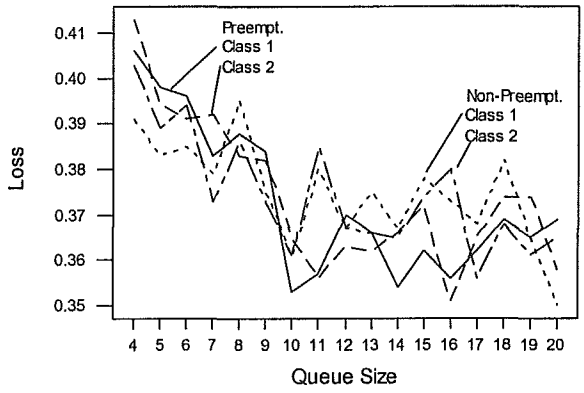

Figure 7.19: Model II Loss by Queue Size

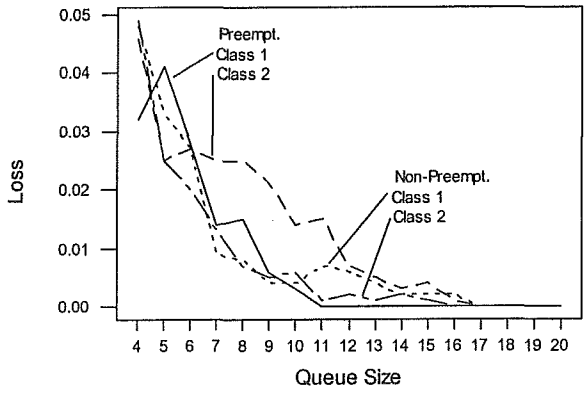

Figure 7.20: Model III Loss by Queue Size

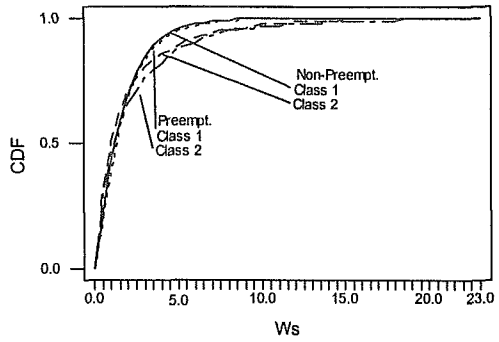

Figure 7.21: CDF of W_s for Model I

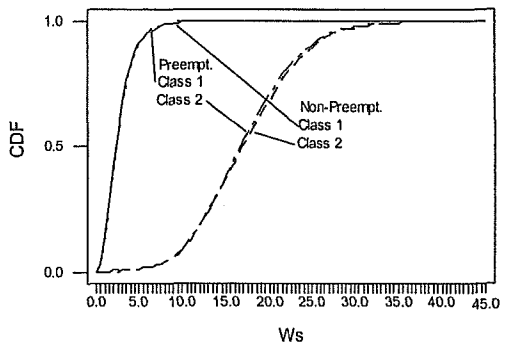

Figure 7.22: CDF of W_s for Model II

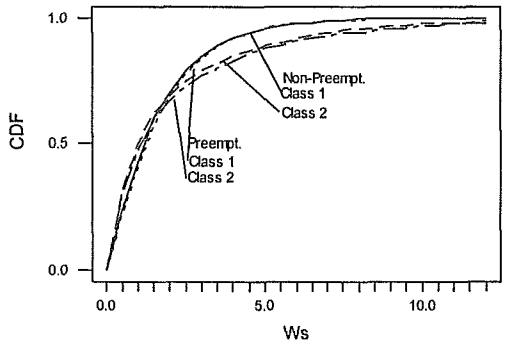

Figure 7.23: CDF of W_s for Model III

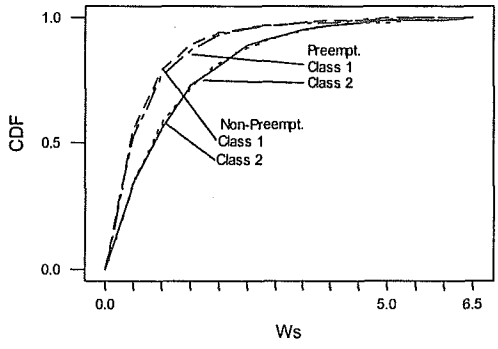

Figure 7.24: CDF of W_s for Model IV

Class 2 service regime	Regression Equation	r^2	p
Non-preemptive	W_s^2= -1.02412 + 0.973284$(c_1 + c_2)$	99.9	0.000
Preemptive	W_s^2=-1.04029+0.97826$(c_1 + c_2)$	100.0	0.000

Table 7.49: Equation for delay by queue size

preemptive Class 1 customers outperform all other customers, however the advantage is marginal (yet statistically significant at 5%) over the preemptive customers. Clearly, Class 2 customers wait longer, as expected, than Class 1. For Models I and III, an increased chance of shorter waiting times towards the Class 2 over Class 1 customers (for Models I and III, for both priority schemes) is seen. For the Class 2 customers, the preemptive regime is the worst in all models but Model II. For communications providers, the MPDQ for two classes of customers, where $\rho \leq 1$, we can expect Class 1 packets to receive superior waiting time over Class 2 customers. The regime to be implemented based on the waiting time CDF's is the non-preemptive service regime. It is marginally better for each of the classes than its interrupt counterpart.

Waiting Time by Queue Size

What is interesting in this analysis is the link between buffer size and waiting time for Class 2 customers. From the simulations undertaken here, in models where $\widehat{\pi}_{Loss} \rightarrow 0$ both waiting times in the system reach constant values. Furthermore in models where loss is non-zero, waiting time in the system for Class 2 traffic increases linearly as queue size increases, and loss for Class 1 customers approaches a constant. As nearly all arriving customers will enter the system, a constant waiting time can be achieved for Class 1 customers due to their class-based queue jumping. The preemptive Class 2 customers is almost identical to the non-preemptive with slight deviations from a straight line the only difference. If a regression line is fitted for Class 2 non-preemptive customers we achieve perfect positive correlation, indicating waiting time for Class 2 customers is related linearly to queue size. We obtain the equations listed in Table 7.49 below using a simple linear regression model. Figure 7.25 illustrates the clear linear trend for the Class 2 customers, and Figure 7.26 is an enlargement of Figure 7.25 for the Class 1 customers.

7.7.4 Conclusion

Through analysis of loss and delay CDF's, the best service regime in terms of QoS is the non-preemptive regime. As the traffic volume increased, the advantages of priority over the non-priority schemes reduce. For many communications applications, a preemptive service regime is harder to implement as it may require more time to

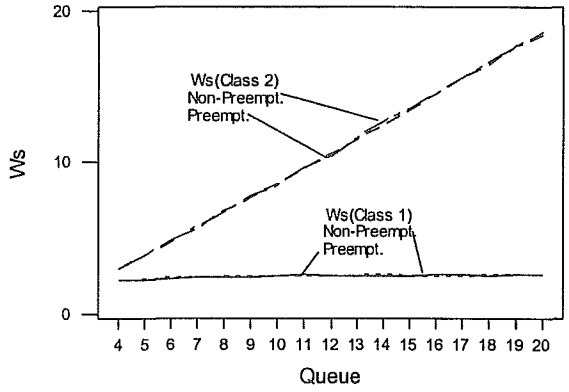

Figure 7.25: Ws by Queue Size Model I

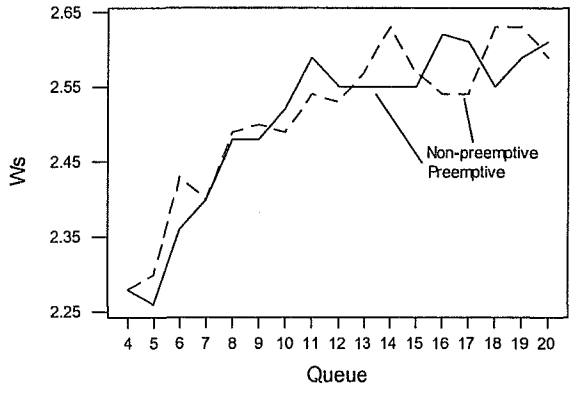

Figure 7.26: Class 1 waiting time in the system Model I

interrupt a packet in service, return it to the head-of-the-line, and recommence service for the higher classed customers than to just let it be served. Certainly, the differences are small in many cases between preemption and non-preemption, however the non-preemption is fairer to the lower classes, and is statistically significant when comparing both W_s^1 and W_s^2.

7.8 Computation of Quality of Service in Networks with Prioritised Traffic involving the MPDQ

With the MPDQ shown to reduce waiting times for multi-class traffic over conventional scheduling disciplines in isolation, its performance in a network is now considered. As the MPDQ improves waiting times, most notably in communication systems, ideally it would be best to use it at every node within a network. However, it may well be the case that the cost associated may be too high for complete network replace. So in this section insight into how the MPDQ behaves within a network where it is not used at every node, or service centre, is investigated. To gain insight into the advantages of a mixed MPDQ / First In First Out network with prioritised traffic, simulations are performed in networks both with and without a MPDQ. Various QoS criteria are compared. As discussed, queueing networks containing differentiated traffic, also known as multi-class networks, are complicated to solve analytically using existing queueing theory techniques. However, some have been solved analytically such as Jackson and Kelly networks [Kelly, 1975], [Kelly, 1976] and the BCMP network. Other analysis of networks has been undertaken using approximate analysis, such as that in [Shi, 1995]. In this section performance statistics are obtained. Further, high-class traffic under different offered loads is investigated, and a comparison of the delay characteristics is provided. The findings in this section provide communication service providers valuable information in determining the improvement in QoS to differentiated networks with the MPDQ. They also continue to highlight the importance of simulation as a tool of evaluation of networks, and the best MPDQ network scenario in terms of waiting times and loss. This is justified through extensive confidence intervals and two-sample tests.

7.8.1 Outline

Firstly the network structures used in this section are introduced. Some key QoS indicators are investigated, namely loss, and adjusted route delay. A comprehensive analysis of the results using the waiting time distributions to determine network delay for the MPDQ follows.

7.8.2 Multi-Priority Dual Queue (MPDQ)

The MPDQ was devised to partition a typical queue into two with finite waiting room to allow some lower class traffic an opportunity to pass through a system even with the existence of a higher class customer. The network model here bring together the ideas of differentiated classes in networks with the improved QoS given by the MPDQ.

As seen in earlier sections, the HCF regime has proven to be the best regime for the MPDQ, outperforming Lowest-Class First (LCF) and FIFO/LIFO regimes. A non-preemptive service regime is dispensed, which out-performs preemptive (See Section 7.7) so there is no interruption to the service of traffic being processed. This is also is line with practical use of the MPDQ, as preemptive is harder to apply due to its interrupt nature.

Steady-state characteristics

As described earlier, obtaining the steady-state probabilities for the MPDQ remains difficult to evaluate due to the complex state transitions. The dual queue requires exhaustive demands on computational resources due to the rapidly expanding size of A as c_1 and c_2 increase. Nonetheless the algorithmic techniques developed in Chapters 5 and 6 make solutions possible using a computer package such as Maple. However, exact analysis cannot be undertaken using present stage of knowledge as we wish to consider more than two classes and a mixed network of MPDQ and FIFO nodes. In turn the QoS criteria of delay (waiting time) is not possible to obtain using exact analysis. A recursive algorithm was applied to determine the moments of the waiting time distribution for each class of customer in a two-class system, but not higher classes at this point. Placing a MPDQ within a network complicates these analyses, so the solutions needed are no longer achievable through these techniques. Fortunately, simulations can be used and are undertaken here.

Classes

With the DQ and MPDQ's versatility for application in communications established, the aim here is to analyse a simple range of networks with and without a MPDQ using differentiated classes of traffic. Furthermore, QoS issues for each traffic class and the implications to each network are investigated. In these trials, three classes of traffic are used . In our previous work [Bedford and Zeephongsekul, 2001a],
[Bedford and Zeephongsekul, 2001b] and in [Ogawa et. al., 2000], this was seen as a practical number of classes in a differentiated network. This also follows using the principals with which the MPDQ was devised as discussed in [Odlyzko, 1999a] and covered in the Introduction.

7.8.3 Networks

The network topology consists of three, four and five node examples. For each network, the simulation includes two distinct findings with the MPDQ at node 1 – all other nodes FIFO compared to all nodes FIFO. The types illustrated in Figure 7.27 can be used

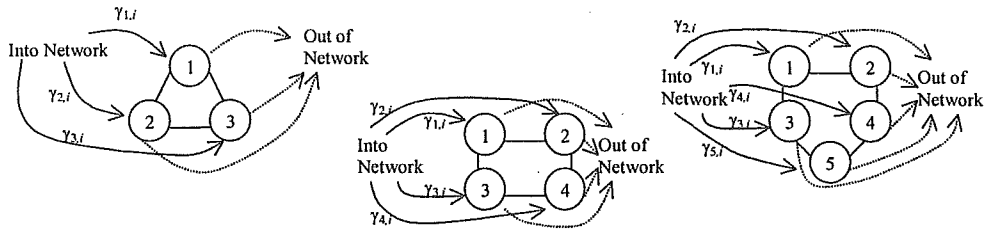

Figure 7.27: Three, four and five node networks analysed

as a blueprint for constructing larger networks. In the figures, $\gamma_{n,i}$ is the arrival rate to node n for traffic of class i.

The traffic waits at each node using the FIFO regime with the exception of node 1 which is the MPDQ (non-preemptive service and HCF queueing regime). All paths between nodes are identical, and let the transit times between nodes to be zero. Traffic is restricted to two adjacent nodes, and may depart the network at any node (i.e. no node is purely transitional). In communications, nodes represent link transmissions.

Route equivalence

After preliminary analysis, the results could be simplified into sets of routes rather than individual routes. This is because analysis showed that even though external arrivals could occur at any node, routes with the same distance in the same network could be considered equivalent for the same class of traffic. Two-sample tests indicated the routes to be statistically significant (all $p - values < 0.05$). For example, a Class 1 customer travelling from node 1 to 3 was found statistically equivalent to a Class 1 customer travelling from node 3 to 1. Two sample analysis confirmed this trend to hold for all like route combinations with the same service queueing regime. There are, at most, four types of paths in these networks, which is discussed next.

Routes and node transitions

For notational simplicity, all references to the MPDQ shall be denoted simply as a DQ node, and, for a FIFO node, an F node. A shortest path algorithm is employed, so traffic cannot take a longer route even in circumstances of congestion, hence the restrictions on possible routes (e.g. traffic shall not travel from node 1 to node 5 via nodes 2 and 4). With this in mind, six networks are given in Table 7.50.

The three networks DQ-F-F, DQ-F-F-F and DQ-F-F-F-F have the dual queue at node 1 and all other nodes are FIFO. As routes can be considered equivalent, all feasible

Node Quantity	All FIFO	MPDQ with FIFO
3-node	F-F-F	DQ-F-F
4-node	F-F-F-F	DQ-F-F-F
5-node	F-F-F-F-F	DQ-F-F-F-F

Table 7.50: Networks Analysed

	Routes			
Network	F-F	DQ-F	F-F-F	DQ-F-F
F-F-F	X			
DQ-F-F	X	X		
F-F-F-F	X		X	
DQ-F-F-F	X	X	X	X
F-F-F-F-F	X		X	
DQ-F-F-F-F	X	X	X	X

Table 7.51: Feasible routes by network

sets of routes by network are now given in Table 7.51. In the networks analysed all feasible routes are combined into the following sets DQ-F, F-F, F-F-F and DQ-F-F-F.

7.8.4 Simulation

Each simulation was run as stipulated in Section 7.3.2. For each network simulation the mean inter-arrival and service times are fixed so that comparisons across networks could be made, and each network model simulation was replicated as described in Table 7.2. Traffic could take any feasible route through a network and no classes were restricted along any route. All mean inter-arrival and process times followed an exponential distribution. These rates were identical irrespective of the node type. In the first performance analysis, the network load is $\rho = 0.525$. Later simulations use a varying inter-arrival mean for Class 1 to evaluate how delay times are effected under different loads. No changes are made to Class 2 or Class 3 traffic.

Define the primary and secondary queue in the dual queue to be of length 5 each, i.e. $c_1 = c_2 = 5$. All other FIFO nodes have single queue length of 10. The mean inter-arrival and process times are given in Table 7.52. Only one DQ node is included as to see the affect it has on the network. More could be added but are left from this sections analysis.

Later simulations use a varying mean for Class 1 to evaluate how delay times are effected at different loads. The simulation results combined 35MB of data.

A screen shot of the DQ-F-F-F example is seen in Figure 7.28. The model is large in size when compared with single MPDQ models.

Class	Inter-arrival	Process
1	10 sec.	4 sec.
2	2.5 sec.	1 sec.
3	5 sec.	2 sec.

Table 7.52: Mean Inter-Arrival/Service times

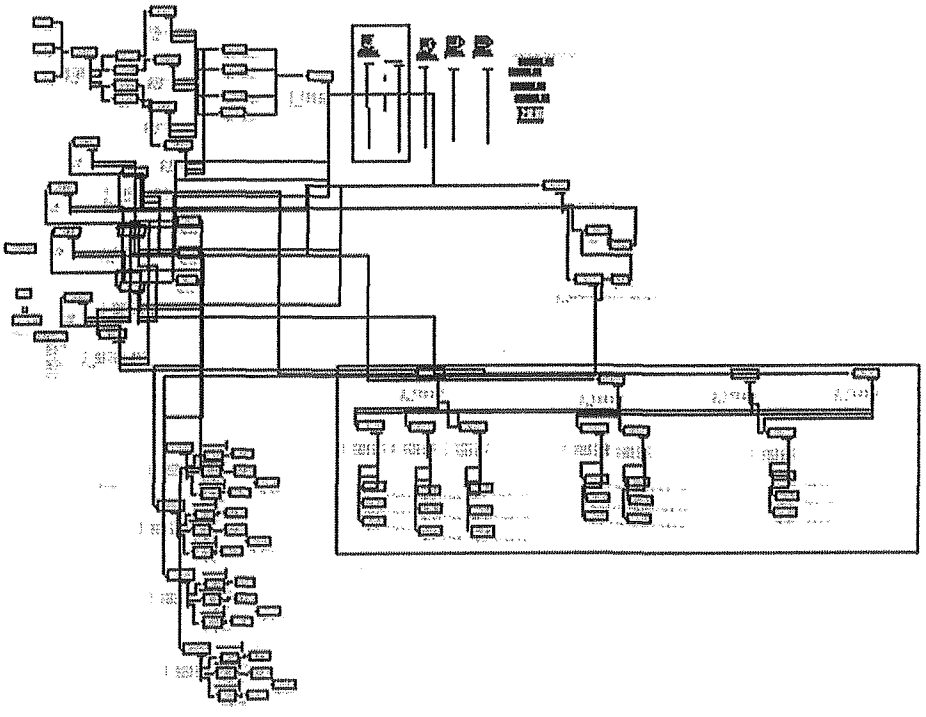

Figure 7.28: DQ-F-F-F Network Arena Code

		Node				
Network	Class	1	2	3	4	5
	1	0.133	0.146	0.141		
F-F-F	2	0.149	0.141	0.148		
	3	0.144	0.141	0.144		
	1	0.129	0.145	0.142		
DQ-F-F	2	0.137	0.150	0.147		
	3	0.146	0.141	0.146		
	1	0.047	0.045	0.052	0.051	
F-F-F-F	2	0.047	0.041	0.051	0.057	
	3	0.051	0.042	0.049	0.053	
	1	0.048	0.042	0.048	0.048	
DQ-F-F-F	2	0.054	0.046	0.052	0.051	
	3	0.053	0.051	0.0.51	0.051	
	1	0.042	0.042	0.042	0.040	0.039
F-F-F-F-F	2	0.039	0.043	0.042	0.0.42	0.041
	3	0.040	0.045	0.043	0.041	0.040
	1	0.037	0.038	0.035	0.035	0.042
DQ-F-F-F-F	2	0.043	0.044	0.043	0.040	0.044
	3	0.037	0.044	0.040	0.042	0.045

Table 7.53: Loss at node by class and network

7.8.5 Performance Evaluation

The first analysis of the QoS for traffic focuses on average network loss and delay. A series of tables with the average statistics for the simulation is presented. Table 7.53 gives the average loss at each node for each network design by class and Table 7.48 the average route time delay. As this analysis is seen as a starting point for the service providers in assessing the QoS a MPDQ may give, time-out thresholds as in [Hayes et. al., 1999] were not employed.

Average node loss

From Table 7.53, the 3-node FIFO network (F-F-F) shows little difference in loss levels for each node, with the exception of Class 1 for node 1. Two sample tests indicate that this is significantly different from the other classes and other nodes within this network ($p-value < 0.05$). This same result holds for the DQ-F-F network. All other networks shows no statistically significant difference in loss levels between each node within the same network.

Route Delay

To account for the service rate difference for the three classes of traffic, a statistic was devised that removes the service rate from the route time. Called the adjusted average network delay (AAND), it is an approximate measure of class-wise traffic delay by route. By removing the service time at each node we are left with the waiting time in the queue. As this traffic is differentiated with independent service rates, this provides a better picture of network performance than an overall average delay. This delay statistic is given by

$$AAND_{c,n,r} = \frac{\sum_{i=1}^{N_{n,r}} \left(W_c^{i,r} - \left(N_{n,r} \mu_c^{-1} \right) \right)_+}{N_{n,r}} \tag{7.5}$$

where $N_{n,r}$ is the number of nodes in the route r in network n, c is the class, μ_c is the mean service rate for class c, $x_+ = \max\{x, 0\}$ and $W_c^{i,r}$ is the mean waiting time in the network for the i^{th} link (node or hop) in the routing set r of class c. So for example the adjusted delay for the route DQ-F in the network DQ-F-F for Class 1 traffic is given by

$$AAND_{1,n,r} = \frac{\left(\left(W_1^{1-2,DQ-F} - 2\mu_1^{-1} \right)_+ + \left(W_1^{1-3,DQ-F} - 2\mu_1^{-1} \right)_+ \right)}{2} \tag{7.6}$$

From the 3-node networks in Tables 7.54 and 7.55, note the influence of the MPDQ on the prioritised traffic. Classes 1 and 2 show tremendous reductions in route time when the dual queue is one of the nodal links. The difference between DQ-F and F-F is statistically significant for Classes 1 and 2 (i.e. less waiting time under the DQ than FIFO). This also does not adversely effect the F-F link, with times statistically equivalent to the F-F-F network. Class 3 is disadvantaged by the dual queue, with the largest delay time of all traffic in these 3-node networks. All FIFO networks show no discrimination in their delay times, with the adjusted averages near identical for the classes.

For the 4-node networks, the dual queue improves QoS indirectly to the FIFO nodes. The DQ-F-F-F shows slightly improved delay times for the F-F route for all classes, however they are considered statistically equivalent. There is greatly improved delay times for the F-F-F route over the FIFO network. A flow-over effect of resorted traffic by the dual queue to the FIFO nodes is emerging. Notably, Class 3 is again disadvantaged in the dual queue network. This may be of concern for service providers if the low class is now receiving worse delay than usually expected. As the networks increase in size, the delay to this traffic is closing in on the FIFO network delay times.

Network	Class	Route F-F	DQ-F	F-F-F	DQ-F-F
F-F-F	1	54.27			
	2	54.47			
	3	56.75			
DQ-F-F	1	55.00	33.50		
	2	54.70	35.20		
	3	57.10	71.50		
F-F-F-F	1	49.08		72.68	
	2	49.02		73.58	
	3	50.83		76.37	
DQ-F-F-F	1	48.36	30.63	63.49	54.13
	2	48.14	31.74	64.77	56.23
	3	50.72	63.41	81.90	89.86
F-F-F-F-F	1	38.89		58.76	
	2	39.15		59.20	
	3	41.20		61.89	
DQ-F-F-F-F	1	39.42	25.78	59.27	44.25
	2	39.97	26.25	59.54	46.47
	3	41.56	51.22	63.01	71.64

Table 7.54: Adjusted Route Delay Time by Class And Network

Network	Class	Route F-F	DQ-F	F-F-F	DQ-F-F
F-F-F	1	(53.07,55.47)			
	2	(53.67,55.27)			
	3	(56.25,57.25)			
DQ-F-F	1	(53.9,56.1)	(32.5,34.5)		
	2	(53.9,55.5)	(34.6,35.8)		
	3	(56.5,57.7)	(70.7,72.3)		
F-F-F-F	1	(47.88,50.28)		(71.08,74.28)	
	2	(48.22,49.82)		(72.48,74.68)	
	3	(50.23,51.43)		(75.57,77.17)	
DQ-F-F-F	1	(46.56,50.16)	(29.53,31.73)	(62.29,64.69)	(52.63,55.63)
	2	(46.94,49.34)	(31.04,32.44)	(63.97,65.57)	(55.23,57.23)
	3	(49.72,51.72)	(62.61,64.21)	(81.4,82.4)	(88.86,90.86)
F-F-F-F-F	1	(37.29,40.49)		(58.16,59.36)	
	2	(38.05,40.25)		(58.1,60.3)	
	3	(40.4,42)		(61.09,62.69)	
DQ-F-F-F-F	1	(38.72,40.12)	(24.48,27.08)	(57.17,61.37)	(42.95,45.55)
	2	(38.77,41.17)	(25.35,27.15)	(58.44,60.64)	(45.57,47.37)
	3	(40.36,42.76)	(50.22,52.22)	(62.01,64.01)	(70.64,72.64)

Table 7.55: Confidence Intervals for the Adjusted Route Delay Time by Class And Network

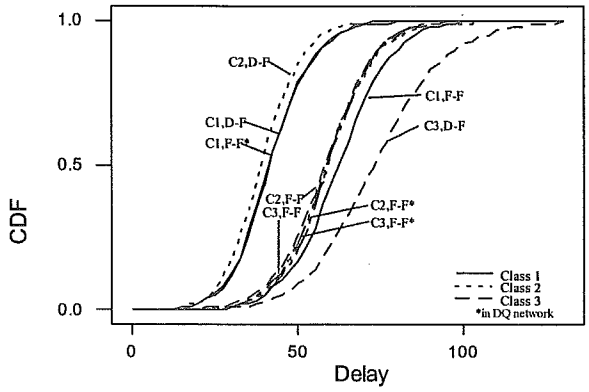

Figure 7.29: 3-node CDF for delay times by class and route

In the 5-node networks, the F-F route is now virtually identical (and of course statistically equivalent) for the two variants, with the dual queue network now a little larger in delay than the FIFO network along the F-F route. This is also the case in the F-F-F route. Also the DQ-F and DQ-F-F routes in the dual queue network continue to deliver lesser delay times than the FIFO. However the margin is also smaller. This may indicate that the effectiveness of the dual queue in re-sorting inter-nodal arrival traffic is diminishing. With more non-sorted departures within the network, the influence of the dual queue, whilst still effective, may not be delivering the requirements needed to guarantee QoS to higher priority traffic. Furthermore the cost of implementing a dual queue may not be justified if the improvements to network congestion are viewed as marginal. Clearly, classes 1 and 2 benefit in the dual queue networks at the expense of class 3 customers.

Cumulative Density Functions of delay

The behaviour of the traffic along all routes is now modelled for each class. The delays are no longer adjusted for their service times as in the previous section as investigation into what is happening to the traffic in its entirety is desired. The Cumulative Density Function (CDF) of delay time is used to model each network, and they are compared on a route basis for each class. In this way, as before, administrators can decide what prioritised customers should expect in their service requirements.

First look at Figure 7.29, which contains the 3-node network CDF's.

The optimal curve is one that reaches a probability of 1 as rapidly as possible. The

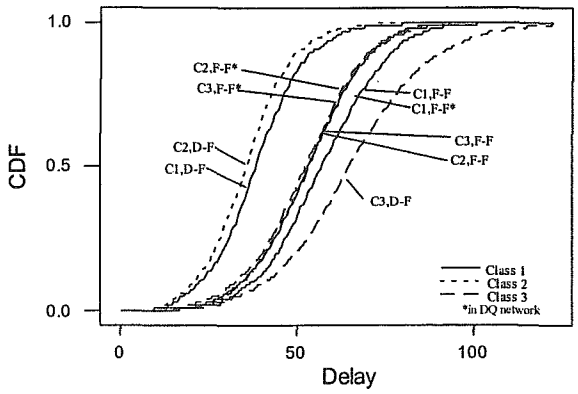

Figure 7.30: 4-node CDF for delay times by class and route (2-nodes)

delay CDF's are the familiar 'S' shape common in delay models. What is noticeable in Figure 7.29 is the desirable characteristics in the first three curves. Class 1 and 2 traffic along the DQ-F route achieves excellent delay times. The sorting influence of the dual queue is also evident, with Class 1 traffic along the F-F route in the dual queue network also achieving excellent delay. This influence is a way of sorting traffic, as it exits the dual queue in class order, and arrives at the next node in class order. The intermixing of this traffic with traffic from other nodes will be in a semi-ordered form. This result is excellent for this class. Class 1 traffic can be guaranteed low delay times irrespective of the route in the 3-node dual queue network. Furthermore, Class 2 and 3 traffic in the same network receives virtually identical delay functions as the FIFO counterparts. The only poor result in the dual queue network is Class 3 traffic. The FIFO network routes all receive near identical delay functions.

We next consider the 4-node networks. To avoid congestion, we have divided the figures into two: one for the routes DQ-F and F-F (Figure 7.30), and the other for DQ-F-F and F-F-F (Figure 7.31). From Figure 7.30, we again see the Class 1 and 2 traffic in the dual queue network receiving excellent service. Class 2 and 3 customers along the F-F route in the dual queue network also do well, with them having near identical delay functions to that of the FIFO network. Unlike in Figure 7.29, Class 1 customers are now delayed substantially more in the dual queue network along the F-F route. The influence of the dual queue may have waned for Class 1 customers, as the dual queue is not necessarily adjacent to both of the F-F routes.

In the 3 node routes in the 4-node network, as seen in Figure 7.31, the delay

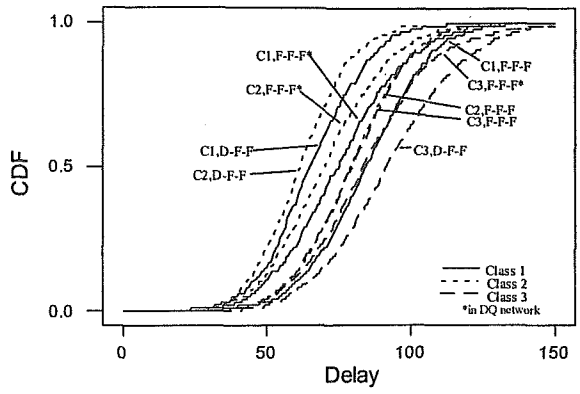

Figure 7.31: 4-node CDF for delay times by class and route (3-nodes)

functions are closer. Notably, the magnitude of the x-axis has increased as analyse now involves traffic through 3 nodes. The results are excellent for the dual queue network, with Class 1 and 2 traffic in this network on both types of routes receiving the best delay times. Class 3 again has the longest delay in the dual queue network along both routes. In the FIFO network, Class 2 and 3 are again superior to Class 1 traffic. This is due to the longer service time of Class 1 traffic.

Finally consider the 5-node networks in Figure 7.32 and Figure 7.33 . From Figure 7.32, again the Class 1 and 2 traffic in the dual queue networks receives the best service. The results are virtually identical to those found in Figure 7.30. The increase in capacity and nodes has little effect on the 2-node routes.

Figure 7.33 shows a distinct advantage of the dual queue network. All six class and route combinations in the dual queue are superior in delay times to the three FIFO combinations. Class 1 and 2 dual queue traffic experience the best delay times, whilst the Class 1 F-F-F traffic is delayed the longest. Through the mid-section of the CDF, the gap is increasing between the priority and non-priority networks. Overall the value to Class 1 and 2 customers in the dual queue networks can be seen. For service providers, the choice to allocate a priority network is dependent upon their willingness to sacrifice service to lower class traffic in order to provide better QoS to higher-class traffic.

The delay analysis concludes by looking at the broad delay CDF's for the networks. This comprises of all classes of traffic from all routes in each network and gives a broad overview to the network behaviour. Figure 7.34 shows the overall delay.

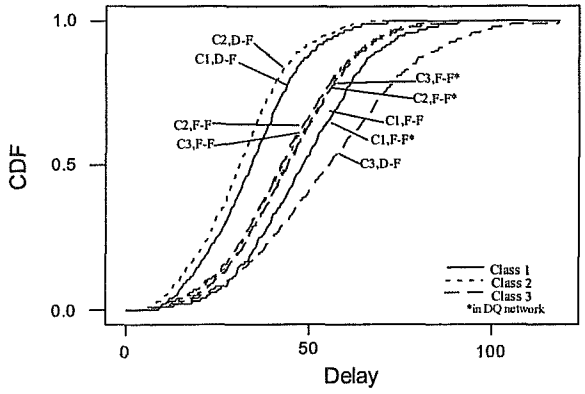

Figure 7.32: 5-node CDF for delay times by class and route (2-node)

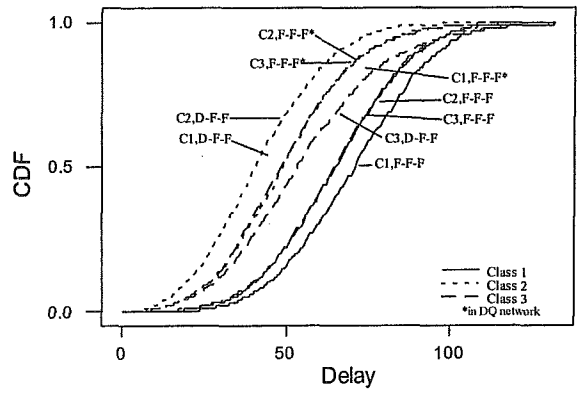

Figure 7.33: 5-node CDF for delay times by class and route (3-node)

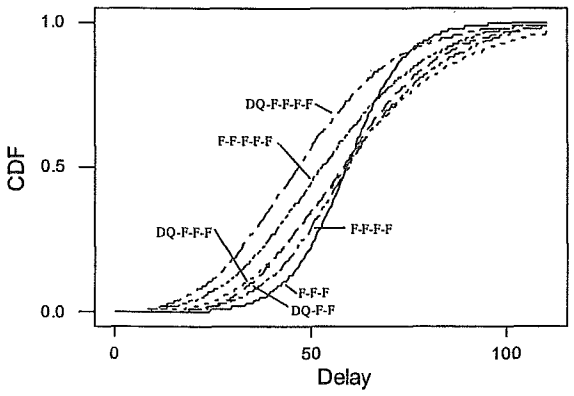

Figure 7.34: CDF for overall delay times by network

All functions have the same shape, with the exception of the F-F-F network. It guarantees the worst probability for short delays but the best in long delays. There is such a slight difference between the DQ-F-F-F and DQ-F-F networks, that in overall terms the set up costs may not be worth the marginal network improvement.

Delay for Class 1 traffic in the 5-node network and average waiting time

The final analysis of the networks looks at the behaviour of first class traffic in the 5-node network for various load levels. The load is varied by adjusting the inter-arrival rate for Class 1 customers, which in turn changes the network load. This was undertaken for the DQ-F-F-F-F network. As seen in Figure 7.35, as the load decreases on the network, the delay time decreases for Class 1 traffic. Furthermore, as the arrival rate increases (and as the load decreases), the gap between CDF's decreases.

Another point of interest is the 95% confidence intervals for the average waiting time in the queue (L_q) at each node, as seen in Table 7.56. The two separate bracketed numbers that appear in the dual queue elements of the Node 1 column are the average number waiting in the primary and secondary queue respectively. The decrease of the number in the queue is approximately one at each node as the network increases by one node. The dual queue outperforms all other nodes in the same network and is the equivalent for the corresponding FIFO network.

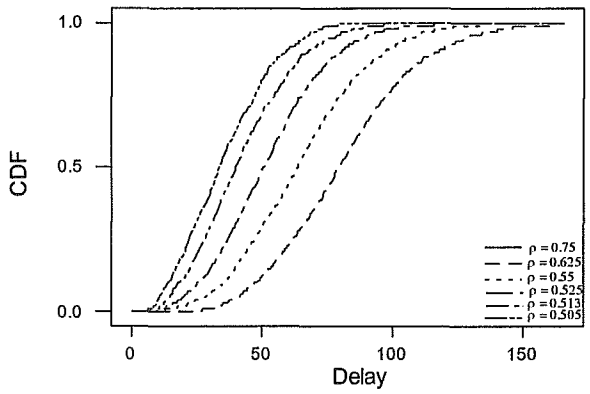

Figure 7.35: CDF for delay times in DQ-F-F-F-F for Class 1 at various loads

	Node				
Network	1	2	3	4	5
F-F-F	7.75, 8.15	7.73, 8.07	7.92,8.16		
DQ-F-F	(4.77,4.85);(3.00,3.20)	7.99,8.20	7.84,8.12		
F-F-F-F	6.92, 7.32	6.66,7.06	6.91,7.31	6.80,7.20	
DQ-F-F-F	(4.46,4.65);(2.34,2.58)	6.70,7.20	6.66,7.08	6.81,7.23	
F-F-F-F-F	5.08,5.68	5.17,5.91	5.29,5.89	5.00,5.64	5.22,5.80
DQ-F-F-F-F	(3.62,3.98);(1.38,1.72)	5.10,5.76	5.28,5.90	5.18,5.88	5.27,5.97

Table 7.56: Average number waiting at a node by network

7.8.6 Conclusion

The implications of a MPDQ in various network scenarios have been explored and delay functions provided. As expected, loss levels decrease as network size or capacity increases. From the $AAND$ statistic, the 4-node DQ network proved best, with a traffic resorting influencing the delay times of FIFO nodes. The delay curves indicated that low class traffic waits longer under the MPDQ, whilst high-class benefits. Further the resorting influence continued in the DQ networks. With FIFO networks, the more nodes, the lower the delay time. All results according to class show that differentiated services benefit in a network with the MPDQ present, and its effectiveness diminishes as we increase the quantity of FIFO nodes within the network.

7.9 Summary

In this chapter, simulation was shown to provide valuable insight into the behaviour of the MPDQ under various queueing regimes and service disciplines.

The overall conclusions of this chapter are

- The MPDQ HCF queueing regime outperforms all other investigated queueing regimes for both preemptive and non-preemptive service disciplines
- The MPDQ reduces the waiting time of first class customers markedly whilst still allowing lower classes an opportunity to pass through the system without adversely long waiting times in comparison to traditional FIFO
- The HCF regime performs best with 3 classes or 4 classes of customers for these examples
- 'Middle class' degraded QoS is seen in some 4 and 5 class models
- The MPDQ in a network scenario was found to improve traffic flow outside of its service in isolation
- The MPDQ for the networks here performed optimally network-wise in the 4-node network

Part V

FINALE

Chapter 8

Conclusions and Further Work

Finite queueing models and their application in communications systems are well known. Novel approaches to reduce congestion in these systems are needed to improve users performance. One such idea is the MPDQ, introduced and shown in this thesis to be a system that improves delay and loss in times of high traffic intensity. A systematic approach was taken to investigate various conventional finite buffer models, both priority and non-priority based alongside the MPDQ. Simulation also proved a valuable tool in determining computations beyond the bounds of analytical solution processes, and further justified consideration of the MPDQ in practical scenarios.

A brief summary of the thesis and its achievements are given below.

PART I - INTRODUCTION TO FINITE QUEUEING MODELS

In Chapter 2 the mathematical preliminaries were detailed, including the birth-death process, balance equation, Laplace transforms and the PASTA property. In Chapter 3 the MPDQ was introduced along with conventional single buffer models. The service disciplines and queueing regimes were discussed in detail with examples given to exhibit the finer workings of the finite buffer models. The four queueing regimes (HCF, LCF, FIFO and LIFO) were shown for a specific arrival pattern, with the differences highlighted through illustrations. This was stratified into preemptive and non-preemptive service disciplines. Table 3.1 detailed the differences in departure from the respective systems assuming the same arrival pattern for seven types. The preemptive MPDQ HCF discipline was the only system seeing the Class 1 customer depart first, with the non-preemptive MPDQ HCF being next. This was an early introduction into the differences in fairness between preemptive and non-preemptive scheduling.

PART II - SINGLE QUEUE MODELS

Chapter 4 began the analytical work in the thesis. A look at past work on finite buffer models was detailed. The performance statistics were given for the classic finite

buffer models. Next a new algorithmic approach to solving the steady-state probabilities and waiting time distributions was shown. The method to solving the steady-state probabilities were shown via simple algorithm. The waiting time was derived using a recursive algorithm also. Both approaches are used later in the derivation of performance characteristics for the MPDQ. The steady-state algorithm design was coded for use in Maple and is included in the Appendix.

PART III - DUAL QUEUE MODELS

This Part was divided into two with Chapter 5 looking at the preemptive MPDQ, and Chapter 6 the non-preemptive MPDQ. In both chapters, algorithms based on global balance were used to solve the steady-state probabilities. In the preemptive case an algorithm for the waiting time of both classes of customers was detailed with the non-preemptive case remaining a future work. The general structure of the systems were given using matrix notation in order to assist in the understanding of the complexities of each system, and ordering the states for solution. The non-preemptive MPDQ states numbered a little over twice the number for the preemptive case due to the need to consider the class of customer in service. The linear systems of steady-state distributions were referenced within the detailed algorithms for both preemptive and non-preemptive cases to illustrate the systematic approach towards a solution. Examples are given illustrating the process of obtaining a solution with some specific values for different sized systems.

PART IV - SIMULATION STUDIES.

Chapter 7 takes the analysis beyond the realm of analytical solutions given in Parts II and III by considering more than two classes of customers. The Arena simulation package was used in all analysis, with the process of setting up simulations and comparisons with sum analytical results given. Analysis of the preemptive MPDQ is considered first. The waiting times indicated that in times of high traffic intensity, the waiting times were far shorter under HCF than for any other queueing discipline, and marginally shorter as the traffic intensity decreased. Further, five classes of customers proved too many when considering delay and loss, with 3 or 4 the ideal amount based on the simulations. Then the non-preemptive MPDQ was investigated showing again the waiting times and loss are far superior under the HCF regime. Comparison between the non-preemptive and preemptive showed non-preemptive to be the 'fairer' of the two. Network analysis highlighted that the ordering of traffic after departing an MPDQ node improved delay times in adjacent nodes that were FIFO. The 4-node network proved the ideal setup (1 MPDQ node and 3 FIFO nodes) in terms of delay.

Further work

Due to the MPDQ performing well primarily in periods of high traffic intensity, the dynamic MPDQ is a planned future work. The dynamic MPDQ is proposed to change its queueing discipline depending on the traffic conditions. If traffic intensity is at a defined low level, the dynamic MPDQ will switch to the conventional FIFO. In this way, it may be possible for this hybrid model to improve the already impressive performance characteristics of the MPDQ seen in this thesis with the non-dynamic MPDQ. Also, the development of computer code to evaluate systems of any size is planned.

More general service distributions in line with communications systems is planned. The exponential distribution provides a good approximation for communications systems, however more precise modeling is needed to replicate existing systems.

Analysis into the waiting time distribution for non-preemptive MPDQ customers is close to completion. This work is analogous to that undertaken in Chapter 5.

Bibliography

[Abate and Whitt, 1997] ABATE, J. and WHITT, W. (1997). Limits and Approximations for the M/G/1 LIFO Waiting-time Distribution, *Operations Research Letters,* **20**, pp. 199-206.

[Adiri and Domb, 1982] ADIRI, I. and DOMB, I. (1982). A Single Server Queueing System Working Under Mixed Priority Disciplines, *Operations Research,* **30**, 1, pp. 97-115.

[Bagwell et. al., 1995] BAGWELL TIPPER, C. and DAIGLE, J.N. (1995). ATM Cell Delay and Loss for Best-Effort TCP in the Presence of Isochronous Traffic, *IEEE Journal on Selected Areas in Communications,* **13**, 8, pp. 1457-1463.

[Bahk and El Zarki, 1992] BAHK, S. and EL ZARKI, M. (1992). Dynamic Multi-path Routing and How it Compares with Other Dynamic Routing Algorithms for High Speed Wide Area Networks, *Proc. ACM SIGCOMM '92, pp. 53-64.*

[Balbo et. al., 1985] BALBO, S., BRUELL, S. and GHANTA, S. (1985). Modelling Priority Schemes, *Journal of ACM,* pp.15-26.

[Bedford, 1999] BEDFORD,A. Single and Multiple Job Classed Closed Queueing Networks, Honours Thesis, RMIT University.

[Bedford and Zeephongsekul, 2001a] BEDFORD, A. and ZEEPHONGSEKUL, P. (2001a). Simulation Studies of Waiting Time

Approximation for the Multi Priority Dual Queue (MPDQ) with Finite Waiting Room and Non-Preemptive Scheduling, *Topics in Applied and Theoretical Mathematics and Computer Science*, WSEAS Press, pp. 220-235.

[Bedford and Zeephongsekul, 2001b] BEDFORD, A. and ZEEPHONGSEKUL, P. (2001b). Simulation Studies on the Performance Characteristics of Multi Priority Dual Queue (MPDQ) with Finite Waiting Room and Non-Preemptive Scheduling, *Topics in Applied and Theoretical Mathematics and Computer Science*, WSEAS Press, pp. 226-231.

[Bedford and Zeephongsekul, 2002] BEDFORD, A. and ZEEPHONGSEKUL, P. (2002). Finite Buffer Queueing System with Preemptive Priority Scheduling, *Research Report No. 9, Department of Mathematics and Statistics*, RMIT University.

[Bedford and Zeephongsekul, 2002a] BEDFORD, A. and ZEEPHONGSEKUL, P. (2002a). Computation of Network Delay With Prioritised Traffic Involving the Multi Priority Dual Queue, *Recent Advances in Computational Science and Engineering*, Imperial College Press, 2002, pp. 683-686.

[Bedford and Zeephongsekul, 2002b] BEDFORD, A. and ZEEPHONGSEKUL, P. (2002b). Simulation Solutions of Networks with Prioritised Traffic Involving the Multi Priority Dual Queue, *Recent Advances in Computational Science and Engineering*, Imperial College Press, 2002, pp. 687-691.

[Bedford and Zeephongsekul, 2003] BEDFORD, A. and ZEEPHONGSEKUL, P. (2003), Simulation Studies of a Multi-Priority Dual Queue (MPDQ) with Preemptive and Non-preemptive Scheduling, *Computational*

Science-ICCS 2003, LNCS 2659, Springer, **3**, pp. 179-189.

[Bedford and Zeephongsekul, 2003a] BEDFORD, A. and ZEEPHONGSEKUL, P. (2003a). On a Dual Queueing System with Preemptive Priority Service Discipline. *In Press, European Journal of Operational Research.*

[Bolot, 1993] BOLOT, J-C. (1993). Characterizing End-to-End Packet Delay and Loss in the Internet, *Journal of High-Speed Networks,* **2**, 3, pp. 305-323.

[Bondi, 1989] BONDI, A. (1989). An Analysis of Finite Capacity Queues with Priority Scheduling and Common or Reserved Waiting Areas, *Computers & Operations Research,* **16**, 3, pp. 217-233.

[Boxma and Down, 1997] BOXMA, O. and DOWN, D.G. (1997). Dynamic Server Assignment in a Two-Queue Model, *European Journal of Operational Research,* **103**, pp. 595-609.

[Boyce and Gaglianello, 1998] BOYCE, J.M. and GAGLIANELLO, R.D. (1998). Packet Loss Effects on MPEG Video Sent Over the Public Internet, *Proceedings of the 6th ACM International Conference on Multimedia,* pp. 181-190.

[Brandt and Brandt, 1999] BRANDT, A. and BRANDT, M. (1999). On a Two-Queue Priority System with Impatience and its Application to a Call Centre, *Methodol. Comput. Appl. Probab.*, **1**, 2, pp. 191–210.

[Bruell and Balbo, 1980] BRUELL, S. and BALBO, G. (1980). *Computational Algorithms for Closed Queueing Networks*, Elsevier, North Holland.

[Bunday, 1996] BUNDAY, B. (1996). *An Introduction to Queueing Theory*, Arnold, London.

[Burke, 1966] BURKE, P.J. (1966). The Output of a Queueing System, *Operations Research*, 4, pp. 699-704.

[Buzen, 1973] BUZEN, J. (1973). Computational Algorithms for Closed Queueing Networks with Exponential Servers, *Communications of the ACM*, **16**, 9, pp. 527-531.

[Chan et. al., 1997] CHAN, K.T., BENSAOU, B. and TSANG, D.H.K. (1997). Credit-Based Fair Queueing (CBFQ), *IEEE Electronics Letters,* **33**, 7, pp. 584-585.

[Chang, 1965] CHANG W. (1965). Preemptive priority queues. *Operations Research*, **13**, pp. 820–827.

[Cidon and Sidi, 1990] CIDON, I., and SIDI, M. (1990). Recursive Computation of Steady-State Probabilities in Priority Queues, *Operations Research Letters,* **9**, pp. 249-256.

[Cidon et. al., 1993] CIDON, I., KHAMISY, A. and SIDI, M. (1993). Analysis of Packet Loss Processes in High-Speed Networks, *IEEE Transactions on Information Theory,* 39, **1**, pp. 93-108.

[Clark and Fang, 1998] CLARK, D.D and FANG, W. (1998). Explicit Allocation of Best-Effort Packet Delivery Service, *IEEE/ACM Transactions on Networking,* **6**, 4, pp. 362-373.

[Cobham, 1954] COBHAM, A. (1954). Priority Assignment in Waiting Line Problems, Operations Research, **2**, pp. 70-76.

[Conway and Georganas, 1989] CONWAY, A.E. and GEORGANAS, N.D. (1989). *Queueing Networks- Exact Computational Algorithms: A Unified Theory Based on Decomposition and Aggregation*, MIT Press, London.

[Daigle and Roughan, 1999] DAIGLE, J.N. and ROUGHAN, M. (1999). Queue-Length Distributions for Multi-Priority Queueing Systems, *IEEE INFOCOM'99,* pp. *641–648.*

[Douvrolis and Ramanathann, 2000] DOUVROLIS, C. and RAMANATHANN, P. (2000). Proportional Differentiated Services, Part II: Loss Rate Differentiation and Packet Dropping, *8th International Workshop on quality of Service,* pp.53-61.

[Feng et. al., 1998] FENG, W., KOWADA, M. and ADACHI, K. (1998). A Two-Queue Model with Bernoulli Service Schedule and Switching Times, *Queueing Systems*, **30**, pp. 405-434.

[Feng et. al., 2000] FENG, W., KOWADA, M. and ADACHI, K. (2000). Analysis of a Multi-Server Queue with Two Priority Classes and (M, N) - Threshold Service Schedule I : Non-Preemptive Priority, *International Transactions in Operations Research*, **7**, pp. 653-671.

[Feng et. al. , 1999] FENG, W., KANDLUR, D.D., SAHA, D. and SHIN, K.G. (1999). Adaptive Packet Marking for Maintaining End-to-End Throughput in a Differentiated Services Internet, IEEE – ACM Transactions on Networking, **7**, 5, pp. 685–697.

[Fritz, 1999] FRITZ, J. (1999). Caught up on Video, *Data Communications,* October 21, pp. 51

[Gail, Hunter and Taylor 1992] GAIL, H.R., HUNTER, S.L. and TAYLOR, B.A.(1992). On a Preemptive Markovian Queue with Multiple Servers and Two Priority Classes. *Mathematics of Operations Research,* **17**, 2, pp. 365-391.

[Goff et. al., 2001] GOFF, T., ABU-GHAZALEH, N.B., PHATAK, D.S. and KAHVECIOGLU, R. (2001). Preemptive Routing in Ad Hoc Networks, *ACM Sigmobile*, pp. 43-52.

[Golestani, 1994] GOLESTANI, S.J. (1994). A Self-Clocked Fair Queueing Scheme for Broadband Applications, Proceedings of the IEEE Infocom, pp. 636-646.

[Gouweleeuw, 1997] GOUWELEEUW, F.N. (1997). The Loss Probability in an Overloaded Queue using the Dual Queue, *Operations Research Letters,* **21**, pp. 101-106.

[Gross and Harris, 1974] GROSS, D. and HARRIS, C.M. (1974). *Fundamentals of Queueing Theory.* John Wiley and Sons, New York.

[Hall et. al., 1997] HALL, N.G., POSNER, M.E. and POTTS, C.N. (1997). Preemptive Scheduling with Finite Capacity Input Buffers, *Annals of Operations Research,* **70**, pp. 399-413.

[Halpin, 1998] HALPIN, J. (1998). Get Your Priorities Straight, *Computer Shopper*, **18**, 9, pp. 312

[Hartanto et. al., 1995] HARTANTO, F., SIRISENA, H. and PAWLIKOWSKI, K. (1995). Protective Buffer Policies at ATM Switches, *IEEE International Conference on Communications*, **2**, pp. 960-964.

[Hayes et. al., 1999] HAYES, D., RUMSEWICZ, M. and ANDREW, L. (1999). Quality of Service Driven Packet Scheduling Disciplines for Real-Time Applications: Looking Beyond Fairness, *IEEE Infocom 1999*, pp. 405-412.

[Hayes, 2001] HAYES, D.A. (2001). *Congestion Management of Packet Switched Networks*, Ph. D. Thesis, Uni. of Melbourne.

[Heathcote, 1960] HEATHCOTE, C.R. (1960). A Simple Queue with Several Preemptive Priority Classes, *Operations Research.* **8**, pp. 630-608.

[Hitchcock, 1997] HITCHCOCK, S.E. (1997). Two Queues With A Single Server, *Communcations in Statistics - Stochastic Models*, **13**, 1, pp. 95-104.

[Jain and Dhyani, 1997] JAIN, M. and DHYANI, I. (1997). A Finite-Multiserver Queueing System with Non-Preemptive Priorities, *Journal of M.A.C.T.*, **30**, pp. 45-55.

[Jang et. al., 1997] JANG, J., SHIM, S., and SHIN, B. (1997). Analysis of DQLT Scheduling for an ATM Multiplexer, *IEEE Communcations Letters*, **1**, 4, pp. 175-177.

[Jang et. al., 2001] JANG, J., SUH, J. and RICHARD LIU, C. (2001). A New Procedure to Estimate Waiting Time in GI/G/2 System by Server Observation, *Computers & Operations Research*, **28**, pp. 597-611.

[Kapadia et. al., 1984] KAPADIA, A., KAZMI, M. and MITCHELL, A. (1984). Analysis of a Finite Capacity Non-Preemptive Priority Queue, *Computers & Operations Research*, **11**, 3, pp. 337-343.

[Karve, 1998] KARVE, A. (1998). Quality-of-Service Options for Real-Time Traffic, June, *Network.*

[Kato et. al., 1999] KATO, J., SHIMIZU, A. and GOTO, S. (1999). End-to-End Delay Distribution on the Internet, *IEICE Transactions Inf. & Syst.*, **E82-D**, 4, pp. 762-768.

[Kelly, 1975] KELLY, F. (1975). Networks of Queues with Customers of Different Types, *J. Appl.Prob.*, **12**, pp. 542-554.

[Kelly, 1976] KELLY, F. (1976). Networks of Queues, *Adv. Appl. Prob.*, 8, pp. 416-432.

[Kelton et. al., 2002] KELTON, W. D., SADOWSKIL, R.P. and SADOWSKI, D.A. (2002). Simulation with Arena, McGraw-Hill, 2nd Ed.

[Kleinrock, 1975a] KLEINROCK, L. (1975a). *Queueing Systems, Volume I: Theory*, John Wiley & Sons.

[Kleinrock, 1975b] KLEINROCK, L. (1975b). *Queueing Systems, Volume II: Computer Applications*, John Wiley & Sons.

[Koole and Mandelbaum, 2002] KOOLE, G. and MANDELBAUM, A. (2002), Queueing Models of Call Centres, *Annals of Operations Research,* **112** and (www.cs.vu.nl/obp/callcenters)

[Kramer, 1987] KRAMER, M. (1987). Waiting Times in a Queueing System with Capacity Constraints and Preemptive Priorities, *OR Spektrum*, **9**, pp. 33-39.

[Labrador and Banerjee, 1999] LABRADOR, M.A. and BANERJEE, S. (1999). Enhancing Application Throughput by Selective Packet Dropping, *Proceedings of IEEE International Conference on Communications (ICC)*, Vancouver, Canada, pp. 1217-1222.

[Latouche and Ramaswami, 1993] LATOUCHE, G. and RAMASWAMI, V. (1993). A logarithmic reduction algorithm for quasi-birth-death processes. *J. App. Prob.*, **30**, pp. 650-674.

[Latouche and Ramaswami, 1999] LATOUCHE, G. and RAMASWAMI, V. (1999). *Introduction to Matrix Analytic Methods in Stochastic Modeling*, ASA-SIAM, Philadelphia.

[Law and Kelton, 1982] LAW, A.M. and KELTON, W.D. (1982). *Simulation Modeling and Analysis,* McGraw-Hill, New York.

[Lee, 1997] LEE, D. (1997). Analysis of a Two-Queue Model with Bernoulli Schedules, *J. Appl. Prob*, **34**, pp. 176-191.

[Lee et. al., 1999] LEE, H.W., LEE, S.H., YOON, S.H., AHN, B.Y. and PARK, N.I. (1999). A Recursive Method for Bernoulli Arrival Queues and its Application to Partial Buffer Sharing in ATM, *Computers and Operations Research,* **26**, pp. 559-581.

[Lee et. al., 2001] LEE, H.W., SEO, W.J., and YOON, S.H. (2001). An Analysis of Multiple-Class Vacation Queues with Individual Thresholds, *Operations Research Letters,* **28**, pp. 35-49.

[Leemans, 2001] LEEMANS, H. (2001). Waiting Time Distribution in a Two-Class Two-Server Heterogeneous Priority Queue, *Performance Evaluation,* **43**, pp. 133-150.

[Mace, 1998] MACE, S. (1998). Breaking Bandwidth Bottlenecks. *Byte*, **23**, 5, pp. 89.

[Machihara, 1995] MACHIHARA, F. (1995). A Bridge Between Preemptive and Non-preemptive Queueing Models, *Performance Evaluation*, **23**, pp. 93-106.

[Maple, 1996] Maple V Help Menu, Waterloo Maple Inc.

[Marks, 1973] MARKS, B.I. (1973). State Probabilities of M/M/1 Priority Queues. *Operations Research,* **21**, pp. 976–987.

[Miller, 1981] MILLER, D.G. (1981). Compuatation of steady-state probabilities for M/M/1 priority queues, *Operations Research,* **29**, pp. 945-958.

[Morris, 1981] MORRIS, R. (1981). Priority Queuing Networks, *Bell System Technical Journal*, **60**, 8, pp. 1745-1769.

[Morse, 1958] MORSE, P.M. (1958). *Queues, Inventories and Maintenance.* Wiley, New York, pp. 121-127.

[Narahari et. al., 1997] NARAHARI, Y., HEMACHANDRA, N. and GAUR, M.S. (1997). Transient Analysis of Multiclass Manufacturing Systems with Priority Scheduling, *Computers and Operations Research,* 24, 5, pp. 387-398.

[Neuts, 1981] NEUTS, M. (1981). *Matrix-Geometric Solutions in Stochastic Models, an Algorithmic Approach.* John Hopkins University Press.

[Neuts, 1989] NEUTS, M. (1989). *Structures Stochastic Matrices of M/G/1 Type and their Applications.* New York: Marcel Dekker.

[Ng, 1996] NG, C. H. (1996). *Queueing theory Fundamentals.* John Wiley & Sons.

[Nunez Queija, 1997] NÚÑEZ QUEIJA, R. (1997). Steady-state Analysis of a Queue with Varying Service Rate, *Research Report, Centrum voor Wiskunde en Informatica, PNA-R9712, ISSN 1386-3711.*

[Odlyzko, 1997] ODLYZKO, A. (1997). A Modest Proposal for Preventing Internet Congestion, Unpublished manuscript, http://www.dtc.umn.edu/~odlyzko/doc/recent.html.

[Odlyzko, 1999a] ODLYZKO, A. (1999a). Paris Metro Pricing for the Internet, *Proc. ACM Conference on Electronic Commerce,* ACM, pp. 140-147.

[Odlyzko, 1999b] ODLYZKO, A. (1999b). The Current State and Likely Evolution of the Internet, *Proc. IEEE GLOBECOM,* pp. 1869-1875.

[Ogawa et. al., 2000] OGAWA, M., SUEOKA, T. and HATTORI, T. (2000), Priority Based Wireless Packet Communication with Admission and Throughput Control, *Proceedings of the 51st IEEE Conference of Vehicular Technology,* pp. 370-374.

[Peden and Weaver, 1987] PEDEN, J.H. and WEAVER, A.C. (1987). Performance of Priorities on an 802.5 Token Ring, *ACM SIGCOMM Computer Communication Review*, **17**, 5, pp. 58-66.

[Philips and Beightler, 1972] PHILIPS, D.T. and BEIGHTLER,C.S (1972). Procedures for Generating Gamma Variates with Non-Integer Parameters Sets, *Jour. Stat. Comp. Simul.*, 1, pp. 197-208.

[Pike and Hill, 1966] PIKE, P.C. and HILL, I.D. (1966). Algorithm 291. Logarithm of the Gamma Function, C.A.C.M., **9**, 9, pp. 684.

[Posafalvi and Sztrik, 1987] PÓSAFALVI, A. and SZTRIK, J. (1987). On the $\left\langle (m,n)\,(\overrightarrow{M}/\overrightarrow{M})1 \right\rangle$ Priority Queues and their applications, *Problems of Control and Information Theory,* **16**, 3, pp. 169-186.

[Ranasinghe et. al., 2001] RANASINGHE, R., ANDREW, L., HAYES, D. and EVERITT, D. (2001). Scheduling Disciplines for Multimedia WLANs: Embedded Round Robin and Wireless Dual Queue, *Proc. IEEE Int. Conf. Commun. (ICC) '01*, Helsinki, Finland, pp. 1243-1248.

[Ramaswami, 1986] RAMASWAMI, V. and LATOUCHE, G. (1986). A General Class of Markov Processes with Explicit Matrix-Geometric Solutions, *OR Spektrum*, 8, pp. 209-218.

[Rege and Sengupta, 1985] REGE, K.M. and SENGUPTA, B. (1985). A Priority-Based Admission Scheme for a Multiclass Queueing System, *AT&T Technical Journal,* **64**, 7, pp. 1731-1753.

[Rolland et. al., 1999] ROLLAND, E., AMIRI, A. and BARKHI, R. (1999). Queueing Delay Guarantees in Bandwidth Packing, *Computers and Operations Research,* **26**, pp. 921-935.

[Rose, 1995] ROSE, O. (1995). Simple and Efficient Models for Variable Bit Rate MPEG Video Traffic, *Performance Evaluation*, **30**, pp. 69-85.

[Ross, 2001] ROSS, S.M (2001). *Introduction to Probability Models*, Academic Press.

[Sharma and Virtamo, 2002] SHARMA, V. and VIRTAMO, J.T. (2002). A Finite Buffer Queue with Priorities, *Performance Evaluation,* **47**, pp. 1-22.

[Shreedhar and Varghese, 1996] SHREEDHAR, M. and VARGHESE, G. (1996). Efficient Fair Queuing using Deficit Round Robin, *IEEE/ACM Trans. Networking*, 4, 3, pp. 375-385.

[Shi, 1995] SHI, L. (1995). Approximate Analysis for Queueing Networks with Finite Capacity and Customer loss, *European Journal of Operational Research,* **85**, pp. 178-191.

[Singh, 1977] SINGH, J. (1977). A Finite Queueing System with Preemptive Priorities and Multiserver Facility, *Pure and Applied Mathmatika Sciences,* **6**, 1-2, pp. 43-46.

[Siripongwutikorn et. al., 2000] SIRIPONGWUTIKORN, P., LABRADOR, M.and ZNATI, T. (2000). A Wireless-Aware Packet Dropping Policy for ATM Networks, *Proc. Comm. Networks and Distributed Systems Modeling and Simulation Conf.*, pp. 155-160.

[Stadje, 1995] STADJE, W.(1995). The Busy Period of some Queueing Systems, *Stochastic Processes and their Applications,* **55**, pp. 159-167.

[Stephan, 1958] STEPHAN, F.F. (1958). Two Queues under Preemptive Priority with Poisson Arrival and Service Rates, *Operations Research*, **6**, pp. 399-418.

[Tadikamalla, 1978] TADIKAMALLA, P.R. (1978). Computer Generation of Gamma Random Variables, C.A.C.M., **21**, 5, pp. 419-422.

[Takahashi et. al., 2000] TAKAHASHI, M., OSAWA, H. and FUJISAWA, T. (2000). On a Synchronization Queue with Two Finite Buffers, *Queueing Systems*, **36**, pp. 107-123.

[Van Dijk and Miyaza, 1997] VAN DIJK, N.M. and MIYAZAWA, M. (1997). Error Bounds on a Practical Approximation for Finite Tandem Queues, *Operations Research Letters*, **21**, pp. 201-208.

[Venkataramani et. al., 1997] VENKATARAMANI, B., BOSE, S.K. and SRIVATHSAN, K.R. (1997). Queuing Analysis of a Non-preemptive MMPP/D/1 Priority System, *Computer Communications*, **20**, pp. 999-1018.

[Venkatesan and Zeephongsekul, 1992] VENKATESAN, G. and ZEEPHONGSEKUL, P. (1992). Cost Optimization and Quality Control of Products from Queues with Preemptive Priority, *Opsearch*, **29**, 4, pp. 261-273.

[Vijaya Laxmi and Gupta, 1999] VIJAYA LAXMI, P. and GUPTA, U.C. (1999). On the Finite-Buffer Bulk-Service Queue with General Independent Arrivals: $GI/M^{[b]}/1/N$, *Operations Research Letters*, **25**, pp. 241-245.

[Wagner and Krieger, 1999] WAGNER, D. and KRIEGER, U. (1999). Analysis of a Finite Buffer with Non-Preemptive Priority Scheduling, *Communications in Statistics-Stochastic Models*, **15**, 2, pp. 345-365.

[Wagner, 1997] WAGNER, D. (1997). Waiting Times of a Finite-Capacity Multi-Server Model with Non-Preemptive Priorities, *European Journal of Operational Research*, **102**, pp. 227-241.

[Wang and Yonatan, 2000] WANG, J. and YONATAN, L. (2000). Managing Performance Using Weighted Round Robin, *IEEE Symposium on Computers and Communications*, pp. 785-792.

[Welch, 1981] WELCH, P.D. (1981). *On the Problem of the Initial Transient in Steady-State Simulation*, IBM Watson Research Center,Yorktown Heights, New York.

[Welch, 1982] WELCH, P.D. (1982). The Statistical Analysis of Simulation Results, *The Computer Performance Modeling Handbook*, pp. 268-328.

[Wheeler, 1975] WHEELER, D.J. (1975) An Approximation for Simulation of Gamma Distributions, *Jour. Stat. Comp. Simul.*, **3**, pp. 225-232.

[White and Christie, 1958] WHITE, H. and CHRISTIE, L.S. (1958). Queueing with Preemptive Priorities or with Breakdowns, *Operations Research*, **6**, pp. 79-95.

[Willmot, 1998] WILLMOT, G.E. (1998). On a Class Approximation for Ruin and Waiting Time Probabilities, *Operations Research Letters*, **22**, pp. 27-32.

[Wolff, 1982] Poisson Arrivals See Time Averages, *Operations Research*, **30**, pp. 223-231.

[Xu and Zhao, 1995] XU, S.H. and ZHAO, Y.Q. (1995). Dynamic Routing and Jockeying Controls in a Two-Station Queueing System, *Adv.Appl. Prob.*, 28, 4, pp. 1201-1226.

[Yau and Pawlikowski, 1992] YAU, V. and PAWLIKOWSKI, K. (1992). Improved Nested-Threshold-Cell-Discard Buffer Management Mechanisms, *IEEE Region 10 Conference, Tencon*, pp. 820-824.

[Yeo, 1963] YEO, G.F. (1963). Preemptive Priority Queues, *Journal of the Australian Mathematical Society,* **3**, pp. 491-502.

[Zeephongsekul and Venkatesan, 1993] ZEEPHONGSEKUL, P., and VENKATESAN, G. (1993). Quality Control of Products From Queues wth a Non-Preemptive Priority Service Discipline, *Asia-Pacific Journal of Operational Research,* **10**, pp. 135-143.

[Zeephongsekul et.al., 2002] ZEEPHONGSEKUL, P., BEDFORD, A. and SAUNDERS, J. (2002). A matrix-algorithmic approach to solving a preemptive priority queue with finite buffer. *Research Report No. 10, Department of Mathematics and Statistics,* RMIT University.

[Zeeephongsekul and Bedford, 2003] ZEEPHONGSEKUL, P., and BEDFORD, A. (2003). Waiting Time Analysis of the Multiple Priority Dual Queue with a Preemptive Priority Service Discipline. *Research Report No. 9, Department of Mathematics and Statistics,* RMIT University.

[Zeephongsekul and Bedford, 2003] Waiting Time Analysis of the Multiple Priority Dual Queue with a Preemptive Priority Service Discipline. *Submitted to EJOR.*

Chapter 9

Appendix

9.0.1 MPDQ Preemptive and non-preemptive algorithms

The simulations were conducted using the Arena Simulation Software package. Arena has provided effective results of queueing disciplines in our prior work and was again chosen here as our tool of analysis. In constructing the Arena model, we first needed to define the MPDQ as an algorithm. Algorithm 1 below is the simpler non-preemptive algorithm stated in generic Arena code.

Algorithm 8 *WHILE (A Customer is generated OR Customer in System) DO*
!Generate Arriving Traffic:
Class 1 : EXPO(λ_1)
Class 2 : EXPO(λ_2)
IF (Num. in Queue 1 < Queue 1 Size) THEN
#1:Sort Queue in High Class First Order
Head-Of-Line enter service
IF Class = 1 THEN
EXPO(λ_1)
ELSE
EXPO(λ_2)
END-IF
Send SIGNAL to release Head-Of-Line Customer from Queue 2
Traffic DEPARTS
ELSE-IF (Num. in Queue 2 < Queue 2 Size) THEN
Sort Queue in High Class First Order
WAIT until SIGNAL then Head-Of-Line customer GOTO #1
ELSE Traffic is lost
END-IF

END-WHILE

The non-preemptive process is quite simple to simulate due to there being no interruption. The above algorithm gives an idea of the simulation process in Arena. The simulation time does not advance until all the criteria above are checked, so an arriving customer can enter during a service period and the queue is reshuffled if required. Next we display the preemptive algorithm, one which is more difficult to code in Arena.

Algorithm 9 *Class1in=0*

WHILE (A Customer is generated OR Customer in System) DO
Class 1 : EXPO(λ_1)
Class 2 : EXPO(λ_2)
IF (Num. in Queue 1 < Queue 1 Size) THEN
IF(Class = 1)THEN
Class1in = Class1in +1;
END-IF
IF(Class = 1.AND.Class1in = 1)THEN
SUB PREEMPT
END-IF
#1:Sort Queue in High Class First Order
Serve Head-Of-Line:
Head-Of-Line enter service
IF Class = 1 THEN
EXPO(λ_1)
ELSE
EXPO(λ_2)
END-IF
Send SIGNAL to release Head-Of-Line Customer from Queue 2
Class1in = Class1in-1;
Traffic DEPARTS
ELSE-IF (Num. in Queue 2 < Queue 2 Size) THEN
Sort Queue in High Class First Order
WAIT until SIGNAL then H-O-L joins Queue 1 GOTO #1;
ELSE Traffic is lost
END-IF
END-WHILE
SUBROUTINE PREEMPT:

IF(Class in Service>Class in Queue) THEN
Swap in Service with Head-Of-Line
RETURN
ELSE
RETURN
END-IF
END-SUBROUTINE

As can be seen from Algorithm 2, the preemption process required a separate statement to check for the quantity of first class customers in the system and the preempt subroutine to expel a lower classed customer from service if a higher classed customer is in the primary queue. Both algorithms are simple to extend to multiple classes of customers. The simulation model developed for the non-preemptive simulation was substantially simpler than the preemptive model. Analytically the reverse holds, with the non-preemptive solution being more difficult to obtain than the preemptive solution using the matrix-analytic computational algorithm solution process.

9.1 Maple Code

This section contains the various maple codes used. The codes given here work with version 8. Older versions shall also run the code, however the operator % must be replaced by "

9.1.1 Preemptive Single Queue Code

```
restart; st := time();
    c:=8;lambda[1]:=2;lambda[2]:=3;mu[1]:=5;mu[2]:=4;
    ****Enter the parameters in the above line****
    lambda[0]:=lambda[1]+lambda[2];
    chi[1,1]:=(lambda[1]/(mu[1]+lambda[0]));
    chi[2,1]:=(lambda[2]/(mu[1]+lambda[0]));
    alpha[1,1]:=(mu[1]/(mu[1]+lambda[0]));
    alpha[2,2]:=(mu[2]/(mu[2]+lambda[0]));
    rho[1]:=lambda[1]/mu[1];
    rho[2]:=lambda[2]/mu[2];
    rho[1]+rho[2];
    pi[c,0]:=rho[1];
    pi[c-1,0]:=1;
    top:=c-2;
```

```
for i from top by -1 to 0 do
  eqn[i]:=pi[i,0]=(chi[1,1]^(-1))*pi[i+1,0]-(rho[1]^(-1))*pi[i+2,0];
od;
eqnset:={seq(eqn[i],i=0..top)};
solset:={seq(pi[i,0],i=0..top)};
s:=solve(eqnset,solset);
assign(s);
******Outer Loop begins,We initialise by setting j0 =1 and i0 =c-1******
j0:=1;i0:=c-1;
for j0 from 1 to c-1 do
  pi[0,j0]:=(-(mu[1]/mu[2])* pi[1,j0-1])+'if'(j0 > 1,(-rho[2]*pi[0,j0-2]+((alpha[2,2])^(-
1))*pi[0,j0-1]),0)+'if'(j0 = 1,((lambda[0]/mu[2])*pi[0,j0-1]),0);
    for i from i0 by -1 to 1 do
    eqn[i]:=pi[i,j0]=chi[2,1]*pi[i,j0-1]+chi[1,1]*pi[i-1,j0]+'if'(i < i0,alpha[1,1]*pi[i+1,j0],0);
    od;
  eqnset:={seq(eqn[i],i=1..i0)};
  solset:={seq(pi[i,j0],i=1..i0)};
  s:=solve(eqnset,solset);
  assign(s);
  i0:=i0-1;
od;
pi[0,c]:=rho[2]*pi[0,c-1];
for i from 0 to c do
  z[i]:=sum('pi[i,j]','j'=0..c-i);
od;
a:=sum('z[i]','i'=0..c);
G:=solve(a*k=1,k);
for i from 0 to c do
  for j from 0 to c-i do
     pi[i,j]:=evalf(pi[i,j]*G);
  od;
od;
pi[Busy]:=1-pi[0,0];
sum('pi[i,0]','i'=1..c);
sum('pi[0,i]','i'=1..c);
evalf(%);
for i from 0 to c do
  j:=c-i; qa[i]:=pi[i,j];
```

```
od;
sum('qa[i]','i'=0..c);
time() - st;
kernelopts( bytesused );
evalf(%/1024);
```

9.1.2 Preemptive MPDQ

```
restart;st := time();
c1:=5;c2:=3;lambda[1]:=1;lambda[2]:=3;mu[1]:=5;mu[2]:=4;
******Enter the parameters in the above line******
lambda[0]:=lambda[1]+lambda[2];
chi[1,1]:=(lambda[1]/(mu[1]+lambda[0]));
chi[2,1]:=(lambda[2]/(mu[1]+lambda[0]));
chi[2,2]:=(lambda[2]/(mu[2]+lambda[0]));
chi[1,2]:=(lambda[1]/(mu[2]+lambda[0]));
alpha[1,1]:=(mu[1]/(mu[1]+lambda[0]));
alpha[2,2]:=(mu[2]/(mu[2]+lambda[0]));
alpha[2,1]:=(mu[2]/(mu[1]+lambda[0]));
alpha[1,2]:=(mu[1]/(mu[2]+lambda[0]));
rho[1]:=lambda[1]/mu[1];
rho[2]:=lambda[2]/mu[2];
rho[1]+rho[2];
pi[c1,c2,0]:=rho[1];
pi[c1,c2-1,0]:=1;
top:=c2-2;
for j from top by -1 to 0 do
  eqn[j]:='if'(j=0,pi[c1,0],0)+'if'(j>0,pi[c1,j,0],0)=(chi[1,1]^(-1))*pi[c1,j+1,0]-(rho[1]^(-
1))*pi[c1,j+2,0];
od;
eqnset:={seq(eqn[j],j=0..top)};
solset:={seq(pi[c1,j,0],j=1..top),pi[c1,0]};
s:=solve(eqnset,solset);
assign(s);
******Outer Loop begins,We initialise by setting j0 =1 and i0 =c-1******
******This solves downwards,substituting solutions as it moves forward******
eqn[0]:=pi[5,0,1]=alpha[1,1]*pi[5,1,1]+chi[2,1]*pi[5,0];
eqn[1]:=pi[5,1,1]=alpha[1,1]*pi[5,2,1]+chi[2,1]*pi[5,1,0]+chi[1,1]*pi[5,0,1];
```

```
eqn[2]:=pi[5,2,1]=(lambda[2]/mu[1])*pi[5,2,0]+rho[1]*pi[5,1,1];
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[5,j,1],j=0..2)};
s:=solve(eqnset,solset);
assign(s);
eqn[0]:=pi[5,0,2]=alpha[1,1]*pi[5,1,2]+chi[2,1]*pi[5,0,1];
eqn[1]:=pi[5,1,2]=(lambda[2]/mu[1])*pi[5,1,1]+rho[1]*pi[5,0,2];
eqnset:={seq(eqn[j],j=0..1)};
solset:={seq(pi[5,j,2],j=0..1)};
s:=solve(eqnset,solset);
assign(s);
pi[5,0,3]:=(lambda[2]/mu[1])*pi[5,0,2];
pi[4,0]:=(chi[1,1]^(-1))*pi[5,0]-(rho[1]^(-1))*pi[5,1,0];
for i from 3 by -1 to 0 do
  pi[i,0]:=(chi[1,1]^(-1))*pi[i+1,0]-(rho[1]^(-1))*pi[i+2,0];
od;
pi[0,1]:=(lambda[0]/mu[2])*pi[0,0]-(mu[1]/mu[2])*pi[1,0];
eqn[0]:=pi[4,1]=chi[2,1]*pi[4,0]+chi[1,1]*pi[3,1]+alpha[1,1]*(pi[4,1,0]+pi[5,0,1]);
for i from 3 by -1 to 1 do
  eqn[4-i]:=pi[i,1]=chi[2,1]*pi[i,0]+chi[1,1]*pi[i-1,1]+alpha[1,1]*pi[i+1,1];
od;
eqnset:={seq(eqn[j],j=0..3)};
solset:={seq(pi[i,1],i=1..4)};
s:=solve(eqnset,solset);
******Above is Result A******
eqn[0]:=pi[4,1,0]=chi[1,1]*pi[4,1]+alpha[1,1]*pi[4,2,0];
eqn[1]:=pi[4,2,0]=chi[1,1]*pi[4,1,0]+alpha[1,1]*pi[4,3,0];
eqn[2]:=pi[4,3,0]=rho[1]*pi[4,2,0];
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[4,i,0],i=1..3)};
t:=solve(eqnset,solset);
solset:={seq(pi[4,i,0],i=1..3),seq(pi[i,1],i=1..4)};
eqnset:=s union t;
u:=solve(eqnset,solset);
assign(u);
eqn[0]:=pi[4,0,1]=alpha[1,1]*pi[4,1,1]+chi[2,1]*pi[4,1]+alpha[1,1]*pi[5,0,2];
eqn[1]:=pi[4,1,1]=alpha[1,1]*pi[4,2,1]+alpha[1,1]*pi[4,0,1];
eqn[2]:=pi[4,2,1]=(lambda[2]/mu[1])*pi[4,2,0]+rho[1]*pi[4,1,1];
```

```
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[4,i,1],i=0..2)};
s:=solve(eqnset,solset);
assign(s);
eqn[0]:=pi[4,0,2]=alpha[1,1]*pi[4,1,2]+chi[2,1]*pi[4,0,1]+alpha[1,1]*pi[5,0,3];
eqn[1]:=pi[4,1,2]=(lambda[2]/mu[1])*pi[4,1,1]+rho[1]*pi[4,0,2];
eqnset:={seq(eqn[j],j=0..1)};
solset:={seq(pi[4,i,2],i=0..1)};
s:=solve(eqnset,solset);
assign(s);
pi[4,0,3]:=(lambda[2]/mu[1])*pi[4,0,2];
pi[0,2]:=-rho[2]*pi[0,0]+(alpha[2,2]^(-1))*pi[0,1]-(mu[1]/mu[2])*pi[1,1];
eqn[0]:=pi[3,2]=chi[2,1]*pi[3,1]+chi[1,1]*pi[2,2]+alpha[1,1]*(pi[3,1,0]+pi[4,0,1]);
for i from 2 by -1 to 1 do
  eqn[3-i]:=pi[i,2]=chi[2,1]*pi[i,1]+chi[1,1]*pi[i-1,2]+alpha[1,1]*pi[i+1,2];
od;
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[i,2],i=1..3)};
s:=solve(eqnset,solset);
******Result A******
eqn[0]:=pi[3,1,0]=chi[1,1]*pi[3,2]+alpha[1,1]*pi[3,2,0];
eqn[1]:=pi[3,2,0]=chi[1,1]*pi[3,1,0]+alpha[1,1]*pi[3,3,0];
eqn[2]:=pi[3,3,0]=rho[1]*pi[3,2,0];
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[3,i,0],i=1..3)};
t:=solve(eqnset,solset);
solset:={seq(pi[3,i,0],i=1..3),seq(pi[i,2],i=1..3)};
eqnset:=s union t;
u:=solve(eqnset,solset);
assign(u);
eqn[0]:=pi[3,0,1]=alpha[1,1]*pi[3,1,1]+chi[2,1]*pi[3,2]+alpha[1,1]*pi[4,0,2];
eqn[1]:=pi[3,1,1]=alpha[1,1]*pi[3,2,1]+chi[1,1]*pi[3,0,1];
eqn[2]:=pi[3,2,1]=(lambda[2]/mu[1])*pi[3,2,0]+rho[1]*pi[3,1,1];
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[3,i,1],i=0..2)};
s:=solve(eqnset,solset);
assign(s);
eqn[0]:=pi[3,0,2]=alpha[1,1]*pi[3,1,2]+chi[2,1]*pi[3,0,1]+alpha[1,1]*pi[4,0,3];
```

```
eqn[1]:=pi[3,1,2]=(lambda[2]/mu[1])*pi[3,1,1]+rho[1]*pi[3,0,2];
eqnset:={seq(eqn[j],j=0..1)};
solset:={seq(pi[3,i,2],i=0..1)};
s:=solve(eqnset,solset);
assign(s);
pi[3,0,3]:=(lambda[2]/mu[1])*pi[3,0,2];
pi[0,3]:=-rho[2]*pi[0,1]+(alpha[2,2]^(-1))*pi[0,2]-(mu[1]/mu[2])*pi[1,2];
eqn[0]:=pi[2,3]=chi[2,1]*pi[2,2]+chi[1,1]*pi[1,3]+alpha[1,1]*(pi[2,1,0]+pi[3,0,1]);
eqn[1]:=pi[1,3]=chi[2,1]*pi[1,2]+chi[1,1]*pi[0,3]+alpha[1,1]*pi[2,3];
eqnset:={seq(eqn[j],j=0..1)};
solset:={seq(pi[i,3],i=1..2)};
s:=solve(eqnset,solset);
eqn[0]:=pi[2,1,0]=chi[1,1]*pi[2,3]+alpha[1,1]*pi[2,2,0];
eqn[1]:=pi[2,2,0]=chi[1,1]*pi[2,1,0]+alpha[1,1]*pi[2,3,0];
eqn[2]:=pi[2,3,0]=rho[1]*pi[2,2,0];
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[2,i,0],i=1..3)};
t:=solve(eqnset,solset);
solset:={seq(pi[2,i,0],i=1..3),seq(pi[i,3],i=1..2)};
eqnset:=s union t;
u:=solve(eqnset,solset);
assign(u);
eqn[0]:=pi[2,0,1]=alpha[1,1]*pi[2,1,1]+chi[2,1]*pi[2,3]+alpha[1,1]*pi[3,0,2];
eqn[1]:=pi[2,1,1]=alpha[1,1]*pi[2,2,1]+chi[1,1]*pi[2,0,1];
eqn[2]:=pi[2,2,1]=(lambda[2]/mu[1])*pi[2,2,0]+rho[1]*pi[2,1,1];
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[2,i,1],i=0..2)};
s:=solve(eqnset,solset);
assign(s);
eqn[0]:=pi[2,0,2]=alpha[1,1]*pi[2,1,2]+chi[2,1]*pi[2,0,1]+alpha[1,1]*pi[3,0,3];
eqn[1]:=pi[2,1,2]=(lambda[2]/mu[1])*pi[2,1,1]+rho[1]*pi[2,0,2];
eqnset:={seq(eqn[j],j=0..1)};
solset:={seq(pi[2,i,2],i=0..1)};
s:=solve(eqnset,solset);
assign(s);
pi[2,0,3]:=(lambda[2]/mu[1])*pi[2,0,2];
pi[0,4]:=-rho[2]*pi[0,2]+((alpha[2,2]^(-1))*pi[0,3])-(mu[1]/mu[2])*pi[1,3];
pi[0,2,0]:=chi[1,2]*pi[0,1,0];
```

```
pi[0,3,0]:=(lambda[1]/mu[2])*chi[1,2]*pi[0,1,0];
pi[0,5]:=(chi[1,2]^(-1))*pi[0,1,0];
pi[1,4]:=(alpha[2,1]^(-1))*pi[0,4]-(mu[2]/mu[1])*pi[0,5]-(lambda[2]/mu[1])*pi[0,3];
******Result 1******
pi[0,1,0]=-(lambda[1]/mu[2])*pi[0,4]+((alpha[2,1]^(-1))*pi[1,4])-(rho[2]*pi[1,3])-
        ((mu[1]/mu[2])*(pi[1,1,0]+pi[2,0,1]));
******Result2(with result 1 substituted)******
pi[0,1,0]:=solve(",pi[0,1,0]);
******above is Result 2 simplified******
pi[1,1,0]=alpha[2,1]*chi[1,2]*pi[0,1,0]+chi[1,1]*pi[1,4]+alpha[1,1]*pi[1,2,0];
******result 3 with result 1 substituted******
pi[1,1,0]:=solve(",pi[1,1,0]);
pi[1,3,0]:=rho[1]*pi[1,2,0];
pi[1,2,0]=((1-(alpha[1,1]*rho[1]))^(-1))*(alpha[2,1]*pi[0,3,0]+chi[1,1]*pi[1,1,0]);
******Result 4******
pi[1,2,0]:=solve(",pi[1,2,0]);
pi[1,3,0];pi[1,1,0];pi[0,1,0];pi[1,4];pi[0,5];
eq[0]:=pi[0,1,1]=chi[1,2]*pi[0,0,1]+chi[2,2]*pi[0,1,0];
eq[1]:=pi[0,2,1]=chi[1,2]*pi[0,1,1]+chi[2,2]*pi[0,2,0];
eq[2]:=pi[1,2,1]=rho[1]*pi[1,1,1]+(lambda[2]/mu[1])*pi[1,2,0];
eq[3]:=pi[1,1,1]=alpha[1,1]*pi[1,2,1]+alpha[1,2]*pi[0,2,1]+chi[1,1]*pi[1,0,1]+chi[2,1]*pi[1,1,0];
eq[4]:=pi[1,0,1]=alpha[1,1]*pi[1,1,1]+alpha[2,1]*pi[0,1,1]+chi[2,1]*pi[1,4]+alpha[1,1]*pi[2,0,2];
eq[5]:=pi[0,0,1]=(alpha[2,2]^(-1))*pi[0,5]-rho[2]*pi[0,4]-(mu[1]/mu[2])*pi[1,0,1];
eqnset:={eq[0],eq[1],eq[2],eq[3],eq[4],eq[5]};
solset:={pi[1,1,1],pi[1,2,1],pi[0,2,1],pi[1,0,1],pi[0,0,1],pi[0,1,1]};
s:=solve(eqnset,solset);
assign(s);
eq[0]:=pi[0,1,2]=rho[2]*pi[0,1,1]+(lambda[1]/mu[2])*pi[0,0,2];
eq[1]:=pi[1,1,2]=(lambda[2]/mu[1])*pi[1,1,1]+rho[1]*pi[1,0,2];
eq[2]:=pi[1,0,2]=alpha[1,1]*pi[1,1,2]+alpha[2,1]*pi[0,1,2]+chi[2,1]*pi[1,0,1]+alpha[1,1]*pi[2,0,3];
eq[3]:=pi[0,0,2]=-rho[2]*pi[0,5]-(mu[1]/mu[2])*pi[1,0,2]+(alpha[2,2]^(-1))*pi[0,0,1];
eqnset:={eq[0],eq[1],eq[2],eq[3]};
solset:={pi[1,1,2],pi[1,0,2],pi[0,0,2],pi[0,1,2]};
s:=solve(eqnset,solset);
assign(s);
pi[1,0,3]:=(lambda[2]/mu[1])*pi[1,0,2];
pi[0,0,c2]:=rho[2]*pi[0,0,c2-1];
sq:=pi[0,0]+pi[0,1]+pi[0,2]+pi[0,3]+pi[0,4]+pi[0,5]+pi[1,0]+pi[1,1]+pi[1,2]+pi[1,3]+pi[1,4]+
```

```
    pi[2,0]+pi[2,1]+pi[2,2]+pi[2,3]+pi[3,0]+pi[3,1]+pi[3,2]+pi[4,0]+pi[4,1]+pi[5,0];
dq0:=pi[0,1,0]+pi[0,0,1]+pi[0,0,2]+pi[0,0,3]+pi[0,1,1]+pi[0,1,2]+pi[0,2,0]+pi[0,2,1]+pi[0,3,0];
dq1:=pi[1,1,0]+pi[1,0,1]+pi[1,0,2]+pi[1,0,3]+pi[1,1,1]+pi[1,1,2]+pi[1,2,0]+pi[1,2,1]+pi[1,3,0];
dq2:=pi[2,1,0]+pi[2,0,1]+pi[2,0,2]+pi[2,0,3]+pi[2,1,1]+pi[2,1,2]+pi[2,2,0]+pi[2,2,1]+pi[2,3,0];
dq3:=pi[3,1,0]+pi[3,0,1]+pi[3,0,2]+pi[3,0,3]+pi[3,1,1]+pi[3,1,2]+pi[3,2,0]+pi[3,2,1]+pi[3,3,0];
dq4:=pi[4,1,0]+pi[4,0,1]+pi[4,0,2]+pi[4,0,3]+pi[4,1,1]+pi[4,1,2]+pi[4,2,0]+pi[4,2,1]+pi[4,3,0];
dq5:=pi[5,1,0]+pi[5,0,1]+pi[5,0,2]+pi[5,0,3]+pi[5,1,1]+pi[5,1,2]+pi[5,2,0]+pi[5,2,1]+pi[5,3,0];
all:=sq+dq0+dq1+dq2+dq3+dq4+dq5;
G:=solve(all*k=1,k);
pi[0,0]:=evalf(G*pi[0,0]);
pi[0,1]:=evalf(G*pi[0,1]);
pi[0,2]:=evalf(G*pi[0,2]);
pi[0,3]:=evalf(G*pi[0,3]);
pi[0,4]:=evalf(G*pi[0,4]);
pi[0,5]:=evalf(G*pi[0,5]);
pi[1,0]:=evalf(G*pi[1,0]);
pi[1,1]:=evalf(G*pi[1,1]);
pi[1,2]:=evalf(G*pi[1,2]);
pi[1,3]:=evalf(G*pi[1,3]);
pi[1,4]:=evalf(G*pi[1,4]);
pi[2,0]:=evalf(G*pi[2,0]);
pi[2,1]:=evalf(G*pi[2,1]);
pi[2,2]:=evalf(G*pi[2,2]);
pi[2,3]:=evalf(G*pi[2,3]);
pi[3,0]:=evalf(G*pi[3,0]);
pi[3,1]:=evalf(G*pi[3,1]);
pi[3,2]:=evalf(G*pi[3,2]);
pi[4,0]:=evalf(G*pi[4,0]);
pi[4,1]:=evalf(G*pi[4,1]);
pi[5,0]:=evalf(G*pi[5,0]);
pi[0,1,0]:=evalf(G*pi[0,1,0]);
pi[0,0,1]:=evalf(G*pi[0,0,1]);
pi[0,0,2]:=evalf(G*pi[0,0,2]);
pi[0,0,3]:=evalf(G*pi[0,0,3]);
pi[0,1,1]:=evalf(G*pi[0,1,1]);
pi[0,1,2]:=evalf(G*pi[0,1,2]);
pi[0,2,0]:=evalf(G*pi[0,2,0]);
pi[0,2,1]:=evalf(G*pi[0,2,1]);
```

```
pi[0,3,0]:=evalf(G*pi[0,3,0]);
pi[1,1,0]:=evalf(G*pi[1,1,0]);
pi[1,0,1]:=evalf(G*pi[1,0,1]);
pi[1,0,2]:=evalf(G*pi[1,0,2]);
pi[1,0,3]:=evalf(G*pi[1,0,3]);
pi[1,1,1]:=evalf(G*pi[1,1,1]);
pi[1,1,2]:=evalf(G*pi[1,1,2]);
pi[1,2,0]:=evalf(G*pi[1,2,0]);
pi[1,2,1]:=evalf(G*pi[1,2,1]);
pi[1,3,0]:=evalf(G*pi[1,3,0]);
pi[2,1,0]:=evalf(G*pi[2,1,0]);
pi[2,0,1]:=evalf(G*pi[2,0,1]);
pi[2,0,2]:=evalf(G*pi[2,0,2]);
pi[2,0,3]:=evalf(G*pi[2,0,3]);
pi[2,1,1]:=evalf(G*pi[2,1,1]);
pi[2,1,2]:=evalf(G*pi[2,1,2]);
pi[2,2,0]:=evalf(G*pi[2,2,0]);
pi[2,2,1]:=evalf(G*pi[2,2,1]);
pi[2,3,0]:=evalf(G*pi[2,3,0]);
pi[3,1,0]:=evalf(G*pi[3,1,0]);
pi[3,0,1]:=evalf(G*pi[3,0,1]);
pi[3,0,2]:=evalf(G*pi[3,0,2]);
pi[3,0,3]:=evalf(G*pi[3,0,3]);
pi[3,1,1]:=evalf(G*pi[3,1,1]);
pi[3,1,2]:=evalf(G*pi[3,1,2]);
pi[3,2,0]:=evalf(G*pi[3,2,0]);
pi[3,2,1]:=evalf(G*pi[3,2,1]);
pi[3,3,0]:=evalf(G*pi[3,3,0]);
pi[4,1,0]:=evalf(G*pi[4,1,0]);
pi[4,0,1]:=evalf(G*pi[4,0,1]);
pi[4,0,2]:=evalf(G*pi[4,0,2]);
pi[4,0,3]:=evalf(G*pi[4,0,3]);
pi[4,1,1]:=evalf(G*pi[4,1,1]);
pi[4,1,2]:=evalf(G*pi[4,1,2]);
pi[4,2,0]:=evalf(G*pi[4,2,0]);
pi[4,2,1]:=evalf(G*pi[4,2,1]);
pi[4,3,0]:=evalf(G*pi[4,3,0]);
pi[5,1,0]:=evalf(G*pi[5,1,0]);
```

```
pi[5,0,1]:=evalf(G*pi[5,0,1]);
pi[5,0,2]:=evalf(G*pi[5,0,2]);
pi[5,0,3]:=evalf(G*pi[5,0,3]);
pi[5,1,1]:=evalf(G*pi[5,1,1]);
pi[5,1,2]:=evalf(G*pi[5,1,2]);
pi[5,2,0]:=evalf(G*pi[5,2,0]);
pi[5,2,1]:=evalf(G*pi[5,2,1]);
pi[5,3,0]:=evalf(G*pi[5,3,0]);
pi[0, 0] := 0.1419486571
sq:=pi[0,0]+pi[0,1]+pi[0,2]+pi[0,3]+pi[0,4]+pi[0,5]+pi[1,0]+pi[1,1]+pi[1,2]+pi[1,3]+pi[1,4]+
    pi[2,0]+pi[2,1]+pi[2,2]+pi[2,3]+pi[3,0]+pi[3,1]+pi[3,2]+pi[4,0]+pi[4,1]+pi[5,0];
dq0:=pi[0,1,0]+pi[0,0,1]+pi[0,0,2]+pi[0,0,3]+pi[0,1,1]+pi[0,1,2]+pi[0,2,0]+pi[0,2,1]+pi[0,3,0];
dq1:=pi[1,1,0]+pi[1,0,1]+pi[1,0,2]+pi[1,0,3]+pi[1,1,1]+pi[1,1,2]+pi[1,2,0]+pi[1,2,1]+pi[1,3,0];
dq2:=pi[2,1,0]+pi[2,0,1]+pi[2,0,2]+pi[2,0,3]+pi[2,1,1]+pi[2,1,2]+pi[2,2,0]+pi[2,2,1]+pi[2,3,0];
dq3:=pi[3,1,0]+pi[3,0,1]+pi[3,0,2]+pi[3,0,3]+pi[3,1,1]+pi[3,1,2]+pi[3,2,0]+pi[3,2,1]+pi[3,3,0];
dq4:=pi[4,1,0]+pi[4,0,1]+pi[4,0,2]+pi[4,0,3]+pi[4,1,1]+pi[4,1,2]+pi[4,2,0]+pi[4,2,1]+pi[4,3,0];
dq5:=pi[5,1,0]+pi[5,0,1]+pi[5,0,2]+pi[5,0,3]+pi[5,1,1]+pi[5,1,2]+pi[5,2,0]+pi[5,2,1]+pi[5,3,0];
sq+dq0+dq1+dq2+dq3+dq4+dq5;
```

9.1.3 Non-preemptive MPDQ

```
restart;
    st := time();
    c1:=4;c2:=4;lambda[1]:=1;lambda[2]:=3;mu[1]:=6;mu[2]:=5;
    ******Enter the parameters in the above line******
    lambda[0]:=lambda[1]+lambda[2];
    chi[1,1]:=(lambda[1]/(mu[1]+lambda[0]));
    chi[2,1]:=(lambda[2]/(mu[1]+lambda[0]));
    chi[2,2]:=(lambda[2]/(mu[2]+lambda[0]));
    chi[1,2]:=(lambda[1]/(mu[2]+lambda[0]));
    alpha[1,1]:=(mu[1]/(mu[1]+lambda[0]));
    alpha[2,2]:=(mu[2]/(mu[2]+lambda[0]));
    alpha[2,1]:=(mu[2]/(mu[1]+lambda[0]));
    alpha[1,2]:=(mu[1]/(mu[2]+lambda[0]));
    rho[1]:=lambda[1]/mu[1];
    rho[2]:=lambda[2]/mu[2];
    rho[1]+rho[2];
    pi[c1,c2,0,2]:=lambda[1]/mu[2];
```

```
pi[c1,c2-1,0,2]:=1;
for i from c2-2 by -1 to 1 do
   pi[c1,i,0,2]:=chi[1,2]*pi[c1,i+1,0,2];
od;
pi[c1,0,2]:=chi[1,2]*pi[c1,1,0,2];
pi[4,0,1,2]:=chi[2,2]*pi[4,0,2];
loop i'=1,j'=1
pi[4,1,1,2]:=chi[1,2]*pi[4,0,1,2]+chi[2,2]*pi[4,1,0,2];
pi[4,2,1,2]:=chi[1,2]*pi[4,1,1,2]+chi[2,2]*pi[4,2,0,2];
pi[4,3,1,2]:=(lambda[1]/rho[2])*pi[4,2,1,2]+rho[2]*pi[4,3,0,2];
i'=0,j'=2
pi[4,0,2,2]:=chi[2,2]*pi[4,0,1,2];
******loop i'=1..2 , j'=2******
pi[4,1,2,2]:=chi[1,2]*pi[4,0,2,2]+chi[2,2]*pi[4,1,1,2];
pi[4,2,2,2]:=(lambda[1]/rho[2])*pi[4,1,2,2]+rho[2]*pi[4,2,1,2];
******i'=0,j'=3******
pi[4,0,3,2]:=chi[2,2]*pi[4,0,2,2];
******i'=1,j'=3******
pi[4,1,3,2]:=(lambda[1]/rho[2])*pi[4,0,3,2]+rho[2]*pi[4,1,2,2];
******finally*****
pi[4,0,4,2]:=rho[2]*pi[4,0,3,2];
******The above solves all (4,i',j',2)******
for i from 3 by -1 to 0 do
   pi[i,0,2]:=(1/chi[1,2])*pi[i+1,0,2];
od;
eqn[1]:=pi[4,4,0,1]=rho[1]*pi[4,3,0,1];
eqn[2]:=pi[4,3,0,1]=alpha[1,1]*pi[4,4,0,1]+chi[1,1]*pi[4,2,0,1]+alpha[2,1]*pi[4,4,0,2];
eqn[3]:=pi[4,2,0,1]=alpha[1,1]*pi[4,3,0,1]+chi[1,1]*pi[4,1,0,1]+alpha[2,1]*pi[4,3,0,2];
eqn[4]:=pi[4,1,0,1]=alpha[1,1]*pi[4,2,0,1]+chi[1,1]*pi[4,0,1]+alpha[2,1]*pi[4,2,0,2];
eqn[5]:=pi[0,0,0]=(mu[1]/lambda[0])*pi[0,0,1]+(mu[2]/lambda[0])*pi[0,0,2];
eqn[6]:=pi[0,0,1]=chi[1,1]*pi[0,0,0]+alpha[1,1]*pi[1,0,1]+alpha[2,1]*pi[1,0,2];
eqn[7]:=pi[1,0,1]=alpha[1,1]*pi[2,0,1]+chi[1,1]*pi[0,0,1]+alpha[2,1]*pi[2,0,2];
eqn[8]:=pi[2,0,1]=alpha[1,1]*pi[3,0,1]+chi[1,1]*pi[1,0,1]+alpha[2,1]*pi[3,0,2];
eqn[9]:=pi[3,0,1]=alpha[1,1]*pi[4,0,1]+chi[1,1]*pi[2,0,1]+alpha[2,1]*pi[4,0,2];
eqn[10]:=pi[4,0,1]=alpha[1,1]*pi[4,1,0,1]+chi[1,1]*pi[3,0,1]+alpha[2,1]*pi[4,1,0,2];
eqnset:={seq(eqn[j],j=1..10)};
solset:={seq(pi[c1,j,0,1],j=1..4),seq(pi[j,0,1],j=0..4),pi[0,0,0]};
s:=solve(eqnset,solset);
```

```
assign(s);
******Outer Loop begins,We initialise by setting j0 =1 and i0 =c-j0******
******This solves downwards,substituting solutions as it moves forward******
******j=1,i=3,2.,0******
eqn[0]:=pi[4,3,1,1]=rho[1]*pi[4,3,1,1]+(lambda[2]/mu[1])*pi[4,3,0,1];
eqn[1]:=pi[4,2,1,1]=alpha[1,1]*pi[4,3,1,1]+chi[1,1]*pi[4,1,1,1]
+chi[2,1]*pi[4,2,0,1]+alpha[2,1]*pi[4,3,1,2];
eqn[2]:=pi[4,1,1,1]=alpha[1,1]*pi[4,2,1,1]+chi[1,1]*pi[4,0,1,1]
+chi[2,1]*pi[4,1,0,1]+alpha[2,1]*pi[4,2,1,2];
eqn[3]:=pi[4,0,1,1]=alpha[1,1]*pi[4,1,1,1]+chi[2,1]*pi[4,0,1]+alpha[2,1]*pi[4,1,1,2];
eqnset:={seq(eqn[j],j=0..3)};
solset:={seq(pi[4,j,1,1],j=0..3)};
s:=solve(eqnset,solset);
assign(s);
******j=2,i=0,1,2******
eqn[0]:=pi[4,2,2,1]=rho[1]*pi[4,1,2,1]+(lambda[2]/mu[1])*pi[4,2,1,1];
eqn[1]:=pi[4,1,2,1]=alpha[1,1]*pi[4,2,2,1]+chi[1,1]*pi[4,0,2,1]
+chi[2,1]*pi[4,1,1,1]+alpha[2,1]*pi[4,2,2,2];
eqn[2]:=pi[4,0,2,1]=alpha[1,1]*pi[4,1,2,1]+chi[2,1]*pi[4,0,1,1]+alpha[2,1]*pi[4,1,2,2];
eqnset:={seq(eqn[j],j=0..2)};
solset:={seq(pi[4,j,2,1],j=0..2)};
s:=solve(eqnset,solset);
assign(s);
******j=3,i=0,1******
eqn[0]:=pi[4,1,3,1]=rho[1]*pi[4,0,3,1]+(lambda[2]/mu[1])*pi[4,1,2,1];
eqn[1]:=pi[4,0,3,1]=alpha[1,1]*pi[4,1,3,1]+chi[2,1]*pi[4,0,2,1]+alpha[2,1]*pi[4,1,3,2];
eqnset:={seq(eqn[j],j=0..1)};
solset:={seq(pi[4,j,3,1],j=0..1)};
s:=solve(eqnset,solset);
assign(s);
pi[4,0,4,1]:=(lambda[2]/mu[1])*pi[4,0,3,1];
******All 4,i,j,1 is now solved******
for i from 1 to 3 do
  pi[3,i,0,2]:=(chi[1,2]^(i))*pi[3,1,2];
od;
pi[3,4,0,2]:=(lambda[2]/mu[1])*(chi[1,2]^(3))*pi[3,1,2];
******Simplification B******
for j from 1 to 3 do
```

```
pi[3,0,j,2]:=(chi[2,2]^(j))*pi[3,1,2];
od;
pi[3,0,4,2]:=(rho[2])*(chi[2,2]^(3))*pi[3,1,2];
pi[3,1,1,2]:=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[3,0,1,2]+lambda[2]*pi[3,1,0,2]);
pi[3,1,2,2]:=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[3,0,2,2]+lambda[2]*pi[3,1,1,2]);
pi[3,1,3,2]:=(1/(mu[2]+0))*(lambda[1]*pi[3,0,3,2]+lambda[2]*pi[3,1,2,2]);
pi[3,2,1,2]:=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[3,1,1,2]+lambda[2]*pi[3,2,0,2]);
pi[3,2,2,2]:=(1/(mu[2]+0))*(lambda[1]*pi[3,1,2,2]+lambda[2]*pi[3,2,1,2]);
pi[3,3,1,2]:=(1/(mu[2]+0))*(lambda[1]*pi[3,2,1,2]+lambda[2]*pi[3,3,0,2]);
pi[3,4,0,2]:=(lambda[1]/mu[2])*pi[3,3,0,2];
******Simplification C******
******i*=3******
eq[0]:=pi[1,1,2]=chi[1,2]*pi[0,1,2]+chi[2,2]*pi[1,0,2];
eq[1]:=pi[2,1,2]=chi[1,2]*pi[1,1,2]+chi[2,2]*pi[2,0,2];
eq[2]:=pi[3,1,2]=chi[1,2]*pi[2,1,2]+chi[2,2]*pi[3,0,2];
******Simplification D******
eq[3]:=pi[3,4,0,1]=rho[1]*pi[3,3,0,1];
eq[4]:=pi[3,3,0,1]=alpha[1,1]*pi[3,4,0,1]+chi[1,1]*pi[3,2,0,1]+alpha[2,1]*pi[3,4,0,2];
eq[5]:=pi[3,2,0,1]=alpha[1,1]*pi[3,3,0,1]+chi[1,1]*pi[3,1,0,1]+alpha[2,1]*pi[3,3,0,2];
eq[6]:=pi[3,1,0,1]=alpha[1,1]*pi[3,2,0,1]+chi[1,1]*pi[3,1,1]+alpha[2,1]*pi[3,2,0,2];
eq[7]:=pi[3,1,1]=chi[1,1]*pi[2,1,1]+chi[2,1]*pi[3,0,1]+alpha[1,1]*(pi[3,1,0,1]+pi[4,0,1,1])
      +alpha[2,1]*(pi[3,1,0,2]+pi[4,0,1,2]);
eq[8]:=pi[2,1,1]=chi[1,1]*pi[1,1,1]+chi[2,1]*pi[2,0,1]+alpha[1,1]*pi[3,1,1]+alpha[2,1]*pi[3,1,2];
eq[9]:=pi[1,1,1]=chi[1,1]*pi[0,1,1]+chi[2,1]*pi[1,0,1]+alpha[1,1]*pi[2,1,1]+alpha[2,1]*pi[2,1,2];
eq[10]:=pi[0,1,1]=chi[2,1]*pi[0,0,1]+alpha[1,1]*pi[1,1,1]+alpha[2,1]*pi[1,1,2];
******Result A******
eq[11]:=pi[0,1,2]=-rho[2]*pi[0,0,0]+(mu[1]/mu[2])*pi[0,1,1]+((alpha[2,2])^(-1))*pi[0,0,2];
eqnset:={seq(eq[j],j=0..11)};
solset:={seq(pi[3,j,0,1],j=1..4),seq(pi[i,1,1],i=0..3),seq(pi[i,1,2],i=0..3)};
s:=solve(eqnset,solset);
assign(s);
******j'=1,i'=3******
eq[1]:=pi[3,3,1,1]=rho[1]*pi[3,2,1,1]+(lambda[2]/mu[1])*pi[3,3,0,1];
eq[2]:=pi[3,2,1,1]=chi[1,1]*pi[3,1,1,1]+chi[2,1]*pi[3,2,0,1]+alpha[1,1]*pi[3,3,1,1]
+alpha[1,1]*pi[3,3,1,2];
eq[3]:=pi[3,1,1,1]=chi[1,1]*pi[3,0,1,1]+chi[2,1]*pi[3,1,0,1]+alpha[1,1]*pi[3,2,1,1]
+alpha[1,1]*pi[3,2,1,2];
eq[4]:=pi[3,0,1,1]=chi[2,1]*pi[3,1,1]+alpha[1,1]*(pi[3,1,1,1]+pi[4,0,2,1])
```

```
        +alpha[2,1]*(pi[3,1,1,2]+pi[4,0,2,2]);
    eqnset:={seq(eq[j],j=1..4)};
    solset:={seq(pi[3,j,1,1],j=0..3)};
    s:=solve(eqnset,solset);
    assign(s);
    eq[1]:=pi[3,2,2,1]=rho[1]*pi[3,1,2,1]+(lambda[2]/mu[1])*pi[3,2,1,1];
    eq[2]:=pi[3,1,2,1]=chi[1,1]*pi[3,0,2,1]+chi[2,1]*pi[3,1,1,1]+alpha[1,1]*pi[3,2,2,1]
+alpha[1,1]*pi[3,2,2,2];
    eq[3]:=pi[3,0,2,1]=chi[2,1]*pi[3,0,1,1]+alpha[1,1]*(pi[3,1,2,1]+pi[4,0,3,1])
+alpha[2,1]*(pi[3,1,2,2]+pi[4,0,3,2]);
    eqnset:={seq(eq[j],j=1..3)};
    solset:={seq(pi[3,j,2,1],j=0..2)};
    s:=solve(eqnset,solset);
    assign(s);
    eq[1]:=pi[3,1,3,1]=rho[1]*pi[3,0,3,1]+(lambda[2]/mu[1])*pi[3,1,2,1];
    eq[2]:=pi[3,0,3,1]=chi[2,1]*pi[3,0,2,1]+alpha[1,1]*(pi[3,1,3,1]+pi[4,0,4,1])
+alpha[2,1]*(pi[3,1,3,2]+pi[4,0,4,2]);
    eqnset:={seq(eq[j],j=1..2)};
    solset:={seq(pi[3,j,3,1],j=0..1)};
    s:=solve(eqnset,solset);
    assign(s);
    pi[3,0,4,1]:=(lambda[2]/mu[1])*pi[3,0,3,1];
    ******All of i*=3 is solved...go back to alpha large looping a stage******
    for i from 1 to 3 do
      pi[2,i,0,2]:=(chi[1,2]^(i))*pi[2,2,2];
    od;
    pi[2,4,0,2]:=(lambda[2]/mu[1])*(chi[1,2]^(3))*pi[2,2,2];
    ******Simplification B******
    for j from 1 to 3 do
      pi[2,0,j,2]:=(chi[2,2]^(j))*pi[2,2,2];
    od;
    pi[2,0,4,2]:=(rho[2])*(chi[2,2]^(3))*pi[2,2,2];
    pi[2,1,1,2]:=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[2,0,1,2]+lambda[2]*pi[2,1,0,2]);
    pi[2,1,2,2]:=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[2,0,2,2]+lambda[2]*pi[2,1,1,2]);
    pi[2,1,3,2]:=(1/(mu[2]+0))*(lambda[1]*pi[2,0,3,2]+lambda[2]*pi[2,1,2,2]);
    pi[2,2,1,2]:=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[2,1,1,2]+lambda[2]*pi[2,2,0,2]);
    pi[2,2,2,2]:=(1/(mu[2]+0))*(lambda[1]*pi[2,1,2,2]+lambda[2]*pi[2,2,1,2]);
    pi[2,3,1,2]:=(1/(mu[2]+0))*(lambda[1]*pi[2,2,1,2]+lambda[2]*pi[2,3,0,2]);
```

```
pi[2,4,0,2]:=(lambda[1]/mu[2])*pi[2,3,0,2];
******Simplification C******
****** i*=2 ******
eq[0]:=pi[1,2,2]=chi[1,2]*pi[0,2,2]+chi[2,2]*pi[1,1,2];
eq[1]:=pi[2,2,2]=chi[1,2]*pi[1,2,2]+chi[2,2]*pi[2,1,2];
******Simplification D******
eq[2]:=pi[2,4,0,1]=rho[1]*pi[2,3,0,1];
eq[3]:=pi[2,3,0,1]=alpha[1,1]*pi[2,4,0,1]+chi[1,1]*pi[2,2,0,1]+alpha[2,1]*pi[2,4,0,2];
eq[4]:=pi[2,2,0,1]=alpha[1,1]*pi[2,3,0,1]+chi[1,1]*pi[2,1,0,1]+alpha[2,1]*pi[2,3,0,2];
eq[5]:=pi[2,1,0,1]=alpha[1,1]*pi[2,2,0,1]+chi[1,1]*pi[2,2,1]+alpha[2,1]*pi[2,2,0,2];
******Solving Stage******
eq[6]:=pi[2,2,1]=chi[1,1]*pi[1,2,1]+chi[2,1]*pi[2,0,1]+alpha[1,1]*(pi[2,1,0,1]+pi[3,0,1,1])
    +alpha[2,1]*(pi[2,1,0,2]+pi[3,0,1,2]);
eq[7]:=pi[1,2,1]=chi[1,1]*pi[0,2,1]+chi[2,1]*pi[1,1,1]+alpha[1,1]*pi[2,2,1]+alpha[2,1]*pi[2,1,2];
eq[8]:=pi[0,2,1]=chi[2,1]*pi[0,1,1]+alpha[1,1]*pi[1,2,1]+alpha[2,1]*pi[1,2,2];
eq[9]:=pi[0,2,2]=-rho[2]*pi[0,0,2]+(mu[1]/mu[2])*pi[0,2,1]+((alpha[2,2])^(-1))*pi[0,1,2];
eqnset:={seq(eq[j],j=0..9)};
solset:={seq(pi[2,j,0,1],j=1..4),seq(pi[i,2,1],i=0..2),seq(pi[i,2,2],i=0..2)};
s:=solve(eqnset,solset);
assign(s);
******j'=1,i'=3******
eq[1]:=pi[2,3,1,1]=rho[1]*pi[2,2,1,1]+(lambda[2]/mu[1])*pi[2,3,0,1];
eq[2]:=pi[2,2,1,1]=chi[1,1]*pi[2,1,1,1]+chi[2,1]*pi[2,2,0,1]+alpha[1,1]*pi[2,3,1,1]
+alpha[1,1]*pi[2,3,1,2];
eq[3]:=pi[2,1,1,1]=chi[1,1]*pi[2,0,1,1]+chi[2,1]*pi[2,1,0,1]+alpha[1,1]*pi[2,2,1,1]
+alpha[1,1]*pi[2,2,1,2];
eq[4]:=pi[2,0,1,1]=chi[2,1]*pi[2,2,1]+alpha[1,1]*(pi[2,1,1,1]+pi[3,0,2,1])
+alpha[2,1]*(pi[2,1,1,2]+pi[3,0,2,2]);
eqnset:={seq(eq[j],j=1..4)};
solset:={seq(pi[2,j,1,1],j=0..3)};
s:=solve(eqnset,solset);
assign(s);
eq[1]:=pi[2,2,2,1]=rho[1]*pi[2,1,2,1]+(lambda[2]/mu[1])*pi[2,2,1,1];
eq[2]:=pi[2,1,2,1]=chi[1,1]*pi[2,0,2,1]+chi[2,1]*pi[2,1,1,1]+alpha[1,1]*pi[2,2,2,1]
+alpha[1,1]*pi[2,2,2,2];
eq[3]:=pi[2,0,2,1]=chi[2,1]*pi[2,0,1,1]+alpha[1,1]*(pi[2,1,2,1]+pi[3,0,3,1])
+alpha[2,1]*(pi[2,1,2,2]+pi[3,0,3,2]);
eqnset:={seq(eq[j],j=1..3)};
```

```
solset:={seq(pi[2,j,2,1],j=0..2)};
s:=solve(eqnset,solset);
assign(s);
eq[1]:=pi[2,1,3,1]=rho[1]*pi[2,0,3,1]+(lambda[2]/mu[1])*pi[2,1,2,1];
eq[2]:=pi[2,0,3,1]=chi[2,1]*pi[2,0,2,1]+alpha[1,1]*(pi[2,1,3,1]+pi[3,0,4,1])
+alpha[2,1]*(pi[2,1,3,2]+pi[3,0,4,2]);
eqnset:={seq(eq[j],j=1..2)};
solset:={seq(pi[2,j,3,1],j=0..1)};
s:=solve(eqnset,solset);
assign(s);
pi[2,0,4,1]:=(lambda[2]/mu[1])*pi[2,0,3,1];
******Simplification Stage alpha0,alpha1******
for i from 1 to 3 do
  pi[0,i,0,1]:=(chi[1,1]^(i))*pi[0,4,1];
od;
pi[0,4,0,1]:=(rho[1])*(chi[1,1]^(3))*pi[0,4,1];
******j'=1,i'=3,...,1******
eq[0]:=pi[0,3,1,1]=rho[1]*pi[0,2,1,1]+(lambda[2]/mu[1])*pi[0,3,0,1];
eq[1]:=pi[0,2,1,1]=chi[1,1]*pi[0,1,1,1]+chi[2,1]*pi[0,2,0,1];
eq[2]:=pi[0,1,1,1]=chi[1,1]*pi[0,0,1,1]+chi[2,1]*pi[0,1,0,1];
eqnset:={seq(eq[j],j=0..2)};
solset:={seq(pi[0,j,1,1],j=1..3)};
s:=solve(eqnset,solset);
assign(s);
******j'=2,i'=2,...,1******
eq[0]:=pi[0,2,2,1]=rho[1]*pi[0,1,2,1]+(lambda[2]/mu[1])*pi[0,2,1,1];
eq[1]:=pi[0,1,2,1]=chi[1,1]*pi[0,0,2,1]+chi[2,1]*pi[0,1,1,1];
eqnset:={seq(eq[j],j=0..1)};
solset:={seq(pi[0,j,2,1],j=1..2)};
s:=solve(eqnset,solset);
assign(s);
pi[0,1,3,1]:=chi[1,1]*pi[0,0,3,1]+chi[2,1]*pi[0,1,2,1];
for i from 1 to 3 do
  pi[0,i,0,2]:=(chi[1,2]^(i))*pi[0,4,2];
od;
pi[0,4,0,2]:=(lambda[1]/mu[2])*(chi[1,2]^(3))*pi[0,4,2];
******j'=1,i'=1,...,3******
eq[0]:=pi[0,1,1,2]=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[0,0,1,2]+lambda[2]*pi[0,1,0,2]);
```

```
eq[1]:=pi[0,2,1,2]=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[0,1,1,2]+lambda[2]*pi[0,2,0,2]);
eq[2]:=pi[0,3,1,2]=(1/(mu[2]))*(lambda[1]*pi[0,2,1,2]+lambda[2]*pi[0,3,0,2]);
eqnset:={seq(eq[j],j=0..2)};
solset:={seq(pi[0,j,1,2],j=1..3)};
s:=solve(eqnset,solset);
assign(s);
******j'=2,i'=1..2******
eq[0]:=pi[0,1,2,2]=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[0,0,2,2]+lambda[2]*pi[0,1,1,2]);
eq[1]:=pi[0,2,2,2]=(1/(mu[2]))*(lambda[1]*pi[0,1,2,2]+lambda[2]*pi[0,2,1,2]);
eqnset:={seq(eq[j],j=0..1)};
solset:={seq(pi[0,j,2,2],j=1..2)};
s:=solve(eqnset,solset);
assign(s);
pi[0,1,3,2]:=(1/(mu[2]))*(lambda[1]*pi[0,0,3,2]+lambda[2]*pi[0,1,2,2]);
******Simplification F******
eq[1]:=pi[1,4,0,2]=rho[2]*pi[1,3,0,2];
eq[2]:=pi[1,3,0,2]=chi[1,2]*pi[1,2,0,2]+alpha[2,2]*pi[0,4,0,2]+alpha[1,2]*pi[0,4,0,1];
eq[3]:=pi[1,2,0,2]=chi[1,2]*pi[1,1,0,2]+alpha[2,2]*pi[0,3,0,2]+alpha[1,2]*pi[0,3,0,1];
eq[4]:=pi[1,1,0,2]=chi[1,2]*pi[1,3,2]+alpha[2,2]*pi[0,2,0,2]+alpha[1,2]*pi[0,2,0,1];
eqnset:={seq(eq[j],j=1..4)};
solset:={seq(pi[1,j,0,2],j=1..4)};
s:=solve(eqnset,solset);
assign(s);
eq[1]:=pi[1,4,0,1]=rho[1]*pi[1,3,0,1];
eq[2]:=pi[1,3,0,1]=chi[1,1]*pi[1,2,0,1]+alpha[2,1]*pi[1,4,0,2]+alpha[1,1]*pi[1,4,0,1];
eq[3]:=pi[1,2,0,1]=chi[1,1]*pi[1,1,0,1]+alpha[2,1]*pi[1,3,0,2]+alpha[1,1]*pi[1,3,0,1];
eq[4]:=pi[1,1,0,1]=chi[1,1]*pi[1,3,1]+alpha[2,1]*pi[1,2,0,2]+alpha[1,1]*pi[1,2,0,1];
eqnset:={seq(eq[j],j=1..4)};
solset:={seq(pi[1,j,0,1],j=1..4)};
s:=solve(eqnset,solset);
assign(s);
******SImplfication G******
******j'=1,i'=1,2,3******
eq[1]:=pi[1,1,1,2]=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[1,0,1,2]+lambda[2]*pi[1,1,0,2])
+alpha[2,2]*pi[0,2,1,2]+alpha[1,2]*pi[0,2,1,1];
eq[2]:=pi[1,2,1,2]=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[1,1,1,2]+lambda[2]*pi[1,2,0,2])
+alpha[2,2]*pi[0,3,1,2]+alpha[1,2]*pi[0,3,1,1];
eq[3]:=pi[1,3,1,2]=(1/(mu[2]))*(lambda[1]*pi[1,2,1,2]+lambda[2]*pi[1,3,0,2]);
```

```
eqnset:={seq(eq[j],j=1..3)};
solset:={seq(pi[1,j,1,2],j=1..3)};
s:=solve(eqnset,solset);
assign(s);
******j'=2,i'=1,2******
eq[4]:=pi[1,1,2,2]=(1/(mu[2]+lambda[0]))*(lambda[1]*pi[1,0,2,2]+lambda[2]*pi[1,1,1,2])
+alpha[2,2]*pi[0,2,2,2]+alpha[1,2]*pi[0,2,2,1];
eq[5]:=pi[1,2,2,2]=(1/(mu[2]))*(lambda[1]*pi[1,1,2,2]+lambda[2]*pi[1,2,1,2]);
eqnset:={seq(eq[j],j=4..5)};
solset:={seq(pi[1,j,2,2],j=1..2)};
s:=solve(eqnset,solset);
assign(s);
******j'=3,i'=1******
pi[1,1,3,2]:=(1/(mu[2]))*(lambda[1]*pi[1,0,3,2]+lambda[2]*pi[1,1,2,2]);
******j'=1,i'=1,2,3******
eq[7]:=pi[1,1,1,1]=(1/(mu[1]+lambda[0]))*(lambda[1]*pi[1,0,1,1]+lambda[2]*pi[1,1,0,1])
+alpha[2,1]*pi[0,2,1,2]+alpha[1,1]*pi[0,2,1,1];
eq[8]:=pi[1,2,1,1]=(1/(mu[1]+lambda[0]))*(lambda[1]*pi[1,1,1,1]+lambda[2]*pi[1,2,0,1])
+alpha[2,1]*pi[0,3,1,2]+alpha[1,1]*pi[0,3,1,1];
eq[9]:=pi[1,3,1,1]=(1/(mu[1]))*(lambda[1]*pi[1,2,1,1]+lambda[2]*pi[1,3,0,1]);
eqnset:={seq(eq[j],j=7..9)};
solset:={seq(pi[1,j,1,1],j=1..3)};
s:=solve(eqnset,solset);
assign(s);
******j'=2,i'=1,2******
eq[10]:=pi[1,1,2,1]=(1/(mu[1]+lambda[0]))*(lambda[1]*pi[1,0,2,1]+lambda[2]*pi[1,1,1,1])
+alpha[2,1]*pi[0,2,2,2]+alpha[1,1]*pi[0,2,2,1];
eq[11]:=pi[1,2,2,1]=(1/(mu[1]))*(lambda[1]*pi[1,1,2,1]+lambda[2]*pi[1,2,1,1]);
eqnset:={seq(eq[j],j=10..11)};
solset:={seq(pi[1,j,2,1],j=1..2)};
s:=solve(eqnset,solset);
assign(s);
******j'=3,i'=1******
pi[1,1,3,1]:=(1/(mu[1]))*(lambda[1]*pi[1,0,3,1]+lambda[2]*pi[1,1,2,1]);
pi[0,3,1]:=chi[2,1]*pi[0,2,1]+alpha[1,1]*pi[1,3,1]+alpha[2,1]*pi[1,3,2];
pi[0,3,2]:=chi[2,2]*pi[0,2,2]+alpha[1,2]*pi[0,4,1]+alpha[2,1]*pi[0,4,2];
eq[15]:=pi[1,3,1]=chi[1,1]*pi[0,3,1]+chi[2,1]*pi[1,2,1]+alpha[1,1]*(pi[1,1,0,1]+pi[2,0,1,1])
+alpha[2,2]*(pi[1,1,0,2]+pi[2,0,1,2]);
```

```
solve(%,pi[1,3,1]);
pi[1,3,1]:=%;
pi[1,3,2]:=chi[2,2]*pi[1,2,2]+chi[1,2]*pi[0,3,2]+alpha[1,2]*pi[0,1,0,1]+alpha[2,2]*pi[0,1,0,2];
eq[17]:=pi[0,2,2]=chi[2,2]*pi[0,1,2]+alpha[2,2]*pi[0,3,2]+alpha[1,2]*pi[0,3,1];
solve(%,pi[0,4,2]);
pi[0,4,2]:=%;
pi[1,3,1];
pi[1,3,2];
eq[18]:=pi[0,4,1]=chi[2,1]*pi[0,3,1]+alpha[1,1]*pi[1,0,1,1]+alpha[2,1]*pi[1,0,1,2];
pi[0,4,1]:=solve(%,pi[0,4,1]);
eq[19]:=pi[0,4,2]=chi[2,2]*pi[0,3,2]+alpha[2,2]*pi[0,0,1,2]+alpha[1,2]*pi[0,0,1,1];
eq[20]:=pi[1,0,1,2]=alpha[1,2]*pi[0,1,1,1]+alpha[2,2]*pi[0,1,1,2]+chi[2,2]*pi[1,3,2];
eq[21]:=pi[1,0,1,1]=chi[2,1]*pi[1,3,1]+alpha[1,1]*(pi[1,1,1,1]+pi[2,0,2,1])
+alpha[2,1]*(pi[1,1,1,2]+pi[2,0,2,2]);
eq[22]:=pi[0,0,1,1]=chi[2,1]*pi[0,4,1]+alpha[1,1]*pi[1,0,2,1]+alpha[2,1]*pi[1,0,2,2];
eq[23]:=pi[0,0,1,2]=alpha[1,2]*pi[0,0,2,1]+alpha[2,2]*pi[0,0,2,2]+chi[2,2]*pi[0,4,2];
eqnset:={seq(eq[j],j=19..23)};
solset:={pi[0,0,1,2],pi[0,0,2,1],pi[0,0,2,2],pi[1,0,1,1],pi[1,0,1,2],pi[0,0,1,1]};
s:=solve(eqnset,solset);
assign(s);
eq[24]:=pi[0,0,2,2]=chi[2,2]*pi[0,0,1,2]+alpha[2,2]*pi[0,0,3,2]+alpha[1,2]*pi[0,0,3,1];
eq[25]:=pi[0,0,3,2]=chi[2,2]*pi[0,0,2,2]+alpha[2,2]*pi[0,0,4,2]+alpha[1,2]*pi[0,0,4,1];
eq[26]:=pi[0,0,4,2]=rho[2]*pi[0,0,3,2];
eqnset:={seq(eq[j],j=24..26)};
solset:={seq(pi[0,0,j,2],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
eq[27]:=pi[1,0,2,1]=chi[2,1]*pi[1,0,1,1]+alpha[1,1]*(pi[1,1,2,1]+pi[2,0,3,1])
+alpha[2,1]*(pi[1,1,2,2]+pi[2,0,3,2]);
eq[28]:=pi[1,0,3,1]=chi[2,1]*pi[1,0,2,1]+alpha[1,1]*(pi[1,1,3,1]+pi[2,0,4,1])
+alpha[2,1]*(pi[1,1,3,2]+pi[2,0,4,2]);
eq[29]:=pi[1,0,4,1]=(lambda[2]/mu[1])*pi[1,0,3,1];
eqnset:={seq(eq[j],j=27..29)};
solset:={seq(pi[1,0,j,1],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
eq[30]:=pi[1,0,2,2]=chi[2,2]*pi[1,0,1,2]+alpha[1,2]*pi[0,1,2,1]+alpha[2,2]*pi[0,1,2,2];
eq[31]:=pi[1,0,3,2]=chi[2,2]*pi[1,0,2,2]+alpha[1,2]*pi[0,1,3,1]+alpha[2,2]*pi[0,1,3,2];
```

```
eq[32]:=pi[1,0,4,2]=rho[2]*pi[1,0,3,2];
eqnset:={seq(eq[j],j=30..32)};
solset:={seq(pi[1,0,j,2],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
eq[33]:=pi[0,0,2,1]=chi[2,1]*pi[0,0,1,1]+alpha[1,1]*pi[1,0,3,1]+alpha[2,1]*pi[1,0,3,2];
eq[34]:=pi[0,0,3,1]=chi[2,1]*pi[0,0,2,1]+alpha[1,1]*pi[1,0,4,1]+alpha[2,1]*pi[1,0,4,2];
eq[35]:=pi[0,0,4,1]=(lambda[2]/mu[1])*pi[0,0,3,1];
eqnset:={seq(eq[j],j=33..35)};
solset:={seq(pi[0,0,j,1],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
sq[01]:=pi[0,0,0]+pi[0,0,1]+pi[0,1,1]+pi[0,2,1]+pi[0,3,1]+pi[0,4,1]+pi[1,0,1]+pi[1,1,1]
+pi[1,2,1]+pi[1,3,1]+pi[2,0,1]+pi[2,1,1]+pi[2,2,1]+pi[3,0,1]+pi[3,1,1]+pi[4,0,1];
sq[02]:=pi[0,0,2]+pi[0,1,2]+pi[0,2,2]+pi[0,3,2]+pi[0,4,2]+pi[1,0,2]+pi[1,1,2]+pi[1,2,2]
pi[1,3,2]+pi[2,0,2]+pi[2,1,2]+pi[2,2,2]+pi[3,0,2]+pi[3,1,2]+pi[4,0,2];
dq[0]:=pi[0,0,1,1]+pi[0,0,2,1]+pi[0,0,3,1]+pi[0,0,4,1]+pi[0,1,0,1]+pi[0,1,1,1]+pi[0,1,2,1]
+pi[0,1,3,1]+pi[0,2,0,1]+pi[0,2,1,1]+pi[0,2,2,1]+pi[0,3,0,1]+pi[0,3,1,1]+pi[0,4,0,1];
dq[1]:=pi[1,0,1,1]+pi[1,0,2,1]+pi[1,0,3,1]+pi[1,0,4,1]+pi[1,1,0,1]+pi[1,1,1,1]+pi[1,1,2,1]
+pi[1,1,3,1]+pi[1,2,0,1]+pi[1,2,1,1]+pi[1,2,2,1]+pi[1,3,0,1]+pi[1,3,1,1]+pi[1,4,0,1];
dq[2]:=pi[2,0,1,1]+pi[2,0,2,1]+pi[2,0,3,1]+pi[2,0,4,1]+pi[2,1,0,1]+pi[2,1,1,1]+pi[2,1,2,1]
+pi[2,1,3,1]+pi[2,2,0,1]+pi[2,2,1,1]+pi[2,2,2,1]+pi[2,3,0,1]+pi[2,3,1,1]+pi[2,4,0,1];
dq[3]:=pi[3,0,1,1]+pi[3,0,2,1]+pi[3,0,3,1]+pi[3,0,4,1]+pi[3,1,0,1]+pi[3,1,1,1]+pi[3,1,2,1]
+pi[3,1,3,1]+pi[3,2,0,1]+pi[3,2,1,1]+pi[3,2,2,1]+pi[3,3,0,1]+pi[3,3,1,1]+pi[3,4,0,1];
dq[4]:=pi[4,0,1,1]+pi[4,0,2,1]+pi[4,0,3,1]+pi[4,0,4,1]+pi[4,1,0,1]+pi[4,1,1,1]+pi[4,1,2,1]
+pi[4,1,3,1]+pi[4,2,0,1]+pi[4,2,1,1]+pi[4,2,2,1]+pi[4,3,0,1]+pi[4,3,1,1]+pi[4,4,0,1];
dq[20]:=pi[0,0,1,2]+pi[0,0,2,2]+pi[0,0,3,2]+pi[0,0,4,2]+pi[0,1,0,2]+pi[0,1,1,2]+pi[0,1,2,2]
+pi[0,1,3,2]+pi[0,2,0,2]+pi[0,2,1,2]+pi[0,2,2,2]+pi[0,3,0,2]+pi[0,3,1,2]+pi[0,4,0,2];
dq[21]:=pi[1,0,1,2]+pi[1,0,2,2]+pi[1,0,3,2]+pi[1,0,4,2]+pi[1,1,0,2]+pi[1,1,1,2]+pi[1,1,2,2]
+pi[1,1,3,2]+pi[1,2,0,2]+pi[1,2,1,2]+pi[1,2,2,2]+pi[1,3,0,2]+pi[1,3,1,2]+pi[1,4,0,2];
dq[22]:=pi[2,0,1,2]+pi[2,0,2,2]+pi[2,0,3,2]+pi[2,0,4,2]+pi[2,1,0,2]+pi[2,1,1,2]+pi[2,1,2,2]
+pi[2,1,3,2]+pi[2,2,0,2]+pi[2,2,1,2]+pi[2,2,2,2]+pi[2,3,0,2]+pi[2,3,1,2]+pi[2,4,0,2];
dq[23]:=pi[3,0,1,2]+pi[3,0,2,2]+pi[3,0,3,2]+pi[3,0,4,2]+pi[3,1,0,2]+pi[3,1,1,2]+pi[3,1,2,2]
+pi[3,1,3,2]+pi[3,2,0,2]+pi[3,2,1,2]+pi[3,2,2,2]+pi[3,3,0,2]+pi[3,3,1,2]+pi[3,4,0,2];
dq[24]:=pi[4,0,1,2]+pi[4,0,2,2]+pi[4,0,3,2]+pi[4,0,4,2]+pi[4,1,0,2]+pi[4,1,1,2]+pi[4,1,2,2]
+pi[4,1,3,2]+pi[4,2,0,2]+pi[4,2,1,2]+pi[4,2,2,2]+pi[4,3,0,2]+pi[4,3,1,2]+pi[4,4,0,2];
all:=sq[01]+sq[02]+dq[0]+dq[1]+dq[2]+dq[3]+dq[4]+dq[20]+dq[21]+dq[22]+dq[23]+dq[24];
c:=solve(all*k=1,k);
```

```
c*all;
pi[0,0,0]:=evalf(pi[0,0,0]*c);
pi[0,0,1]:=evalf(pi[0,0,1]*c);
pi[0,1,1]:=evalf(pi[0,1,1]*c);
pi[0,2,1]:=evalf(pi[0,2,1]*c);
pi[0,3,1]:=evalf(pi[0,3,1]*c);
pi[0,4,1]:=evalf(pi[0,4,1]*c);
pi[1,0,1]:=evalf(pi[1,0,1]*c);
pi[1,1,1]:=evalf(pi[1,1,1]*c);
pi[1,2,1]:=evalf(pi[1,2,1]*c);
pi[1,3,1]:=evalf(pi[1,3,1]*c);
pi[2,0,1]:=evalf(pi[2,0,1]*c);
pi[2,1,1]:=evalf(pi[2,1,1]*c);
pi[2,2,1]:=evalf(pi[2,2,1]*c);
pi[3,0,1]:=evalf(pi[3,0,1]*c);
pi[3,1,1]:=evalf(pi[3,1,1]*c);
pi[4,0,1]:=evalf(pi[4,0,1]*c);
pi[0,0,2]:=evalf(pi[0,0,2]*c);
pi[0,1,2]:=evalf(pi[0,1,2]*c);
pi[0,2,2]:=evalf(pi[0,2,2]*c);
pi[0,3,2]:=evalf(pi[0,3,2]*c);
pi[0,4,2]:=evalf(pi[0,4,2]*c);
pi[1,0,2]:=evalf(pi[1,0,2]*c);
pi[1,1,2]:=evalf(pi[1,1,2]*c);
pi[1,2,2]:=evalf(pi[1,2,2]*c);
pi[1,3,2]:=evalf(pi[1,3,2]*c);
pi[2,0,2]:=evalf(pi[2,0,2]*c);
pi[2,1,2]:=evalf(pi[2,1,2]*c);
pi[2,2,2]:=evalf(pi[2,2,2]*c);
pi[3,0,2]:=evalf(pi[3,0,2]*c);
pi[3,1,2]:=evalf(pi[3,1,2]*c);
pi[4,0,2]:=evalf(pi[4,0,2]*c);
pi[0,0,1,1]:=evalf(pi[0,0,1,1]*c);
pi[0,0,2,1]:=evalf(pi[0,0,2,1]*c);
pi[0,0,3,1]:=evalf(pi[0,0,3,1]*c);
pi[0,0,4,1]:=evalf(pi[0,0,4,1]*c);
pi[0,1,0,1]:=evalf(pi[0,1,0,1]*c);
pi[0,1,1,1]:=evalf(pi[0,1,1,1]*c);
```

```
pi[0,1,2,1]:=evalf(pi[0,1,2,1]*c);
pi[0,1,3,1]:=evalf(pi[0,1,3,1]*c);
pi[0,2,0,1]:=evalf(pi[0,2,0,1]*c);
pi[0,2,1,1]:=evalf(pi[0,2,1,1]*c);
pi[0,2,2,1]:=evalf(pi[0,2,2,1]*c);
pi[0,3,0,1]:=evalf(pi[0,3,0,1]*c);
pi[0,3,1,1]:=evalf(pi[0,3,1,1]*c);
pi[0,4,0,1]:=evalf(pi[0,4,0,1]*c);
pi[1,0,1,1]:=evalf(pi[1,0,1,1]*c);
pi[1,0,2,1]:=evalf(pi[1,0,2,1]*c);
pi[1,0,3,1]:=evalf(pi[1,0,3,1]*c);
pi[1,0,4,1]:=evalf(pi[1,0,4,1]*c);
pi[1,1,0,1]:=evalf(pi[1,1,0,1]*c);
pi[1,1,1,1]:=evalf(pi[1,1,1,1]*c);
pi[1,1,2,1]:=evalf(pi[1,1,2,1]*c);
pi[1,1,3,1]:=evalf(pi[1,1,3,1]*c);
pi[1,2,0,1]:=evalf(pi[1,2,0,1]*c);
pi[1,2,1,1]:=evalf(pi[1,2,1,1]*c);
pi[1,2,2,1]:=evalf(pi[1,2,2,1]*c);
pi[1,3,0,1]:=evalf(pi[1,3,0,1]*c);
pi[1,3,1,1]:=evalf(pi[1,3,1,1]*c);
pi[1,4,0,1]:=evalf(pi[1,4,0,1]*c);
pi[2,0,1,1]:=evalf(pi[2,0,1,1]*c);
pi[2,0,2,1]:=evalf(pi[2,0,2,1]*c);
pi[2,0,3,1]:=evalf(pi[2,0,3,1]*c);
pi[2,0,4,1]:=evalf(pi[2,0,4,1]*c);
pi[2,1,0,1]:=evalf(pi[2,1,0,1]*c);
pi[2,1,1,1]:=evalf(pi[2,1,1,1]*c);
pi[2,1,2,1]:=evalf(pi[2,1,2,1]*c);
pi[2,1,3,1]:=evalf(pi[2,1,3,1]*c);
pi[2,2,0,1]:=evalf(pi[2,2,0,1]*c);
pi[2,2,1,1]:=evalf(pi[2,2,1,1]*c);
pi[2,2,2,1]:=evalf(pi[2,2,2,1]*c);
pi[2,3,0,1]:=evalf(pi[2,3,0,1]*c);
pi[2,3,1,1]:=evalf(pi[2,3,1,1]*c);
pi[2,4,0,1]:=evalf(pi[2,4,0,1]*c);
pi[3,0,1,1]:=evalf(pi[3,0,1,1]*c);
pi[3,0,2,1]:=evalf(pi[3,0,2,1]*c);
```

```
pi[3,0,3,1]:=evalf(pi[3,0,3,1]*c);
pi[3,0,4,1]:=evalf(pi[3,0,4,1]*c);
pi[3,1,0,1]:=evalf(pi[3,1,0,1]*c);
pi[3,1,1,1]:=evalf(pi[3,1,1,1]*c);
pi[3,1,2,1]:=evalf(pi[3,1,2,1]*c);
pi[3,1,3,1]:=evalf(pi[3,1,3,1]*c);
pi[3,2,0,1]:=evalf(pi[3,2,0,1]*c);
pi[3,2,1,1]:=evalf(pi[3,2,1,1]*c);
pi[3,2,2,1]:=evalf(pi[3,2,2,1]*c);
pi[3,3,0,1]:=evalf(pi[3,3,0,1]*c);
pi[3,3,1,1]:=evalf(pi[3,3,1,1]*c);
pi[3,4,0,1]:=evalf(pi[3,4,0,1]*c);
pi[4,0,1,1]:=evalf(pi[4,0,1,1]*c);
pi[4,0,2,1]:=evalf(pi[4,0,2,1]*c);
pi[4,0,3,1]:=evalf(pi[4,0,3,1]*c);
pi[4,0,4,1]:=evalf(pi[4,0,4,1]*c);
pi[4,1,0,1]:=evalf(pi[4,1,0,1]*c);
pi[4,1,1,1]:=evalf(pi[4,1,1,1]*c);
pi[4,1,2,1]:=evalf(pi[4,1,2,1]*c);
pi[4,1,3,1]:=evalf(pi[4,1,3,1]*c);
pi[4,2,0,1]:=evalf(pi[4,2,0,1]*c);
pi[4,2,1,1]:=evalf(pi[4,2,1,1]*c);
pi[4,2,2,1]:=evalf(pi[4,2,2,1]*c);
pi[4,3,0,1]:=evalf(pi[4,3,0,1]*c);
pi[4,3,1,1]:=evalf(pi[4,3,1,1]*c);
pi[4,4,0,1]:=evalf(pi[4,4,0,1]*c);
pi[0,0,1,2]:=evalf(pi[0,0,1,2]*c);
pi[0,0,2,2]:=evalf(pi[0,0,2,2]*c);
pi[0,0,3,2]:=evalf(pi[0,0,3,2]*c);
pi[0,0,4,2]:=evalf(pi[0,0,4,2]*c);
pi[0,1,0,2]:=evalf(pi[0,1,0,2]*c);
pi[0,1,1,2]:=evalf(pi[0,1,1,2]*c);
pi[0,1,2,2]:=evalf(pi[0,1,2,2]*c);
pi[0,1,3,2]:=evalf(pi[0,1,3,2]*c);
pi[0,2,0,2]:=evalf(pi[0,2,0,2]*c);
pi[0,2,1,2]:=evalf(pi[0,2,1,2]*c);
pi[0,2,2,2]:=evalf(pi[0,2,2,2]*c);
pi[0,3,0,2]:=evalf(pi[0,3,0,2]*c);
```

```
pi[0,3,1,2]:=evalf(pi[0,3,1,2]*c);
pi[0,4,0,2]:=evalf(pi[0,4,0,2]*c);
pi[1,0,1,2]:=evalf(pi[1,0,1,2]*c);
pi[1,0,2,2]:=evalf(pi[1,0,2,2]*c);
pi[1,0,3,2]:=evalf(pi[1,0,3,2]*c);
pi[1,0,4,2]:=evalf(pi[1,0,4,2]*c);
pi[1,1,0,2]:=evalf(pi[1,1,0,2]*c);
pi[1,1,1,2]:=evalf(pi[1,1,1,2]*c);
pi[1,1,2,2]:=evalf(pi[1,1,2,2]*c);
pi[1,1,3,2]:=evalf(pi[1,1,3,2]*c);
pi[1,2,0,2]:=evalf(pi[1,2,0,2]*c);
pi[1,2,1,2]:=evalf(pi[1,2,1,2]*c);
pi[1,2,2,2]:=evalf(pi[1,2,2,2]*c);
pi[1,3,0,2]:=evalf(pi[1,3,0,2]*c);
pi[1,3,1,2]:=evalf(pi[1,3,1,2]*c);
pi[1,4,0,2]:=evalf(pi[1,4,0,2]*c);
pi[2,0,1,2]:=evalf(pi[2,0,1,2]*c);
pi[2,0,2,2]:=evalf(pi[2,0,2,2]*c);
pi[2,0,3,2]:=evalf(pi[2,0,3,2]*c);
pi[2,0,4,2]:=evalf(pi[2,0,4,2]*c);
pi[2,1,0,2]:=evalf(pi[2,1,0,2]*c);
pi[2,1,1,2]:=evalf(pi[2,1,1,2]*c);
pi[2,1,2,2]:=evalf(pi[2,1,2,2]*c);
pi[2,1,3,2]:=evalf(pi[2,1,3,2]*c);
pi[2,2,0,2]:=evalf(pi[2,2,0,2]*c);
pi[2,2,1,2]:=evalf(pi[2,2,1,2]*c);
pi[2,2,2,2]:=evalf(pi[2,2,2,2]*c);
pi[2,3,0,2]:=evalf(pi[2,3,0,2]*c);
pi[2,3,1,2]:=evalf(pi[2,3,1,2]*c);
pi[2,4,0,2]:=evalf(pi[2,4,0,2]*c);
pi[3,0,1,2]:=evalf(pi[3,0,1,2]*c);
pi[3,0,2,2]:=evalf(pi[3,0,2,2]*c);
pi[3,0,3,2]:=evalf(pi[3,0,3,2]*c);
pi[3,0,4,2]:=evalf(pi[3,0,4,2]*c);
pi[3,1,0,2]:=evalf(pi[3,1,0,2]*c);
pi[3,1,1,2]:=evalf(pi[3,1,1,2]*c);
pi[3,1,2,2]:=evalf(pi[3,1,2,2]*c);
pi[3,1,3,2]:=evalf(pi[3,1,3,2]*c);
```

```
pi[3,2,0,2]:=evalf(pi[3,2,0,2]*c);
pi[3,2,1,2]:=evalf(pi[3,2,1,2]*c);
pi[3,2,2,2]:=evalf(pi[3,2,2,2]*c);
pi[3,3,0,2]:=evalf(pi[3,3,0,2]*c);
pi[3,3,1,2]:=evalf(pi[3,3,1,2]*c);
pi[3,4,0,2]:=evalf(pi[3,4,0,2]*c);
pi[4,0,1,2]:=evalf(pi[4,0,1,2]*c);
pi[4,0,2,2]:=evalf(pi[4,0,2,2]*c);
pi[4,0,3,2]:=evalf(pi[4,0,3,2]*c);
pi[4,0,4,2]:=evalf(pi[4,0,4,2]*c);
pi[4,1,0,2]:=evalf(pi[4,1,0,2]*c);
pi[4,1,1,2]:=evalf(pi[4,1,1,2]*c);
pi[4,1,2,2]:=evalf(pi[4,1,2,2]*c);
pi[4,1,3,2]:=evalf(pi[4,1,3,2]*c);
pi[4,2,0,2]:=evalf(pi[4,2,0,2]*c);
pi[4,2,1,2]:=evalf(pi[4,2,1,2]*c);
pi[4,2,2,2]:=evalf(pi[4,2,2,2]*c);
pi[4,3,0,2]:=evalf(pi[4,3,0,2]*c);
pi[4,3,1,2]:=evalf(pi[4,3,1,2]*c);
pi[4,4,0,2]:=evalf(pi[4,4,0,2]*c);
all:=sq[01]+sq[02]+dq[0]+dq[1]+dq[2]+dq[3]+dq[4]+dq[20]+dq[21]
+dq[22]+dq[23]+dq[23]+dq[24];
evalf(%);
idle:=pi[0,0,0];
busy=1-%;
```

```
solve(%,pi[1,3,1]);
pi[1,3,1]:=%;
pi[1,3,2]:=chi[2,2]*pi[1,2,2]+chi[1,2]*pi[0,3,2]+alpha[1,2]*pi[0,1,0,1]+alpha[2,2]*pi[0,1,0,2];
eq[17]:=pi[0,2,2]=chi[2,2]*pi[0,1,2]+alpha[2,2]*pi[0,3,2]+alpha[1,2]*pi[0,3,1];
solve(%,pi[0,4,2]);
pi[0,4,2]:=%;
pi[1,3,1];
pi[1,3,2];
eq[18]:=pi[0,4,1]=chi[2,1]*pi[0,3,1]+alpha[1,1]*pi[1,0,1,1]+alpha[2,1]*pi[1,0,1,2];
pi[0,4,1]:=solve(%,pi[0,4,1]);
eq[19]:=pi[0,4,2]=chi[2,2]*pi[0,3,2]+alpha[2,2]*pi[0,0,1,2]+alpha[1,2]*pi[0,0,1,1];
eq[20]:=pi[1,0,1,2]=alpha[1,2]*pi[0,1,1,1]+alpha[2,2]*pi[0,1,1,2]+chi[2,2]*pi[1,3,2];
eq[21]:=pi[1,0,1,1]=chi[2,1]*pi[1,3,1]+alpha[1,1]*(pi[1,1,1,1]+pi[2,0,2,1])
+alpha[2,1]*(pi[1,1,1,2]+pi[2,0,2,2]);
eq[22]:=pi[0,0,1,1]=chi[2,1]*pi[0,4,1]+alpha[1,1]*pi[1,0,2,1]+alpha[2,1]*pi[1,0,2,2];
eq[23]:=pi[0,0,1,2]=alpha[1,2]*pi[0,0,2,1]+alpha[2,2]*pi[0,0,2,2]+chi[2,2]*pi[0,4,2];
eqnset:={seq(eq[j],j=19..23)};
solset:={pi[0,0,1,2],pi[0,0,2,1],pi[0,0,2,2],pi[1,0,1,1],pi[1,0,1,2],pi[0,0,1,1]};
s:=solve(eqnset,solset);
assign(s);
eq[24]:=pi[0,0,2,2]=chi[2,2]*pi[0,0,1,2]+alpha[2,2]*pi[0,0,3,2]+alpha[1,2]*pi[0,0,3,1];
eq[25]:=pi[0,0,3,2]=chi[2,2]*pi[0,0,2,2]+alpha[2,2]*pi[0,0,4,2]+alpha[1,2]*pi[0,0,4,1];
eq[26]:=pi[0,0,4,2]=rho[2]*pi[0,0,3,2];
eqnset:={seq(eq[j],j=24..26)};
solset:={seq(pi[0,0,j,2],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
eq[27]:=pi[1,0,2,1]=chi[2,1]*pi[1,0,1,1]+alpha[1,1]*(pi[1,1,2,1]+pi[2,0,3,1])
+alpha[2,1]*(pi[1,1,2,2]+pi[2,0,3,2]);
eq[28]:=pi[1,0,3,1]=chi[2,1]*pi[1,0,2,1]+alpha[1,1]*(pi[1,1,3,1]+pi[2,0,4,1])
+alpha[2,1]*(pi[1,1,3,2]+pi[2,0,4,2]);
eq[29]:=pi[1,0,4,1]=(lambda[2]/mu[1])*pi[1,0,3,1];
eqnset:={seq(eq[j],j=27..29)};
solset:={seq(pi[1,0,j,1],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
eq[30]:=pi[1,0,2,2]=chi[2,2]*pi[1,0,1,2]+alpha[1,2]*pi[0,1,2,1]+alpha[2,2]*pi[0,1,2,2];
eq[31]:=pi[1,0,3,2]=chi[2,2]*pi[1,0,2,2]+alpha[1,2]*pi[0,1,3,1]+alpha[2,2]*pi[0,1,3,2];
```

```
eq[32]:=pi[1,0,4,2]=rho[2]*pi[1,0,3,2];
eqnset:={seq(eq[j],j=30..32)};
solset:={seq(pi[1,0,j,2],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
eq[33]:=pi[0,0,2,1]=chi[2,1]*pi[0,0,1,1]+alpha[1,1]*pi[1,0,3,1]+alpha[2,1]*pi[1,0,3,2];
eq[34]:=pi[0,0,3,1]=chi[2,1]*pi[0,0,2,1]+alpha[1,1]*pi[1,0,4,1]+alpha[2,1]*pi[1,0,4,2];
eq[35]:=pi[0,0,4,1]=(lambda[2]/mu[1])*pi[0,0,3,1];
eqnset:={seq(eq[j],j=33..35)};
solset:={seq(pi[0,0,j,1],j=2..4)};
s:=solve(eqnset,solset);
assign(s);
sq[01]:=pi[0,0,0]+pi[0,0,1]+pi[0,1,1]+pi[0,2,1]+pi[0,3,1]+pi[0,4,1]+pi[1,0,1]+pi[1,1,1]
+pi[1,2,1]+pi[1,3,1]+pi[2,0,1]+pi[2,1,1]+pi[2,2,1]+pi[3,0,1]+pi[3,1,1]+pi[4,0,1];
sq[02]:=pi[0,0,2]+pi[0,1,2]+pi[0,2,2]+pi[0,3,2]+pi[0,4,2]+pi[1,0,2]+pi[1,1,2]+pi[1,2,2]
pi[1,3,2]+pi[2,0,2]+pi[2,1,2]+pi[2,2,2]+pi[3,0,2]+pi[3,1,2]+pi[4,0,2];
dq[0]:=pi[0,0,1,1]+pi[0,0,2,1]+pi[0,0,3,1]+pi[0,0,4,1]+pi[0,1,0,1]+pi[0,1,1,1]+pi[0,1,2,1]
+pi[0,1,3,1]+pi[0,2,0,1]+pi[0,2,1,1]+pi[0,2,2,1]+pi[0,3,0,1]+pi[0,3,1,1]+pi[0,4,0,1];
dq[1]:=pi[1,0,1,1]+pi[1,0,2,1]+pi[1,0,3,1]+pi[1,0,4,1]+pi[1,1,0,1]+pi[1,1,1,1]+pi[1,1,2,1]
+pi[1,1,3,1]+pi[1,2,0,1]+pi[1,2,1,1]+pi[1,2,2,1]+pi[1,3,0,1]+pi[1,3,1,1]+pi[1,4,0,1];
dq[2]:=pi[2,0,1,1]+pi[2,0,2,1]+pi[2,0,3,1]+pi[2,0,4,1]+pi[2,1,0,1]+pi[2,1,1,1]+pi[2,1,2,1]
+pi[2,1,3,1]+pi[2,2,0,1]+pi[2,2,1,1]+pi[2,2,2,1]+pi[2,3,0,1]+pi[2,3,1,1]+pi[2,4,0,1];
dq[3]:=pi[3,0,1,1]+pi[3,0,2,1]+pi[3,0,3,1]+pi[3,0,4,1]+pi[3,1,0,1]+pi[3,1,1,1]+pi[3,1,2,1]
+pi[3,1,3,1]+pi[3,2,0,1]+pi[3,2,1,1]+pi[3,2,2,1]+pi[3,3,0,1]+pi[3,3,1,1]+pi[3,4,0,1];
dq[4]:=pi[4,0,1,1]+pi[4,0,2,1]+pi[4,0,3,1]+pi[4,0,4,1]+pi[4,1,0,1]+pi[4,1,1,1]+pi[4,1,2,1]
+pi[4,1,3,1]+pi[4,2,0,1]+pi[4,2,1,1]+pi[4,2,2,1]+pi[4,3,0,1]+pi[4,3,1,1]+pi[4,4,0,1];
dq[20]:=pi[0,0,1,2]+pi[0,0,2,2]+pi[0,0,3,2]+pi[0,0,4,2]+pi[0,1,0,2]+pi[0,1,1,2]+pi[0,1,2,2]
+pi[0,1,3,2]+pi[0,2,0,2]+pi[0,2,1,2]+pi[0,2,2,2]+pi[0,3,0,2]+pi[0,3,1,2]+pi[0,4,0,2];
dq[21]:=pi[1,0,1,2]+pi[1,0,2,2]+pi[1,0,3,2]+pi[1,0,4,2]+pi[1,1,0,2]+pi[1,1,1,2]+pi[1,1,2,2]
+pi[1,1,3,2]+pi[1,2,0,2]+pi[1,2,1,2]+pi[1,2,2,2]+pi[1,3,0,2]+pi[1,3,1,2]+pi[1,4,0,2];
dq[22]:=pi[2,0,1,2]+pi[2,0,2,2]+pi[2,0,3,2]+pi[2,0,4,2]+pi[2,1,0,2]+pi[2,1,1,2]+pi[2,1,2,2]
+pi[2,1,3,2]+pi[2,2,0,2]+pi[2,2,1,2]+pi[2,2,2,2]+pi[2,3,0,2]+pi[2,3,1,2]+pi[2,4,0,2];
dq[23]:=pi[3,0,1,2]+pi[3,0,2,2]+pi[3,0,3,2]+pi[3,0,4,2]+pi[3,1,0,2]+pi[3,1,1,2]+pi[3,1,2,2]
+pi[3,1,3,2]+pi[3,2,0,2]+pi[3,2,1,2]+pi[3,2,2,2]+pi[3,3,0,2]+pi[3,3,1,2]+pi[3,4,0,2];
dq[24]:=pi[4,0,1,2]+pi[4,0,2,2]+pi[4,0,3,2]+pi[4,0,4,2]+pi[4,1,0,2]+pi[4,1,1,2]+pi[4,1,2,2]
+pi[4,1,3,2]+pi[4,2,0,2]+pi[4,2,1,2]+pi[4,2,2,2]+pi[4,3,0,2]+pi[4,3,1,2]+pi[4,4,0,2];
all:=sq[01]+sq[02]+dq[0]+dq[1]+dq[2]+dq[3]+dq[4]+dq[20]+dq[21]+dq[22]+dq[23]+dq[24];
c:=solve(all*k=1,k);
```

```
c*all;
pi[0,0,0]:=evalf(pi[0,0,0]*c);
pi[0,0,1]:=evalf(pi[0,0,1]*c);
pi[0,1,1]:=evalf(pi[0,1,1]*c);
pi[0,2,1]:=evalf(pi[0,2,1]*c);
pi[0,3,1]:=evalf(pi[0,3,1]*c);
pi[0,4,1]:=evalf(pi[0,4,1]*c);
pi[1,0,1]:=evalf(pi[1,0,1]*c);
pi[1,1,1]:=evalf(pi[1,1,1]*c);
pi[1,2,1]:=evalf(pi[1,2,1]*c);
pi[1,3,1]:=evalf(pi[1,3,1]*c);
pi[2,0,1]:=evalf(pi[2,0,1]*c);
pi[2,1,1]:=evalf(pi[2,1,1]*c);
pi[2,2,1]:=evalf(pi[2,2,1]*c);
pi[3,0,1]:=evalf(pi[3,0,1]*c);
pi[3,1,1]:=evalf(pi[3,1,1]*c);
pi[4,0,1]:=evalf(pi[4,0,1]*c);
pi[0,0,2]:=evalf(pi[0,0,2]*c);
pi[0,1,2]:=evalf(pi[0,1,2]*c);
pi[0,2,2]:=evalf(pi[0,2,2]*c);
pi[0,3,2]:=evalf(pi[0,3,2]*c);
pi[0,4,2]:=evalf(pi[0,4,2]*c);
pi[1,0,2]:=evalf(pi[1,0,2]*c);
pi[1,1,2]:=evalf(pi[1,1,2]*c);
pi[1,2,2]:=evalf(pi[1,2,2]*c);
pi[1,3,2]:=evalf(pi[1,3,2]*c);
pi[2,0,2]:=evalf(pi[2,0,2]*c);
pi[2,1,2]:=evalf(pi[2,1,2]*c);
pi[2,2,2]:=evalf(pi[2,2,2]*c);
pi[3,0,2]:=evalf(pi[3,0,2]*c);
pi[3,1,2]:=evalf(pi[3,1,2]*c);
pi[4,0,2]:=evalf(pi[4,0,2]*c);
pi[0,0,1,1]:=evalf(pi[0,0,1,1]*c);
pi[0,0,2,1]:=evalf(pi[0,0,2,1]*c);
pi[0,0,3,1]:=evalf(pi[0,0,3,1]*c);
pi[0,0,4,1]:=evalf(pi[0,0,4,1]*c);
pi[0,1,0,1]:=evalf(pi[0,1,0,1]*c);
pi[0,1,1,1]:=evalf(pi[0,1,1,1]*c);
```

```
pi[0,1,2,1]:=evalf(pi[0,1,2,1]*c);
pi[0,1,3,1]:=evalf(pi[0,1,3,1]*c);
pi[0,2,0,1]:=evalf(pi[0,2,0,1]*c);
pi[0,2,1,1]:=evalf(pi[0,2,1,1]*c);
pi[0,2,2,1]:=evalf(pi[0,2,2,1]*c);
pi[0,3,0,1]:=evalf(pi[0,3,0,1]*c);
pi[0,3,1,1]:=evalf(pi[0,3,1,1]*c);
pi[0,4,0,1]:=evalf(pi[0,4,0,1]*c);
pi[1,0,1,1]:=evalf(pi[1,0,1,1]*c);
pi[1,0,2,1]:=evalf(pi[1,0,2,1]*c);
pi[1,0,3,1]:=evalf(pi[1,0,3,1]*c);
pi[1,0,4,1]:=evalf(pi[1,0,4,1]*c);
pi[1,1,0,1]:=evalf(pi[1,1,0,1]*c);
pi[1,1,1,1]:=evalf(pi[1,1,1,1]*c);
pi[1,1,2,1]:=evalf(pi[1,1,2,1]*c);
pi[1,1,3,1]:=evalf(pi[1,1,3,1]*c);
pi[1,2,0,1]:=evalf(pi[1,2,0,1]*c);
pi[1,2,1,1]:=evalf(pi[1,2,1,1]*c);
pi[1,2,2,1]:=evalf(pi[1,2,2,1]*c);
pi[1,3,0,1]:=evalf(pi[1,3,0,1]*c);
pi[1,3,1,1]:=evalf(pi[1,3,1,1]*c);
pi[1,4,0,1]:=evalf(pi[1,4,0,1]*c);
pi[2,0,1,1]:=evalf(pi[2,0,1,1]*c);
pi[2,0,2,1]:=evalf(pi[2,0,2,1]*c);
pi[2,0,3,1]:=evalf(pi[2,0,3,1]*c);
pi[2,0,4,1]:=evalf(pi[2,0,4,1]*c);
pi[2,1,0,1]:=evalf(pi[2,1,0,1]*c);
pi[2,1,1,1]:=evalf(pi[2,1,1,1]*c);
pi[2,1,2,1]:=evalf(pi[2,1,2,1]*c);
pi[2,1,3,1]:=evalf(pi[2,1,3,1]*c);
pi[2,2,0,1]:=evalf(pi[2,2,0,1]*c);
pi[2,2,1,1]:=evalf(pi[2,2,1,1]*c);
pi[2,2,2,1]:=evalf(pi[2,2,2,1]*c);
pi[2,3,0,1]:=evalf(pi[2,3,0,1]*c);
pi[2,3,1,1]:=evalf(pi[2,3,1,1]*c);
pi[2,4,0,1]:=evalf(pi[2,4,0,1]*c);
pi[3,0,1,1]:=evalf(pi[3,0,1,1]*c);
pi[3,0,2,1]:=evalf(pi[3,0,2,1]*c);
```

```
pi[3,0,3,1]:=evalf(pi[3,0,3,1]*c);
pi[3,0,4,1]:=evalf(pi[3,0,4,1]*c);
pi[3,1,0,1]:=evalf(pi[3,1,0,1]*c);
pi[3,1,1,1]:=evalf(pi[3,1,1,1]*c);
pi[3,1,2,1]:=evalf(pi[3,1,2,1]*c);
pi[3,1,3,1]:=evalf(pi[3,1,3,1]*c);
pi[3,2,0,1]:=evalf(pi[3,2,0,1]*c);
pi[3,2,1,1]:=evalf(pi[3,2,1,1]*c);
pi[3,2,2,1]:=evalf(pi[3,2,2,1]*c);
pi[3,3,0,1]:=evalf(pi[3,3,0,1]*c);
pi[3,3,1,1]:=evalf(pi[3,3,1,1]*c);
pi[3,4,0,1]:=evalf(pi[3,4,0,1]*c);
pi[4,0,1,1]:=evalf(pi[4,0,1,1]*c);
pi[4,0,2,1]:=evalf(pi[4,0,2,1]*c);
pi[4,0,3,1]:=evalf(pi[4,0,3,1]*c);
pi[4,0,4,1]:=evalf(pi[4,0,4,1]*c);
pi[4,1,0,1]:=evalf(pi[4,1,0,1]*c);
pi[4,1,1,1]:=evalf(pi[4,1,1,1]*c);
pi[4,1,2,1]:=evalf(pi[4,1,2,1]*c);
pi[4,1,3,1]:=evalf(pi[4,1,3,1]*c);
pi[4,2,0,1]:=evalf(pi[4,2,0,1]*c);
pi[4,2,1,1]:=evalf(pi[4,2,1,1]*c);
pi[4,2,2,1]:=evalf(pi[4,2,2,1]*c);
pi[4,3,0,1]:=evalf(pi[4,3,0,1]*c);
pi[4,3,1,1]:=evalf(pi[4,3,1,1]*c);
pi[4,4,0,1]:=evalf(pi[4,4,0,1]*c);
pi[0,0,1,2]:=evalf(pi[0,0,1,2]*c);
pi[0,0,2,2]:=evalf(pi[0,0,2,2]*c);
pi[0,0,3,2]:=evalf(pi[0,0,3,2]*c);
pi[0,0,4,2]:=evalf(pi[0,0,4,2]*c);
pi[0,1,0,2]:=evalf(pi[0,1,0,2]*c);
pi[0,1,1,2]:=evalf(pi[0,1,1,2]*c);
pi[0,1,2,2]:=evalf(pi[0,1,2,2]*c);
pi[0,1,3,2]:=evalf(pi[0,1,3,2]*c);
pi[0,2,0,2]:=evalf(pi[0,2,0,2]*c);
pi[0,2,1,2]:=evalf(pi[0,2,1,2]*c);
pi[0,2,2,2]:=evalf(pi[0,2,2,2]*c);
pi[0,3,0,2]:=evalf(pi[0,3,0,2]*c);
```

```
pi[0,3,1,2]:=evalf(pi[0,3,1,2]*c);
pi[0,4,0,2]:=evalf(pi[0,4,0,2]*c);
pi[1,0,1,2]:=evalf(pi[1,0,1,2]*c);
pi[1,0,2,2]:=evalf(pi[1,0,2,2]*c);
pi[1,0,3,2]:=evalf(pi[1,0,3,2]*c);
pi[1,0,4,2]:=evalf(pi[1,0,4,2]*c);
pi[1,1,0,2]:=evalf(pi[1,1,0,2]*c);
pi[1,1,1,2]:=evalf(pi[1,1,1,2]*c);
pi[1,1,2,2]:=evalf(pi[1,1,2,2]*c);
pi[1,1,3,2]:=evalf(pi[1,1,3,2]*c);
pi[1,2,0,2]:=evalf(pi[1,2,0,2]*c);
pi[1,2,1,2]:=evalf(pi[1,2,1,2]*c);
pi[1,2,2,2]:=evalf(pi[1,2,2,2]*c);
pi[1,3,0,2]:=evalf(pi[1,3,0,2]*c);
pi[1,3,1,2]:=evalf(pi[1,3,1,2]*c);
pi[1,4,0,2]:=evalf(pi[1,4,0,2]*c);
pi[2,0,1,2]:=evalf(pi[2,0,1,2]*c);
pi[2,0,2,2]:=evalf(pi[2,0,2,2]*c);
pi[2,0,3,2]:=evalf(pi[2,0,3,2]*c);
pi[2,0,4,2]:=evalf(pi[2,0,4,2]*c);
pi[2,1,0,2]:=evalf(pi[2,1,0,2]*c);
pi[2,1,1,2]:=evalf(pi[2,1,1,2]*c);
pi[2,1,2,2]:=evalf(pi[2,1,2,2]*c);
pi[2,1,3,2]:=evalf(pi[2,1,3,2]*c);
pi[2,2,0,2]:=evalf(pi[2,2,0,2]*c);
pi[2,2,1,2]:=evalf(pi[2,2,1,2]*c);
pi[2,2,2,2]:=evalf(pi[2,2,2,2]*c);
pi[2,3,0,2]:=evalf(pi[2,3,0,2]*c);
pi[2,3,1,2]:=evalf(pi[2,3,1,2]*c);
pi[2,4,0,2]:=evalf(pi[2,4,0,2]*c);
pi[3,0,1,2]:=evalf(pi[3,0,1,2]*c);
pi[3,0,2,2]:=evalf(pi[3,0,2,2]*c);
pi[3,0,3,2]:=evalf(pi[3,0,3,2]*c);
pi[3,0,4,2]:=evalf(pi[3,0,4,2]*c);
pi[3,1,0,2]:=evalf(pi[3,1,0,2]*c);
pi[3,1,1,2]:=evalf(pi[3,1,1,2]*c);
pi[3,1,2,2]:=evalf(pi[3,1,2,2]*c);
pi[3,1,3,2]:=evalf(pi[3,1,3,2]*c);
```

```
    pi[3,2,0,2]:=evalf(pi[3,2,0,2]*c);
    pi[3,2,1,2]:=evalf(pi[3,2,1,2]*c);
    pi[3,2,2,2]:=evalf(pi[3,2,2,2]*c);
    pi[3,3,0,2]:=evalf(pi[3,3,0,2]*c);
    pi[3,3,1,2]:=evalf(pi[3,3,1,2]*c);
    pi[3,4,0,2]:=evalf(pi[3,4,0,2]*c);
    pi[4,0,1,2]:=evalf(pi[4,0,1,2]*c);
    pi[4,0,2,2]:=evalf(pi[4,0,2,2]*c);
    pi[4,0,3,2]:=evalf(pi[4,0,3,2]*c);
    pi[4,0,4,2]:=evalf(pi[4,0,4,2]*c);
    pi[4,1,0,2]:=evalf(pi[4,1,0,2]*c);
    pi[4,1,1,2]:=evalf(pi[4,1,1,2]*c);
    pi[4,1,2,2]:=evalf(pi[4,1,2,2]*c);
    pi[4,1,3,2]:=evalf(pi[4,1,3,2]*c);
    pi[4,2,0,2]:=evalf(pi[4,2,0,2]*c);
    pi[4,2,1,2]:=evalf(pi[4,2,1,2]*c);
    pi[4,2,2,2]:=evalf(pi[4,2,2,2]*c);
    pi[4,3,0,2]:=evalf(pi[4,3,0,2]*c);
    pi[4,3,1,2]:=evalf(pi[4,3,1,2]*c);
    pi[4,4,0,2]:=evalf(pi[4,4,0,2]*c);
    all:=sq[01]+sq[02]+dq[0]+dq[1]+dq[2]+dq[3]+dq[4]+dq[20]+dq[21]
+dq[22]+dq[23]+dq[23]+dq[24];
    evalf(%);
    idle:=pi[0,0,0];
    busy=1-%;
```