Analyzing queues with customer differentiation using matrix analytic methods

Universiteit Antwerpen

Faculteit Wetenschappen

Departement Wiskunde & Informatica

Analyzing Queues with Customer Differentiation using Matrix Analytic Methods

Een analyse van wachtrijsystemen met differentiatie in klanten gebruikmakend van matrix analytische methodes

Proefschrift voorgelegd tot het behalen van de graad van
doctor in de Wetenschappen aan de Universiteit Antwerpen,
te verdedigen door

Jeroen VAN VELTHOVEN

Promotor: Prof. dr. Benny Van Houdt
Copromotor: Prof. dr. Chris Blondia

Antwerpen, 2008

Contents

List of Figures

List of Algorithms

To Kristof, My Brother,

Preface

In general, building a telecommunication system is a quite expensive task. If afterward it turns out that the implementation that was chosen does not meet the expectations, we will need to change the system, which is in many cases even more expensive than building an entire system from scratch. Therefore we generally would like to have an estimate of the system's performance before it is actually implemented and this can be done by modeling the system and analyzing this model. Moreover, an analytical model can help us to understand the influence of changes in the environment on the system's behavior.

Once the model is constructed and the accompanying algorithms are identified, we can verify if the system will perform as we would expect. We can also use it to compare different design alternatives, to answer dimensioning issues, to optimize the system's performance, etc. Depending on what we want from the analysis, we can define our performance measures of interest. In a queueing system, we are typically interested in the time customers have to wait or the total time customers have to spend in the system. These times are referred to as the waiting time, respectively response time, of a customer.

Together with these time measures we might be interested in a distribution for the queue length which can be used together with a threshold for the fraction of lost customers to dimension the buffers in the queueing system. Indeed, we can only provide finite storage for the waiting customers and as a consequence it is possible that an arriving customer cannot be admitted to the buffer, since it is entirely occupied by other customers. This customer will then be considered lost. In a typical system, we want to minimize the number of lost customers, while keeping the total cost as small as possible.

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. First of all, I would like to thank my adviser prof. dr. Benny Van Houdt for the excellent guidance he gave me during the past four years, for introducing me into the field of matrix analytic methods and providing me with stimulating suggestions. I am also very grateful to prof. dr. Chris Blondia who gave me the opportunity to perform the research that has led to this thesis at the PATS research group.

This work was only possible thanks to the great support of my family and friends. Thank you Niki, for being that very special part of my life, for your support and your love. My parents earn a special word of appreciation for everything they have done for me. Not only during the last few years, but ever since I entered into their lives. Even in the most difficult times their love and support were never absent for even a single minute. I could never have established this without you, mama and papa.

Thanks to my colleagues at the University of Antwerp, to Joke Lambert for the constructive discussions and the joint organization of the FriS and WIS seminars. I want to thank Gino Peeters for helping me out with all kind of computer related issues. My thanks also go to Joris Van Geet and Juan F. Pérez for the helpful answers and useful suggestions, and to Niels Van Eetvelde and Frank Altheide for the pleasant atmosphere in office G3.19.

Thank you Tom Hofkens, Nele Dexters, Sabine Verboven, Jeroen Avonts, and many other colleagues for your cooperation in making different activities such as “de Kinderuniversiteit” en “het Wetenschapsfeest” truly unforgettable. Finally, I would like to thank everyone else that contributed to this work in any way or another.

Notations and Acronyms

Before we proceed to the analysis of the queueing systems and the presentation of the algorithms to obtain the performance measures of interest, let us first introduce some notations and acronyms that will be used throughout this thesis.

Notations

I_n	$n \times n$ unity matrix
I_n^+	$n \times n$ zero matrix with ones on the upper diagonal and at entry (n, n)
$I_{mn}^=$	$m \times n$ zero matrix with ones on the main diagonal
I_{mn}^-	$m \times n$ zero matrix with ones on the lower diagonal
I_{mn}^{--}	$m \times n$ zero matrix with ones on the diagonal below the lower diagonal
e_n	$n \times 1$ column vector filled with ones
$e_n^{(i)}$	$n \times 1$ row zero vector with only its i-th entry equal to one
$e_n^{(i)t}$	$n \times 1$ column zero vector with only its i-th entry equal to one
x^t	transposed vector of x
x^+	$\max(x, 0)$
$1_A(x)$	characteristic function: 1 if $x \in A$, 0 otherwise
$diag[X]$	block diagonal matrix composed from the block vector X
$sp(X)$	spectral radius of the matrix X
$A \,.\, B$	point-wise matrix product of the matrices A and B
$A \otimes B$	Kronecker product of the matrices A and B
$A \oplus B$	Kronecker sum of the matrices A and B, i.e., $A \otimes I + I \otimes B$
$A^{(s)*}$	s-fold convolution of A
$X \leq_{cx} Y$	X precedes Y in the convex ordering sense

Remark, when referring to a square matrix of dimension n, we will write I_n^* instead of I_{nn}^*, for $* \in \{+, -, --\}$. Note also that the dimension of the vectors and matrices will be omitted if this is unambiguously determined by their context.

Acronyms

btbT	block triangular block Toeplitz
CPU	Central Processing Unit
CTM	Capetanakis-Tsybakov-Mikhailov random access protocol

D-BMAP	Discrete-time Batch Markovian Arrival Process
D-MAP	Discrete-time Markovian Arrival Process
DFT	Discrete Fourier Transform
DTMC	Discrete-time Markov Chain
FCFS	First-Come First-Served
FIFO	First-In-First-Out
Geo	Geometric arrival or service process
GI	General Independent
i.i.d	independent and identically distributed
I/O	Input / Output
IAT	Inter Arrival Time
LJG	Latouche-Jacobs-Gaver algorithm
LIFO	Last-In-First-Out
MAP	Markovian Arrival Process
MMAP[K]	Markovian Arrival Process with Marked Jobs
MMPP	Markov Modulated Poisson Process
MC	Markov Chain
MST	Maximum Stable Throughput
NBD	Negative Binomially Distributed
PH	Phase-Type distribution
QBD	Quasi-Birth-Death
QBD^m	Quasi-Birth-Death Markov chain with Marked Time Epochs
RR	Round Robin
SS	System with limited Sojourn time
SW	System with limited Waiting time

Summary and Organization

This thesis presents different analytical models and accompanying algorithms that can be used to analyze parts of a telecommunication system where customers require a certain kind of processing. Customers that cannot be processed immediately, have to wait in a buffer or are lost when the buffer space is entirely occupied. In Part I, a short introduction to queueing theory is presented, together with some basic principles of Markov chains and *matrix analytic methods*, a technique whereupon we rely when designing our models. When analyzing these models, we exploit the structure appearing in the matrices to reduce the computational complexity as much as possible. In the introductory part, we also present some terminology that is used throughout this thesis and a few basic results on the subject under consideration, without going into much detail.

The work presented in this thesis is divided into two main parts. Part II concentrates on the steady state performance measures of a variety of queues with customer differentiation, while Part III focuses on transient analysis. In *Chapter 3* a model to analyze a D-MAP/PH/1 buffer with *impatient customers* is presented. We allow the patience distribution of arriving customers to be general, with the only restriction that there is some maximum amount of patience a customer can have. Three types of customer impatience are considered. Customers can be impatient only when they are waiting or they can remain impatient while being processed. The second kind of impatience can be further divided into systems where customers are aware of their required processing time and systems where customers do not know in advance how long their service will take. The basic algorithm to determine the response time distribution makes use of a GI/M/1 type Markov chain that keeps track of the age of the customer in service. The disadvantage of this algorithm is that the time needed to compute the response time distribution is a square function of the maximum amount of patience. This computation time can be reduced to a linear function of the maximum amount of patience by slightly increasing the space complexity and making use of so-called quasi-birth-death processes. The introduction of a number of additional states to the state space of the Markov chain increases the number of zero blocks in its transition matrix. The tridiagonal block structure of the resulting transition matrix induces the development of a significantly faster algorithm, as is illustrated by some numerical examples at the end of Section 3.2.

These examples also indicate a possible relation between the variance of the patience distribution of a customer and the number of customers that are lost due to buffer overflow. In Section 3.3, the influence of the patience distribution on the *loss probability* is further investigated. To be able to draw general conclusions about this possible influence, a simpler arrival and service process is considered in this section. It is proved that a smaller patience distribution in the convex ordering sense results in a smaller abandonment probability in a queue with geometric inter arrival times and geometric service times. This result holds for both the

system where customers are only impatient while waiting and for the system with customers that remain impatient as long as their processing is not completed. It is also proved that in a queue with geometric service times there are fewer abandonments in the system where customers are only impatient while waiting compared to the system where customers remain impatient during their service.

The next two chapters consider *priority queues* where a distinction between the arriving customers is made in terms of importance. Customers with a high priority level will be served before customers belonging to a lower priority class. Each class has its own finite buffer to store waiting packets. To simplify the analysis of a (priority) system, in analytical models buffers are often assumed to be infinite. As in the systems with impatient customers, the number of lost customers is also an important performance measure of priority systems. If the model uses infinite buffers, the loss rate cannot be computed exactly, it has to be approximated. *Chapter 4* attempts to expose potential consequences one might experience when assuming one or two buffers infinite on the approximation of the loss rate in a two-class priority system. For this purpose five different ways to approximate the loss rate are discussed together with an exact approach. The exact approach, i.e., the system with two finite buffers, uses an M/G/1 type Markov chain where the G matrix, needed to compute the steady state probabilities, is known explicitly. It is observed that assuming the low priority buffer infinite results in an overestimate of the accompanying loss rate, while the infiniteness of the high priority buffer has only a very limited impact on the low priority loss rate. Relying on the actual steady state probabilities or the asymptotic tail behavior seems to make little difference if the high priority queue is finite. However, in case both queues are infinite very inaccurate loss probabilities are observed when making use of the asymptotic tail behavior, especially in the area near the transition point as illustrated by an arbitrary numerical example.

An alternative approach for modeling a priority system with an arbitrary number of priority classes and more complex arrival and service processes, i.e., the MMAP[K]/PH[K]/1 priority queue, is presented in *Chapter 5*. In contrast to the other chapters, this chapter deals with a continuous-time setting. After performing a uniformization, the corresponding discrete-time queue is modeled as a tree-structured process. Depending on the type of the customer in service, we keep track of the amount of certain types of customers present in the queue, while other customers are stored on a stack. The introduction of some additional states and transitions between these states, transforms the resulting Markov chain into a *tree-like process*. The additional states are required to remove transitions where multiple customers are removed from, or pushed on the stack. They provide the matrices with a nice structure that can be exploited when computing the steady state probabilities.

In a priority queue, customers get assigned their priority level upon arrival. There are however situations in which we might prefer to modify the priority of a customer depending on its evolution in the queue. When considering an operating system that needs to schedule the allocation of different jobs to the CPU, one might prefer to schedule the shortest job first to minimize the average response time. Unfortunately, the required processing time is usually not known in advance and therefore the priority level of a job cannot be predetermined. The *multilevel feedback queue*, analyzed in *Chapter 6*, tries to cope with this problem by splitting the processing time of each job into small time units and at any time serving the job that has received the least service time so far. To analyze this queue, a new class of tree-structured processes is introduced, namely the *quasi-birth-death processes on a binomial-like tree*. In a

binomial-like tree, each i-th child of a node has exactly i children of its own. A QBD process on such a tree allows us to efficiently capture the behavior of a multilevel feedback queue where jobs arrive in batches and have generally distributed service times with finite support. Methods to obtain performance measures, like the average queue length and the average response time of a customer, are discussed and illustrated by some numerical examples. In these examples we furthermore investigate the influence of (i) the batch size distribution of arriving customers and (ii) the processing time distribution on (a) the average queue length and (b) the coefficient of variation of the queue length distribution.

Part III deals with the *transient performance analysis* of two general classes of Markov chains. In a variety of situations, it is preferable to perform a transient study instead of a steady state analysis, e.g., when examining systems that do not have a steady state, or simply to understand the initial behavior of the system, especially when it takes a long time before the steady state is eventually reached. In contrast to the models presented in Part II, we do not consider a specific telecommunication (sub)system here. Instead, we present a general framework that can be applied to perform a transient analysis of any system that can be modeled as a QBD Markov chain or as a tree-like process. Many existing approaches are very powerful in solving transient problems over short time scales, but have difficulties obtaining transient performance measures for epochs further in time. The main reason is that, in those approaches, the computation times increase as a function of the time epoch of interest. By converting the transient problem into a steady state analysis we are able to efficiently obtain performance measures over much longer time scales. This conversion is accomplished by applying a discrete Erlangization and constructing a reset Markov chain. For this construction, we introduce a so-called *reset counter*. This counter increases the number of states in the Markov chain, nevertheless, it also introduces an appealing structure in the transition matrix that can be exploited to reduce the time complexity of our algorithms.

Chapter 7 discusses the transient analysis of a QBD Markov chain using of the notion of a *quasi-birth-death process with marked time epochs*. In such a QBD part of the time epochs get marked according to the performance measure we want to study, so that our research question translates into obtaining the system state at the n-th marking. In transient studies, the original state can have an important influence on the performance results. For this reason, there is a need for an efficient manner of comparing the results for different initial configurations. A very time consuming approach would be to compute the results for a specific initial state, then to change this state and redo all computations. Instead, we propose a level based recursion that reuses intermediate results for initial states belonging to levels $0, 1, \ldots, r-1$ to compute the transient results when the initial state is part of level r. A main obstacle when identifying the invariant vector of a QBD Markov chain is the computation of a set of R-matrices. If the initial state is part of level zero, this involves determining the solution of a quadratic matrix equation. This can however be simplified to solving a quadratic matrix equation of a much smaller dimension and a set of Sylvester matrix equations. Relying on their probabilistic interpretation, we observe that the R-matrices for a situation with an initial state that is part of level one, can be obtained directly from the results for level zero and the computation of just one additional block row. This strategy can be continued by increasing the initial level one at a time, each time computing one additional block row, using a very simple recursive formula. Moreover, the computations for all states belonging to a certain level can be done simultaneously. The efficiency of this level based recursion is illustrated by a numerical example at the end of Chapter 7.

Summary and Organization

The idea of marking time epochs, applying a discrete-Erlangization and constructing a reset Markov chain is generalized to the class of tree-like processes in *Chapter 8*. Analogous to the construction presented in Chapter 7, we define a *tree-like process with marked time epochs* and discuss how the transient analysis can be converted into a steady state problem. We present different algorithms to compute the V-matrix, which is essential when solving a tree-like process, and compare them with relation to their computation times and the convergence of the algorithm. To show the applicability of this framework, we explore the transient behavior of the Capetanakis-Tsybakov-Mikhailov protocol for random access. We present how this protocol can be modeled as a tree-like process and study its initial behavior under both normal and overload conditions. With overload conditions we denote situations with an input rate above the maximum stable throughput, which is the highest possible input rate for which a packet has a finite delay with probability one. The numerical examples show among others that a Poisson input rate above the maximum stable throughput does not result in an output rate above this throughput. Initially, these input rates can result in a rather high throughput, but after a few slots many stations will get backlogged, considerably lowering the output rate. To conclude, we discuss and illustrate how the average time until the n-th success, or until the n-th collision, can be obtained from the computed steady state probabilities.

Part I

Introduction

1

Queueing Theory & Markov Chains

Queues can be encountered almost everywhere in our daily life. Sometimes we even do not realize that we are in a queue, while at other times, we are very aware of having to wait in line. Probably, no one really likes to be in a queue, certainly not when it is a long one. However, given the fact that we have to spend a significant amount of time in certain queues, we would prefer to have some kind of *fair* scheduling policy that guarantees that everyone is able to leave the queue after a reasonable amount of time compared to the other queue occupiers. As will be discussed further, this is not always equivalent to try to have the same waiting time for everyone. We can find an everyday example of such a scheduling discipline on the street where priority rules schedule the order in which cars can cross an intersection. On places where this priority rules would cause a lot of problems, traffic lights are placed in order to introduce a more fair scheduling discipline. In a traffic jam, we would expect that cars that arrived first would also leave the queue before the others, however, it seems logical that an ambulance heading to the location of the accident can pass the stationary vehicles without having to wait its turn.

In telecommunication systems this is not really different, except that cars on the road are actually packets that are transmitted on a shared link, jobs running on the same CPU, etc. Depending on the type of the packets or the importance of a job, we again need to design a *fair* scheduling discipline. The problem here is that it is not easy to give an indisputable definition of the word "fair". One might for example choose to give time sensitive traffic priority over the other traffic on the same link, however, if there is too much time sensitive traffic, the other traffic might experience intolerable delays. A similar reasoning can be made with relation to jobs requiring a certain processing time by the CPU. If there is a short and a long job waiting, we would prefer to process the short job first, since its influence on the long job is smaller than vice versa. Unfortunately, the required processing time of a job is generally not known in advance, making the real situation somewhat more complex.

Throughout this thesis we will construct a number of Markov chains to model queueing systems using different scheduling policies and present efficient algorithms to obtain the performance measures of interest. This first chapter provides, besides general queueing theory information, also a short introduction to Markov chains and some accompanying terminology.

1.1 Markov Chains

We can look at a certain queueing system as being a stochastic process $\{X_t, t \in \mathcal{T}\}$, where we think of t as time, $\mathcal{T}$ as a set of points in time and X_t as the state of the stochastic process at time t. Take for example a simple queueing system, where the state is given by the number of customers in this system. When a new customer arrives or a customer completes his service, the total number of customers in the queue will change. This corresponds to a transition to a different state in the stochastic process. Stochastic processes are classified according to time, in the sense that we refer to a discrete-time process if $\mathcal{T}$ is discrete and to a continuous-time process if $\mathcal{T}$ is continuous.

An important subset of stochastic processes are the *Markov processes*. Informally, one could say that a Markov process is a stochastic process with a limited form of historical dependency. This dependency will be formalized in Definition 1. Notice, some authors incorrectly denote the discrete-time Markov processes as *Markov chains* and the continuous-time counterpart as *Markov processes*, while one should use the terms *discrete-time Markov chain* and *continuous-time Markov chain*. Since we will mainly use discrete-time processes, we start by defining a discrete-time Markov chain. The definition of a Markov chain in continuous-time is similar and will be mentioned further in this section.

Definition 1 A stochastic process $\{X_n, n \geq 0\}$ on a finite or countably infinite state space is called a discrete-time Markov chain (DTMC) if the following property holds

$$P[X_{n+1} = j | X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, X_n = i] = P[X_{n+1} = j | X_n = i], \tag{1.1}$$

for all time epochs n and all states $i_0, \ldots, i_{n-1}, i, j$.

Property (1.1) is known as the Markov property. It expresses that, given the value of X_t, the values of X_s with $s > t$ are not influenced by the values of X_u, for $u < t$. The next state therefore only depends on the current state and not on the previous states of the process. We will use $p_{i,j}(n)$ to denote the probability $P[X_{n+1} = j | X_n = i]$. A DTMC $\{X_n, n \geq 0\}$ is called time-homogeneous if the conditional probabilities $P[X_{n+1} = j | X_n = i]$ are independent of n. That is, a DTMC is time-homogeneous if $p_{i,j}(n) = p_{i,j}, \ \forall n \geq 0$. In this thesis, all Markov chains we will encounter are time-homogeneous. Therefore, we will assume that the Markov chains that will occur in the rest of this section are time-homogeneous. Hence, with this assumption, we can write

$$p_{i,j} = P[X_{n+1} = j | X_n = i]. \tag{1.2}$$

For a time-homogeneous MC we can construct the transition matrix containing the probabilities that the chain moves from one state to another as

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} & \cdots \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots \\ p_{n,1} & p_{n,2} & \cdots & p_{n,n} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots \end{bmatrix}. \tag{1.3}$$

Given that the initial state of the MC is represented by the distribution x_0, the system state at the n-th time epoch is given by $x_n = x_0 P^n$. The steady state distribution (also called the

invariant distribution) of the MC is the solution to $\pi P = \pi$, with $\pi e = 1$. Of course, not every Markov chain has such a steady state distribution. We will briefly state the conditions that are required for an MC to have a steady state distribution as a detailed discussion can be found in almost every book touching the field of Markov chains, e.g., [32, 89, 76, 39, 57, 94, 26]. In order to do this, we first present some basic terminology.

- A state j is said to be *accessible* form state i if there exists an integer k for which $P_{i,j}^k > 0$. Two states i and j *communicate* if i is accessible from j and vice versa.

- A Markov chain is *irreducible* if all pairs of states communicate.

- The *period* of a state is defined as the greatest common divisor d of all integers $k > 0$ for which $P_{i,i}^k > 0$. If $d = 1$, the state is called *aperiodic*. A Markov chain is said to be aperiodic if all states are aperiodic.

- A state i is recurrent if the probability that the chain eventually returns to this state equals one. A recurrent state is *positive recurrent* if $E[T_i] < \infty$, where T_i is the time between two visits to state i, otherwise it is called *null recurrent*. If all the states of MC are (positive or null) recurrent, the MC itself is said to be (positive or null) recurrent.

- A state that is not recurrent is called a *transient* state.

- An *ergodic* state is a state that is positive recurrent and aperiodic.

Theorem 1. *If a Markov chain is irreducible and positive recurrent, there exists a unique solution to the linear system $\pi P = \pi$, $\pi e = 1$. If moreover the chain is aperiodic the probabilities $P[X_n = i]$ will converge to π_i as $n \to \infty$.*

In Part II we will concentrate on the steady state performance of different Markov chains that are constructed to represent various types of queueing systems one can encounter in a telecommunication system. In Part III we will study the transient behavior of two general classes of Markov chains. In that part, we do not lay the emphasis on a particular system. On the contrary, we will present a general framework that can be applied to any kind of system, the analytical model of which is covered by one of these two classes. Before moving on to the queueing theory part of this chapter, we first mention the continuous-time MC that will be used rather sporadically in this work. We will not include a detailed description of continuous-time Markov chains since this is analogous to its discrete-time counterpart. In the case of a continuous-time MC $\{X(t), t \geq 0\}$, the Markov property translates into

$$P[X(t+s) = j | X(s) = i, X(u) : 0 \leq u < s] = P[X(t+s) = j | X(s) = i], \tag{1.4}$$

for all $t, s \geq 0$.
In this thesis we will demonstrate how Markov chains can be applied to model various aspects of a telecommunication system. However, the applicability of Markov chains is certainly not limited to this type of problems. They can be used within a wide range of research areas like physics, economics, genetics, neurology, etc.

1.2 Queueing Theory

Informally we could say that a queueing system consists of customers arriving at random times to some facility where they receive service and then depart. Figure 1.1 shows a simple queueing system in which new customers arrive in the buffer where they have to wait some time before they can be served. When the service facility becomes available, a scheduling discipline decides which customer will be served next. This customer then moves to the service facility and after receiving its required processing it will leave the queueing system. Depending on the context, we will sometimes use different terms to express a specific part of the system. We shall for example use *waiting room*, *buffer*, *queue* to denote the part where customers have to wait before entering the *service facility* or *server*. When considering tasks that have to be processed by a CPU, we use *job size* or *job duration* instead of the traditional *packet length* or *service time*, etc. Nevertheless, the basic principles described above remain unchanged in every context.

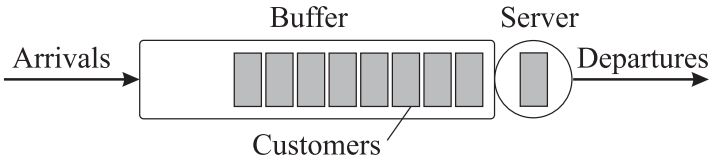


Figure 1.1: A basic queueing system

Queueing systems are classified according to the input process, the service time distribution, the size of the buffer(s), the number of servers and the scheduling discipline. They can be represented using the Kendall notation that bundles the various characteristics of the system. A queueing system can be described as $A/B/C/K/N/D$, where A denotes the arrival process, B the service time distribution, C the number of servers, K the size of the buffer, N the size of the population and D the scheduling discipline. Throughout this thesis we will assume that $N = \infty$ and we will mainly use the shorter notation $A/B/C$, while describing the other characteristics in the text. In the next sections we will pay some attention to the different parts of a general queueing system as presented in Figure 1.1.

1.2.1 Arrival Process

In this section we will describe some frequently used arrival processes. These are very elementary arrival processes that can be used to model certain simple systems. However, they are insufficient to cover more complex arrival streams of customers which often occur in telecommunication networks. A more general arrival process is discussed in Section 2.2.1.

1.2.1.1 Deterministic Arrivals

The first one, the deterministic arrival process, is without any doubt the simplest process to describe. In this process the time between two successive arrivals is constant. The deterministic arrival process can be used in a continuous-time as well as a discrete-time setting. A typical example where this arrival process can be applied is a source that generates packets at a constant rate, commonly referred to as a constant bit rate (CBR) source. One can think for example of a simple voice application where the time between the generation of two successive packets is constant.

1.2.1.2 Poisson Arrivals

Although the deterministic arrival process can be applied to some specific cases, it is inadequate to model arrivals in the major part of the applications. Another relatively simple, but maybe the most traditional arrival processes in queueing theory is the Poisson process. It has the property that the inter arrival times (IATs), i.e., the time between two successive arrivals, follow an exponential distribution. That is, the probability that the time between two arrivals is at most t equals $1 - e^{-\lambda t}$, where λ is the mean arrival rate. The probability of having i arrivals in a time interval T is given by

$$\frac{(\lambda T)^i}{i!} e^{-\lambda T}. \tag{1.5}$$

The Poisson arrival process has a number of useful properties that make it interesting to use this process for modeling purposes. First, the process is memoryless, meaning that the probability of having a new arrival does not depend on the previous arrivals. So, at an arbitrary point in time, events in the future will occur independently of any past events. It is obvious that this property can simplify the analysis of the system under consideration, yet one should verify whether the assumption of having independent arrivals is appropriate, i.e., whether the Poisson process is suitable to model the arriving customers. Typically, Poisson processes are encountered when the population is sufficiently large and arrivals from this population occur separately and independently from each other.

Another property of this process is that the superposition of two Poisson processes with mean arrival rates λ_1 and λ_2 is again a Poisson process with rate $\lambda_1 + \lambda_2$. Analogously, the inverse property holds, i.e., the decomposition (using probabilities p_1 and p_2, with $p_1 + p_2 = 1$) of a Poisson process with rate λ results in two independent Poisson processes with rate $p_1\lambda$ and $p_2\lambda$.

1.2.1.3 Geometric Arrivals

The discrete-time counterpart of the Poisson arrival process is the Bernoulli arrival process, where the IATs are geometrically distributed. As opposed to the continuous-time process, we now assume that time is slotted into unit intervals and at most one customer can arrive in each slot. Arrivals are assumed to be random and occur in each slot independently with some probability α. Thus, the probability that the time between two arrivals equals n time slots can be expressed as

$$(1 - \alpha)^{n-1}\alpha. \tag{1.6}$$

Like with the Poisson arrival process, the memoryless property also holds for the geometric arrivals. It is also easy to see that the number of arrivals in a set of k slots is distributed as a binomial distributed random variable with parameters (k, α).

1.2.2 Scheduling Discipline

All customers that arrive in a queueing system, require some kind of processing. The order in which customers are processed is determined by the scheduling discipline. There is a large variety of scheduling policies, where each basic policy often has different variants itself. The classic scheduling discipline is the first-in-first-out (FIFO) discipline, where the customers

are served in the order they arrived. We can compare this to a single waiting line at the checkout in a supermarket where people join at the end of the line and the person first in line will be served next. This policy shall often be referred to as the first-come first-served (FCFS) discipline. The opposite policy is referred to as the last-in-first-out (LIFO) scheduling discipline. With this policy the customer that arrived last will be served first. This strategy is comparable to a stack mechanism, where the last element that is pushed onto the stack will be the first one to be removed.

In the previous disciplines a job is processed entirely and afterward it leaves the system with probability one. When dealing with large jobs, it might be a good idea to split the processing of a job into different parts. The scheduling discipline that arises from this thought is the so-called round robin (RR) discipline. Upon entering the service facility a customer is allocated a quantum of service. After this, there are two possibilities: if his processing is not yet finished, he joins the queue again, waiting for a next quantum of service, otherwise he leaves the system. When these quanta shrink to zero, we denote the resulting discipline as processor sharing. Because jobs can be interrupted by other jobs and the processing is split into different quanta, the round robin and the processor sharing scheduling disciplines avoid that large jobs are being favored.

If more than one buffer is present, the scheduling discipline also needs to specify in which order the different buffers are visited. Here, a straightforward technique consists of assigning to each buffer a different priority class. The customer that will be served when the service facility becomes available is the next customer in the nonempty buffer with the highest priority. The "next" customer in this buffer is identified by following the scheduling discipline within each buffer. There is a further distinction between preemptive and nonpreemptive priority scheduling. With the first technique customers will be interrupted when a job of higher priority arrives, whereas with the nonpreemptive priority scheduling the newly arriving job needs to wait until the job in the server finishes its processing.

Other scheduling disciplines are, e.g., *shortest processing time* where the jobs with the smallest processing time can enter the service facility first, *shortest remaining processing time* that is comparable to the shortest processing time discipline, but that allows preemption, and *shortest elapsed time* where the processing time of a job is divided in different quanta like with the RR discipline, but each time the server becomes available, the job that has received the shortest processing time so far will be processed first. This discipline is also known as the multilevel feedback queue, which will be modeled and analyzed in Chapter 6.

1.2.3 Service Process

Like the various arrival processes presented in Section 1.2.1, we will also present a very short overview of some basic, frequently used, service time distributions in the modeling of telecommunication systems. Within the context of matrix analytic methods, a less restrictive service time distribution will be discussed in Section 2.2.2.

1.2.3.1 Deterministic Service Time

The simplest service time distribution is of course the deterministic distribution. In this case every customer requires the same amount of service. Analogous to the deterministic arrival process, we can apply the deterministic service time distribution to continuous as well as to discrete time.

1.2.3.2 Geometric and Exponential Service Time

Depending whether we are working in a discrete-time or a continuous-time setting, we can assume a geometric, resp. exponential, service time distribution. If the length of a service time is geometrically distributed with parameter β, the probability that an arbitrary customer requires n time slots of service is given by

$$(1-\beta)^{n-1}\beta. \tag{1.7}$$

This is equivalent with stating that the service of a customer ends in the current time slot with probability β. For an exponential distribution with a mean service time equal to $1/\mu$, the probability that the service time of a customer is at most t equals

$$1-e^{-\mu t}. \tag{1.8}$$

1.2.4 Some Basic Queueing Systems

By combining these arrival and service processes, we can construct the well known M/M/1, M/D/1, M/G/1, Geo/Geo/1, ... queues. Here, M denotes the exponential (memoryless) distribution, Geo the geometric distribution, D the deterministic distribution and G a general distribution. Hence, in an M/G/1 queue we have Poisson arrivals and general service times, where the service process is represented by an arbitrary distribution. Results about the waiting times, response times, etc. in these these queueing systems are included in almost any book about queueing theory, e.g., [56, 37, 76].

Of course, the deterministic and exponential distributions (or geometric in a discrete-time setting) are quite limited to model a realistic queueing system, since arrivals might be correlated, there could be more than one arrival at the same time, there can be multiple types of customers in the system, and the service time distribution might be more complex than a simple deterministic or exponential distribution. In Section 2.2 we mention a more complex arrival and service process, which will be used in different sections of this thesis.

1.2.5 Customer Differentiation

As in daily life, not all customers in telecommunication systems are equal. Customer differentiation is the main reason why scheduling disciplines were introduced. Customers might differ in packet size (or job duration), patience, importance, etc. In Part II, we will consider three types of customer differentiation. Chapter 3 discusses systems with different types of impatient customers, while Chapters 4 and 5 consider priority queues. We will pay attention to the traditional two-class priority queue as well as the general system with more priority classes. Finally, in Chapter 6 we will model and analyze the multilevel feedback queue which was also mentioned in Section 1.2.2.

2

Matrix Analytic Methods

A strategy to analyze Markov processes is to rely on matrix analytic methods. They are extensively used in the modeling and performance analysis of computer systems, telecommunication networks, network protocols and many other stochastic systems of current commercial and engineering interest. The main idea is to represent such a process using a matrix notation (as was illustrated in the previous section with the transition matrix of a general time-homogeneous Markov chain) and to take advantage of special structures appearing in this matrix. Exploiting these structures while computing performance measures or designing algorithms can result in a huge gain in computation time as well as memory usage. Another advantage of working with matrix analytic methods is that one can easily understand the matrix representation with relation to its probabilistic interpretation.

A pioneer in this field of research is Marcel Neuts [77, 78]. His research led among others to the matrix geometric distribution and the Phase-Type (PH) processes. Over the last few decades, broad classes of frequently encountered queueing models have been analyzed by matrix analytic methods [13, 62]. The embedded Markov chains in these models are two-dimensional generalizations of the classic M/G/1 and GI/M/1 queues, and quasi-birth-death processes. Matrix analytic models include notions such as the Markovian arrival process and the PH distribution, both in discrete and continuous time. Considerable efforts have been put into the development of efficient and numerically stable methods for their analysis [13]. There is also an active search for accurate matching algorithms for both PH distributions and MAP arrivals when modeling communication systems [8, 18, 19, 22, 25, 45, 47, 48, 87, 91, 104].

The basic mathematical ideas and algorithms of the matrix analytic theory are presented in a comprehensible manner in [62]. In this book the authors mainly address the QBD processes after which they also present a short discussion of the generalization to the M/G/1 and GI/M/1 type Markov chains. This chapter contains a description of these three types of Markov chains, without elaborating on the methods that were developed to analyze them. Tree-like processes, one of the more recent paradigms within the field of matrix analytic methods, will be discussed in more detail. We also introduce the D-MAP/PH/1 queueing system that will be used several times in this thesis. This system allows more general arrivals and service times compared to the queues mentioned in Section 1.2.4.

2.1 Structured Processes

The step from the stochastic processes presented in the previous chapter to those appearing in matrix analytic theory is mainly a step from scalars to matrices. Instead of keeping track of individual states and transition probabilities between these states we will now cluster states into blocks of states where transition submatrices describe the transitions between these blocks. In other words, one can say that these are processes in two dimensions, the *level* and the *state* within that level. In this section, we consider some structured Markov chains that will be used to model the systems appearing in this thesis.

2.1.1 M/G/1 and GI/M/1 Type Markov Chains

M/G/1 type Markov chains have transition matrices of the same form as the embedded Markov chain of the M/G/1 queue, except that the entries are matrices, rather than scalars. The same holds for the GI/M/1 type Markov chain with relation to the GI/M/1 queue. Note, to construct the embedded Markov chain for the M/G/1 queue, one has to keep track of the number of customers present in the system at service completion epochs only, since the stochastic process that holds the number of customers at each time epoch does not fulfill the Markov property (1.1). For the same reason, the embedded Markov chain for the GI/M/1 queue keeps track of the number of customers present at arrival epochs.

In an M/G/1-type MC, there is only one nonzero block diagonal below the main diagonal. The other entries are positioned so that the blocks appearing on the different diagonals of the matrix are all equal, except for the first block row and the first block column. Similarly, in an GI/M/1-type MC there is only one block diagonal above the main diagonal that differs from zero. The other entries are again positioned in such a way that all blocks on the diagonals are equal (except for the first block row and column). The following equations present the structure of the transition matrices $P_{M/G/1}$ and $P_{GI/M/1}$ that represent an MC of the M/G/1, resp. GI/M/1 type:

$$P_{M/G/1} = \begin{bmatrix} B_0 & C_1 & C_2 & C_3 & \cdots \\ B_1 & A_1 & A_2 & A_3 & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad P_{GI/M/1} = \begin{bmatrix} B_0 & C_1 & 0 & 0 & \cdots \\ B_1 & D_1 & D_0 & 0 & \cdots \\ B_2 & D_2 & D_1 & D_0 & \cdots \\ B_3 & D_3 & D_2 & D_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

As illustrated by these equations, the default M/G/1 and GI/M/1 type MCs are level independent, that is, the entries of the associated transition matrices are the same on each diagonal. Further in this work, we will encounter the level dependent variant of these MCs where the entries on the diagonals depend on the block row on which they are located.

2.1.2 Quasi-Birth-Death Markov Chains

A more simple type of Markov chains that is often used in the modeling of telecommunication systems, are Quasi-Birth-Death (QBD) Markov chains. In this type of Markov chain the only transitions that are possible from some level i are transitions to level $i-1$, to level i itself or to level $i+1$. QBDs are matrix generalizations of simple birth-and-death processes, where transitions are only allowed from state n to one of the states $n-1$, n or $n+1$. As the M/G/1 and GI/M/1 type MCs could be seen as generalizations of the embedded MC for the M/G/1

and the GI/M/1 queue, the QBD can be considered to be the generalization of the embedded MC for the M/M/1 queue. The transition matrix of a QBD has the following structure:

$$P_{QBD} = \begin{bmatrix} B_1 & B_0 & 0 & 0 & \cdots \\ B_2 & A_1 & A_0 & 0 & \ddots \\ 0 & A_2 & A_1 & A_0 & \ddots \\ 0 & 0 & A_2 & A_1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{2.1}$$

The entries in A_0 and B_0 on the upper block diagonal correspond to the *births*, whereas the blocks A_2 and B_2 on the lower diagonal hold the probabilities of a *death*. A graphical representation of a QBD is given in Figure 2.1.

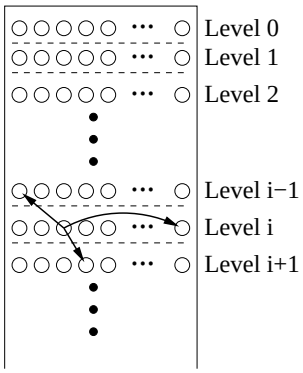


Figure 2.1: A Quasi-Birth-Death Markov chain

Let ϕ be the stationary probability vector of $A = A_0 + A_1 + A_2$, i.e., $\phi A = \phi$ and $\phi e = 1$. Under the conditions that $\phi A_2 e > \phi A_0 e$, we can compute the steady state distribution π of the QBD Markov chain, i.e., the solution to the linear system $\pi P_{QBD} = \pi$, $\pi e = 1$ [62]. Therefore, we need to find the so-called R-matrix. The (i, j)-th entry of this matrix contains the expected number of visits to state j of level $i+1$, before returning to level i again. It can be found as the smallest nonnegative solution to the quadratic matrix equation

$$R = A_0 + RA_1 + R^2 A_2. \tag{2.2}$$

Having computed R, an equivalent stability condition can given as $sp(R) < 1$. Alternative stability conditions can be found in [63]. After having computed the matrix $R_0 = B_0(I - A_1 - RA_2)^{-1}$, the steady state distribution can be obtained from

$$\pi_0 = \pi_0(B_1 + R_0 B_2), \tag{2.3}$$

$$\pi_1 = \pi_0 R_0, \tag{2.4}$$

$$\pi_n = \pi_{n-1} R, \quad \text{for } n > 1, \tag{2.5}$$

with $\pi_0 e + \pi_0 R_0 (I - R)^{-1} e = 1$. As with the two types of MCs presented in the previous section, we will also encounter the level dependent variant of this QBD further in the thesis. The reason that this type of Markov chain is very popular is of course the useful structure that can be exploited during the computations. In Section 3.2.5 we will for example present a method to convert a level-dependent GI/M/1 type Markov chain into a QBD. The analysis of the transient behavior of a QBD will be discussed in Chapter 7.

2.1.3 Tree-Like Processes

One of the more recent paradigms within the field of matrix analytic methods has been the generalization to discrete-time bivariate Markov chains $\{(X_t, N_t), t \geq 0\}$, in which the values of X_t are the nodes of a d-ary tree (and N_t takes integer values between 1 and some value h).

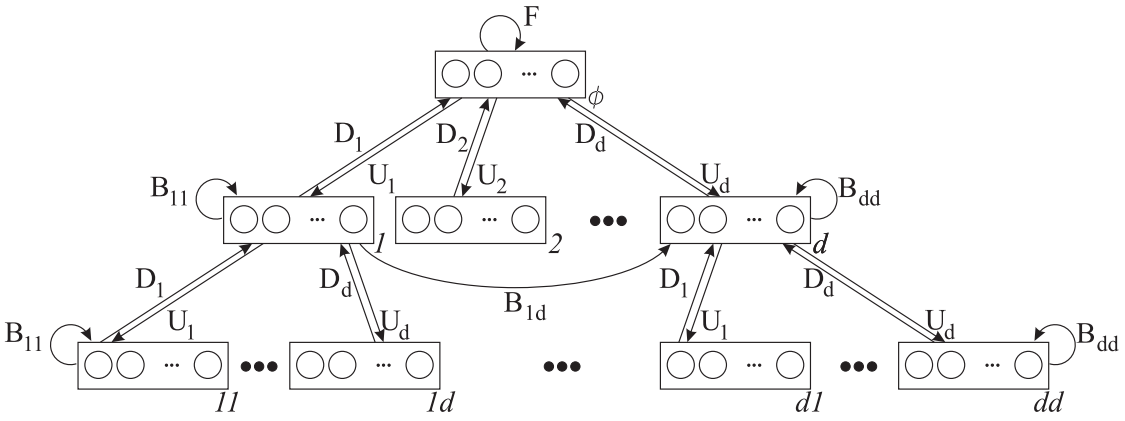


Figure 2.2: Structure of the first three levels of a tree-structured Markov chain

Figure 2.2 illustrates the structure of a general tree-structured Markov chain together with the matrices describing the transitions between the different nodes of the tree. The (i, j)-th entry of the matrix D_k contains the probability of a transition from state i of node $J + k$ to state j of its parent node J. The (i, j)-th entry of the matrix U_k $(1 \leq k \leq d)$ contains the probability of a transition from state i of node J to state j of node $J + k$. The transitions between two sibling nodes $J + k$ and $J + l$ are enclosed in the B_{kl} matrices $(1 \leq k, l \leq d)$ and in the matrix F for the root node.

This root node is denoted as $\emptyset$ and the remaining nodes are denoted as strings of integers, where each integer takes a value between 1 and d, as shown in Figure 2.2 at the right bottom corner of the nodes. For instance, the k-th child of the root node is represented by k, the l-th child of the node k by kl, and so on. In the context of tree-structured processes, we will use '+' to denote the concatenation on the right and '-' to represent the deletion from the right. For example, if $J = k_1 k_2 \ldots k_n$, then $J + k = k_1 k_2 \ldots k_n k$. Let $f(J, k)$, for $J \neq \emptyset$, denote the k rightmost elements of the string J, then $J - f(J, 1)$ represents the parent node of J.

Depending on the transitions allowed in such a tree-structured MC, several classes have been identified: (a) M/G/1 type [103], (b) GI/M/1 type [118], (c) QBD type [117] and (d) tree-like processes [15], where the latter two classes were shown to be equivalent. Namely, it has been shown that any tree-structured QBD can be reduced to a tree-like process [107]. Moreover, every tree-like process can be embedded in a binary tree-like Markov chain with a special structure [98]. In each of these models the transitions fulfill a spacial homogeneity property, that is, the transition probability between two nodes depends only on the spacial relationship between the two nodes and not on their specific values.

Let us focus a bit on tree-like processes since it is this class of tree-structured MCs that we will use throughout this thesis. They are defined as a tree-structured QBD MC with the additional requirement that there are no transitions between sibling nodes, i.e., $B_{ij} = 0$ if

$i \neq j$ and that

$$B_{11} = B_{22} = \ldots = B_{dd} = B.$$

That is, at each step the chain can only make a transition to its parent (i.e., $X_{t+1} = X_t - f(X_t, 1)$, for $X_t \neq \emptyset$), to itself ($X_{t+1} = X_t$), or to one of its own children ($X_{t+1} = X_t + s$ for some $1 \leq s \leq d$). Moreover, the chain's state at time $t+1$ is determined as follows:

$$P[(X_{t+1}, N_{t+1}) = (J', j) | (X_t, N_t) = (J, i)] =$$
$$\begin{cases} f^{i,j} & J' = J = \emptyset, \\ b^{i,j} & J' = J \neq \emptyset, \\ d_k^{i,j} & J \neq \emptyset, f(J,1) = k, J' = J - f(J,1), \\ u_s^{i,j} & J' = J + s, s = 1, \ldots, d, \\ 0 & otherwise. \end{cases}$$

From these transition probabilities, we can define the $h \times h$ matrices D_k, B, F and U_s with respective (i,j)-th elements given by $d_k^{i,j}$, $b^{i,j}$, $f^{i,j}$ and $u_s^{i,j}$, as illustrated in Figure 2.2. This completes the description of the tree-like process. Notice, a tree-like process is fully characterized by the matrices D_k, B, U_s and F.

Let us first introduce a number of matrices that play a crucial role when studying the stability and stationary behavior of a tree-like process. The fundamental period of a tree-like process starting in state $(J+k, i)$ is defined as the first passage time from the state $(J+k, i)$ to one of the states (J, j), for $j = 1, \ldots, h$. Let G_k, for $1 \leq k \leq d$, denote the matrix whose (i,v)-th element is the probability that the MC is in state (J, v) at the end of a fundamental period which started in state $(J+k, i)$. Let the (i,v)-th element of the matrix R_k denote the expected number of visits to state $(J+k, v)$ before visiting node J again, given that $(X_0, N_0) = (J, i)$.

Finally, let V denote the matrix whose (i,v)-th element is the taboo probability that starting from state $(J+k, i)$, the process eventually returns to node $J+k$ by visiting $(J+k, v)$, under the taboo of the node J. Notice, due to the restrictions on the transition probabilities, the matrix V does not depend on k. It is easy to see that the following expressions hold for these matrices [117]:

$$G_k = (I - V)^{-1} D_k, \tag{2.6}$$
$$R_k = U_k (I - V)^{-1}, \tag{2.7}$$
$$V = B + \sum_{s=1}^{d} U_s G_s. \tag{2.8}$$

Combining these equations, we have the following relation:

$$V = B + \sum_{s=1}^{d} U_s (I - V)^{-1} D_s. \tag{2.9}$$

Provided that the tree-like process $\{(X_t, N_t), t \geq 0\}$ is ergodic, which is the case if and only if the all the G_k matrices are stochastic (note, all the G_k matrices can be stochastic while the MC is null recurrent, however, these situations barely occur in practice) or likewise if and only if the spectral radius of $R = R_1 + \ldots + R_d$ is less than one, define

$$\pi_i(J) = \lim_{t \to \infty} P[X_t = J, N_t = i],$$
$$\pi(J) = (\pi_1(J), \pi_2(J), \ldots, \pi_h(J)).$$

The vectors $\pi(J)$ can be computed from $\pi(\emptyset)$ using the relation

$$\pi(J+k) = \pi(J)R_k \tag{2.10}$$

and $\pi(\emptyset)$ is found by solving the boundary condition

$$\pi(\emptyset) = \pi(\emptyset)\left(\sum_{k=1}^{d} R_k D_k + F\right) \tag{2.11}$$

with the normalizing restriction that

$$\pi(\emptyset)(I-R)^{-1}e = 1. \tag{2.12}$$

Notice the similarity (both with relation to the presented equations as in the probabilistic interpretation of the matrices) with the QBD Markov chains, where the steady state probabilities could be computed from the R-matrices using equations (2.3), (2.4), and (2.5). Moreover, by setting $d = 1$ in a tree-like process we end up with a basic QBD Markov chain.

Various iterative algorithms to determine, among others, the steady state probability vector of these types of Markov chains have been developed. As opposed to the standard M/G/1, GI/M/1 and QBD type chains such algorithms exhibit either only linear convergence or are quadratic, but require an iterative procedure at each step or the solution of a large system of linear equations (e.g., [15]).

2.2 The D-MAP/PH/1 Queue

The arrival and service processes presented in Section 1.2 allow us to build some basic queueing systems. Nevertheless, more complex processes are needed to be able to model a typical telecommunication application. Within the field of matrix analytic methods, the Markovian arrival process and the Phase-Type service time distribution have been introduced, naturally leading to the D-MAP/PH/1 queue. We will return several times to this queueing system in the course of this manuscript.

2.2.1 The Discrete-Time Markovian Arrival Process

Although the Poisson and geometric arrival processes have a lot of advantages, they do not allow to model correlated arrivals. If we want to model burstiness, periodicity, etc., we will need a different and more detailed arrival process, which is still analytically tractable. For this reason, the batch Markovian arrival process (BMAP) was introduced in [65, 67]. A special case of this process is the MAP which does not allow batch arrivals. With a batch arrival we refer to the arrival of multiple customers at the same time instant. The discrete-time equivalent of the (B)MAP is the D-(B)MAP, which was introduced in [16, 17].

The D-MAP process is characterized by two $m \times m$ matrices D_0 and D_1, with m a positive integer. These matrices contain the transition probabilities of the underlying Markov chain when either a customer arrives (covered by D_1) or not (D_0). For example, the entry (j_1, j_2) of D_1 represents the probability that there is an arrival and the underlying Markov chain makes a transition from state j_1 to state j_2. The matrix D_0 covers the case in which there is no arrival. Remark, for the more general D-BMAP arrival process with batch arrivals we

additionally have some matrices D_i, with $i = 2, \ldots, k$ describing the probabilities of having a batch arrival of size $2, \ldots, k$.

The matrix D represents the stochastic $m \times m$ transition matrix of the underlying Markov chain and is defined by

$$D = D_0 + D_1. \tag{2.13}$$

Throughout this thesis, we will assume that D is irreducible. Denote θ as the solution to $\theta D = \theta$ and $\theta e = 1$, where e is a vector with all its entries equal to one of the appropriate dimension. That is, θ is the stationary probability vector of D. Then, the stationary arrival rate is given by

$$\lambda = \theta D_1 e. \tag{2.14}$$

As a small illustrative example we present a two-state D-MAP process in which the arrival probability in state one, resp. state two, equals 0.2 customers per time slot, resp. 0.4 customers per time slot, while a transition from state one to state two occurs with probability 0.3 and the opposite transition occurs with probability 0.8. The matrices D_0 and D_1 are then given by

$$D_0 = \begin{bmatrix} 0.56 & 0.24 \\ 0.48 & 0.12 \end{bmatrix}, \quad D_1 = \begin{bmatrix} 0.14 & 0.06 \\ 0.32 & 0.08 \end{bmatrix},$$

and the stationary arrival rate equals 2.55 customers per time slot.

2.2.2 Phase-Type Service Time

As with the arrival processes, the service time distributions presented in the Section 1.2.3 impose certain limitations on the model. In everyday situations a lot of services can be identified that do not fit these distributions. The Phase-Type (PH) distributions however, cover a much wider range of applications. More specifically, the class of PH distributions is dense in the field of all distributions with positive support [62]. PH distributions can be used both in a discrete-time and a continuous-time setting. Discrete-time PH distributions are matrix generalizations of the geometric distribution in the same way as QBDs are matrix generalizations of a simple birth-and-death process. Naturally, for continuous-time PH distributions the same can be noted with relation to the exponential distribution.

Let us introduce the PH distributions with relation to the service time of a customer as this will be their only application throughout this thesis, keeping in mind that PH distributions are of course more generally applicable. A discrete-time PH distribution is characterized by its matrix representation (m_{ser},α,T), where m_{ser} is a positive integer. The $1 \times m_{ser}$ vector α contains the probabilities that the service of a customer starts in a given phase. The probability that a customer continues his service in phase j at time $n+1$, given the phase at time n equals i, is represented by the (i,j)-th entry of the $m_{ser} \times m_{ser}$ substochastic matrix T. Finally, define

$$t = e - Te, \tag{2.15}$$

that is, t is a vector which contains the probabilities that a customer completes his service, given the current phase of the service process. Notice, the service time of a customer equals

k time units with probability $\alpha T^{k-1}t$ and the mean service time is given by

$$\frac{1}{\mu} = \alpha(I-T)^{-1}e. \tag{2.16}$$

Phase-Type distributions will be used frequently in this thesis, since they are well suited for representing most of the types of services encountered in communication systems [59, 53].

2.2.3 Modeling the D-MAP/PH/1 Queue

The D-MAP/PH/1 queue forms a QBD Markov chain, the transition matrix of which is given by

$$P = \begin{bmatrix} B_1 & B_0 & 0 & 0 & \cdots \\ B_2 & A_1 & A_0 & 0 & \ddots \\ 0 & A_2 & A_1 & A_0 & \ddots \\ 0 & 0 & A_2 & A_1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix},$$

with

$$\begin{array}{lcllcl} B_0 & = & D_1 \otimes \alpha, & A_0 & = & D_1 \otimes T, \\ B_1 & = & D_0, & A_1 & = & (D_0 \otimes T) + (D_1 \otimes t\alpha), \\ B_2 & = & D_0 \otimes t, & A_2 & = & D_0 \otimes t\alpha. \end{array}$$

We furthermore assume that $\alpha e = 1$. In Section 3.2 we will study the steady state behavior of a D-MAP/PH/1 queue with impatient customers. Here, customers can leave the queue early when they have to wait for a long time. We present formulas for the waiting time distribution, the queue length distribution and the loss probability. A comparison between three types of impatient customers is made. In Section 7 we present an algorithm that can be used for a transient analysis of a QBD Markov chain and we present some numerical examples in which we apply our methods to an elementary D-MAP/PH/1 queue to show the efficiency of our algorithm.

Part II

Steady State Analysis

3

Impatient Customers

This chapter addresses a first type of customer differentiation where every customer that enters the queue has a particular amount of patience. Such a customer is willing to wait a certain amount of time to receive service, however, if he does not receive his required service before this time, he will abandon the system without being served. Impatient customers can for example be encountered in real-time applications. These applications often include strict timing constraints, i.e., packets become worthless unless they reach their destination before a certain deadline. As such, these packets should be considered impatient when setting up an appropriate queueing model. Queueing systems with impatient customers can also be applied to, for instance, telephone systems, where people have to wait for a dial tone [120], call centers [34], where the customers are only willing to wait a certain amount of time before they can be served, data communication networks with a time-out protocol, etc. Other examples of queues with impatient customers can be found in manufacturing and service industries, e.g., inventory systems with perishable goods.

In Section 3.2 different types of impatient customers are discussed and compared with each other. We consider the D-MAP/PH/1 queue with three types of impatient customers. That is, a customer can be impatient during his entire sojourn time or only while waiting. For customers with a limited sojourn time, we can further distinguish between those customers that are aware of their service time and those who are not. We start with constructing a GI/M/1 type Markov chain by keeping track of the age of the customer in service and the state of the D-MAP arrival process immediately after the customer in service arrived. To reduce the computational complexity, we propose a reduction to a QBD afterward.

Section 3.3 presents some results about the relation between the patience distribution and the abandonment probability in a Geo/Geo/1 queue. We will show that a smaller patience distribution in the convex-ordering sense leads to a lower abandonment probability, irrespective of whether customers become patient when entering the service facility. Moreover, we will prove that in a queue with geometric service times, there are fewer abandonments in the system where customers are only impatient while waiting compared to the system where customers remain impatient during their entire service.

3.1 Related Work

The study of single-server queues with impatient customers has a long history. It seems Palm [81] was the first to consider customer impatience. Barrar [10] analyzed the M/M/1+D system. A key reference for the general GI/GI/1+GI is Baccelli *et al.* [9]. In this work a stability condition was established for the general case, while for the M/GI/1+GI queue the virtual waiting time was studied. Markovian arrivals were considered by Combé [28], who studied the MAP/G/1+M queue and derived an expression for the transform of the virtual waiting time and rejection probability of a customer. Most of these studies assume that a customer becomes patient when entering the server and set up a Markov process using the virtual (offered) waiting time. Van Houdt *et al.* [114] developed an algorithm to compute the response time distribution in a D-MAP/PH/1+D queue, by setting up a finite GI/M/1 type Markov chain. In Section 3.2, the work in [114] is generalized in a number of ways. First, we allow the patience distribution to be general, as opposed to deterministic. Second, we also consider the system where customers only enter the server if they are able to complete their service before reaching their maximum amount of patience. Third, a QBD reduction procedure is included to improve the time complexity of the algorithms, and finally, we will show that apart from the response time distribution and the rejection probability, the queue length distribution can easily be obtained as a by-product.

Bhattacharya & Ephremides [12] showed, for the GI/GI/m+GI queue, that the number of customers abandoning the system over any time interval decreases stochastically when the patience distribution becomes stochastically larger (i.e., when the patience distribution X_1 is replaced by X_2 with $P[X_2 \geq x] \geq P[X_1 \geq x]$, for all $x \geq 0$) . Thus, as intuitively expected, more patience leads to fewer abandonments. An interesting open question therefore is: What can be said about the the probability of abandonment for different patience distributions X with the same mean, finite amount of patience $m = E[X]$? Making use of the results by Brandt & Brandt [21, Theorem 3.1] or Boxma & De Waal [20] it can be shown easily that in an M/M/1+GI queue, where customers become patient when entering the service facility, smaller distributions (in the convex-ordering sense) give rise to fewer abandonments. In both these papers an expression for the probability of abandonment is established via the steady-state probabilities of the virtual offered waiting time. To the best of our knowledge this is the only result of this type available in the literature. In Section 3.3 the discrete-time counterpart of the M/M/1+GI will be proved, as well as the more difficult system where customers remain impatient after entering the service facility.

3.2 General Customer Impatience in the D-MAP/PH/1 Queue

In this section, two methods to compute the response time distribution of impatient customers in the D-MAP/PH/1 queue are presented. The first approach uses a GI/M/1 type Markov chain and may be regarded as a generalization of the procedure presented in [114] for the D-MAP/PH/1 queue, where every customer has the same amount of patience. The key construction in order to obtain the response time distribution is to set up a Markov chain based on the age of the customer being served, together with the state of the D-MAP process immediately after the arrival of this customer.

The disadvantage of this algorithm is that the time needed to calculate the response time distribution is a square function of the maximum amount of patience r that a customer can

have. A solution to this problem is given by the introduction of the second algorithm which is based on a QBD reduction. As a result, the time complexity becomes a linear function of the maximum amount of patience r. On the other hand, an implementation of the QBD based algorithm requires somewhat more memory in comparison with the GI/M/1 approach. We also show that apart from the response time distribution and the rejection probability, the queue length distribution can easily be obtained as a by-product.

Consider a patience distribution X that contains the probability that an arriving customer has a certain amount of patience. We allow X to be generally distributed, except that we assume that there exists some $r \geq 0$ sufficiently large, such that $P[X > r] = 0$, i.e., the maximum amount of patience that a customer can have is bounded above by some constant r. We consider three different systems: (i) a customer is impatient during his sojourn time (waiting + service) and may thus be partially served, (ii) customers are aware of their service time and only enter the service facility if their amount of patience is sufficient to complete service, (iii) a customer can only run out of patience while waiting, i.e., he becomes patient as soon as he enters the service facility. We shall denote each of these three D-MAP/PH/1 queues as D-MAP/PH/1 + GI, to reflect the general nature of the patience distribution. It should be noted that in the literature this notation is mostly used for systems where the customers have a limited waiting time (case (iii)).

3.2.1 The Discrete-Time D-MAP/PH/1+GI Queue

The arrival process of the queueing system of interest is the discrete-time Markovian arrival process discussed in Section 2.2.1. Recall that this process is characterized by two $m \times m$ matrices D_0 and D_1 and that the stationary arrival rate is given by $\lambda = \theta D_1 e$, with θ the stationary probability vector of $D = D_0 + D_1$. The arriving customers are being served in a FCFS order, except for those leaving the waiting room when they reach their critical age. The service time of a customer has a common discrete-time PH distribution with a matrix representation (m_{ser},α,T). As in Section 2.2.2, we define $t = e - Te$ as the vector containing the probabilities that a customer completes his service. Furthermore, set

$$m_{tot} = m_{ser} m. \tag{3.1}$$

Each customer has a finite amount of patience that is i.i.d. and arbitrarily distributed according to some random variable X. The patience distribution X is characterized by the stochastic vector

$$\tilde{a} = (a_1, a_2, \ldots, a_r),$$

where a_i is the probability $P[X = i]$ that the amount of patience of a customer equals i time units, while r is the maximum amount of patience a customer can have. Let p_i be the probability $P[X \leq i]$ that the patience of a customer is at most i time units, hence,

$$p_i = \sum_{j=1}^{i} a_j. \tag{3.2}$$

In order to simplify the notation, we define $p_0 = 0$, which we may regard as the probability that the amount of patience of a customer equals zero time units. Because such customers are never served, we do not consider them here. Nevertheless, such customers can easily be

taken into account by modifying the characteristics of the arrival process. The amount of patience of a customer is also referred to as his critical age.

Finally, if there is a departure and an arrival at the same time, we assume that the departure occurs first. Also, when a customer arrives while the server is idle, his service starts immediately. In each of the models presented, we will consider the system just prior to possible arrivals, departures or phase changes. Thus, if we refer to the system state at time n, such events happening at time n are not yet taken into account. For instance, when a customer starts his service at time n, the probability that the phase of the service process equals i at time $n+1$ is given by the i-th component of the stochastic vector α. Also, the minimum age of a customer in the service facility is one time unit.

3.2.2 Impatient Customers in the System

First, we consider the D-MAP/PH/1+GI queue with customers who are impatient in the entire system, that is, all customers are impatient irrespective of whether they are being served or not. A customer reaching his critical age will leave the queue without starting or completing his service. This system is also referred to as the "limitation on sojourn time with unaware customers" [9], as customers are impatient during their entire sojourn time and are not aware, when entering the queue, whether their total sojourn time will be larger than their patience.

Consider the following Markov chain (MC) with $rm_{tot}+m$ states. Denote level zero of the MC as the set of states $\{1,\ldots,m\}$ and level i as the set of states $\{(i-1)m_{tot}+m+1,\ldots,im_{tot}+m\}$, for $0<i\leq r$. The states of level $i>0$ are labeled as (s,j), with $1\leq s\leq m_{ser}$ and $1\leq j\leq m$. Let n be the current time instant and let state (s,j) of level i (with $0<i\leq r$) correspond to the situation in which the age of the customer in service equals i, the service process is currently in phase s and the D-MAP was in state j at time $n-i+1$. We say that the age of a customer equals i when he arrived in the system i time units ago. Also, let state j of level zero correspond to the situation in which the server is idle and the current state of the arrival process is j. Then, the system can be described by a transition matrix P with a lower block-Hessenberg structure, i.e.,

$$P=\begin{bmatrix} B_1 & B_0 & 0 & 0 & \cdots & 0 & 0 \\ B_2 & A_1^1 & A_0^1 & 0 & \cdots & 0 & 0 \\ B_3 & A_2^2 & A_1^2 & A_0^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{r-1} & A_{r-2}^{r-2} & A_{r-3}^{r-2} & A_{r-4}^{r-2} & \cdots & A_0^{r-2} & 0 \\ B_r & A_{r-1}^{r-1} & A_{r-2}^{r-1} & A_{r-3}^{r-1} & \cdots & A_1^{r-1} & A_0^{r-1} \\ E & C_r & C_{r-1} & C_{r-2} & \cdots & C_2 & C_1 \end{bmatrix}.$$

Here, B_1 is an $m\times m$ matrix, B_0 is an $m\times m_{tot}$ matrix, B_i $(i>1)$ and E are $m_{tot}\times m$ matrices and A_k^i and C_i are $m_{tot}\times m_{tot}$ matrices. We will now derive an expression for each of these matrices.

Let us start with the first level of P, level zero. If the MC is in a state of this level, the server is idle and only two events can take place: either a customer arrives or not. If there is no arrival, the MC remains at level zero and makes a transition to some state of this level, according to the transition of the underlying Markov chain of the arrival process. Hence,

$$B_1=D_0. \tag{3.3}$$

If a customer does arrive at the current time instant n, a transition from level zero to level one occurs, this new arrival immediately enters the service facility and starts his service in a phase determined by the vector α, so

$$B_0 = \alpha \otimes D_1. \tag{3.4}$$

Remark, in this chapter we set the state of the arrival process, resp. the phase of the service process, as the inner variable, resp. the outer variable. This is different from the formulas presented in Section 2.2.3, however, this has of course no influence on the results. If the MC is in state (s_1,j_1) of level i at time n, there is a transition to level zero if the customer in service leaves the system and there is no customer present in the waiting room at time $n+1$. There are two possible causes for the customer in service to leave the service facility: either he completes his service or he reaches his critical age. Therefore, the customer in service leaves the system with a probability

$$t_{s_1} + \frac{a_i}{1-p_{i-1}}\left(1 - t_{s_1}\right).$$

Let us now derive the probability that the system will be empty at time $n+1$, provided that an age i customer leaves the system at time n. Unlike the case in which all of the customers have the same amount of patience [114], it is possible that a customer arrival occurred during some of the time instants $n-i+1, n-i+2, \ldots, n-1$, who left before or at time n due to impatience. Notice, a customer arriving at time n is always present in the system at time $n+1$ as the minimum amount of patience is one time unit. If the critical age of a customer arriving at time $n-i+k$, for $k = 1, \ldots, i-1$, is at most $i-k$, which is the case with probability p_{i-k}, this customer is no longer in the queue at time $n+1$. Obviously, there could be multiple customers of this kind. When we combine these probabilities, we find

$$B_{i+1} = \left(t + \frac{a_i}{1-p_{i-1}}\left(e - t\right)\right) \otimes \prod_{k=1}^{i}\left(D_0 + p_{i-k}D_1\right), \tag{3.5}$$

for $0 < i < r$ (recall, $p_0 = 0$). A transition from level i to level $i+1$ occurs when the customer in service has not reached his critical age nor completed his service. In this case the customers stays in the system and the state of the D-MAP remains the same. Hence,

$$A_0^i = \frac{1-p_i}{1-p_{i-1}} T \otimes I_m. \tag{3.6}$$

The only remaining transitions from level i are those to some lower level, $i-l$ $(0 < l < i)$. They occur when the customer in service leaves the system and the first customer in the waiting room arrived at time $n-i+l+1$. Notice, this customer is still patient, otherwise he would have left the system before time $n+1$. This yields,

$$A_{l+1}^i = \left(t + \frac{a_i}{1-p_{i-1}}\left(e - t\right)\right) \alpha \otimes \left(\prod_{k=1}^{l}\left(D_0 + p_{i-k}D_1\right)\right)\left(1 - p_{i-l-1}\right) D_1. \tag{3.7}$$

Notice that there could have been arrivals on some of the time instants $n-i+k$, with $0 < k \leq l$, who reached there critical age before time $n+1$. When the next customer with a sufficient amount of patience arrived at time $n-i+1$, the MC remains at the same level. In this case we define the subexpression $\prod_{k=1}^{0}(\ldots)$ equal to one.

Finally, consider the situation where the age of the customer in service equals r. This customer will leave the system, whether he finishes his service or not. If there are no customers in the waiting room at time $n+1$, the MC will make a transition to level zero. In view of Eqn. (3.5) it follows

$$E = e \otimes \prod_{k=1}^{r} (D_0 + p_{r-k} D_1)\,. \tag{3.8}$$

Otherwise the first customer in the waiting room, who has clearly not yet reached his critical age, will enter the server. Implying, for $0 \leq i < r$,

$$C_{i+1} = e\alpha \otimes \left(\prod_{k=1}^{i} (D_0 + p_{r-k} D_1) \right) (1 - p_{r-i-1})\, D_1. \tag{3.9}$$

Notice, it is also possible to write the matrices E and C_{i+1} as B_{r+1}, resp. A^r_{i+1}, since $t + \frac{a_r}{1-p_{r-1}}(e-t) = e$. The transition matrix P is a finite level dependent GI/M/1 type Markov chain, the steady state vector of which can be computed efficiently by the Latouche-Jacobs-Gaver (LJG) algorithm [60], which computes the solution of $\pi Q = 0$, where Q is an infinitesimal generator matrix with a lower block-Hessenberg form. By rewriting $\pi P = \pi$ as $\pi(P - I_{rm_{tot}+m}) = 0$, we see that $(P - I_{rm_{tot}+m})$ is such a matrix. Hence, the vector π can be calculated by applying the LJG algorithm on this matrix as illustrated in Algorithm 3.1. The time and memory complexity of this algorithm equal $O(m_{tot}^3 r^2)$ and $O(m_{tot}^2 r)$, respectively. The matrices G_s and $\Pi_{s,k}$ can all be stored in a single $m_{tot} \times (m + r m_{tot})$ matrix W by using the following method. After step 2a, we set

$$W = [\Pi_{r,0} \;\; \Pi_{r,1} \;\; \ldots \;\; \Pi_{r,r-1} \;\; (-G_r)^{-1}].$$

In the next step, we replace $\Pi_{s+1,s}$ by $(-G_s)^{-1}$ and $\Pi_{s+1,k}$ by $\Pi_{s,k}$, for $s = r-1, \ldots, 1$. After step 2b, the matrix W reads

$$W = [\Pi_{1,0} \;\; (-G_1)^{-1} \;\; \ldots \;\; (-G_{r-1})^{-1} \;\; (-G_r)^{-1}].$$

Finally, in 2c, we replace $\Pi_{1,0}$ by the $m \times m$ matrix G_0, hence,

$$W = [G_0 \;\; (-G_1)^{-1} \;\; \ldots \;\; (-G_{r-1})^{-1} \;\; (-G_r)^{-1}].$$

This matrix can now be used to compute the vector π in step 3.

Having calculated the steady state vector $\pi = (\pi_0, \pi_1, \ldots)$ of the transition matrix P, we want to determine the probability $P[Z = i]$ that a customer receives a complete service and that his response time equals i time units. This probability is given by the expected number of customers who complete their service i time units after they entered the queue, divided by the expected number of customers who leave the system at an arbitrary time instant. Denote the probability vector π_i that the MC is at level i at an arbitrary time as

$$\pi_i = ((\pi_i)_{(1,1)}, (\pi_i)_{(1,2)}, \ldots, (\pi_i)_{(1,m)}, (\pi_i)_{(2,1)}, \ldots, (\pi_i)_{(m_{ser},m)}).$$

Thus, $\sum_{j=1}^{m} (\pi_i)_{(s,j)}$ is the probability that at an arbitrary time instant an age i customer is in the service facility, with the phase of the service process equaling s. Also, $(t)_s$ is the probability that such a customer completes his service. Hence, we obtain, for $0 < i \leq r$

$$P[Z = i] = \frac{1}{\lambda} \sum_{s=1}^{m_{ser}} (t)_s \sum_{j=1}^{m} (\pi_i)_{(s,j)}. \tag{3.10}$$

Algorithm 3.1 Latouche-Jacobs-Gaver algorithm for a GI/M/1 type Markov chain

1. Input:
   - the matrices D_0 and D_1 of the arrival process,
   - the PH distribution, characterized by the vector α and the matrix T,
   - the patience distribution $\tilde{a}$.
2. (a) First, calculate the matrices E and C_i $(1 \leq i \leq r)$ by means of the equations in Section 3.2.2. Then, define
$$\begin{aligned} G_r &= C_1 - I_{m_{tot}}, \\ \Pi_{r,0} &= -(G_r)^{-1}E, \\ \Pi_{r,k} &= -(G_r)^{-1}C_{r-k+1}, \text{ for } k = 1, 2, \ldots, r-1. \end{aligned}$$
   (b) For $s = r-1, r-2 \ldots, 1$ calculate the matrices B_{s+1} and A_k^s, for $k = 0, \ldots, s$ and define:
$$\begin{aligned} G_s &= A_1^s - I_{m_{tot}} + A_0^s \Pi_{s+1,s}, \\ \Pi_{s,0} &= -(G_s)^{-1}\left(B_{s+1} + A_0^s \Pi_{s+1,0}\right), \\ \Pi_{s,k} &= -(G_s)^{-1}\left(A_{s-k+1}^s + A_0^s \Pi_{s+1,k}\right), \text{ for } k = 1, 2, \ldots, s-1. \end{aligned}$$
   (c) Finally, calculate B_1 and B_0 and define $G_0 = B_1 - I_m + B_0\Pi_{1,0}$.
3. The steady state vector $\pi = (\pi_0, \pi_1, \ldots, \pi_r)$ can be computed by means of the following relations, where π_0 is a $1 \times m$ vector and π_i $(0 < i \leq r)$ is a $1 \times m_{tot}$ vector.
$$\begin{aligned} \pi_0 G_0 &= 0, \\ \pi_1 &= \pi_0 B_0 (-G_1)^{-1}, \\ \pi_s &= \pi_{s-1} A_0^{s-1} (-G_s)^{-1}, \end{aligned}$$
   for $s = 2, \ldots, r$.
4. Finally, one normalizes π, such that $\pi e = 1$.

The rejection probability P_{out}, being the probability that a customer leaves the system without entering or completing his service, reads

$$P_{out} = 1 - \sum_{i=1}^{r} P[Z = i]. \tag{3.11}$$

Although we focus mainly on the response time, we can also obtain the queue length distribution as a by-product. To do so, we define the $m \times 1$ vector $h_{k,d}$ as the probability that in an interval of length d, k customers arrive, who are still in the queue at the end of this interval. The j-th entry of this vector $h_{k,d}$ represents the case in which the state of the D-MAP process equals j at the start of the interval. The following relation can be used to compute these probability vectors:

$$h_{k,0} = 1_{\{k=0\}} e, \tag{3.12}$$

$$h_{0,d} = \left(\prod_{i=1}^{d}(D_0 + p_{d-i}D_1)\right)e, \tag{3.13}$$

$$h_{k,d} = (1-p_{d-1})D_1 h_{k-1,d-1} + (D_0 + p_{d-1}D_1)h_{k,d-1}. \tag{3.14}$$

Denote $P[Q = q]$ as the probability that there are q customers in the waiting room at an arbitrary time instant. This situation is only possible when a customer with an age of at least $q+1$ time units is in the service facility. Therefore, we have

$$P[Q=q] = \sum_{i=q+1}^{r}\sum_{j=1}^{m}(h_{q,i-1})_j \sum_{s=1}^{m_{ser}}(\pi_i)_{(s,j)}, \text{ for } 0 < q < r, \tag{3.15}$$

$$P[Q=0] = \sum_{j=1}^{m}(\pi_0)_j + \sum_{i=1}^{r}\sum_{j=1}^{m}(h_{0,i-1})_j \sum_{s=1}^{m_{ser}}(\pi_i)_{(s,j)}. \tag{3.16}$$

A similar method can be used to obtain the queue length distribution of the systems discussed in Sections 3.2.3 and 3.2.4.

Remark 1. In the special case that the service time distribution is geometric with parameter μ, we immediately have

$$(1 - P_{out}) = \frac{\mu}{\lambda}(1 - P_{idle}), \tag{3.17}$$

where $P_{idle} = 1 - P_{busy}$ is the probability that the server is idle. We can rewrite this as

$$P_{busy} = \frac{\lambda}{\mu}(1 - P_{out}). \tag{3.18}$$

This last equation seems remarkable since one might incorrectly conclude that the server is only busy serving customers who are successful. For more details about this somewhat unexpected result, we refer to the remark on Page 46.

3.2.3 Service Time Aware Impatient Customers

In the system studied in the previous section a customer who does not complete his service before reaching his critical age, is still allowed to enter the server. In this way, such a customer wastes some service capacity which could have been used by other customers. In this section we consider a system where customers are only allowed to enter the server, if they manage to compete service before reaching their critical age. As such, we might refer to this model as the "limitation on sojourn time with service time aware customers", as customers are still impatient during the entire sojourn time, but are aware of their service time. Therefore they can decide to leave the system, when the service facility becomes available to them, if they notice that their remaining amount of patience does not suffice to complete service. As long as we are concerned with the rejection probability or the response time (of successful customers), this system is equivalent to the "limitation on sojourn time with aware customers" [9]. In such a system, customers are assumed to be aware of their required sojourn time upon arrival to the queue and immediately leave if their amount of patience does not suffice. The queue length distribution is however somewhat different as aware customers will often not enter the waiting line at all, whereas in our case some unsuccessful customers still spend some time waiting in the queue.

We can construct a similar MC as in the previous section and denote its $(m + rm_{tot}) \times (m + rm_{tot})$ transition matrix as

$$\hat{P} = \begin{bmatrix} \hat{B}_1 & \hat{B}_0 & 0 & 0 & \cdots & 0 & 0 \\ \hat{B}_2 & \hat{A}_1^1 & \hat{A}_0^1 & 0 & \cdots & 0 & 0 \\ \hat{B}_3 & \hat{A}_2^2 & \hat{A}_1^2 & \hat{A}_0^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hat{B}_{r-1} & \hat{A}_{r-2}^{r-2} & \hat{A}_{r-3}^{r-2} & \hat{A}_{r-4}^{r-2} & \cdots & \hat{A}_0^{r-2} & 0 \\ \hat{B}_r & \hat{A}_{r-1}^{r-1} & \hat{A}_{r-2}^{r-1} & \hat{A}_{r-3}^{r-1} & \cdots & \hat{A}_1^{r-1} & \hat{A}_0^{r-1} \\ \hat{E} & \hat{C}_r & \hat{C}_{r-1} & \hat{C}_{r-2} & \cdots & \hat{C}_2 & \hat{C}_1 \end{bmatrix}.$$

Let n be the current time instant and assume some customer c leaves the server at time n. Then, the first customer that arrived after customer c at some time $n-k$ $(0 \leq k < r)$ does not enter the server at time n if he has already reached his critical age or if his remaining patience is not sufficient to be served entirely. This probability is given by

$$q_k = p_k + \sum_{i=k+1}^{r} a_i \alpha T^{i-k} e. \tag{3.19}$$

Denote the probability that a customer of age k is being served at time n, with the phase of the service process equal to s as the s-th component of the $m_{ser} \times 1$ vector v^k. This can only be the case when the sum of his age and his remaining service time does not exceed his critical age, hence,

$$(v^k)_s \;=\; \sum_{i=k}^{r} a_i \sum_{j=0}^{i-k} (T^j t)_s = 1 - p_{k-1} - \sum_{i=k}^{r} a_i (T^{i-k+1} e)_s, \tag{3.20}$$

using the relation $t = e - Te$. Hence,

$$v^k = e - p_{k-1} e - \sum_{i=k}^{r} a_i T^{i-k+1} e. \tag{3.21}$$

Let u^k, a column vector of dimension m_{ser}, represent the probability that a customer of age k completes his service. Since this customer was admitted by the server, we know that his critical age is at least as much as the sum of his age (k) and his remaining service time. Thus, for $1 \leq k \leq r$ and for $1 \leq s \leq m_{ser}$ we have,

$$(u^k)_s = \frac{a_k (t)_s + a_{k+1} (t)_s + \ldots + a_r (t)_s}{(v^k)_s} = \frac{(1 - p_{k-1})(t)_s}{(v^k)_s}. \tag{3.22}$$

Consider an age k customer in phase s of the service process at time n. The probability that this customer continues his service in phase s' is represented by the (s, s')-th entry of the $m_{ser} \times m_{ser}$ matrix U^k. Notice, since the customer was admitted in the server, he only leaves the system when his service finishes. Hence, this probability is given by

$$(U^k)_{ss'} = \frac{\sum_{h=k+1}^{r} a_h (T)_{ss'} \sum_{l=1}^{h-k} (T^{l-1} t)_{s'}}{(v^k)_s} = \frac{(T)_{ss'} (v^{k+1})_{s'}}{(v^k)_s}, \tag{3.23}$$

for $1 \leq k \leq r$ and $1 \leq s, s' \leq m_{ser}$.

We are now in a position to set up an equation for each of the transition blocks appearing in the matrix $\hat{P}$. For the matrices on level zero, the only difference with the system discussed in Section 3.2.2 is that an arriving customer who finds the server idle, does not necessarily enter the service facility. If he has not enough patience to stay in the system until his service has completed, he abandons the system. This leads to

$$\hat{B}_1 = D_0 + q_0 D_1, \tag{3.24}$$

$$\hat{B}_0 = (1 - q_0) D_1. \tag{3.25}$$

The MC makes a transition from level i to level zero if the customer in service completes his service and there is no customer who enters the server. Note that, due to the definition of the levels of the MC, the age of the customer that completes his service equals i time units. For $0 < i < r$, the expression of the $m_{tot} \times m$ matrix $\hat{B}_{i+1}$ is given by the following equation:

$$\hat{B}_{i+1} = u^i \otimes \prod_{k=1}^{i} (D_0 + q_{i-k} D_1). \tag{3.26}$$

Remember, the continuation and completion of the service of a customer with an age equal to k time units, is given by the matrix U^k and the vector u^k, respectively. Arguments similar to those presented in Section 3.2.2 yield the following expressions:

$$\hat{A}_0^i = U^i \otimes I_m, \tag{3.27}$$

$$\hat{A}_{l+1}^i = u^i \alpha \otimes \left(\prod_{k=1}^{l} (D_0 + q_{i-k} D_1) \right) (1 - q_{i-l-1}) D_1, \tag{3.28}$$

$$\hat{E} = e \otimes \prod_{k=1}^{r} (D_0 + q_{r-k} D_1), \tag{3.29}$$

$$\hat{C}_{i+1} = e\alpha \otimes \left(\prod_{k=1}^{i} (D_0 + q_{r-k} D_1) \right) (1 - q_{r-i-1}) D_1. \tag{3.30}$$

As the transition matrices P and $\hat{P}$ have the same form, we can make use of the LJG algorithm (Algorithm 3.1) to obtain the steady state vector $\hat{\pi}$ of $\hat{P}$. An expression for the probability that a customer receives a complete service and has a response time of i time units is then established as

$$P[\hat{Z} = i] = \frac{1}{\lambda} \sum_{s=1}^{m_{ser}} (u^i)_s \sum_{j=1}^{m} (\hat{\pi}_i)_{(s,j)}, \tag{3.31}$$

for $0 < i \leq r$. The rejection probability $\hat{P}_{out}$, being the probability that a customer abandons the system without entering the service facility, is given by

$$\hat{P}_{out} = 1 - \sum_{i=1}^{r} P[\hat{Z} = i]. \tag{3.32}$$

3.2.4 Impatient Customers in the Waiting Room

In Sections 3.2.2 and 3.2.3, we considered customers who remain impatient even when they have already entered the service facility. In this section we assume that a customer is no

longer impatient during his sojourn time, but only while waiting. Once a customer enters the server, he remains there until his service is completed. When focusing on the rejection probabilities or on the response time (of successful customers), this system coincides with both the "limitation on waiting time with aware or unaware customers" [9] as in neither system abandoning customers use any service capacity. We assume that a customer who reaches his critical age at the exact moment that the server becomes available to him, leaves the system.

If we define an MC analogue to the previous sections, the corresponding transition matrix P' is either finite or infinite, depending on the service time distribution. For unbounded service times, we get an infinite matrix P' as there is no limit on the age of the customer in service. In order to find the steady state vector of P', we introduce a new MC with an $(m + rm_{tot}) \times (m + rm_{tot})$ transition matrix $\bar{P}$, on which we can apply the LJG algorithm to find its steady state vector. The transition matrix $\bar{P}$ corresponds to the MC that we get when censoring the MC P' on the set of states for which the age of the customer in service is at most r:

$$\bar{P} = \begin{bmatrix} B_1 & B_0 & 0 & 0 & \cdots & 0 & 0 \\ \bar{B}_2 & \bar{A}_1^1 & \bar{A}_0^1 & 0 & \cdots & 0 & 0 \\ \bar{B}_3 & \bar{A}_2^2 & \bar{A}_1^2 & \bar{A}_0^2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \bar{B}_{r-1} & \bar{A}_{r-2}^{r-2} & \bar{A}_{r-3}^{r-2} & \bar{A}_{r-4}^{r-2} & \cdots & \bar{A}_0^{r-2} & 0 \\ \bar{B}_r & \bar{A}_{r-1}^{r-1} & \bar{A}_{r-2}^{r-1} & \bar{A}_{r-3}^{r-1} & \cdots & \bar{A}_1^{r-1} & \bar{A}_0^{r-1} \\ \bar{E} & \bar{C}_r & \bar{C}_{r-1} & \bar{C}_{r-2} & \cdots & \bar{C}_2 & \bar{C}_1 \end{bmatrix}.$$

Notice, the matrices corresponding to level zero are identical to those in Section 3.2.2, as the distinction between this model and the one in Section 3.2.2 lies only in the customer behavior inside the service facility. The expressions for the matrices, representing a transition from level i $(0 < i < r)$ are altered, because the probability that a customer leaves (remains in) the server no longer depends upon the age of this customer. However apart from the vector and the matrix holding the probabilities that a service completion and continuation occurs, respectively, the system evolves identical to the one in Section 3.2.2:

$$\bar{B}_{i+1} = t \otimes \prod_{k=1}^{i} \left(D_0 + p_{i-k} D_1\right), \tag{3.33}$$

$$\bar{A}_0^i = T \otimes I_m, \tag{3.34}$$

$$\bar{A}_{l+1}^i = t\alpha \otimes \left(\prod_{k=1}^{l} \left(D_0 + p_{i-k} D_1\right) \right) \left(1 - p_{i-l-1}\right) D_1, \tag{3.35}$$

for $1 \leq i < r$ and $0 \leq l < i$. Next, let us derive an expression for the matrices at level r. The matrix $\bar{E}$ contains the probabilities that the MC makes a transition from state (s_1, j_1) at level r to state j_2 at level zero. Consider a customer c of age r who is being served at time n. Customers present in the service facility are patient and remain in the system until their service is completed. Thus, with probability $(T^k t)_{s_1}$, customer c will remain in the server for another k time units, meaning the next time instant observed by the censored chain $\bar{P}$ is time $n + k + 1$. When customer c leaves the server at time $n + k$, the next customer, if present, can start his service. Because no customer has a critical age of more than r time units, customers who arrived at some time $n - r + i$, with $0 < i \leq k$, will leave the waiting room before or at time $n + k$. In order to have a transition to level zero, some arrivals may occur at one of the

time instants $n-r+k+1,\ldots,n+k-1$, as long as these customers reach their critical age before or at time $n+k$. This probability is given by $\prod_{h=1}^{r-1}(D_0+p_{r-h}D_1)$ and therefore the probability of having an empty waiting room at time $n+k+1$ equals $D^k\prod_{h=1}^{r}(D_0+p_{r-h}D_1)$. Recall, a customer who reaches his critical age on the exact moment that the server becomes available, leaves the system nevertheless. This leads to

$$\bar{E}=\left(\sum_{k=0}^{\infty}(T^kt)\otimes D^k\right)\prod_{h=1}^{r}(D_0+p_{r-h}D_1)\,. \tag{3.36}$$

The MC makes a transition to level $r-i$ from level r if there are no arrivals at time $n-r+k+1,\ldots,n-r+k+i$ of customers who are still in the waiting room at time $n+k$ and a customer c' with a critical age larger than $r-i-1$ arrives at time $n-r+k+i+1$. If the customer in service leaves the system, customer c' enters the server and will remain there until his service is completed. For $0\le i<r$, we get:

$$\bar{C}_{i+1}=\left(\sum_{k=0}^{\infty}(T^kt)\otimes D^k\right)\left(\alpha\otimes\left(\prod_{h=1}^{i}(D_0+p_{r-h}D_1)\right)(1-p_{r-i-1})\,D_1\right). \tag{3.37}$$

If the service time is unbounded, an infinite sum occurs in both Eqn. (3.36) and (3.37). This sum can be truncated after k' terms if

$$\sum_{k=k'}^{\infty}(T^kt)\otimes D^k<\epsilon, \tag{3.38}$$

for some ϵ small. Because T, resp. D, is a substochastic, resp. stochastic matrix, we can always find such a k'.

Algorithm 3.1 can be used to find the steady state vector $\bar{\pi}=(\bar{\pi}_0,\bar{\pi}_1,\ldots,\bar{\pi}_r)$ of $\bar{P}$. The steady state vector $\pi'=(\pi'_0,\pi'_1,\ldots)$ of P' obeys the following relation:

$$\pi'_i = \bar{\pi}_i/c, \tag{3.39}$$
$$\pi'_{r+j} = \bar{\pi}_r(T^j\otimes I_m)/c, \tag{3.40}$$

for $0\le i<r$ and $j\ge 0$. The normalization constant c can be calculated by

$$c=\sum_{i=0}^{r-1}\bar{\pi}_i e+\bar{\pi}_r\big((I_{m_{ser}}-T)^{-1}\otimes I_m\big)e. \tag{3.41}$$

As before, the probability that a customer receives a complete service and has a response time of i time units is found as

$$P[Z'=i]=\frac{1}{\lambda}\sum_{s=1}^{m_{ser}}(t)_s\sum_{j=1}^{m}(\pi'_i)_{(s,j)}, \tag{3.42}$$

and the rejection probability equals

$$P'_{out}=1-\sum_{i>0}P[Z'=i]. \tag{3.43}$$

Remark 2. In this case one easily obtains the relation

$$1 - P'_{idle} = \frac{\lambda}{\mu}(1 - P'_{out}), \tag{3.44}$$

as $\sum_{i,j}(\pi'_i)_{(s,j)} = \beta_s$, for $i > 0$, and where $\beta(T + t\alpha) = \beta$, $\beta e = 1$ and $\beta t = 1/\mu$. This result is also intuitively clear as the server only serves successful customers and the mean service time of such a customer is $1/\mu$.

3.2.5 Quasi-Birth-Death Reduction

In the previous three sections, we constructed a level dependent GI/M/1 type Markov chain to compute the response time distribution of the D-MAP/PH/1+GI queue. A limitation of this method lies in the fact that the time complexity of the algorithm is a square function of r, the maximum tolerable customer patience. In this section, we introduce a different approach, using QBDs, which speeds up the computational process. The idea behind this reduction was also applied in [109] to calculate the response time distribution in a queue with multiple types of customers (i.e., the MMAP[K]/PH[K]/1 queue).

In order to construct the level dependent QBD we add m states to each level i, for $0 < i < r$, to the state space of the transition matrix of interest (P, $\hat{P}$ or $\bar{P}$). These additional states are referred to as artificial states. The basic idea behind this construction is to replace a transition from level i to $i - k$, for $k \geq 0$, by $k + 1$ transitions, where for each of the first k transitions we decrease the level by one, while for the $(k + 1)^{st}$ transition the level remains identical. Thus, instead of making a transition from level i to $i - k$ at once, the new MC will visit k intermediate states, which shall all be artificial states. Actually, for $i - k = 0$ we split the transition into k steps instead of $k + 1$. We will only need m artificial states due to the specific form of the matrices that correspond to transitions causing the level to decrease. In the worst case, i.e., when $m_{ser} = 1$, this method requires four times as much memory as the one discussed in the previous sections. However, the QBD method offers a significant gain in computation time, as will be discussed later in this section.

Roughly speaking, we can explain the idea behind the QBD reduction as follows. A transition from level i to $i - k < i$ can be regarded as *scanning* the $k + 1$ time instants $n-i+1, \ldots, n-i+k+1$ for a customer arrival who is still present in the system at time $n+1$ (where n is the current time instant). The first k of these scans results in a negative result, whereas the last one is successful (unless $i - k = 0$). Whether the l-th scan is successful, is solely determined by the D-MAP state at time $n - i + l$, say state j_l, and the amount of patience $i - l + 1$ needed to be in the system at time $n + 1$.

Thus, instead of going directly from some state (s, j_1) of level i to a state of the form (s', j') of level $i - k$, we make a series of $k+1$ transitions: the first one to artificial state j_2 of level $i - 1$, the l-th, for $l = 2, \ldots, k$, going from artificial state j_l of level $i - l + 1$ to state j_{l+1} of level $i - l$ and the last one from artificial state j_{k+1} of level $i - k$ to state (s', j') of level $i - k$. We do not need to keep track the phase s' as this is determined by the vector α. If $i - k = 0$, we have only k transitions, the k-th one going from artificial state j_k at level one to state j_{k+1} at level zero.

We only discuss the system presented in Section 3.2.2 in detail, the QBD Markov chain for the other two systems can be set up in an analogue way. First, we add m additional states to every level, except the first and the last one (level zero and level r). As explained above,

these states will correspond to those of the D-MAP arrival process. Define

$$\begin{aligned}\Omega_i &= \{j \mid 1 \leq j \leq m\}, \quad 0 \leq i < r,\\ \Psi_i &= \{(s,j) \mid 1 \leq s \leq m_{ser}, 1 \leq j \leq m\}, \quad 0 < i \leq r.\end{aligned}$$

Using these definitions, the states of level zero are denoted as Ω_0, those of level i $(0 < i < r)$ as $\Omega_i \cup \Psi_i$ and the states of level r are denoted as Ψ_r. The states of Ω_0 and Ψ_i $(0 < i \leq r)$ correspond to those of the Markov chain presented in Section 3.2.2. The states of Ω_i $(0 < i < r)$ are those that have been added, which we call the artificial states. For our convenience, let us call the other states, original states and let us define

$$m_{art} = m_{tot} + m. \tag{3.45}$$

Define the QBD using the following $rm_{art} \times rm_{art}$ matrix P^*:

$$P^* = \begin{bmatrix} B_1^* & B_0^* & 0 & 0 & \cdots & 0 & 0 \\ B_2^* & A_1^{1^*} & A_0^{1^*} & 0 & \cdots & 0 & 0 \\ 0 & A_2^{2^*} & A_1^{2^*} & A_0^{2^*} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_2^{r-1^*} & A_1^{r-1^*} & C_0^* \\ 0 & 0 & 0 & \cdots & 0 & C_2^* & C_1^* \end{bmatrix},$$

where B_1^* is an $m \times m$ matrix, B_2^* an $m_{art} \times m$ matrix, B_0^* an $m \times m_{art}$ matrix and $A_0^{i^*}$, $A_1^{i^*}$ $(0 < i < a)$ and $A_2^{i^*}$ are $m_{art} \times m_{art}$ matrices. Also, the matrix C_0^* is an $m_{art} \times m_{tot}$ matrix, C_1^* an $m_{tot} \times m_{tot}$ and C_2^* is an $m_{tot} \times m_{art}$ matrix.

The matrices B_1^* and B_0^* represent the situation in which the QBD is at level zero, that is, the system is idle. As the transitions from level 0 are unaltered, the only difference with the expressions for B_1 and B_0 is caused by the presence of the artificial states added to level one, therefore

$$B_1^* = B_1 \tag{3.46}$$

$$B_0^* = \begin{bmatrix} 0 & B_0 \end{bmatrix}. \tag{3.47}$$

The expression for the $m_{art} \times m_{tot}$ matrix B_2^* can be divided into two parts. The first m rows correspond to the situation in which the QBD is in an artificial state of level one at time n. Notice, we are referring to the time epochs of the QBD Markov chain. If no customer arrives at this moment, the QBD makes a transition to level zero. These probabilities are, by definition, given by the matrix D_0. The other rows represent the situation in which the age of the customer in service equals one time unit, meaning that the QBD is in an original state of level one at time n. In this case, the QBD makes a transition to level zero if this customer leaves the server and there is no arrival at the current time instant. Notice, the waiting room is empty because the customer in service arrived at time $n-1$. Hence,

$$B_2^* = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} = \begin{bmatrix} D_0 \\ (t + a_1(e-t)) \otimes D_0 \end{bmatrix}. \tag{3.48}$$

The same arguments can be used to determine an expression for the matrices $A_2^{i^*}$ $(0 < i < r)$, which contain the probabilities of a transition from level i to level $i-1$. In order to have such a transition, there can be no arrival $i-1$ time units ago of a customer with a critical

age greater than or equal to i time units (that is, the *scan* of time instant $n-i+1$ ought to be unsuccessful). When the QBD makes a transition to a lower level, the resulting state is always an artificial state. Therefore, all the entries in the last m_{tot} columns are equal to zero and we get,

$$A_2^{i^*} = \begin{bmatrix} D_0 + p_{i-1}D_1 & 0 \\ \left(t + \frac{a_i}{1-p_{i-1}}(e-t)\right) \otimes (D_0 + p_{i-1}D_1) & 0 \end{bmatrix}. \tag{3.49}$$

The transitions to a higher level are identical to those in Section 3.2.2. Moreover, we never increase the level from an artificial state, implying

$$A_0^{i^*} = \begin{bmatrix} 0 & 0 \\ 0 & A_0^i \end{bmatrix}, \tag{3.50}$$

$$C_0^* = \begin{bmatrix} 0 \\ A_0^{r-1} \end{bmatrix}. \tag{3.51}$$

The probabilities of a transition between two original states of level i, are given by the matrix A_1^i, for $0 < i < r$ and by C_1 for $i = r$. Apart from the transitions between original states, a transition from an artificial to an original state at level i can occur when a customer with a critical age of at least i time units arrives (that is, the *scan* at time $n-i+1$ is a success). This yields,

$$A_1^{i^*} = \begin{bmatrix} 0 & (1-p_{i-1})\,\alpha \otimes D_1 \\ 0 & \left(t + \frac{a_i}{1-p_{i-1}}(e-t)\right) \alpha \otimes (1-p_{i-1})\,D_1 \end{bmatrix}, \tag{3.52}$$

$$C_1^* = e\alpha \otimes a_r D_1. \tag{3.53}$$

Finally, the probabilities that the QBD makes a transition from level r to level $r-1$, can be found in the $m_{tot} \times m_{art}$ matrix C_2^*. The customer in service leaves the system, whether he finishes his service or not. There is a transition to level $r-1$ if no customer of age r will be in the queue at the next time instant. Hence,

$$\tilde{C}_2^* = \begin{bmatrix} e \otimes (D_0 + (1-a_r)D_1) & 0 \end{bmatrix}. \tag{3.54}$$

The matrix P^* is a finite level-dependent QBD matrix, therefore, its steady vector $\pi^* = (\pi_0^*, \pi_1^*, \ldots, \pi_r^*)$ can be computed by a variant on the LJG algorithm, described in [35]. That is, the matrix $Q^* = P^* - I$ is a tridiagonal infinitesimal generator block matrix. Thus, to obtain the steady state distribution, we can apply the algorithm presented in [35], the time and space complexity of which equal $O(rm_{art}^3)$ and $O(rm_{art}^2)$. From π^* we can then extract the steady state vector π of the original transition matrix P as explained in step 4 of Algorithm 3.2. Having found π we can apply the formulas given at the end of Section 3.2.2 to obtain the performance measures of interest.

We can use a single $m_{art} \times rm_{art}$ matrix to save the intermediate results, similar to the procedure given by Algorithm 3.1. Let us now compare the time and memory complexity of this algorithm with those of Algorithm 3.1, based on the GI/M/1 Markov chain. The QBD approach requires slightly more memory, due to the addition of the artificial states. As only m states are added per level, m_{art} is at most twice as large as m_{tot} (if $m_{ser} = 1$), the QBD requires at most 4 times as much memory. The advantage however, lies in the difference in the time complexity. The time needed by the algorithm using QBDs is only linear in the maximum critical age r of a customer, whereas with the GI/M/1 type MC it is a square function of r.

Algorithm 3.2 Computing the steady state vector π of P via the QBD MC P^*

1. Input:
   - the matrices D_0 and D_1 of the arrival process,
   - the PH distribution, characterized by α and T,
   - the patience distribution $\tilde{a}$.
2. Calculate the matrices G_i, for $0 \leq i \leq r$, by means of the equations:

$$\begin{aligned} G_0 &= B_1^* - I_m, \\ G_1 &= A_1^{1^*} - I_{m_{art}} + B_2^*(-G_0^{-1})B_0^*, \\ G_i &= A_1^{i^*} - I_{m_{art}} + A_2^{i^*}(-G_{i-1}^{-1})A_0^{i-1^*} \quad (\text{for } 1 < i < r), \\ G_r &= C_1^* - I_{m_{tot}} + C_2^*(-G_{r-1}^{-1})C_0^* \end{aligned}$$

3. The steady state probabilities are found using the following expressions:

$$\begin{aligned} \pi_r^* G_r &= 0, \text{ with } \pi_r^* e = 1, \\ \pi_{r-1}^* &= \pi_r^* C_2^*(-G_{r-1}^{-1}), \\ \pi_i^* &= \pi_{i+1}^* A_2^{i+1^*}(-G_i^{-1}), \text{ for } i = r-2, \ldots, 1, \\ \pi_0^* &= \pi_1^* B_2^*(-G_0^{-1}), \end{aligned}$$

   and $\sum_{i=0}^{r} \pi_i^* e = 1$.

4. Finally, compute the steady state vector π of the original transition matrix P as follows. Denote π_i^* $(0 < i < r)$ as $(\pi_i^*(m), \pi_i^*(m_{tot}))$, where the probabilities of the row vector $\pi_i^*(m)$ of size m correspond to the artificial states and those of the $1 \times m_{tot}$ vector $\pi_i^*(m_{tot})$ to the original states. Then, the steady state vector π can be computed by

$$\begin{aligned} \pi_0 &= \pi_0^*/(1-d), \\ \pi_i &= \pi_i^*(m_{tot})/(1-d), \text{ for } 1 \leq i < r, \\ \pi_r &= \pi_r^*/(1-d), \end{aligned}$$

   where $d = \sum_{i=1}^{r-1} \pi_i^*(m)e$.

3.2.6 Numerical Examples

In this section, we discuss some fairly arbitrary numerical examples of systems with impatient customers which provide, among others, some insight on the influence of the patience distribution X. The D-MAP arrival process is characterized by the following two matrices:

$$D_0 = \begin{bmatrix} 0.76 & 0.19 \\ 0.09 & 0.81 \end{bmatrix} \qquad \text{and} \qquad D_1 = \begin{bmatrix} 0.04 & 0.01 \\ 0.01 & 0.09 \end{bmatrix}.$$

This arrival process has a mean arrival rate $\lambda = 1/12$. The service time of a customer consists of two components. Every customer needs an initial service time of 3 time units. On top of that, with probability 0.4, 0.4 and 0.2 a customers needs some extra, geometrically distributed, service with an average of 5, 10 and 20 time units, respectively. We have $\alpha = (1, 0, 0, 0, 0, 0)$ and

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0.4 & 0.2 \\ 0 & 0 & 0 & 0.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.95 \end{bmatrix}.$$

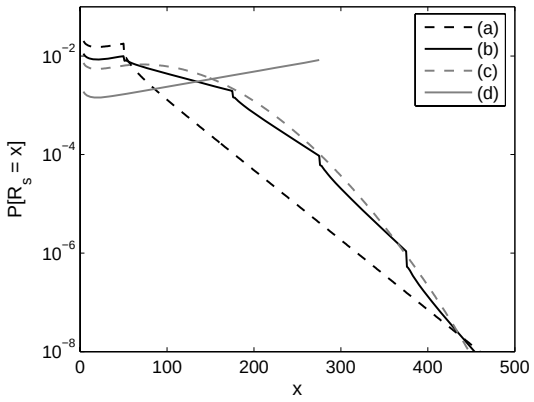


Figure 3.1: Response time distribution of successful impatient customers

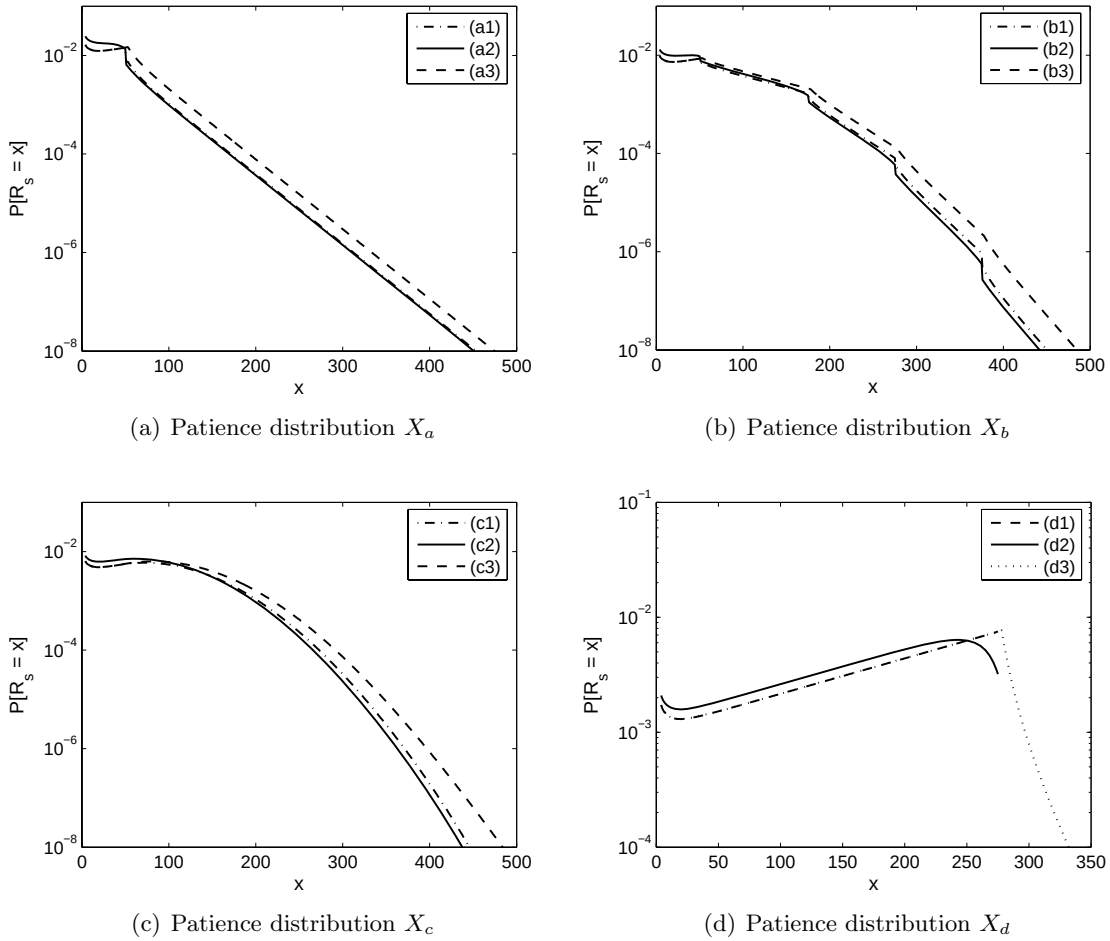
The mean service time $1/\mu$ equals 13 time units, meaning without the customer impatience we would have an unstable system as $\lambda/\mu = 13/12 > 1$. We consider 4 possible patience distributions X, each having the same mean $E[X]$ of 275 time units:

X_a: One half of the customers runs out of patience after 50 time units, whereas the other half will be patient for 500 time units, i.e., $a_{50} = a_{500} = 0.5$.

X_b: The critical age of a customer equals 50, 175, 275, 375 or 500 time units, each with a probability 0.2.

X_c: The patience distribution is uniformly distributed between 50 and 500 time units.

X_d: All customers have the same amount of patience, being 275 time units.



(a) Patience distribution X_a

(b) Patience distribution X_b

(c) Patience distribution X_c

(d) Patience distribution X_d

Figure 3.2: Response time distributions of successful service time (un)aware customers that are impatient in the entire system or only in the waiting room

For each of the patience distributions considered, Figure 3.1 shows us the response time distribution R_s of a successful customer, who remains impatient in the entire system (i.e., Section 3.2.2). The behavior of these curves can be understood as follows. When considering the probability that a successful customer has a response time of x time units, one might intuitively say that the load of the system

$$\rho_x \approx \frac{\lambda}{\mu} P[X \geq x]. \tag{3.55}$$

Notice, ρ_x decreases as a function of x and equals 13/12 for $x = 1$. As such, we expect that the curves in Figure 3.1 start to decrease as soon as ρ_x drops below 1. Moreover, the smaller ρ_x becomes the sharper the decrease. For the distributions X_a and X_b, the value $P[X \geq x]$ decreases only in a few steps, which correspond to the different 'stairs' in those curves. The curve corresponding to X_c reaches a maximum at $x = 72$, even though $\rho_{72} \approx 1.03 > 1$. This might be explained by the fact that the mean time that a customer spends in the server is less than $1/\mu$, due to the impatience (i.e., ρ_x slightly overestimates the load).

distribution	impatient in server	server time aware	patient in server
X_a	0.1885	0.1437	0.1673
X_b	0.1415	0.1286	0.1324
X_c	0.1192	0.1047	0.1144
X_d	0.0918	0.0334	0.0873

Figure 3.3: Expected number of lost customers

In Figure 3.2, we compare the distribution R_s of the three systems discussed in Sections 3.2.2-3.2.4, for each of the patience distributions considered. The system with impatient customers in the entire system is represented by a dash-dotted line (Section 3.2.2), service time aware customers correspond to a full line (Section 3.2.3), whereas the situation in which the customers are only impatient in the waiting room is given by a dashed line (Section 3.2.4). The computation times for the curves in this figure, using an AMD Athlon 2.0 GHz processor with 512 Mb of memory, are about 173 seconds for the method using the GI/M/1 type Markov chains and about 21 seconds when applying the QBD approach (see Section 3.2.5). Because the maximum critical age of a customer equals either 275 or 500 time units and the number of states of the arrival process is relatively small, the method using QBDs takes remarkably less time while the memory requirements only increase with a factor 1.36.

distribution	impatient in server	rejected by server	patient in server
X_a	38.9042	34.4974	46.1523
X_b	73.4010	68.1397	80.1541
X_c	90.4167	83.9073	98.0788
X_d	179.8562	171.7786	189.5891

Figure 3.4: Expected response time of a successful impatient customer

Figure 3.3 represents the probabilities that a customer abandons the system without receiving a complete service. If we compare the four patience distributions, the probability that a customer leaves the system early decreases together with the variance of the patience distribution X, while the average response time of a customer (given by Figure 3.4) increases. Whether becoming patient while entering the server, reduces the rejection probability (as is the case in our numerical example) depends to a great extent on variation of the service time distribution. In [114] a numerical example, where all customers have the same amount of patience a, illustrates that for small variations in the service time, the system with a limited waiting time performs better than the system with a limited sojourn time, and this with relation to the rejection probability. For larger variations in the service time the opposite result is obtained, that is, in those cases it is more efficient to drop the customers who require more processing. Intuitively, this can be explained as follows. Consider a system in which the service time is deterministic. In this situation, finishing the service of a customer that has already been processed for some time requires less time than completely serving a new customer. This is not necessarily the case when the service time is highly variable, since the required processing time of the next customer might be significantly less than the remaining service time of the customer being served at this moment. Then, it is more efficient to drop the customer in service when he reaches his maximum amount of patience. In the next section we will address the question which patience distribution results in the lowest rejection probability P_{out} of all patience distributions X with a fixed mean $E[X] = m_u$.

3.3 Minimizing the Abandonment Probability

From the numerical results presented in the previous section, we could observe that a patience distribution with a higher variance resulted in a higher abandonment probability. To determine the influence the patience distribution has on the number of abandonments, we will consider the Geo/Geo/1+GI queue with impatient customers. Obviously, this queue is less general than the D-MAP/PH/1+GI queue discussed in Section 3.2, however, this allows us to prove some interesting results about customer impatience. We show that systems with a smaller patience distribution X in the convex-ordering sense, give rise to fewer abandonments (due to impatience), irrespective of whether customers become patient when entering the service facility. As a consequence, given a fixed amount of patience m, the patience distribution with mean m that achieves the lowest probability of abandonment is the deterministic distribution.

We prove the discrete-time counterpart of the M/M/1+GI result mentioned in Section 3.1 as well as the more difficult system where customers remain impatient after entering the service facility, that is, the Geo/Geo/1+GI queue with a limited waiting or sojourn time. In the latter case customers are assumed to be unaware of the required sojourn time upon arrival and may therefore receive partial service (and waste some of the service capacity). Furthermore, we demonstrate that the number of abandonments in a single-server queue with impatient customers and geometric service times, i.e., the $\cdot$/Geo/1+GI queue, is higher for systems with a limited sojourn time as opposed to a limited waiting time.

Let X be a general, discrete patience distribution on the nonnegative integers and assume without loss of generality that $P[X = 0] = 0$. Further assume that for each X there exists some $r \geq 0$ sufficiently large, such that $P[X > r] = 0$, i.e., the maximum amount of patience that a customer can have is bounded above by some constant r (notice, we do not assume that a single r exists for all distributions X, but given a particular X such an r can be found). The results presented here can be generalized to include distributions for which such an r does not exist ($P[X = \infty] = 0$ for such distributions as $m = E[X]$ is finite, implying that the Geo/Geo/1+X queue is stable [9]). In Sections 3.3.1 and 3.3.2 we show that the smaller distributions (in the convex-ordering sense) induce a lower probability of abandonment in a Geo/Geo/1+GI system with a limited sojourn and waiting time, respectively. The lower number of abandonments achieved by $\cdot$/Geo/1+GI systems with limited waiting times, as opposed to limited sojourn times, is proved in Section 3.3.3.

3.3.1 The Geo/Geo/1 Queue with a Limited Sojourn Time

Consider the Geo/Geo/1 queue with

$$P[arrival] = \alpha, \tag{3.56}$$
$$P[departure] = \beta, \tag{3.57}$$

and with impatient customers. As in Section 3.2.2 we assume that customers are impatient irrespective of whether they are being served or not. Let X represent the patience distribution of the customers and denote $P[X = k]$ as $a_k(X)$ and $P[X \leq k]$ as $p_k(X)$. To simplify the notation, we shall write

$$x_k(X) = (1 - p_k(X))\alpha, \tag{3.58}$$
$$y_k(X) = \beta + \frac{a_k(X)}{1 - p_{k-1}(X)}(1 - \beta) = \frac{a_k(X) + (1 - p_k(X))\beta}{1 - p_{k-1}(X)}, \tag{3.59}$$

for $k = 0, \ldots, r$. When there is no ambiguity as to which distribution X is meant, we will drop the X to simplify the notation. Similarly we will add an X (or Y) to any other variable when there is a need to clarify the patience distribution at hand. Notice, x_k is the arrival rate (probability) of customers whose patience is more than k, while y_k is the probability that a customer with age k leaves the server (due to either impatience or a service completion) given that an age k customer occupied the server. Let A_n be the age of the customer in service just prior to time n, where A_n is said to be zero if the system is idle. As in Section 3.2, all events such as service completions, arrivals, etc. are assumed to occur immediately after time n, implying amongst others that the age of a customer in the service facility is at least one. As discussed below, $(A_n)_{n\geq 0}$ is a Markov chain, the $(r+1) \times (r+1)$ transition matrix of which is given by

$$P = \begin{bmatrix} b_1 & b_0 & 0 & 0 & \cdots & 0 & 0 \\ b_2 & a_1^1 & a_0^1 & 0 & \cdots & 0 & 0 \\ b_3 & a_2^2 & a_1^2 & a_0^2 & \ddots & 0 & 0 \\ b_4 & a_3^3 & a_2^3 & a_1^3 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ b_r & a_{r-1}^{r-1} & a_{r-2}^{r-1} & a_{r-3}^{r-1} & \ddots & a_1^{r-1} & a_0^{r-1} \\ b_{r+1} & a_r^r & a_{r-1}^r & a_{r-2}^r & \cdots & a_2^r & a_1^r \end{bmatrix},$$

with

$$\begin{aligned} b_0 &= x_0, && (3.60)\\ b_1 &= 1 - x_0, && (3.61)\\ b_{i+1} &= y_i \prod_{k=1}^{i} (1 - x_{i-k}), \text{ for } 1 < i \leq r, && (3.62)\\ a_0^i &= 1 - y_i, \text{ for } 1 \leq i < r, && (3.63)\\ a_{l+1}^i &= y_i x_{i-l-1} \prod_{k=1}^{l} (1 - x_{i-k}), \text{ for } 0 < i \leq r \text{ and } 0 \leq l < i. && (3.64) \end{aligned}$$

As mentioned before, we assume that $(\prod_{k=i}^{j} \ldots) = 1$ for $i > j$. We will very briefly discuss these expressions as they can be obtained directly from the formulas presented in Section 3.2.2. The expressions for b_0, b_1 and a_0^i follow in a straightforward manner from the definitions of x_k and y_k. Denote n as the current time instant and assume an age i customer is in service, that is, $A_n = i$, then a_{l+1}^i equals the probability that this customer leaves the system (which happens with a probability y_i, notice that $y_r = 1$ since there can never be a customer with an age larger than r in the queue) and there will be a customer of age $i - l$ in service at time $n + 1$. Therefore, every customer who arrived at some time $n - i + k$, with $0 < k \leq l$, should reach his critical age no later than time n (this occurs with probability $\prod_{k=1}^{l}(1 - x_{i-k})$), and finally, with probability x_{i-l-1} there is an arrival at time $n - i + l + 1$ of a customer whose patience is at least $i - l$ time units. A similar argument can be given with relation to the expression for b_{i+1}. The only difference being that b_{i+1} corresponds to a transition to an idle queue and hence no customer will be served at time instant $n + 1$.

Lemma 1.

$$\prod_{k=k_{min}}^{k_{max}} (1 - z_k) = 1 - \sum_{k=k_{min}}^{k_{max}} z_k \prod_{m=k_{min}}^{k-1} (1 - z_m),$$

for $1 \leq k_{min} \leq k_{max}$ *and* $z_k \in \mathbb{R}$.

Proof. The statement follows by induction on k_{max}. First consider the situation in which $k_{max} = k_{min} + 1$:

$$\begin{aligned}(1 - z_{k_{min}})(1 - z_{k_{min}+1}) &= 1 - z_{k_{min}} - z_{k_{min}+1} + z_{k_{min}} z_{k_{min}+1} \\ &= 1 - z_{k_{min}} - z_{k_{min}+1}(1 - z_{k_{min}}).\end{aligned}$$

Assuming that for any $k_{max} \leq t$ the equation holds, we can prove it for $k_{max} = t + 1$:

$$\begin{aligned}1 - \sum_{k=k_{min}}^{t+1} z_k \prod_{m=k_{min}}^{k-1} (1 - z_m) &= 1 - \sum_{k=k_{min}}^{t} z_k \prod_{m=k_{min}}^{k-1} (1 - z_m) - z_{t+1} \prod_{m=k_{min}}^{t} (1 - z_m) \\ &= \prod_{k=k_{min}}^{t} (1 - z_k) - z_{t+1} \prod_{m=k_{min}}^{t} (1 - z_m) \\ &= \prod_{k=k_{min}}^{t+1} (1 - z_k).\end{aligned}$$

□

Theorem 2. *The steady-state probabilities of the Markov chain* $(A_n)_{n\geq 0}$ *under consideration are given by*

$$\pi_0 = \frac{1}{N}, \tag{3.65}$$

$$\pi_i = \frac{1}{N} \left(\prod_{k=1}^{i-1} \frac{1 - y_k}{1 - x_{k-1}} \right) \frac{x_0}{1 - x_{i-1}}, \tag{3.66}$$

for $i = 1, \ldots, r$ *and with*

$$N = 1 + \sum_{i=1}^{r} \left(\prod_{k=1}^{i-1} \frac{1 - y_k}{1 - x_{k-1}} \right) \frac{x_0}{1 - x_{i-1}}. \tag{3.67}$$

Proof. Clearly, $\pi = (\pi_0, \pi_1, \ldots, \pi_r)$ is a stochastic vector, therefore, it suffices to check whether π is an invariant vector of P. By making use of Eqns. (3.61), (3.62), (3.65) and (3.66) in the first balance equation

$$\sum_{i=0}^{r} \pi_i b_{i+1} = \pi_0$$

and by eliminating the common factors in the numerator and denominator we have

$$(1 - x_0) + \sum_{i=1}^{r} x_0 y_i \prod_{k=1}^{i-1} (1 - y_k) = 1 \Leftrightarrow \sum_{i=1}^{r} y_i \prod_{k=1}^{i-1} (1 - y_k) = 1,$$

which is proved by applying Lemma 1 with $k_{min} = 1, k_{max} = r$ and $z_k = y_k$, because $y_r = 1$. Inserting the expressions for b_0, a_i^i and π_i in the second balance equation

$$\pi_0 b_0 + \sum_{i=1}^{r} \pi_i a_i^i = \pi_1$$

allows us to rewrite this as

$$x_0 + \sum_{i=1}^{r} \frac{x_0^2 y_i}{1-x_0} \prod_{k=1}^{i-1}(1-y_k) = \frac{x_0}{1-x_0} \Leftrightarrow \sum_{i=1}^{r} y_i \prod_{k=1}^{i-1}(1-y_k) = 1.$$

Finally, the $(s+1)$-th balance equation

$$\sum_{i=s-1}^{r} \pi_i a_{i-s+1}^i = \pi_s,$$

for $1 < s \leq r$, is given by

$$\begin{aligned} & \pi_{s-1}(1-y_{s-1})N + \sum_{i=s}^{r} \left(\prod_{k=1}^{i-1} \left(\frac{1-y_k}{1-x_{k-1}} \right) \frac{x_0 y_i x_{s-1}}{1-x_{i-1}} \right) \prod_{m=1}^{i-s}(1-x_{i-m}) = \pi_s N \\ \Leftrightarrow \quad & \left(\prod_{k=1}^{s-1} \frac{1-y_k}{1-x_{k-1}} \right) x_0 + \sum_{i=s}^{r} \left(\frac{x_0 x_{s-1} y_i \prod_{k=1}^{i-1}(1-y_k)}{\prod_{k=1}^{s}(1-x_{k-1})} \right) = \pi_s N. \end{aligned}$$

By definition of π_s we have

$$\begin{aligned} & (1-x_{s-1})\pi_s N + \frac{x_0 x_{s-1} \prod_{k=1}^{s-1}(1-y_k)}{\prod_{k=1}^{s}(1-x_{k-1})} \left(\sum_{i=s}^{r} y_i \prod_{k=s}^{i-1}(1-y_k) \right) = \pi_s N \\ \Leftrightarrow \quad & (1-x_{s-1})\pi_s + x_{s-1}\pi_s = \pi_s, \end{aligned}$$

due to Lemma 1 with $k_{min} = s$, $k_{max} = r$ and $z_k = y_k$, as $y_r = 1$. □

Having determined the steady-state probabilities of the Markov chain $(A_n)_{n\geq 0}$, equation (3.11) shows the expression for the rejection probability P_{out} that a customer leaves the system before his service is completed/started, i.e.,

$$P_{out} = 1 - \sum_{i=1}^{r} P[Z = i] \tag{3.68}$$

Here, $P[Z = i]$ is the probability that a customer is completely served and has a response time (i.e., waiting time + service time) equal to i time units. This probability is given by

$$P[Z = i] = \frac{\beta}{\alpha}\pi_i, \tag{3.69}$$

as $\beta\pi_i$ is the probability that at an arbitrary time epoch, an age i customer leaves the system after completing service, while α is the probability that an arrival occurs in an arbitrary slot. Using the stochastic nature of the vector π, it follows that

$$P_{out} = 1 - \frac{\beta}{\alpha}(1-\pi_0). \tag{3.70}$$

Keeping α and β fixed, P_{out} is a function of the patience distribution X only, as such we refer to it as $P_{out}(X)$. We will prove that $P_{out}(X) \leq P_{out}(Y)$ if X is smaller than Y in the convex-ordering sense, i.e., $X \leq_{cx} Y$ [95]. As a consequence, given a mean m, $P_{out}(X)$ is minimal for the deterministic distribution with mean m among all discrete distributions X on $\{1, 2, \ldots\}$ with $E[X] = m$ (provided that m is an integer, otherwise X with $P[X = \lfloor m \rfloor] = \lceil m \rceil - m$ and $P[X = \lceil m \rceil] = m - \lfloor m \rfloor$ realizes the lowest $P_{out}(X)$).

Definition 2 Consider two random variables X and Y. Then X is said to precede Y in the convex-ordering sense ($X \leq_{cx} Y$) if and only if

$$E[X] = E[Y], \tag{3.71}$$
$$E[(X - d)^+] \leq E[(Y - d)^+], \tag{3.72}$$

for $-\infty < d < \infty$.

Definition 3 Let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ be vectors of dimension n. Then y weakly majorizes x in the sense of Marshall and Olkin [69] if

$$\sum_{i=1}^{k} x_i^{\downarrow} \leq \sum_{i=1}^{k} y_i^{\downarrow}, \tag{3.73}$$

for $k = 1, \ldots, n$, where $x^{\downarrow}$ denotes the vector obtained by rearranging the entries of x in nonincreasing order. That is, the sum of the k largest components of the vector y is greater than the sum of the k largest components of x.

Lemma 2. *If $X \leq_{cx} Y$, then*

$$\sum_{k=0}^{i-1} x_k(X) \geq \sum_{k=0}^{i-1} x_k(Y),$$

for all $i \geq 1$.

Proof. From Definition 2 we know that $E[X] = E[Y]$, i.e.,

$$\sum_{k=0}^{r(X)} (1 - p_k(X)) = \sum_{k=0}^{r(Y)} (1 - p_k(Y)). \tag{3.74}$$

Setting $d = i$ in (3.72) results in

$$\sum_{k=i}^{r(X)} (1 - p_k(X)) \leq \sum_{k=i}^{r(Y)} (1 - p_k(Y)). \tag{3.75}$$

Combining equations (3.74) and (3.75) leads to

$$\sum_{k=0}^{i-1} (1 - p_k(X)) \geq \sum_{k=0}^{i-1} (1 - p_k(Y)). \tag{3.76}$$

□

Lemma 3. *If $X \leq_{cx} Y$, then*

$$\prod_{k=0}^{i-1}(1-x_k(X)) \leq \prod_{k=0}^{i-1}(1-x_k(Y)).$$

Proof. From Lemma 2 we know that $X \leq_{cx} Y$ leads to

$$\sum_{k=0}^{i-1} x_k(X) \geq \sum_{k=0}^{i-1} x_k(Y),$$

for all $i \geq 1$. Moreover, $x_k(X)$ and $x_k(Y)$, $k \geq 0$, are both nonincreasing rows. Thus, the vector $(x_0(X), \ldots, x_{i-1}(X))$ weakly majorizes $(x_0(Y), \ldots, x_{i-1}(Y))$. A theorem by Tomić [69, Proposition 4.B.2] implies that

$$\sum_{k=0}^{i-1} g(x_k(X)) \geq \sum_{k=0}^{i-1} g(x_k(Y)),$$

for any continuous increasing convex function g. Letting $g(x) = -\log(1-x)$ proves the lemma since

$$\begin{aligned} & \sum_{k=0}^{i-1} -\log(1-x_k(X)) \geq \sum_{k=0}^{i-1} -\log(1-x_k(Y)) \\ \Leftrightarrow \quad & \log\left(\prod_{k=0}^{i-1}(1-x_k(X))\right) \leq \log\left(\prod_{k=0}^{i-1}(1-x_k(Y))\right) \end{aligned}$$

□

Theorem 3. *Let $X \leq_{cx} Y$, then $P_{out}(X) \leq P_{out}(Y)$, meaning a smaller patience distribution (in the convex-ordering sense) achieves a lower probability of abandonment in a Geo/Geo/1 queue with impatient customers in the system.*

Proof. Based on Theorem 2 and the expression found for P_{out}, it is sufficient to show $\Theta(X) \geq \Theta(Y)$, where Θ is defined as

$$\Theta(X) = \sum_{i=1}^{r(X)} \left(\prod_{k=1}^{i-1} \frac{1-y_k(X)}{1-x_{k-1}(X)}\right) \frac{x_0(X)}{1-x_{i-1}(X)} \tag{3.77}$$

$$= \sum_{i=1}^{r(X)} \frac{(1-p_{i-1}(X))(1-\beta)^{i-1}x_0(X)}{\prod_{k=1}^{i}(1-x_{k-1}(X))} \tag{3.78}$$

$$= \sum_{i=1}^{r(X)} \frac{x_{i-1}(X)(1-\beta)^{i-1}}{\prod_{k=0}^{i-1}(1-x_k(X))}, \tag{3.79}$$

where equation (3.78) is obtained by rewriting $(1-y_k(X))$ as,

$$\frac{(1-\beta)(1-p_k(X))}{1-p_{k-1}(X)}$$

and taking into account that $p_0(X) = 0$ as discussed on Page 40. By observing that $x_0(X) = \alpha$ we obtain equation (3.79). Having $X \leq_{cx} Y$ yields $r(X) \leq r(Y)$. Rewrite $\Theta(X) \geq \Theta(Y)$ by

subtracting the first $r(X)$ terms of $\Theta(Y)$ in both sides of the inequality and by dividing them by $(1-\beta)^{r(X)}$. This results in

$$\sum_{i=1}^{r(X)} \left(\frac{x_{i-1}(X)}{\prod_{k=0}^{i-1}(1-x_k(X))} - \frac{x_{i-1}(Y)}{\prod_{k=0}^{i-1}(1-x_k(Y))} \right) (1-\beta)^{i-1-r(X)} \geq \sum_{i=r(X)+1}^{r(Y)} \frac{x_{i-1}(Y)(1-\beta)^{i-1-r(X)}}{\prod_{k=0}^{i-1}(1-x_k(Y))}. \tag{3.80}$$

As $0 < \beta < 1$, we know that $(1-\beta)^i \leq 1$ for $i \geq 0$ and $(1-\beta)^i \geq 1$ for $i \leq 0$, therefore it suffices to show

$$\sum_{i=1}^{r(X)} \frac{x_{i-1}(X)}{\prod_{k=0}^{i-1}(1-x_k(X))} \geq \sum_{i=1}^{r(Y)} \frac{x_{i-1}(Y)}{\prod_{k=0}^{i-1}(1-x_k(Y))}. \tag{3.81}$$

Dividing both sides of the equality given in Lemma 1 by $\prod_{m=k_{min}}^{k_{max}}(1-z_m)$ results in

$$\sum_{k=k_{min}}^{k_{max}} \frac{z_k}{\prod_{m=k}^{k_{max}}(1-z_m)} = \frac{1}{\prod_{k=k_{min}}^{k_{max}}(1-z_k)} - 1. \tag{3.82}$$

Consider the following instance of this last equation (with $k_{min}=1, k_{max}=r(X)$ (and $r(Y)$), and $z_k = x_{r(X)-k}$ (and $x_{r(Y)-k}$))

$$\sum_{k=1}^{r(X)} \frac{x_{r(X)-k}}{\prod_{m=k}^{r(X)}(1-x_{r(X)-m})} = \frac{1}{\prod_{k=1}^{r(X)}(1-x_{r(X)-k})} - 1. \tag{3.83}$$

By combining this with (3.81), we find

$$\frac{1}{\prod_{k=0}^{r(X)-1}(1-x_k(X))} \geq \frac{1}{\prod_{k=0}^{r(Y)-1}(1-x_k(Y))}, \tag{3.84}$$

and this inequality is valid because of Lemma 3 (notice $1-x_k(X)=1$, for $k \geq r(X)$). □

In the next section we prove that this theorem is also valid for the Geo/Geo/1 queue where impatient customers become patient when entering the service facility.

Remark 3. The formula

$$(1-P_{out}) = \frac{\beta}{\alpha}(1-\pi_0), \tag{3.85}$$

which we can rewrite as

$$\frac{\alpha(1-P_{out})}{\beta} = 1-\pi_0, \tag{3.86}$$

is somewhat unexpected as one might get the incorrect impression that the server is only busy serving customers who are successful, that is, manage to complete their service before running out of patience. This is however not the case, as some customers get partially served. The

reason that we still get this expression can be explained by noticing that the mean service time $1/\beta_{suc}$ of a customer who is successful (arriving with a rate $\alpha(1-P_{out})$) is less than $1/\beta$, while the remaining service capacity is wasted by customers leaving the system prematurely. Actually, if we denote X_{rem} as the stationary distribution of the remaining patience when a customer enters the server, one can easily prove that the mean time $1/\beta_{in}$ that a customer spends in the server (irrespective of whether he is successful) equals

$$\frac{1}{\beta_{in}} = \frac{1}{\beta}\left(1 - \sum_{j=1}^{\infty} P[X_{rem} = j](1-\beta)^j\right) = \frac{1}{\beta}P_{suc|in}, \tag{3.87}$$

where $P_{suc|in}$ is the probability that a customer is successful provided that he entered the server. This can be understood as follows:

$$\begin{aligned}
\frac{1}{\beta_{in}} &= \sum_{i=1}^{\infty}(1-\beta)^{i-1}\beta P[X_{rem} \geq i]i + \sum_{i=1}^{\infty}(1-\beta)^i P[X_{rem} = i]i \\
&= \sum_{j=1}^{\infty} P[X_{rem} = j]\left(\beta\sum_{i=1}^{j} i(1-\beta)^{i-1} + j(1-\beta)^j\right) \\
&= \sum_{j=1}^{\infty} P[X_{rem} = j]\left(\beta\left(\frac{1}{\beta^2} - \frac{(j+1)(1-\beta)^j}{\beta} - \frac{(1-\beta)^{j+1}}{\beta^2}\right) + j(1-\beta)^j\right) \\
&= \frac{1}{\beta}\sum_{j=1}^{\infty} P[X_{rem} = j]\left(1-(1-\beta)^j\right).
\end{aligned}$$

Clearly,

$$1 - \pi_0 = \frac{\alpha P_{in}}{\beta_{in}} \tag{3.88}$$

where P_{in} is the probability that a customer enters the server. Thus, we get

$$1 - \pi_0 = \frac{\alpha}{\beta}P_{in}P_{suc|in} = \frac{\alpha}{\beta}(1 - P_{out}), \tag{3.89}$$

as required. Remark that it is not necessary for the IATs to be geometrically distributed, hence, equation (3.86) also holds for the D-MAP/Geo/1 queue with $\alpha = \theta D_1 e$. Recall from Section 2.2.1, θ is the stationary probability vector of $D = D_0 + D_1$. The equality is however not valid for general Phase-Type distributions, for instance if the service is deterministic then $1/\beta_{suc} = 1/\beta$ and

$$1 - \pi_0 \geq \frac{\alpha}{\beta}(1 - P_{out}). \tag{3.90}$$

3.3.2 The Geo/Geo/1 Queue with a Limited Waiting Time

Consider the same Geo/Geo/1 queue as in Section 3.3.1, but instead of having a limited sojourn time we assume that the limitation applies to the waiting time. This implies that once a customer has entered the service facility, he remains there until his service is completed. We further assume that a customer abandons the system even if he reaches his critical age

at the exact time instant that the server becomes available to him. Let $\hat{A}_n$ be the age of the customer in service just prior to time n, where $\hat{A}_n$ is said to be zero if the system is idle. Notice, due to the patient nature of the customers in the service facility and the unbounded geometric service time, the state space of the process $(\hat{A}_n)_{n\geq 0}$ is infinite and equals $\{0, 1, 2, \ldots\}$.

It is easily seen that $(\hat{A}_n)_{n\geq 0}$ is a Markov chain. Moreover, if we censor this Markov chain on the state space $\{0, 1, 2, \ldots, r\}$ its transition matrix $\hat{P}_r$ is identical to P if, for $i = 1, \ldots, r-1$, we replace y_i by β in the various expressions of Eqn. (7.1) (recall, $y_r = 1$). Indeed, the probability that an age k customer leaves the service facility equals β as opposed to y_i as in Section 3.3.1 and the probability that the Markov chain $(\hat{A}_n)_{n\geq 0}$ makes a transition from some state $r + k$, for $k > 0$, to a state $i \in \{0, \ldots, r\}$ does not depend upon k, meaning we can act as if age r customers simply leave the system with probability 1 when censoring on $\{0, 1, 2, \ldots, r\}$.

Theorem 4. *The steady-state probabilities of the Markov chain $(\hat{A}_n)_{n\geq 0}$ under consideration are given by*

$$\hat{\pi}_0 = \frac{1}{\hat{N}} \tag{3.91}$$

$$\hat{\pi}_i = \frac{1}{\hat{N}} \frac{(1-\beta)^{i-1} x_0}{\prod_{k=1}^{i}(1 - x_{k-1})} \tag{3.92}$$

for $i \geq 0$ and with

$$\hat{N} = 1 + \sum_{i=1}^{\infty} \left(\frac{(1-\beta)^{i-1} x_0}{\prod_{k=1}^{i}(1 - x_{k-1})} \right). \tag{3.93}$$

Proof. For $0 \leq i \leq r$, define $\pi_i(\beta, y)$ as π_i with y_k, for $1 \leq k \leq r-1$, replaced by β (notice, $y_r = 1$ is not to be replaced by β). Due to the censoring argument presented above we have that $(\hat{\pi}_0, \ldots, \hat{\pi}_r)$ is proportional to $(\pi_0(\beta, y), \ldots, \pi_r(\beta, y))$. Clearly,

$$\hat{\pi}_i = \hat{\pi}_r (1-\beta)^{i-r},$$

for $i \geq r$. Hence, for $i \geq r$, $\hat{\pi}_i$ is proportional to

$$\pi_r(\beta, y)(1-\beta)^{i-r} = \frac{\pi_r(\beta, y)(1-\beta)^{i-r}}{\prod_{k=r+1}^{i}(1 - x_{k-1})},$$

as $x_k = 0$ for $k > r$. Using the expression for π_i in Eqn. (3.66), we find that $\hat{\pi}_i$, for $i \geq 0$, is proportional to

$$\frac{(1-\beta)^{i-1} x_0}{\prod_{k=1}^{i}(1 - x_{k-1})},$$

which proves the theorem as $(\hat{\pi}_0, \hat{\pi}_1, \hat{\pi}_2, \ldots)$ is a stochastic vector. □

For α and β fixed, the probability of abandonment is a function of the patience distribution X only, therefore, we refer to it as $\hat{P}_{out}(X)$.

Theorem 5. *Let $X \leq_{cx} Y$, then $\hat{P}_{out}(X) \leq \hat{P}_{out}(Y)$, meaning a smaller patience distribution (in the convex-ordering sense) achieves a lower probability of abandonment in a Geo/Geo/1 queue with impatient customers in the waiting room.*

Proof. Analogously to Section 3.3.1, we can establish the following relation

$$\hat{P}_{out}(X) = 1 - \frac{\beta}{\alpha}(1 - \hat{\pi}_0(X)).$$

Hence, $\hat{P}_{out}(X) \leq \hat{P}_{out}(Y)$ if $\Phi(X) \geq \Phi(Y)$, which we define as:

$$\Phi(X) = \sum_{i=1}^{\infty} \left(\frac{(1-\beta)^{i-1}}{\prod_{k=1}^{i}(1 - x_{k-1}(X))} \right).$$

This inequality can be written as

$$\sum_{i=1}^{\infty} (1-\beta)^{i-1} \left(\frac{1}{\prod_{k=0}^{i-1}(1 - x_k(X))} - \frac{1}{\prod_{k=0}^{i-1}(1 - x_k(Y))} \right) \geq 0,$$

which holds due to Lemma 3. □

Remark 4. Theorem 5 also applies for the Geo/Geo/1 queue with waiting time aware customers as the loss probability in a GI/GI/1+GI queue is not affected by the awareness [9]. Waiting time aware customers only enter the waiting room provided that they can be served without losing patience.

3.3.3 Queues with Impatient Customers and Geometric Service Times

Consider a discrete-time FCFS queueing system S with a single server that serves customers in a geometric time. Label the arriving, impatient customers in order of their arrival starting from 0 and let X be their patience distribution. We shall refer to the queueing system S in which customers remain impatient during their sojourn time as SS and the one where customers are only impatient while waiting as SW.

Lemma 4. *Consider an arbitrary realization ω of S, then if customer i is being served in the SS system at time n, there will be a customer j present in the service facility of the SW system at time n, with $j \leq i$.*

Proof. The result is immediate from a sample-path argument. □

Theorem 6. *The loss probability $P_{out}(SW) \leq P_{out}(SS)$ for any patience distribution X, meaning the $\cdot$/Geo/1 queue where customers are only impatient while waiting (as opposed to during their sojourn time) achieves fewer abandonments.*

Proof. Lemma 4 shows us that if the SS system is busy, then so is the SW system, i.e.,

$$P_{idle}(SW) \leq P_{idle}(SS). \tag{3.94}$$

In Section 3.2 it is shown that the relation

$$P_{out} = 1 - \frac{\beta}{\lambda}(1 - P_{idle}), \tag{3.95}$$

holds both in the SS and SW system, where λ is the stationary arrival rate, yielding

$$P_{out}(SW) \leq P_{out}(SS). \tag{3.96}$$

□

Remark 5. As mentioned at the end of Section 3.2.6, theorem 6 does not generally apply to any queue with impatient customers (see [114] for an example).

4

Priority Scheduling

In this chapter we will study queues in which every arriving customer has a certain importance, referred to as his priority level. Customers with a higher priority level will be served before customers with a lower priority and as a consequence low priority customers will only be served if there are no higher priority customers present in the queue. Priority queues can for example be useful in modern communication networks to support the combination of multiple types of traffic with different Quality of Service requirements. A simple classification can be the distinction between delay sensitive and delay tolerant traffic. A typical example of the delay sensitive traffic is real-time traffic, such as video and voice. As mentioned before, these applications have strict requirements considering delay and delay jitter, therefore, packets that arrive at the receiver too late have become worthless to the application. For the delay tolerant traffic the loss rate due to buffer finiteness is usually a more important performance measure. In this chapter, we will pay attention to the computation of this loss rate.

We consider a priority system with two types of customers. Each class has its own finite capacity buffer to store packets that have to wait before they can be served. We discuss five different ways to approximate the loss rate as well as an exact approach, which allows us to identify the impact of assuming one or both buffers infinite on this loss rate. Furthermore, we investigate whether asymptotic based results can achieve the same level of accuracy as those based on the actual steady state probabilities. Three novel priority queueing models are introduced and efficient algorithms, relying on matrix analytic methods, are developed within this context.

If we would like to extend this model to more general service times and a larger number of priority classes, the dimension of the matrices involved in the computations would become very large. In other words, the disadvantage of this method is its high space complexity, since one has to keep track of the number of customers of each priority class present in the system. Additionally, the discrete-time setting which allows for multiple events to occur at the same time also contributes to the high complexity, together with the manner in which the Markov chain is set up. A novel approach to model a priority queue with K priority classes, MMAP[K] arrivals and PH service times in a continuous-time setting, using tree-like processes, will be discussed in Chapter 5.

4.1 Related Work

The study of priority queues has a long history and is often motivated by their common occurrence in communication networks [115, 116, 7, 5, 50], where they can be used to model Random Access Memory (RAM) buffers and in service part logistics [96, 97]. One of the key performance measures of such a buffer is the loss rate induced by their finite capacity as this strongly affects the network performance. From an analytical point of view, dealing with finite capacity queues is often more troublesome compared to infinite size buffers. Therefore, it is a common practice to analyze the infinite capacity system first and afterward to apply a heuristic method to obtain an estimate of the loss probability for the finite capacity problem (e.g., the probability of having more than C customers in the infinite case is frequently used as an approximation to the loss rate in a finite capacity C setting [38]).

Although this approach has been shown to be fruitful for many queueing systems, more recent results may question such an approach when applied to the (low priority) loss rate in a priority queueing system. More specifically, in [2, 58, 115, 116] it is shown that the tail behavior of the low priority buffer occupation might be nongeometric when both the low and high priority buffer is of infinite capacity. Earlier results (e.g., [30]), however, have shown that one typically has geometric tails when the high priority buffer capacity is finite (and arbitrarily large). One does not expect a substantial difference between having an infinite or a very large finite buffer for the high priority traffic (i.e., any simulation run attains some finite maximum queue length). As such, the correspondence between the infinite and finite capacity C system should grow as C increases. However, the tail behavior of both systems, for any finite C, follows a very different regime, implying that blindly trusting upon asymptotic results may lead to substantial errors. The opposite modeling approach, where infinite size queueing systems are studied by truncation to accomplish a numerical evaluation, also exists [7, 5], further motivating our interest in this subject.

4.2 The Impact of Finite Buffers

In this section, we will analyze a fundamental discrete-time queueing system with two priority classes, where each priority class has its own waiting room. To study the impact of the buffer finiteness, we introduce three novel discrete-time queueing models with batch arrivals: one to analyze the system where both queues (low and high priority) are finite and two models that evaluate the systems where either one of the buffers is finite. The objective lies in identifying the approaches that may cause poor estimates. The arrival process considered allows correlation between the number of arrivals of each priority class. There is, however, no correlation between the number of arrivals during consecutive time slots. We further assume a deterministic service time of one time slot for all packets. Although this model is a rather restrictive one, it allows us to isolate the impact of assuming one (or two) infinite size buffers on the accuracy of the loss rate obtained.

A variety of matrix analytic techniques are exploited to assess the (estimated) loss rate for each of the three models with at least one finite capacity buffer. Especially useful is the observation that the system with two finite capacity buffers can be captured by the paradigm developed in [49] for an M/G/1 type Markov chain with some regenerative structure, as well as the explicit knowledge of the G matrix appearing in the M/G/1 type Markov chain for the finite capacity high priority buffer. For the setup where both queues are of infinite size we

can rely on existing results involving generating functions [116] to obtain numerical results. In case the low priority traffic has an infinite size capacity buffer, we develop two estimates for the loss rate: one based on a numerical evaluation of the steady state probabilities and another that uses an asymptotic description of the tail behavior. This leads to a total of six different approaches to gather the loss rate of a system with two finite buffers (including five approximations). Notice, although the methods developed in [7] are closely related to the model with an infinite high and finite low capacity buffer, they do not apply directly as batch arrivals are not considered in [7].

4.2.1 System Characteristics

We consider a discrete-time single-server multi-class queueing system with a priority scheduling discipline. We consider a system with two priority classes, denoted as the high (class-1) and the low (class-2) priority class. The arrival process is chosen as in [58, 115, 116] and is characterized by the probabilities

$$a(i_1, i_2) = Prob[a_1 = i_1, a_2 = i_2],$$

where a_j denotes the number of arriving packets of class-j during a time slot. The corresponding joint probability generating function is given by

$$A(z_1, z_2) = E\left[z_1^{a_1} z_2^{a_2}\right] = \sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} a(i_1, i_2) z_1^{i_1} z_2^{i_2}. \tag{4.1}$$

Notice that the number of arrivals from different classes in one slot can be correlated. There is however no correlation between the number of arrivals during consecutive time slots. For further use, let

$$\begin{array}{ll} a_1(i) = \sum_{i_2=0}^{\infty} a(i, i_2), & a_1^*(i) = \sum_{k=i}^{\infty} a_1(k), \\ a_2(i) = \sum_{i_1=0}^{\infty} a(i_1, i), & a_2^*(i) = \sum_{k=i}^{\infty} a_2(k). \end{array}$$

The class-i arrival rate λ_i can then be expressed as

$$\lambda_i = \sum_{k=1}^{\infty} a_i^*(k). \tag{4.2}$$

We assume a deterministic service time of one time slot for all the packets. Although this assumption is rather strong, it allows us to isolate the impact of assuming one (or two) infinite size buffers on the accuracy of the loss rate obtained. There are two buffers, one for the high and one for the low priority traffic. If an arriving packet finds the server busy, it joins the appropriate buffer. The class-1 packets have priority over these of class-2 and within each class the service discipline is assumed to be FCFS. Therefore, when a packet completes its service, the class-1 packet with the longest waiting time will be served. If there are no high priority packets available, the oldest low priority packet is selected for service.

In the next sections, we discuss four different cases, where the buffer size of the two buffers is either finite or infinite. For each situation, we determine the steady state probabilities of the system contents distribution, which can, among others be used to calculate the loss probability of the class-2 packets. In each of these models, all events such as arrivals, service completions and packet losses are assumed to occur at instants immediately after the discrete time epochs. We further assume that departures occur before arrivals.

4.2.2 Finite High Priority Buffer

Let us first discuss the above-mentioned queueing system provided that the class-1 buffer is finite, with a capacity H, and the class-2 buffer is infinite. We can model this system using an M/G/1 type Markov chain represented by the following transition matrix:

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \cdots \\ A_0 & A_1 & A_2 & A_3 & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

We denote the states of this Markov chain as $\langle i, j \rangle$, where the level $i \geq 0$ denotes the number of low priority packets in the queueing system and $j = 0, \ldots, H+1$ reflects the number of high priority packets. An expression for the $(H+2) \times (H+2)$ matrices A_i $(i = 0, 1, \ldots)$ is given first. A transition to a lower level can only occur, if there are no high priority packets present in the system, otherwise such a packet is served, preventing any low priority packet from leaving the system. As a consequence only the first row of the matrix A_0 contains nonzero probabilities. A second condition in order to have a transition to a lower level is that no low priority packets arrive during the current time slot. Hence,

$$A_0 = e^{(1)t}(a(0,0), a(1,0), a(2,0), \ldots, a^*(H+1,0)), \tag{4.3}$$

where $a^*(i,j) = \sum_{k=i}^{\infty} a(k,j)$ and $e^{(1)t}$ is a column vector with all its entries equal to zero, except for the first which equals one. The transitions from state $\langle i, j \rangle$ to state $\langle i+k, j' \rangle$ are covered by the matrix A_{k+1}, for $i \geq 1$ and $k \geq 0$. We distinguish two cases: $j = 0$ and $j > 0$. In the first case, a low priority packet is in service; hence, $k+1$ low priority packets need to arrive in order to get a transition to level $i+k$. In the latter case, a class-1 packet occupies the server. A transition to level $i+k$ thus occurs if k class-2 packets arrive. This yields,

$$A_{k+1} = \begin{bmatrix} a(0,k+1) & a(1,k+1) & a(2,k+1) & \cdots & a^*(H+1,k+1) \\ a(0,k) & a(1,k) & a(2,k) & \cdots & a^*(H+1,k) \\ 0 & a(0,k) & a(1,k) & \cdots & a^*(H,k) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a(0,k) & a^*(1,k) \end{bmatrix}. \tag{4.4}$$

Finally, the matrix B_k contains the probabilities of having a transition from level zero to level k. Level zero corresponds to having zero class-2 packets in the system, implying that k low priority packets must arrive to enter a level k state, for $k \geq 0$,

$$B_k = \begin{bmatrix} a(0,k) & a(1,k) & a(2,k) & \cdots & a^*(H+1,k) \\ a(0,k) & a(1,k) & a(2,k) & \cdots & a^*(H+1,k) \\ 0 & a(0,k) & a(1,k) & \cdots & a^*(H,k) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a(0,k) & a^*(1,k) \end{bmatrix}. \tag{4.5}$$

To calculate the steady state vector $x = (x_0, x_1, x_2, \ldots)$, with x_k a $1 \times (H+2)$ vector for $k \geq 0$, of P, i.e., the joint system contents distribution, Ramaswami's formula [83, 78, 70] can be

used. This formula requires x_0 and a (stochastic) matrix G, being the smallest nonnegative solution of

$$G = \sum_{k=0}^{\infty} A_k G^k, \tag{4.6}$$

as its input. The (j,k)-th entry of this matrix represents the probability that, starting from state $\langle i+1, j\rangle$, the Markov chain visits the set of states $\{\langle i,0\rangle, \ldots, \langle i, H+1\rangle\}$ the first time by entering the state $\langle i,k\rangle$. Finding G is often by far the bottleneck when computing the invariant vector of an M/G/1 type MC. However, since the rank of A_0 equals one, the matrix G also has rank one [64]. More specifically, provided that the steady state of the MC exists, all the rows of G are identical and are given explicitly by the vector

$$\alpha = (a(0,0), a(1,0), a(2,0), \ldots, a^*(H+1,0))\,/a^*(0,0). \tag{4.7}$$

Notice that $G^k = G = e\alpha$ for $k > 0$ and $g = \alpha$, where g is the unique solution of $gG = g$, with $ge = 1$ and e a column vector of ones. Combining [78, Chapter 3] and the structure of the A_k and B_k matrices with these properties, Algorithm 4.1 to compute x can be devised using the equations

$$\sum_{k=1}^{\infty} B_k G^{k-1} = B_1 + a_2^*(2) e\alpha, \tag{4.8}$$

$$\sum_{k=2}^{\infty} A_k G^{k-1} = (a_2^*(1)e - a_2(1)e^{(1)t})\alpha, \tag{4.9}$$

$$\sum_{k=1}^{\infty} (B_k - B_k G^{k-1}) = (B - B_0 - B_1)(I - e\alpha), \tag{4.10}$$

and the following lemmas:

Lemma 5.

$$\sum_{k=1}^{\infty} k A_k e = (1+\lambda_2)e - e^{(1)t}$$

Proof. We can rewrite the left hand side of the equation as

$$\sum_{k=1}^{\infty} k A_k e = \sum_{k=1}^{\infty} k \left(a_2(k-1)e - (a_2(k-1) - a_2(k))e^{(1)t} \right). \tag{4.11}$$

Using the equations

$$\sum_{k=1}^{\infty} k a_2(k) = \lambda_2 \text{ and } \sum_{k=0}^{\infty} a_2(k) = 1, \tag{4.12}$$

we can can prove the lemma, since

$$\sum_{k=1}^{\infty} k a_2(k-1) = 1 + \lambda_2. \tag{4.13}$$

□

Algorithm 4.1 Computing the steady state vector in a H/∞ priority queue

1. Input:
   - the probabilities $a(i_1, i_2)$ for $0 \leq i_1$ and $0 \leq i_2$, concerning the arrival process,
   - the capacity H of the buffer for the high priority traffic.
2. Determine the matrices A_k and B_k $(k \geq 0)$ using Eqn. (4.3), (4.4) and (4.5).
3. Calculate $\rho = \pi\beta$, where π is the vector representing the invariant vector of the stochastic matrix $A = \sum_{k=0}^{\infty} A_k$ and $\beta = (1 + \lambda_2)e - e^{(1)t}$.
4. Next, set

   $$\tilde{\kappa}_1 = \psi_2 + (B_1 + a_2^*(2)e\alpha)\,(I - A_1 - a_2^*(1)e\alpha + a_2(1)e^{(1)t}\alpha)^{-1}\psi_1.$$

   The vectors ψ_1 and ψ_2 are given by the following expressions:

   $$\begin{aligned} \psi_1 &= (I - A_0 - A_1)\,(I - e\alpha)\,(I - A + (e - \beta)\alpha)^{-1}\,e + (1 - \rho)^{-1}a_2(0)e^{(1)t}, \\ \psi_2 &= (B - B_0 - B_1)\,(I - e\alpha)\,(I - A + (e - \beta)\alpha)^{-1}\,e + (1 - \rho)^{-1}(\lambda_2 - \rho + a_2(0))e, \end{aligned}$$

   where $B = \sum_{k=0}^{\infty} B_k$.
5. The vector x_0 containing the steady state probabilities that there are no low priority packets in the system, is given by

   $$x_0 = (\kappa\tilde{\kappa}_1)^{-1}\kappa,$$

   with κ the invariant probability vector of $K = B_0 + (I - B_0)e\alpha$.
6. Finally, the following recursion is used to calculate the remaining vectors x_i of the steady state distribution:

   $$x_i = \left(x_0\bar{B}_i + \sum_{j=1}^{i-1} x_j\bar{A}_{i+1-j} \right) \left(I - \bar{A}_1\right)^{-1}, \qquad i > 0.$$

   In this expression we have

   $$\begin{aligned} \bar{A}_k &= A_k + (a_2^*(k)e - a_2(k)e^{(1)t})\alpha, \\ \bar{B}_k &= B_k + a_2^*(k+1)e\alpha, \end{aligned}$$

   for $k \geq 0$.

Lemma 6.

$$e + (1-\rho)^{-1} \sum_{k=1}^{\infty} (k-1) B_k e = (1-\rho)^{-1} (\lambda_2 - \rho + a_2(0)) e$$

Proof. Using both equations in (4.12), we obtain

$$\begin{aligned} e + (1-\rho)^{-1} \sum_{k=1}^{\infty} (k-1) B_k e &= e + (1-\rho)^{-1} (\lambda_2 e - (e - a_2(0)e)) \\ &= (1-\rho)^{-1} (1 - \rho + \lambda_2 - 1 + a_2(0)) e \end{aligned}$$

which proves the lemma. □

Remark 6. The matrices A_k, B_k, $\bar{A}_k$, $\bar{B}_k$, etc. used in Algorithm 4.1 are fully characterized by their first (or first two) rows; hence, there is no need to store more than one (two) rows for each of these matrices.

In this section, we assumed an infinite size low priority buffer. In practice, buffers are finite and some low priority losses can occur. High priority buffers are usually dimensioned such that hardly any losses occur, therefore, we focus on the low priority packets. To estimate the loss probability of the class-2 packets, given the maximum capacity L of the corresponding buffer, we can use the following standard approach in queueing. This approach approximates the packet loss in a finite size L buffer, by the expected value of max(0,number of packets waiting $-L$) in an infinite size system:

$$P_{loss} \approx \sum_{k=L+1}^{\infty} (k-L) x_k e - x_{L+1}(0), \tag{4.14}$$

where $x_{L+1} = (x_{L+1}(0), x_{L+1}(1), \ldots, x_{L+1}(H))$. The last term in (4.14) originates from the possibility of having $L+1$ low priority packets in the system when no high priority packets are present. The accuracy of this estimate is studied in Section 4.2.6. Apart from computing the steady state vector $x = (x_0, x_1, x_2, \ldots)$ in an exact manner via Algorithm 4.1, we can also rely on a theorem by Falkenberg [30, Theorem 3.5], that describes the tail behavior of an M/G/1 type MC, to approximate x_k for k large. This theorem states that the tail will typically decay geometrically, with parameter τ. This parameter is the solution $\tau > 1$ to

$$\xi(A(z)) = z, \text{ with } A(z) = \sum_{k=0}^{\infty} A_k z^k, \tag{4.15}$$

and where $\xi(X)$ represents the Perron-Frobenius eigenvalue of the matrix X. In [30] it was proved that (4.15) has a unique solution with $z \in]1, R_A[$, where R_A denotes the radius of convergence of $A(z)$. Also, the function $\xi(A(z))$ is strictly increasing and $\xi'(A(1)) < 1$ (see Figure 4.1). Therefore, τ can be computed by a simple bisection algorithm. Algorithm 4.2 describes the computation of the approximated x_k values, which can be plugged in (4.14) to find an alternative estimate for the class-2 loss probability. We will refer to this approach as the H/∞_t approach (as opposed to the H/∞ approach of Algorithm 4.1).

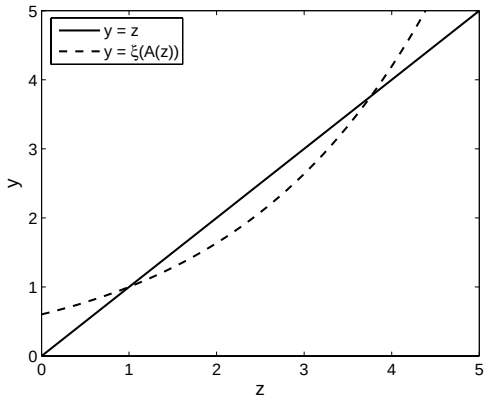


Figure 4.1: Illustration of the functions z and $\xi(A(z))$, for $z \in \mathbb{R}^+$

Algorithm 4.2 Computing the steady state vector in a H/∞_t priority queue

1. Compute the vectors x_0 and x_1 as explained in Algorithm 4.1, and the parameter τ as indicated in (4.15) using the following bisection algorithm:
   (a) Determine the smallest integer value $z_2 > 1$ for which $\xi(A(z_2)) > z_2$.
   (b) Set $z_1 = z_2 - 1$, $\tau = z_2 - 1/2$ and perform the following iteration until $z_2 - z_1 < \epsilon$:
      i. if $A(\tau) < \tau$ then $z_1 = \tau$ else $z_2 = \tau$
      ii. $\tau = (z_1 + z_2)/2$
2. Denote $u^T(\tau)$ and $v(\tau)$ as the left, resp. right eigenvector of $A(\tau)$ with corresponding eigenvalue τ and let
$$C = \frac{(x_0 B_1(\tau) - x_1 A_0)v(\tau)}{u^T(\tau)A'(\tau)v(\tau) - 1}, \tag{4.16}$$
   with $B_1(\tau) = \sum_{i=1}^{\infty} B_i \tau^i$ and $A'(\tau) = \sum_{i=1}^{\infty} i A_i \tau^{i-1}$.
3. Then, the vector x_n can be approximated by
$$x_n \approx C\tau^{-n} u^T(\tau). \tag{4.17}$$

4.2.3 Finite Low Priority Buffer

Consider the same system as in Section 4.2.2, but with an infinite buffer for the high priority traffic and a finite one of size L for the low priority traffic. As before, we start by setting up an M/G/1 type Markov chain to describe the system, where the level, resp. the state within a level, now denotes the number of high priority customers, resp. the number of low priority

customers. The transition matrix of this Markov chain is given by

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \cdots \\ C_0 & A_1 & A_2 & A_3 & \cdots \\ 0 & A_0 & A_1 & A_2 & \cdots \\ 0 & 0 & A_0 & A_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

with A_k ($k \geq 0$) an $(L+1) \times (L+1)$ matrix, B_k ($k > 0$) an $(L+2) \times (L+1)$ matrix, B_0 an $(L+2) \times (L+2)$ matrix and C_0 an $(L+1) \times (L+2)$ matrix. The different dimensions originate from the fact that there can be $L+1$ low priority packets in the system only if there are no packets of high priority present. Within a level, the states of this Markov chain correspond to the number of low priority packets; thus, level zero contains one additional state. Arguments similar to the one presented in Section 4.2.2 yield the following expressions:

$$A_k = \begin{bmatrix} a(k,0) & a(k,1) & \cdots & \bar{a}(k,L) \\ 0 & a(k,0) & \cdots & \bar{a}(k,L-1) \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \bar{a}(k,0) \end{bmatrix}, \quad k \geq 0, \tag{4.18}$$

$$B_0 = \begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & \cdots & \bar{a}(0,L+1) \\ a(0,0) & a(0,1) & a(0,2) & \cdots & \bar{a}(0,L+1) \\ 0 & a(0,0) & a(0,1) & \cdots & \bar{a}(0,L) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a(0,0) & \bar{a}(0,1) \end{bmatrix}, \tag{4.19}$$

$$B_k = \begin{bmatrix} a(k,0) & a(k,1) & \cdots & \bar{a}(k,L) \\ a(k,0) & a(k,1) & \cdots & \bar{a}(k,L) \\ 0 & a(k,0) & \cdots & \bar{a}(k,L-1) \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \bar{a}(k,0) \end{bmatrix}, \quad k > 0 \tag{4.20}$$

and

$$C_0 = \begin{bmatrix} a(0,0) & a(0,1) & a(0,2) & \cdots & \bar{a}(0,L+1) \\ 0 & a(0,0) & a(0,1) & \cdots & \bar{a}(0,L) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a(0,0) & \bar{a}(0,1) \end{bmatrix}, \tag{4.21}$$

where $\bar{a}(i,j) = \sum_{k=j}^{\infty} a(i,k)$. Given these expressions, we only need to find x_0 and the matrix G before we can apply Ramaswami's formula to compute $x = (x_0, x_1, \ldots)$. For this setup, there is no explicit expression for G. However, various iterative algorithms can be used to compute G [14]. A low memory implementation can be achieved using the following basic scheme, dating back to Neuts [78]:

$$\begin{aligned} G_0 &= I, \\ G_n &= \sum_{k=0}^{\infty} A_k G_{n-1}^k. \end{aligned}$$

The time needed to execute one iteration can be reduced by observing that only the first row has to be calculated for the entire matrix to be known. That is, the matrix G_n is a triangular matrix with the following structure (due to the probabilistic interpretation of G) :

$$G_n = \begin{bmatrix} G(0) & G(1) & \cdots & G(L) \\ 0 & G(0) & \ddots & G^*(L-1) \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & G^*(0) \end{bmatrix},$$

where $G^*(i) = \sum_{k=i}^{L} G(k)$. Moreover, since the entries $G(i)$ of the matrix G are scalars, the triangular structure allows us to compute these entries recursively as illustrated by Algorithm 4.3. The invariant vector of the stochastic matrix G is given by

$$g = (0, 0, \ldots, 1). \tag{4.22}$$

Algorithm 4.3 Computing the entries $G(i)$ of the matrix G recursively

1. Compute $G(0)$ iteratively:

$$\begin{aligned} G_0(0) &= 1, \\ G_n(0) &= \sum_{k=0}^{\infty} a(k,0)(G_{n-1}(0))^k. \end{aligned}$$

2. Set $DG = 1 - \sum_{k=0}^{\infty} k a(k,0)(G(0))^{k-1}$.

3. To compute $G(m)$ from $G(0), \ldots, G(m-1)$, for $m = 1, \ldots, L$, determine

$$NG(m) = a(0,m) + \sum_{j=1}^{m} \sum_{k=1}^{\infty} a(k,j) G^{(k)*}(m-j) + \sum_{k=2}^{\infty} a(k,0) G^{(k)*}(m),$$

where $G^{(k)*}$ denotes the k-fold convolution of $[G(0) \ \ldots \ G(m-1)]$. Then, the value of $G(m)$ can be obtained by

$$G(m) = \frac{NG(m)}{DG}.$$

As $A = \sum_k A_k$ is also triangular, its invariant vector $\pi = (0, 0, \ldots, 1)$ as well. Furthermore, the matrices A_k, B_k and $C_0 (k \geq 0)$ can be represented by their first row and both $A_k e$ and $B_k e$ equal $a_1(k)e$ (for $k \geq 0$). This leads to the following simplifications:

$$\beta = \lambda_1 e, \tag{4.23}$$

$$\rho = \lambda_1, \tag{4.24}$$

$$\psi_1 = \psi_2 = a_1(0)(1-\lambda_1)^{-1} e, \tag{4.25}$$

$$\tilde{\kappa}_1 = (1-\lambda_1)^{-1} e. \tag{4.26}$$

These expressions can be obtained from [78, Chapter 3] by noticing that

$$\begin{aligned}(I - A + (e - \beta)g)^{-1}e &= \sum_{k=0}^{\infty}(A - (e - \beta)g)^k e \\ &= \sum_{k=0}^{\infty}\lambda_1^k e \\ &= (1 - \lambda_1)^{-1}e.\end{aligned}$$

Therefore, Algorithm 4.4 can be used to compute $x = (x_0, x_1, x_2, \ldots)$. As A_k, B_k and G are fully characterized by their first row, so are the $\bar{A}_k$ and $\bar{B}_k$ matrices, allowing a significant reduction in the computing time and storage space needed to implement Ramaswami's formula (i.e., step 4 of the algorithm).

Having found the steady state probabilities, $x_j(k)$ denotes the steady state probability of having j high and k low priority packets in the system. Define

$$\bar{a}^*(i,j) = \sum_{k=i}^{\infty}\sum_{l=j}^{\infty} a(k,l). \tag{4.27}$$

Let us now take a look at the calculation of the loss rate of class-2 packets. Low priority packets are lost when the buffer has reached its maximum capacity upon their arrival. This happens in the following two cases:

- The system contains $j = 0, 1$ class-1 packets, i class-2 packets ($0 \leq i \leq L + 1 - j$) and

  (a) at least one high and $L + 1 - (i - \bar{j})^+$ low priority packets arrive or

  (b) no high and at least $L + 2 - (i - \bar{j})^+$ low priority packets arrive,

  where $\bar{j} = j + 1 \mod 2$. Notice, $(i - \bar{j})^+$ represents the number of class-2 packets left behind by the possible departure and seen by the new arrivals. The expected number of losses due to these cases corresponds to

$$\sum_{j=0}^{1}\sum_{i=0}^{L+1-j} x_j(i)\left(\sum_{k=L+1-(i-\bar{j})^+}^{\infty}\bar{a}^*(1,k) + \sum_{k=L+2-(i-\bar{j})^+}^{\infty}\bar{a}(0,k)\right).$$

- There are j ($j > 1$) class-1, i ($0 \leq i \leq L$) class-2 packets and more than $L - i$ low priority packets arrive. The expected number of losses caused by these cases equals

$$\sum_{j=2}^{\infty}\sum_{i=0}^{L} x_j(i)\left(\sum_{k=L+1-i}^{\infty}\bar{a}_2^*(k)\right).$$

The loss rate of the class-2 traffic can now be calculated by taking the sum of these two expressions. We expect that this approach provides us with a more accurate estimation than the one presented in the previous section, keeping in mind that the high priority queue is typically dimensioned sufficiently large such that hardly any losses occur. In Section 4.2.6 we will give some numerical examples in which both approaches are compared.

Algorithm 4.4 Computing the steady state vector in a ∞/L priority queue

1. Input:
   - the probabilities $a(i_1, i_2)$ for $0 \leq i_1$ and $0 \leq i_2$, concerning the arrival process,
   - the capacity L of the buffer for the class-2 traffic.
2. Determine the matrices A_k, B_k ($k \geq 0$) and C_0 using Equations (4.18), (4.19), (4.20) and (4.21).
3. Set
   $$x_0 = (\kappa\tilde{\kappa}_1)^{-1}\kappa = (1-\lambda_1)\kappa$$
   with κ the invariant probability vector of the matrix K:
   $$K = B_0 + \left(\sum_{k=1}^{\infty} B_k G^{k-1}\right)\left(I - \sum_{k=1}^{\infty} A_k G^{k-1}\right)^{-1} C_0.$$
4. Finally, we can use the following iteration to calculate the other vectors of the steady state distribution:
   $$x_i = \left(x_0\bar{B}_i + \sum_{j=1}^{i-1} x_j \bar{A}_{i+1-j}\right)\left(I - \bar{A}_1\right)^{-1}, \qquad i > 0,$$
   where
   $$\begin{aligned} \bar{A}_k &= \sum_{i=k}^{\infty} A_i G^{i-k}, \\ \bar{B}_k &= \sum_{i=k}^{\infty} B_i G^{i-k}, \end{aligned}$$
   for $k \geq 0$.

Remark 7. To increase the readability of this chapter, we use the same notations for the transition matrices in the different sections. That is, independent of the (in)finiteness of the buffers, we use the notations A_i, B_i and C_i, although the matrices themselves are of course different depending on the specific situation.

4.2.4 Two Finite Buffers

This section focuses on the system with both a finite, size L low and finite, size H high priority traffic buffer. In practice, all buffers are finite, thus the results obtained in this section are the most relevant. The system state, captured by the number of low and high priority customers in the queue, can be described by a Markov chain with the following transition matrix P:

$$P = \begin{bmatrix} B_0 & B_1 & \cdots & B_{L-1} & D_L & C_L \\ A_0 & A_1 & \cdots & A_{L-1} & D_{L-1} & C_{L-1} \\ 0 & A_0 & \ddots & A_{L-2} & D_{L-2} & C_{L-2} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \ddots & \ddots & A_0 & D_0 & C_0 \\ 0 & \cdots & \cdots & 0 & F & E \end{bmatrix}. \tag{4.28}$$

As in Section 4.2.2, the states are labeled as $\langle i, j \rangle$, with i and j reflecting the number of low and high priority customers, respectively. Notice that the states $\langle L+1, j \rangle$ can only be reached if $j = 0$. Otherwise, a high priority customer will occupy the server, leaving only L buffer places available for the low priority traffic. As a consequence C_i $(0 \leq i \leq L)$ are column vectors, F is a row vector, and E is a scalar.

In many applications, the dimension of the buffer for the class-1 traffic is significantly smaller than the class-2 buffer. Keeping this in mind, choosing the representation above allows us to work with smaller matrices compared to the model where the order of both variables is switched. Moreover, this choice causes P to have a useful regenerative structure. The expressions for the matrices A_k and B_k $(0 \leq k < L)$ are identical to those given in Section 4.2.2 and as a consequence the first L balance equations are also identical.

Let us now determine the expressions for the matrices C_k, D_k, E and F. First, the matrix C_k $(0 \leq k \leq L)$ contains the probabilities of having a transition to level $L+1$, which can only occur when there are no high priority packets in the system during the next time slot. Meaning, C_k is a column vector, the first two entries of which only differ form zero:

$$C_L = \begin{bmatrix} \bar{a}(0, L+1) \\ \bar{a}(0, L+1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \text{ and } C_k = \begin{bmatrix} \bar{a}(0, k+2) \\ \bar{a}(0, k+1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ 0 \leq k < L. \tag{4.29}$$

A similar argument can be used to find

$$E = \bar{a}(0, 1). \tag{4.30}$$

The transitions to level L are described by D_k $(0 \leq k \leq L)$ and F, and can be written as:

$$D_L = \begin{bmatrix} a(0,L) & \bar{a}(1,L) & \bar{a}(2,L) & \cdots & \bar{a}^*(H+1,L) \\ a(0,L) & \bar{a}(1,L) & \bar{a}(2,L) & \cdots & \bar{a}^*(H+1,L) \\ 0 & \bar{a}(0,L) & \bar{a}(1,L) & \cdots & \bar{a}^*(H,L) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \bar{a}(0,L) & \bar{a}^*(1,L) \end{bmatrix}, \tag{4.31}$$

$$D_k = \begin{bmatrix} a(0,k+1) & \bar{a}(1,k+1) & \bar{a}(2,k+1) & \cdots & \bar{a}^*(H+1,k+1) \\ a(0,k) & \bar{a}(1,k) & \bar{a}(2,k) & \cdots & \bar{a}^*(H+1,k) \\ 0 & \bar{a}(0,k) & \bar{a}(1,k) & \cdots & \bar{a}^*(H,k) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \bar{a}(0,k) & \bar{a}^*(1,k) \end{bmatrix}, \tag{4.32}$$

and

$$F = (a(0,0), \bar{a}(1,0), \bar{a}(2,0), \ldots, \bar{a}^*(H+1,0)), \tag{4.33}$$

Now that we have derived an expression for the building blocks of the transition matrix P, we are in a position to calculate its steady state distribution $x = (x_0, x_1, \ldots, x_{L+1})$. P is a downward skip-free finite transition matrix with a special regenerative structure, in [49, Theorem 4.1] Ishizaki introduced an efficient algorithm (similar to Ramaswami's formula) to compute the steady state vector of such a matrix P. More specifically, this algorithm computes the stationary distribution of a matrix with the following structure:

$$T = \begin{bmatrix} A_{0,0} & A_{0,1} & A_{0,2} & \cdots & A_{0,N-1} & A^*_{0,N} \\ A_{1,0} & A_{1,1} & A_{1,2} & \cdots & A_{1,N-1} & A^*_{1,N} \\ 0 & A_{2,0} & A_{2,1} & \cdots & A_{2,N-2} & A^*_{2,N-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & A_{N-1,0} & A_{N-1,1} & A^*_{N-1,2} \\ 0 & 0 & \cdots & 0 & A_{N,0} & A^*_{N,1} \end{bmatrix}, \tag{4.34}$$

where $A^*_{i,j} = \sum_{l=j}^{\infty} A_{i,l}$ and where there exists a probability vector a_i such that $A_{i,0} = A_{i,0} e a_i$, for $i = 1, \ldots, N$. Applying this algorithm to our setting and using the same notations as in Section 4.2.2, we can calculate the steady state probabilities by means the set of equations presented in Algorithm 4.5.

Observe that we compute $(x_0, \ldots, x_{L-1})$ in exactly the same way as in Section 4.2.2, except that x_0 is not normalized. Normalization occurs after computing x_L and x_{L+1}. Thus, obtaining results for the system with two finite buffers is almost computationally equivalent to solving the finite/infinite system. This is exceptional as finite buffer systems typically demand more computational power. Using these steady state probabilities, the loss probability of the class-2 packets can be calculated in the same way as in Section 4.2.3.

4.2.5 Two Infinite Buffers

To analyze the system where both buffers are of infinite size, we can rely on some existing results in the literature. From [116], it follows that the probability generating function $Q_2(z)$ of the number of class-2 packets waiting in the queue can be written as

$$Q_2(z) = (1-\lambda)\frac{(z-1)(Y(z)-1)}{(z-Y(z))(A(1,z)-1)}, \tag{4.35}$$

where $Y(z)$ is implicitly defined by $Y(z) = A(Y(z), z)$ and where $A(z_1, z_2)$ was defined in (4.1). From Rouché's theorem, it can be seen that there is exactly one solution for $Y(z)$, with $|Y(z)| \leq 1$ for $|z| < 1$ [116]. There are two approaches to retrieve an estimate for the class-2 loss probability from (4.35).

Algorithm 4.5 Computing the steady state vector in an H/L priority queue

1. Input:
   - the probabilities $a(i_1, i_2)$ for $0 \leq i_1$ and $0 \leq i_2$, concerning the arrival process,
   - both buffer capacities L and H.
2. Determine the matrices A_k, B_k $(0 \leq k \leq L-1)$, C_k, D_k $(0 \leq k \leq L)$, E and F using Eqns. (4.3–4.5) and (4.29–4.33).
3. Let x_0 be the stochastic solution of $x_0 = x_0 K$, where $K = B_0 + (I - B_0)e\alpha$.
4. Set
$$x_i = \left(x_0 \bar{B}_i + \sum_{k=1}^{i-1} x_k \bar{A}_{i-k+1} \right) \left(I - \bar{A}_1 \right)^{-1},$$
for $i = 1, \ldots, L-1$, where $\bar{A}_k$ and $\bar{B}_k$ were defined in step 6 of Algorithm 4.1.
5. Let
$$x_L = \left(\sum_{k=0}^{L-1} x_k (D_{L-k} + C_{L-k} F^*) \right) (I - \bar{D}_0)^{-1},$$
where $F^* = F/(Fe)$ and $\bar{D}_0 = D_0 + C_0 F^*$.
6. Compute
$$x_{L+1} = \left(\sum_{i=0}^{L} x_i C_{L-i} \right) (1 - E)^{-1}.$$
7. Normalize $x = (x_0, x_1, \ldots, x_{L+1})$ such that $\sum_{i=0}^{L+1} x_i e = 1$.

4.2.5.1 Numerical Inversion

The first approach involves a numerical inversion of the generating function $Q_2(z)$ to obtain an approximation for the distribution of the number of class-2 packets present in the buffer. The inversion is realized using a discrete Fourier transform method (DFT), where a damping parameter $0 < r < 1$ is used [1]. We make use of a damping parameter such that when evaluating $Q_2(z)$ at $r\omega_N^s$, where ω_N^s for $s = 0, \ldots, N-1$ are the N-th roots of unity, $Y(z)$ is uniquely defined by Rouché's theorem as $|r\omega_N^s| < 1$. This leads to Algorithm 4.6.

Algorithm 4.6 Computing the steady state vector in an ∞/∞ priority queue

1. Input: the probabilities $a(i_1, i_2)$ for $0 \leq i_1$ and $0 \leq i_2$, concerning the arrival process.
2. Evaluate $Q_2(z)$ at $r\omega_N^s$, where ω_N^s for $s = 0, \ldots, N-1$ are the N-th roots of unity (where N is a power of 2 sufficiently large). This entails that we have to determine the unique solution of

   $$Y(z) = A(Y(z), z),$$

   with $|Y(z)| < 1$, for each $z = r\omega_N^s$. This step can be executed using software that allows symbolic computation, such as Maple or Mathematica.
3. Compute q_k, for $k = 0, \ldots, N-1$, via the inverse DFT. That is, q_k can be written as

   $$q_k = \frac{1}{N} \sum_{n=0}^{N-1} Q_2(e^{2\pi i n/N}) e^{2\pi i k n/N}.$$

   The values q_k can then be used as an approximation to the probability of having k buffered class-2 packets.

In [54], it is argued that as long as enough numerical precision is used, the desired probabilities can be obtained to any given accuracy. Therefore, it is advised to use a software package that supports high numerical precision when implementing this algorithm (e.g., Maple or Mathematica). The class-2 loss probability can be estimated as

$$P_{loss} \approx \sum_{k=L+1}^{\infty} (k - L) q_k. \tag{4.36}$$

4.2.5.2 Tail Behavior

A second approach is to rely on the tail behavior of (4.35) to get an alternate approximation q'_k for the probability of having k class-2 packets buffered. A description of the tail behavior of interest can be found easily from [116, Eqn. (21)]. The key in generating numerical results from these expressions is the computation of the values $z_T > 1$ and $z_B > 1$. These numbers are the real solutions to $A(z, z) = z$, resp. $A^{(1)}(Y(z), z) = 1$, with the smallest norm (and strictly larger than one). Here, $A^{(1)}(z_1, z_2)$ denotes the first partial derivative of $A(z_1, z_2)$.

As $A(z, z)$ is a convex function with $A(1, 1) = 1$ and $\left.\frac{dA(z,z)}{dz}\right|_{z=1} = \lambda < 1$ (otherwise the system would be unstable), we can apply a simple bisection algorithm to find z_T (see step

1 of Algorithm 4.2 and Figure 4.1). The value of z_B can be found using Algorithm 4.7. In [116] three cases for the tail behavior are distinguished. To identify these cases, we consider another potential singularity z_L of $Q_2(z)$, i.e., the positive zero of $z - Y(z)$ on the real axis. If z_L exists, it can be proved to be equal to z_T. The tail behavior of the class-2 queue contents is then determined by the dominant singularity, which brings us to the following cases:

- if $z_L = z_T < z_B$, the tail exhibits a typical geometric behavior,
- if $z_L = z_T = z_B$, the tail exhibits a typical nongeometric behavior,
- if z_L does not exists, we have a so-called transition type tail behavior.

More specifically, an approximation q'_k for the probability of having k class-2 packets buffered is given by the following expressions [116]:

$$q'_k = \begin{cases} \frac{(1-\lambda)A(1,z_T)(z_T-1)^2 z_T^{-k-1}}{(A(1,z_T)-1)(Y'(z_T)-1)}, & \text{if } z_L = z_T < z_B, \\ \frac{(1-\lambda)A(1,z_B)(z_B-1)^2 n^{-1/2} z_B^{-k}}{K_Y\sqrt{z_B\pi}(A(1,z_B)-1)}, & \text{if } z_L = z_T = z_B, \\ \frac{(1-\lambda)K_Y A(1,z_B)(z_B-1)^2 n^{-3/2} z_B^{-k}}{2\sqrt{\pi/z_B}(A(1,z_B)-1)(z_B-Y(z_B))^2}, & \text{if } z_L \text{ does not exist,} \end{cases}$$

with

$$K_Y = \sqrt{\frac{2A^{(2)}(Y(z_B),z_B)}{A^{(11)}(Y(z_B),z_B)}}.$$

Algorithm 4.7 Computing the value of z_B in an ∞/∞_t priority queue

1. Set $z_{2:min} = 1$ and $z_{2:max} = 2$. Determine $z_1 > 1$ via a bisection algorithm such that

   $$A^{(1)}(z_1, z_{2:max}) = 1.$$

   As long as $A(z_1, z_{2:max})$ is less than z_1, increase $z_{2:min}$ and $z_{2:max}$ by one.

2. Next, let

   $$z_{2:new} = \frac{z_{2:min} + z_{2:max}}{2}.$$

   Determine $z_1 > 1$ via a bisection algorithm such that $A^{(1)}(z_1, z_{2:new}) = 1$.
   If $z_1 < A(z_1, z_{2:new})$, assign $z_{2:new}$ to $z_{2:max}$, else $z_{2:min} = z_{2:new}$.

Repeat step 2 until $z_{2:max} - z_{2:min} < 10^{-14}$.

4.2.6 Numerical Examples

In this section we will compare the discussed approaches to estimate the loss probability of the low priority traffic. Let us first describe the arrival process under consideration. The number

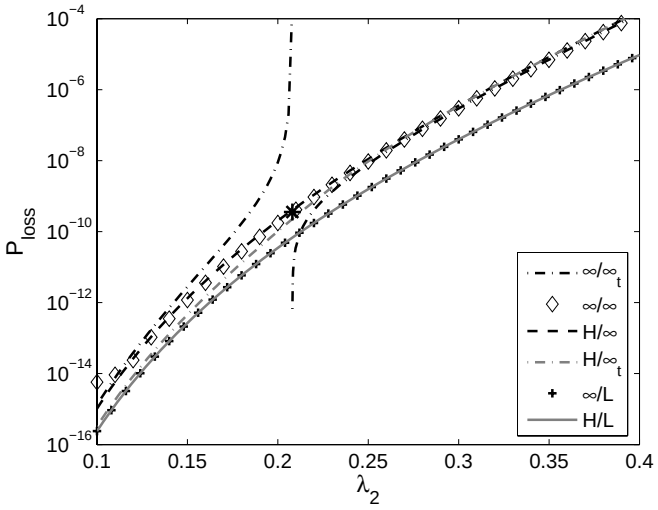


Figure 4.2: Comparison of the loss rate of low priority packets for each of the six approaches with $\lambda_1 = 0.4, H = 25, L = 20$ and a varying rate λ_2

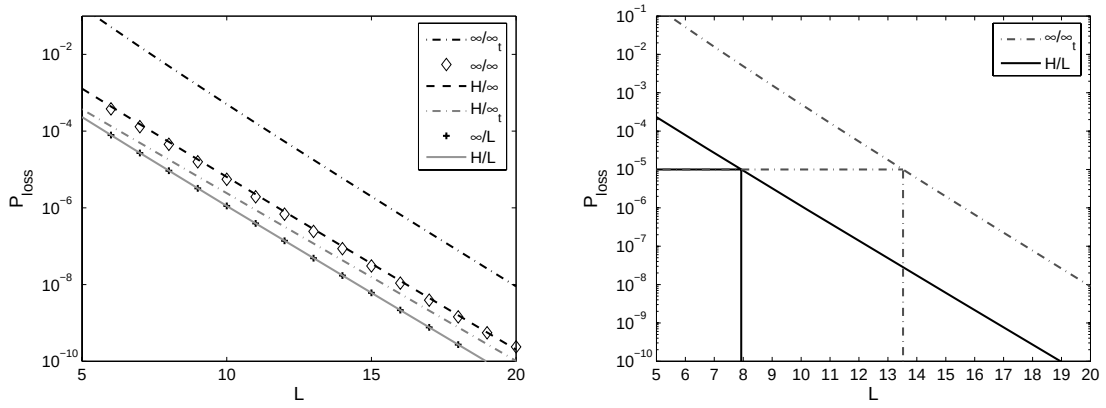
of arrivals during one time slot is bounded by N and is generated by a Bernoulli process with rate λ_T/N, where an arriving packet belongs to class-j ($j = 1, 2$), with a probability λ_j/λ_T (with $\lambda_1 + \lambda_2 = \lambda_T$). This arrival process is characterized by the joint probability generating function

$$A(z_1, z_2) = \left(1 + \sum_{j=1}^{2} \frac{\lambda_j}{N}(z_j - 1)\right)^N .$$

It was also used in [115] where a nonblocking output-queueing switch with N inlets and N outlets was given as a possible application.

More specifically, we assume the maximum number of simultaneously arriving packets to be 16. The probability that a class-1 packet arrives is fixed throughout this section at $\lambda_1 = 0.4$, while the buffer for the high priority traffic has a size $H = 25$ packets. By dimensioning the high priority buffer like this, the probability that a class-1 packet is dropped due to buffer overflow is in the order of 10^{-20}. Figure 4.2 represents, for each of the discussed approaches, the loss rate of the class-2 packets where the corresponding buffer has a size $L = 20$ packets and $\lambda_2 = 0.1, \ldots, 0.4$.

The exact results obtained via the system with two finite buffers, is denoted by the full line. It can be seen that the ∞/L results are very accurate, meaning there is no harm in assuming an infinite size high priority buffer. The other four approximation approaches give rise to higher loss probabilities. This difference is caused by the heuristic calculation used to estimate the loss probability. Whenever an infinite buffer is used for the low priority traffic, the estimate for the loss probability is based on the probability that the number of packets in the buffer exceeds L. In general, this causes an overestimation as packets that would be dropped earlier by the finite capacity system may still reside in the infinite buffer setup when the next arrival(s) occur.



(a) Comparison of the loss rate

(b) Buffer dimensioning

Figure 4.3: Comparison of the low priority loss rate obtained by each of the six approaches for $\lambda_1 = 0.4, \lambda_2 = 0.2, H = 25$ and a varying capacity L

In case both queues are assumed to be infinite, we observe some poor results around $\lambda_2 = 0.21$ for the ∞/∞_t approach, which relies on the asymptotic tail behavior of the class-2 queue. This is caused by the fact that the tail transition point is situated at $p_t = 0.208060765$: for $\lambda_2 \leq p_t$ the tail is nongeometric, whereas for $\lambda_2 > p_t$, we have a geometric tail. When $\lambda_2 < p_t$, the asymptotic regime is dominated by a branch point, whereas for $\lambda_2 > p_t$ there exists a dominant pole. When we approach the transition point, the domination becomes less severe and significant errors occur as shown in this example. The loss rate obtained for $\lambda_2 = p_t$ is quite accurate as indicated by the star on the plot.

Figure 4.3(a) illustrates the influence of the buffer capacity for the low priority traffic on the loss probability of this traffic in the case where $\lambda_2 = 0.2$. As could be expected, the loss probability of the class-2 packets decreases when this buffer becomes larger. If we compare the different approaches, we notice the same behavior as in Figure 4.2. Because taking $\lambda_2 = 0.2$ brings us relatively close to the transition point, a significant error is introduced by assuming both queues infinite and relying on the tail behavior. That is, the loss probability obtained by this approach overestimates the actual loss rate by a factor of 100 to 1000. If, for example, we would use the ∞/∞_t approximation to dimension the class-2 buffer such that the loss probability is less than 10^{-5}, we would need a buffer of 14 packets, whereas a buffer of only 8 packets suffices if we consider the H/L approach as illustrated by Figure 4.3(b).

In Figure 4.4 we compare the two approaches based on the tail behavior of the low priority queue. In fact, the full line represents the geometric decay parameter in function of the arrival rate λ_2 of the low priority traffic. The dashed line represents the parameter z_T, described in Algorithm 4.7. On the figure, the transition point is indicated by the vertical line. As mentioned before, on the left of this line the tail for the class-2 queue obtained by algorithm 4.7 is nongeometric, whereas on the right of the transition point the tails are geometric. It can be noticed that the values indicated by the two curves, converge to the same value as λ_2 reaches the transition point.

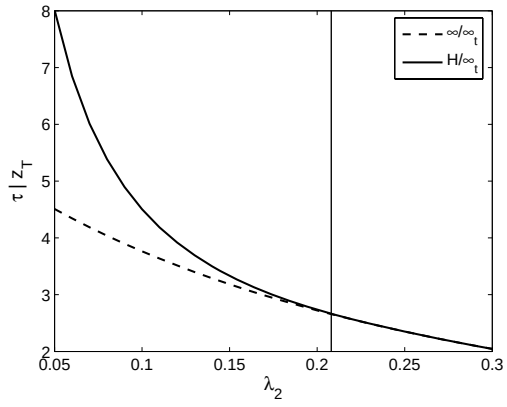


Figure 4.4: Comparison of the coefficient for the tail approaches, $\lambda_1 = 0.4, H = 25, L = 20$

4.2.7 Conclusions

If we compare the different approximations to compute the low priority loss rate in a queueing system with two priority classes, the most accurate results were generated by the approach in which only the high priority traffic is considered as infinite. Moreover, given that the size of the high priority buffer is chosen sufficiently large, the distinction with exact results is negligible. When the low priority queue is assumed to be infinite, an overestimated loss rate is observed. Relying on the actual steady state probabilities or the asymptotic tail behavior seems to make little difference if the high priority queue is finite. However, in case both queues are infinite very inaccurate loss probabilities were observed when we made use of the asymptotic tail behavior, especially in the area near the transition point.

5

A Tree-Like Process Approach to Priority Modeling

In this chapter we will present an alternative strategy to analyze priority queueing systems. Instead of remembering the number of customers of each type, we will only keep track of some types of customers, while storing the other customers on a stack. The introduction of this additional stack mechanism allows us to turn to tree-like processes for modeling the system. As a consequence, the size of the transition blocks that capture the system's behavior is reduced significantly. The queueing system that will be analyzed in this chapter is more general compared to the one that was analyzed in Chapter 4 in a variety of ways. First of all, the number of priority classes will not be restricted to two, but the analytical model holds for an arbitrary number K of priority classes. Also, the process describing the arrivals of new customers in the queue is the MMAP[K] process. It is a very general set of arrival processes that allows correlated inter-arrival times and correlation between the classes of consecutive jobs. Moreover, we will turn to a continuous-time setting, avoiding transitions in the Markov chain that are caused by simultaneous events.

We start by introducing the queueing model under consideration in Section 5.2. In Section 5.3 we shortly explain how the priority system can be modeled as a QBD process. In this section, the size of the transition blocks of the matrices that appear in the traditional approach is identified so that this can be compared to the number of states needed for the tree-like process presented in Section 5.4. In Section 5.4.1 we define a continuous-time Markov chain on a $(K - \eta + 1)$-ary tree that describes the evolution of the priority system. However, this process contains transitions that are not allowed in a tree-like process, where transitions can only take place from a certain node to its parent, to the node itself or to one of its children. Adding a new variable allows us to transform the process into a tree-like process, as explained in Section 5.4.3. Section 5.5 presents expressions for the transition matrices characterizing this process, together with an efficient computation of the V-matrix that exploits the structure of the different matrices. After the calculation of the steady state probabilities in Section 5.5.3, we conclude this chapter with a formula to compute the loss rates and illustrate this with some numerical examples in Section 5.6.

5.1 Related Work

In [4, 6] Alfa et al. studied priority queues with $C \geq 2$ classes, MMAP[C] input and PH services by setting up a QBD Markov chain that generalizes Miller's result [73]. Although the queues considered in [4, 6] are infinite, a similar approach can be used to solve the system when some of the queues are finite. Takine and Hasegawa [101] analyzed the preemptive priority queue with $C \geq 2$ service classes, C independent MAP arrival streams and state-dependent service times, using the workload process of the queue. In this work that is especially effective to compute the waiting time distributions, they generalize the model presented by Machihara in [68] that relied on the diagonalization of some matrices. As will be discussed further in this chapter, our setting does not require independent MAP arrivals streams. Takine [100, 102] also studied the nonpreemptive priority MAP/G/1 queue and derived various formulas for the generating function of the queue length and Laplace-Stieltjes transform of the waiting time for each priority class.

The work presented in this chapter is based on the methodology put forward in [113], where a 3-class discrete-time priority queueing system was analyzed using tree-like processes. A drawback of this approach is the complexity in the description of the model introduced by the discrete-time setting. Due to this setting, we have to distinguish all different combinations of multiple events that can happen during a single time slot. In this chapter where we will consider the general case with K priority classes, these simultaneous events are circumvented by constructing a model in a continuous-time setting. This research has been conducted in cooperation with prof. Qi-Ming HE from Dalhousie University, Canada.

5.2 The Queueing Model

We consider a K-class queueing system with the following characteristics. Type one customers have the highest priority, type two customers have the second highest priority, ..., and type K customers have the lowest priority. The scheduling discipline is preemptive-repeat, meaning that the processing of lower priority customers in the server will be interrupted when a customer with a higher priority arrives. When the low priority customer is allowed to enter the server again, he has to restart his service. Hence, the partial service a customer receives gets lost when being interrupted by a higher priority customer.

Customers arrive according to a continuous-time Markovian arrival process with marked jobs, i.e., the MMAP[K] process [40, 44, 41]. The MMAP[K] process is characterized by $K+1$ rate matrices $\{D_0, D_1, D_2, \ldots, D_K\}$ of order m_a. This arrival process can be compared to the BMAP arrival process, with that difference that the matrices D_i from the MMAP[K] process describe the arrivals of type-i customers and not the arrival of a batch of i customers. The matrix D_0 covers the state transitions without any arrivals. As with the general MAP process, we use this arrival process because of its generality. It allows correlated inter-arrival times and correlation between the classes of consecutive jobs. Moreover, the input traffic does not need to consist of K independent streams (i.e., one for each priority class).

The service times of type-k customers have a common continuous-time PH distribution (m_k, α_k, T_k), $1 \leq k \leq K$. The waiting space of type-k customers has size N_k, including the one in service, if any. Hence, as opposed to the model built in Chapter 4, the customer in service also contributes to the queue size. This implies that the actual maximum queue length is N_k if no customer of this type is in service or $N_k - 1$ if one is in service. The numbers

$\{N_1, \ldots, N_K\}$ are finite except that $N_\eta = \infty$, for some $1 \leq \eta \leq K$. Using similar arguments as put forward in Chapter 4 it can be argued that the infinite nature of the type-η queue does not annul the applicability of our model. That is, priority queues are mainly effective if sufficient jobs have the lowest priority. Therefore, hardly any high priority jobs will leave the system without receiving full service.

5.3 A QBD Formulation

Before introducing the tree-like approach to priority systems, let us first show how a K-class priority queue can be modeled as a QBD process. Let $\bar{q}_k(t)$ be the queue length of the type-k customers at time t, including the one in service and denote $S(t)$ as the type of the customer in service, if any, at time t, otherwise, $S(t) = K + 1$. Furthermore let $J_s(t)$ be the phase of the service time, if any, at time t, otherwise, $J_s(t)$ is zero and let $J_a(t)$ be the state of the arrival process at time t. It is easy to see that

$$\{(\bar{q}_\eta(t), (\bar{q}_1(t), \ldots, \bar{q}_{\eta-1}(t), \bar{q}_{\eta+1}(t), \ldots, \bar{q}_K(t), S(t), J_s(t), J_a(t))), t \geq 0\}$$

is a continuous-time Markov chain, where $\bar{q}_\eta(t)$ is the level variable and others are called auxiliary variables. In fact, the Markov chain is a QBD process since the number of type-η customers can at most increase or decrease by one. As discussed before, such a process can be analyzed using matrix analytic methods and the limiting probabilities of the QBD process have a matrix geometric solution. The size of the non-boundary transition blocks ($\bar{q}_\eta(t) > 0$) however equals

$$m_a \left(\prod_{k=\eta+1}^{K} (1 + N_k) \right) \left[m_\eta + \sum_{j=1}^{\eta-1} m_j N_j \left(\prod_{k=j+1}^{\eta-1} (1 + N_k) \right) \right]. \tag{5.1}$$

That is, since $\bar{q}_\eta(t) > 0$, the type of the customer in service will be at most η. This reduces to

$$m_a m_1 \left(\prod_{k=1, k\neq\eta}^{K} (1 + N_k) \right), \tag{5.2}$$

for $m_1 = m_2 = \ldots = m_\eta$. Note that $S(t)$ can be determined by the other variables and $S(t) \leq \eta$ for $\bar{q}_\eta(t) > 0$. The block size can be restrictive for efficient numerical computations. Therefore, in Section 5.4 we will explore an approach that results in a smaller block size.

5.4 A Tree-Like Process Approach

The queueing system of interest can also be formulated as a tree-like Markov chain. In short, this approach can be described as follows. We keep track of the number of some types of customers in the queue-area (Q-area), while the other customers will be stored on a stack. The size of the transition blocks is only influenced by the Q-area and not by the contents of the stack. To distinguish between these two cases, we choose a value η between 1 and K (recall, $N_\eta = \infty$). If there is at least one high priority customer (type $1, \ldots, \eta - 1$) in the system, customers of type $\eta, \ldots, K$ will be stored on the stack. If there is no high priority

customer present, the Q-area will be used to record the number of low priority customers (type $\eta+1,\ldots,K$) that are removed from the stack and if a low priority customer is in service, the stack will be empty. In this section, we first define a continuous-time Markov chain on a tree-structured state space for the queueing process in Section 5.4.1. Then, we present all the transition blocks of this process in Section 5.4.2. Finally, in Section 5.4.3, we transform the Markov chain into a discrete-time tree-like QBD Markov chain.

5.4.1 Definition of a Markov Process on a $(K-\eta+1)$-ary Tree

We define a stochastic process $\{(X(t),(S(t),Q(t),J_s(t),J_a(t))),t\geq 0\}$ where $J_s(t)$ denotes the phase of the service time, if any, at time t, otherwise, $J_s(t)$ is zero and $J_a(t)$ denotes the state of the arrival process at time t. The number of customers of type $1,2,\ldots,\eta-1,\eta,\eta+1,\ldots,K$ present in what we call the Q-area is recorded by

$$Q(t)=(q_1(t),\ldots,q_{\eta-1}(t),q_\eta(t),q_{\eta+1}(t),\ldots,q_K(t)).$$

We mainly distinguish three cases:

- If there are no customers of type $1,\ldots,\eta$ in the system, $Q(t)$ is of the form

$$Q(t)=(0,\ldots,0,0,q_{\eta+1}(t),\ldots,q_K(t)). \tag{5.3}$$

- If there are high priority customers of type $1,\ldots,\eta-1$ in the system, $Q(t)$ is given by

$$Q(t)=(q_1(t),\ldots,q_{\eta-1}(t),0,0,\ldots,0). \tag{5.4}$$

  Customers of lower priority are stored in the node variable $X(t)$ for this case.

- Finally, if a type-η customer is in service $Q(t)$ will be of the form

$$Q(t)=(0,\ldots,0,1,q_{\eta+1}(t),\ldots,q_K(t)). \tag{5.5}$$

It will be made clear that $q_\eta(t),q_{\eta+1}(t),\ldots,q_K(t)$ only represent some of the customers of those types in the system, while $q_1(t),\ldots,$ and $q_{\eta-1}(t)$ represent all of the customers of those types in the system. It will also be made clear that, in the Q-area, if higher priority customers are present (i.e., customer with a priority level smaller than η), there will be no lower priority customers (i.e., with a type $\geq\eta$). Finally, if $X(t)\neq\emptyset$, the type of the customer in service will be at most η. Define $S(Q(t))$ as

$$S(Q(t))=\min_{i=1}^{K}\{i:q_i(t)>0\}. \tag{5.6}$$

Hence, $S(Q(t))$ will reflect the type of the customer in service, if any, at time t, otherwise, $S(Q(t))$ is defined as $K+1$. $X(t)$ is a random variable taking the nodes of a $(K-\eta+1)$-ary tree as its values. The nodes of this tree are labeled as $\eta,\eta+1,\ldots,K$ and represent customers of the corresponding type. The process $\{X(t),(Q(t),J_s(t),J_a(t))\}$ is defined explicitly as follows.

1. If the system is empty, the system state is denoted as $(\emptyset,(\mathbf{0},0,j_a))$. In this chapter, we use $\mathbf{0}$ to make a clear distinction between a vector of zeros and a single zero, simply denoted by 0. If a customer of type i arrives to an empty system at time t, then the customer begins his service immediately, i.e., $Q(t)=e^{(i)}$, and $J_s(t)$ is determined by α_i.

2. If $Q(t) = (q_1(t), \ldots, q_{\eta-1}(t), \mathbf{0}) \neq \mathbf{0}$, i.e., the Q-area contains at least one customer of type $1, 2, \ldots,$ or $\eta - 1$, and a lower priority customer of type i, for $\eta \leq i \leq K$, arrives, the customer joins the node variable, that is, $X(t)$ becomes $X(t) + i$.

3. If $Q(t) = (q_1(t), \ldots, q_{\eta-1}(t), \mathbf{0}) \neq \mathbf{0}$ and a higher priority customer of type i (i.e., $1 \leq i \leq \eta - 1$) arrives, the customer joins the Q-area and $q_i(t)$ is updated to $\min(q_i(t) + 1, N_i)$. If i was smaller than $S(Q(t))$, then the customer who just arrived begins his service immediately, meaning that $J_s(t)$ is determined by α_i. The service of the customer who was in service is interrupted and will restart at a later time.

4. If $Q(t) = (\mathbf{0}, q_\eta(t), q_{\eta+1}(t), \ldots, q_K(t)) \neq \mathbf{0}$ and a lower priority customer of type i, for $\eta + 1 \leq i \leq K$, arrives, then the customer joins the Q-area and $q_i(t)$ is updated to $\min(q_i(t) + 1, N_i)$. Notice, $X(t) \neq \emptyset$ implies that $q_\eta(t) = 1$. If $X(t) = \emptyset$ and if i was smaller than $S(Q(t))$, the customer who just arrived begins his service immediately, while $J_s(t)$ is determined by α_i.

5. If $Q(t) = (\mathbf{0}, q_\eta(t), q_{\eta+1}(t), \ldots, q_K(t)) \neq \mathbf{0}$ and a customer of type η arrives, then $X(t)$ is updated to $X(t) + \eta$, unless $X(t) = \emptyset$ and $q_\eta(t) = 0$, then the type-η customer who just arrived begins his service immediately, $q_\eta(t)$ is updated to 1 and $J_s(t)$ is determined by α_η. The service of the possible customer who was in service is interrupted and will restart at a later time.

6. If $Q(t) = (\mathbf{0}, q_\eta(t), q_{\eta+1}(t), \ldots, q_K(t)) \neq \mathbf{0}$ and a higher priority customer of type i (i.e., $1 \leq i \leq \eta-1$) arrives, then all possible customers in the Q-area are pushed onto the node variable $X(t)$, the customer who just arrived occupies the Q-area and begins his service immediately, hence $Q(t)$ becomes $e^{(i)}$, and $J_s(t)$ is determined by α_i. The lower priority customers who were present in the Q-area join the node variable in ascending order: the type-η customer first, type-$(\eta + 1)$ customers second, ..., and type-K customers last. Thus, $X(t)$ is updated to

$$X(t) + \overbrace{\eta}^{q_\eta(t)} + \overbrace{(\eta+1)\ldots(\eta+1)}^{q_{\eta+1}(t)} + \ldots + \overbrace{K\ldots K}^{q_K(t)}.$$

7. If $S(Q(t)) = i < \eta$ and the service is completed, $q_i(t)$ is updated to $q_i(t) - 1$. Next, if $S(Q(t)) = n < K+1$, $J_s(t)$ is determined by α_n. Otherwise, we move customers from the node variable $X(t)$ to the empty Q-area as follows. Start with the rightmost customer of $X(t)$ and move the customers into the Q-area until we encounter the first type-η customer, who is also transfered to the Q-area (or until the tree becomes empty). Thus, the customers who moved into the Q-area form $Q(t) = (0, \ldots, 0, q_\eta(t), q_{\eta+1}(t), \ldots, q_K(t))$. In case the number of removed type-k customers exceeds N_k, we set $q_k(t)$ equal to N_k. If a type-η customer is found, $X(t)$ might still hold some low priority customers. In this case $J_s(t)$ is determined by α_η. If $X(t)$ became empty without encountering a type-η customer and $S(Q(t)) = n < K + 1$, then $J_s(t)$ is determined by α_n. If the tree $X(t)$ was initially empty and $S(Q(t)) = K + 1$, the system is empty and $J_s(t)$ is updated to zero.

8. If $S(Q(t)) = \eta$ and the service is completed, we start moving the rightmost customers from $X(t)$ into the Q-area until we encounter a type-η customer, who is also transfered

into the Q-area (or until the tree becomes empty). The Q-area $Q(t)$ is updated accordingly by adding those customers removed from $X(t)$. If a type-η customer is met, then $q_\eta(t)$ remains 1 and $J_s(t)$ is determined by α_η. If the tree becomes empty and $S(Q(t)) = n < K+1$, $J_s(t)$ is determined by α_n and $q_\eta(t) = 0$. If the tree was empty and $S(Q(t)) = K+1$, the system is empty and $J_s(t)$ is updated to zero.

9. If $S(Q(t)) = i > \eta$ and the service is completed, then $q_i(t)$ is updated to $q_i(t) - 1$. Note that $X(t) = \emptyset$ in this case. If $S(Q(t)) = n < K+1$, $J_s(t)$ is determined by α_n. Otherwise, the system is empty and $J_s(t)$ is updated to 0.

Event	**Q(t)**	**S(Q(t))**	**X(t)**
	(0,0,0,0,0)	K+1	$\emptyset$
arrival type 2	(0,1,0,0,0)	2	$\emptyset$
arrival type 4	(0,1,0,0,0)	2	4
arrival type 1	(1,1,0,0,0)	1	4
arrival type 3	(1,1,0,0,0)	1	43
service completion	(0,1,0,0,0)	2	43
arrival type 5	(0,1,0,0,0)	2	435
arrival type 5	(0,1,0,0,0)	2	4355
service completion	(0,0,1,0,2)	3	4
arrival type 4	(0,0,1,1,2)	3	4
arrival type 3	(0,0,1,1,2)	3	43
arrival type 1	(1,0,0,0,0)	1	433455
service completion	(0,0,1,1,2)	3	43
service completion	(0,0,1,1,2)	3	4
service completion	(0,0,0,2,2)	4	$\emptyset$
arrival type 4	(0,0,0,3,2)	4	$\emptyset$

Figure 5.1: Illustrative example of the stochastic process capturing the behavior of the 5-class priority queueing system with $\eta = 3$

Figure 5.1 presents a small example that illustrates the behavior of the stochastic process upon arrivals of new customers at the queue and service completions of customers that are being processed, without concerning the phases of the arrival and the service process. It is easy to see that the stochastic process $\{(X(t), (Q(t), J_s(t), J_a(t))), t \geq 0\}$ is a continuous-time Markov chain with a state space

$$
\begin{aligned}
\Omega \;=\; & \{(\emptyset, (\mathbf{0}, 0, j_a)) : j_a \in M_a\} \\
\cup\; & \{(\emptyset, ((\mathbf{0}, q_i, \ldots, q_K), j_s, j_a)) : i > \eta, 0 < q_i \leq N_i, 0 \leq q_j \leq N_j, j = i+1, \ldots, K, \\
& \qquad j_s \in M_i, j_a \in M_a\} \\
\cup\; & \{(J, ((\mathbf{0}, q_i, \ldots, q_{\eta-1}, \mathbf{0}), j_s, j_a)) : 0 < q_i \leq N_i, 0 \leq q_j \leq N_j, j = i+1, \ldots, \eta-1, \\
& \qquad j_s \in M_i, j_a \in M_a\} \\
\cup\; & \{(J, ((\mathbf{0}, 1, q_{\eta+1}, \ldots, q_K), j_s, j_a)) : 0 \leq q_i \leq N_i, i = \eta+1, \ldots, K, j_s \in M_\eta, j_a \in M_a\},
\end{aligned}
$$

with J a string of zero or more integers between η and K, $M_a = \{1, \ldots, m_a\}$ and $M_i = \{1, \ldots, m_i\}$ for $i = 1, \ldots, K$. The number of states associated with $(Q(t), J_s(t), J_a(t))$ for

$X(t) \neq \emptyset$ is equal to

$$m_a \left[m_\eta \left(\prod_{k=\eta+1}^{K} (1+N_k) \right) + \sum_{j=1}^{\eta-1} m_j N_j \left(\prod_{k=j+1}^{\eta-1} (1+N_k) \right) \right], \tag{5.7}$$

which reduces to

$$m_a m_1 \left[\left(\prod_{k=1}^{\eta-1} (1+N_k) \right) + \left(\prod_{k=\eta+1}^{K} (1+N_k) \right) - 1 \right], \tag{5.8}$$

for $m_1 = m_2 = \ldots = m_\eta$. It is clear that the block sizes for the tree-like process can be significantly smaller than the block sizes for the QBD process defined in Section 5.3 (see equations (5.1) and (5.2)). Figure 5.2 compares the block sizes of the matrices appearing in the two approaches for $m_a = 2$, $m_1 = \ldots = m_\eta = 3$ and $N_i = N$, for $i = 1, \ldots, \eta-1, \eta+1, \ldots, K$. This indicates that the tree-like process approach, as compared to the QBD approach, may be numerically more efficient.

Remark 8. Again, we would like to point out that $\{q_{\eta+1}(t), \ldots, q_K(t)\}$ are not the queue lengths. Denote by $q_{j,total}(t)$ the total number of type-j customers in the Q-area and the node variable $X(t)$, then $\min\{q_{j,total}(t), N_j\}$ gives the number of customers of type j in the original queueing model at time t. Notice, we only serve type $j > \eta$ customers when $X(t) = \emptyset$, thereby respecting the finiteness of these queues.

	$N = 5$		$N = 10$	
K	QBD	tree-struct.	QBD	tree-struct.
3	216	66	726	126
5	7776	426	87846	1446
7	279936	2583	1.06e7	15966

Figure 5.2: Block sizes required for the QBD approach and the tree-structured process

5.4.2 The Infinitesimal Generator

The infinitesimal generator of the process $\{(X(t), (Q(t), J_s(t), J_a(t))), t \geq 0\}$ is given in the form of matrix blocks. Mainly, we consider the transition (matrix) rate for $(X(t), Q(t))$, while the transition rates of $J_s(t)$ and $J_a(t)$ are in the matrices themselves. Therefore, the transition matrices are of size m_a or $m_a m_i$ for $1 \leq i \leq K$. If the system is empty, i.e., $X(t) = \emptyset$ and $S(Q(t)) = K+1$, the transition rate without changes is given by D_0. Upon an arrival of type i, for $1 \leq i \leq K$, the Markov chain moves to state $(\emptyset, e^{(i)})$, the rates corresponding to this state transitions are given by the transition (matrix) rate $\alpha_i \otimes D_i$.

If there is at least one customer present in the system, we can distinguish between two cases depending on the type of the customer currently being processed. Section 5.4.2.1 discusses the cases in which a high priority customer (with a priority level $< \eta$) occupies the server, while Section 5.4.2.2 considers the possible events when there are no high priority customers present, inducing that a low priority customer (with a priority level $\geq \eta$) will be served. Note that in this last case $S(Q(t)) = \eta$ if $X(t) \neq \emptyset$.

5.4.2.1 High Priority Customer in Service

Assume that $X(t) = J$, $S(Q(t)) = k \leq \eta - 1$, with $Q(t) = (\mathbf{0}, q_k, \ldots, q_{\eta-1}, \mathbf{0})$. In this case, the transition rate without changes is given by

$$T_k \oplus \left(D_0 + \sum_{\substack{i=k \\ \text{with } q_i = N_i}}^{\eta-1} D_i \right).$$

If there is an arrival of type i, for $1 \leq i \leq k-1$, the Markov chain moves to state $(J, (\mathbf{0}, q_k, \ldots, q_{\eta-1}, \mathbf{0}) + e^{(i)})$ with transition rate $e\alpha_i \otimes D_i$. Analogously, an arrival of type i, for $k \leq i \leq \eta - 1$ and $q_i < N_i$ causes the Markov chain to move to state $(J, (\mathbf{0}, q_k, \ldots, q_{\eta-1}, \mathbf{0}) + e^{(i)})$ with transition rate $I \otimes D_i$. Finally, if there is an arrival of type i, for $\eta \leq i \leq K$, the Markov chain moves to state $(J + i, (\mathbf{0}, q_k, \ldots, q_{\eta-1}, \mathbf{0}))$ with transition rate $I \otimes D_i$.

Besides this, we need to consider the service completions. If the customer being processed completes his service, we set q_k to be $q_k - 1$. If $S(Q(t)) = n < K + 1$, the chain moves to state $(J, (\mathbf{0}, q_n, \ldots, q_{\eta-1}, \mathbf{0}))$ with transition rate $(-T_k e\alpha_n) \otimes I$. If $S(Q(t)) = K + 1$ and $J = \emptyset$, the Markov chain moves to the empty state with transition rate $(-T_k e) \otimes I$. On the other hand, if $J \neq \emptyset$ and $S(Q(t)) = K + 1$, the Markov chain moves to state $(J', (\mathbf{0}, q_\eta, q_{\eta+1}, \ldots, q_K))$ with transition rate $(-T_k e\alpha_{n'}) \otimes I$. The customers of type $q_\eta, q_{\eta+1}, \ldots, q_K$ in the Q-area were the rightmost elements of J up to and including the first η in J. J' is obtained by removing these elements and $S(Q(t)) = n' < K + 1$. If there was no type-η customer in J (or only the leftmost equaled η), then $J' = \emptyset$.

5.4.2.2 Low Priority Customer in Service

Assume that $X(t) = J$, $S(Q(t)) = k \geq \eta$, with $Q(t) = (\mathbf{0}, q_k, \ldots, q_K)$. Recall that $S(Q(t)) = \eta$ if $J \neq \emptyset$. The transition rate without changes in $(X(t), Q(t))$ equals

$$T_k \oplus \left(D_0 + \sum_{\substack{i=k \\ \text{with } q_i = N_i}}^{K} D_i \right).$$

If there is an arrival of type i, for $1 \leq i \leq \eta - 1$, the Markov chain moves to state $(J + k \ldots k(k+1) \ldots (k+1) \ldots K \ldots K, e^{(i)})$ with transition (matrix) rate $e\alpha_i \otimes D_i$. Note that all customers who were in the Q-area moved to the tree $X(t)$, i.e., exactly q_j type-j customers are added to $X(t)$, for $j = k, \ldots, K$. For the other arrivals, we distinguish between two cases: $k > \eta$ and $k = \eta$.

In the first case, an arrival of type i causes a transition to state $(J, (\mathbf{0}, q_k, \ldots, q_K) + e^{(i)})$ with transition (matrix) rate $e\alpha_i \otimes D_i$, for $\eta \leq i < k$. If there is an arrival of type $i > \eta$, for $k \leq i \leq K$ and $q_i < N_i$, there is a transition to state $(J, (\mathbf{0}, q_k, \ldots, q_K) + e^{(i)})$ with transition rate $I \otimes D_i$. In the case where $k = \eta$ the Markov chain moves to state $(J + \eta, (\mathbf{0}, 1, q_{\eta+1}, \ldots, q_K))$ with transition rate $I \otimes D_\eta$ when a customer of type η arrives. When there is an arrival of type $\eta < i \leq K$, there is a transition to state $(J, (\mathbf{0}, 1, q_{\eta+1}, \ldots, q_K) + e^{(i)})$ with transition rate $I \otimes D_i$.

Let us finally describe the service completion events. If there is a service completion, we set q_k to be $q_k - 1$. Let us start with the case where $J = \emptyset$ and $S(Q(t)) = n < K + 1$. The

chain then moves to state $(\emptyset, (\mathbf{0}, q_n, \ldots, q_K))$ with transition rate $(-T_k e \alpha_n) \otimes I$. If $J = \emptyset$ and $S(Q(t)) = K+1$, the Markov chain moves to the empty state with transition rate $(-T_k e) \otimes I$. If $J \neq \emptyset$ and there are customers of type η in J, the Markov chain moves to state

$$(J', (\mathbf{0}, 1, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K)),$$

with transition rate $(-T_\eta e \alpha_\eta) \otimes I$, where customers $r_{\eta+1}, \ldots, r_K$ are the rightmost elements of J appearing before η, J' is obtained by removing these types and η as well. Note that we set the number of type k customers equal to N_k if $q_k + r_k$ exceeds N_k. If $J \neq \emptyset$ and there is no customer of type η in J, the Markov chain moves to state $(\emptyset, (\mathbf{0}, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K))$ with transition rate $(-T_\eta e \alpha_n) \otimes I$, where $n = S(Q(t))$.

5.4.3 Constructing a Tree-Like QBD Process

The Markov chain $\{(X(t), (Q(t), J_s(t), J_a(t))), t \geq 0\}$ does not have a simple transition structure such as a tree-structured M/G/1, GI/M/1, or QBD structure. More specifically, there are two cases in which a transition occurs that does not fit within the tree-like QBD paradigm. First, the arrival of a high priority customer during the service of a low priority one causes the transfer of the entire Q-area contents to the tree and secondly, if $X(t) \neq \emptyset$, a service completion might cause a shift to the Q-area of all customers in $X(t)$ until the first type-η customer (including this last customer). This induces that the level of the Markov process can both increase or decrease with an arbitrary integer value. As an implication the computation of the limiting probabilities will be quite complex. Fortunately, the process can be transformed into a (discrete-time) tree-like QBD process as follows.
As a first step we transform the continuous-time Markov chain $\{(X(t), (Q(t), J_s(t), J_a(t))), t \geq 0\}$ into a discrete-time process by applying a standard uniformization argument. Let Q be the original infinitesimal generator matrix, then P is found by setting

$$P = \frac{Q}{\lambda} + I. \tag{5.9}$$

In this equation λ is the maximum rate at which events occur in Q. This maximum rate λ equals

$$\lambda = -\min_{k=1}^{K} \theta_k - \delta, \tag{5.10}$$

where δ is the smallest diagonal element of D_0 and θ_k the smallest diagonal element of T_k. Notice, both chains have the same steady state, i.e., if $\pi Q = \mathbf{0}$ and $\pi e = 1$, then $\pi P = \pi$ as well.

Let us now reduce this discrete-time chain into a tree-like QBD. To do so, we introduce a new variable $W(t)$ taking three values $\{-, 0, +\}$. The Markov chain $\{(X(t), (Q(t), J_s(t), J_a(t))), t \geq 0\}$ is extended to $\{(X(t), (W(t), Q(t), J_s(t), J_a(t))), t \geq 0\}$. Abusing the notation a little bit, we will use $X(t), Q(t), J_s(t)$, and $J_a(t)$ for both Markov chains. It will become clear that the addition of this variable $W(t)$ increases the number of states for fixed $X(t) \neq \emptyset$ by

$$2m_a \left(\prod_{k=\eta+1}^{K} (1 + N_k) \right).$$

Figure 5.3 presents the block sizes of the matrices representing the tree-like process for the same values of m_a, $m_1, \ldots, m_\eta$ and N_i $(i = 1, \ldots, \eta - 1, \eta + 1, \ldots, K)$ as in Figure 5.2. This figure shows that, although a number of states has been added due to the introduction of $W(t)$, the tree-like approach still requires significantly less states compared to the classical QBD approach.

	$N = 5$		$N = 10$	
K	QBD	tree-like	QBD	tree-like
3	216	90	726	170
5	7776	570	87846	1930
7	279936	3447	1.06e7	21290

Figure 5.3: Block sizes required for the QBD approach and the tree-like process

We consider the following three cases according to the values of $W(t)$. The states with $W(t) = 0$ correspond to the discrete-time Markov chain. The remaining two values $\{-, +\}$ are used to split the two conflicting transitions into multiple steps such that each complies with the transitions allowed in tree-like QBD processes as described in Sections 5.4.3.1 and 5.4.3.2 and illustrated by Figure 5.4.

5.4.3.1 Transferring Customers from the Q-Area to the Tree

If the Markov chain $\{(X(t), (Q(t), J_s(t), J_a(t))), t \geq 0\}$ is transferring customers from the Q-area to the node variable $X(t)$, we set $W(t) = +$. If $W(t) = +$, then the Markov chain will stay in states with $W(t) = +$ until all customers present in the Q-area are transferred to $X(t)$ and afterward $W(t)$ is set to 0. For the extended process, the transfer of customers from the Q-area to $X(t)$ is done one at a time. During this period of time, both the arrival and the service process are frozen. Let $k \geq \eta$ denote the type of the customer in service. Assume that an arrival of a type-i customer, for $1 \leq i \leq \eta - 1$, causes the need to transfer the contents of the Q-area onto $X(t)$. We will first transfer this contents before actually determining the type of the arriving customer or the initial phase of the service. By postponing this decision we need fewer states with $W(t) = +$. Let

$$\Delta = \frac{diag[\sum_{i=1}^{\eta-1} D_i e]}{\lambda}. \tag{5.11}$$

Hence, the j-th diagonal entry of the matrix Δ gives us the probability that such an arrival occurs while the arrival process is in state j. Note, in case some of these values are zero, we can simply remove them from Δ to end up with fewer states with $W(t) = +$. Thus, the Markov chain will first move to a state of the form $(J + k, +, (\mathbf{0}, q_k - 1, \ldots, q_K))$ with probability $e \otimes \Delta$. Then the Markov chain moves to $(J + kk, +, (\mathbf{0}, q_k - 2, \ldots, q_K))$ with probability 1. This process continues until all customers $q_k, \ldots,$ and q_K have moved from the Q-area to $X(t)$, meaning we end up in

$$(J + k \ldots k(k+1) \ldots (k+1) \ldots K \ldots K, +, \mathbf{0}).$$

Next, the Markov chain moves to

$$(J + k \ldots k(k+1) \ldots (k+1) \ldots K \ldots K, 0, e^{(i)})$$

with probability $\alpha_i \otimes \Delta^{-1} D_i / \lambda$.

Event	Q(t)	W(t)	S(Q(t))	X(t)
	(0,0,0,0,0)	0	K+1	$\emptyset$
arrival type 2	(0,1,0,0,0)	0	2	$\emptyset$
arrival type 4	(0,1,0,0,0)	0	2	4
arrival type 1	(1,1,0,0,0)	0	1	4
arrival type 3	(1,1,0,0,0)	0	1	43
service completion	(0,1,0,0,0)	0	2	43
arrival type 5	(0,1,0,0,0)	0	2	435
arrival type 5	(0,1,0,0,0)	0	2	4355
service completion	(0,0,0,0,0)	-		4355
	(0,0,0,0,1)	-		435
	(0,0,0,0,2)	-		43
	(0,0,1,0,2)	0	3	4
arrival type 4	(0,0,1,1,2)	0	3	4
arrival type 3	(0,0,1,1,2)	0	3	43
arrival type 1	(0,0,0,1,2)	+		433
	(0,0,0,0,2)	+		4334
	(0,0,0,0,1)	+		43345
	(0,0,0,0,0)	+		433455
	(1,0,0,0,0)	0	1	433455
service completion	(0,0,0,0,0)	-		433455
	(0,0,0,0,1)	-		43345
	(0,0,0,0,2)	-		4334
	(0,0,0,1,2)	-		433
	(0,0,1,1,2)	0	3	43
service completion	(0,0,0,1,2)	-		43
	(0,0,1,1,2)	0	3	4
service completion	(0,0,0,1,2)	-		4
	(0,0,0,2,2)	-		$\emptyset$
	(0,0,0,2,2)	0	4	$\emptyset$
arrival type 4	(0,0,0,3,2)	0	4	$\emptyset$

Figure 5.4: Illustrative example from Figure 5.1 ($\eta = 3$) revisited after the transformation of the Markov process into a tree-like QBD process

5.4.3.2 Transferring Customers from the Tree to the Q-Area

If the Markov chain $\{(X(t), (Q(t), J_s(t),\ J_a(t))), t \geq 0\}$ is transferring customers from the node variable $X(t)$ to the Q-area, we set $W(t) = -$. If $W(t) = -$, then the Markov chain will stay in states with $W(t) = -$ until a type-η customer is removed or until the tree becomes empty. After that, $W(t) = 0$. Similar to the states with $W(t) = +$, the transfer of customers from $X(t)$ to the Q-area is done one at a time in the extended process and while $W(t) = -$, there is no service and no arrival. Assume that $J \neq \emptyset$ and denote $k \leq \eta$ as the type of the customer that completes his service and that causes us to remove customers from $X(t)$. Let $u_n u_{n-1} \dots u_1$ be the rightmost elements of J that need to be added to the Q-area. Hence, either $J = u_n u_{n-1} \dots u_1$, with $u_1, \dots, u_n > \eta$ or $u_n = \eta$ and $u_1, \dots, u_{n-1} > \eta$.

The original transition we have to split up to turn the process into a tree-like QBD process is a transition from some state $(J, 0, (\mathbf{0}, q_\eta, q_{\eta+1}, \ldots, q_K))$ to

1. a state $(J', 0, (\mathbf{0}, 1, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K))$, where J' is obtained by removing the elements $u_n u_{n-1} \ldots u_1$ from J, if $u_n = \eta$,

2. a state $(\emptyset, 0, (\mathbf{0}, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K))$, if $u_n \neq \eta$.

Here, r_i denotes the number of type i customers in $u_n u_{n-1} \ldots u_1$ $(i = \eta + 1, \ldots, K)$. If the type of the arriving customer $k = \eta$, the chain will first move to state $(J, -, (\mathbf{0}, q_{\eta+1}, \ldots, q_K))$ with probability $(-T_k)e/\lambda \otimes I$, otherwise the chain will move to state $(J, -, \mathbf{0})$ with this probability. Next, with probability 1, we visit

$$(J - u_1, -, (\mathbf{0}, q_{\eta+1}, \ldots, q_K) + e^{(u_1)}),$$

where $q_{\eta+1} = \ldots = q_K = 0$ if $k < \eta$. Afterward we visit

$$(J - u_1 - u_2, -, (\mathbf{0}, q_{\eta+1}, \ldots, q_K) + e^{(u_1)} + e^{(u_2)}),$$

etc. This process continues until all customers $u_1, \ldots, u_{n-1}$ are part of the Q-area, thereby reaching

$$(J' + u_n, -, (\mathbf{0}, 0, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K) - e^{(u_n)}).$$

Finally, if $u_n = \eta$, the Markov chain moves to the state

$$(J', 0, (\mathbf{0}, 1, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K))$$

with probability $\alpha_\eta \otimes I$. Notice, if $u_n = \eta$, the rightmost element of $X(t)$ during the last transition equaled η, which is used as a criteria to switch back to a state with $W(t) = 0$. Otherwise, if $u_n \neq \eta$, i.e., $J' = \emptyset$, we first visit state

$$(\emptyset, -, (\mathbf{0}, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K))$$

with probability 1 and afterward state

$$(\emptyset, 0, (\mathbf{0}, q_{\eta+1} + r_{\eta+1}, \ldots, q_K + r_K))$$

with probability $\alpha_{n'} \otimes I$, where $S(Q(t)) = n' < K + 1$. Moreover, in case the maximum buffer size of some type of customer is reached while removing customers from the string, we do not further increase the number customers of this type in the Q-area.

5.5 K-Class Priority Queue as a Tree-Like Process

From the previous section we learned that the K-class priority queue can be modeled as a tree-like process. For the definition and basic properties of such a process, we refer to Section 2.1.3. Provided that the tree-like process $\{(X_t, (W(t), Q(t), J_s(t), J_a(t))), t \geq 0\}$ is

ergodic, define

$$\begin{aligned}
\bar{\pi}_J(w,q,s,a) &= \lim_{t\to\infty} P[X_t = J, (W(t),Q(t),J_s(t),J_a(t)) = (w,q,s,a)],\\
\bar{\pi}_J(w,q,s) &= (\bar{\pi}_J(w,q,s,1),\bar{\pi}_J(w,q,s,2),\ldots,\bar{\pi}_J(w,q,s,m_a)),\\
\bar{\pi}_J(w,q) &= (\bar{\pi}_J(w,q,1),\bar{\pi}_J(w,q,2),\ldots,\bar{\pi}_J(w,q,m_{S(q)})),
\end{aligned}$$

$$\bar{\pi}_J(w) = \begin{cases}
(\bar{\pi}_J(w,(\mathbf{0},0,\mathbf{0})),\ldots,\bar{\pi}_J(w,(\mathbf{0},0,0,\ldots,N_K)),\\
\bar{\pi}_J(w,(\mathbf{0},0,\ldots,1,0)),\ldots,\bar{\pi}_J(w,(\mathbf{0},0,\ldots,1,N_K)),\\
\ldots,\\
\bar{\pi}_J(w,(\mathbf{0},0,N_{\eta+1},\ldots,N_K))), & \text{for } w=+,-,\\
(\bar{\pi}_J(w,(0,\ldots,1,0,\mathbf{0})),\ldots,\bar{\pi}_J(w,(N_1,\ldots,N_{\eta-1},0,\mathbf{0})),\\
\bar{\pi}_J(w,(\mathbf{0},1,\mathbf{0})),\bar{\pi}_J(w,(\mathbf{0},1,0,\ldots,1)),\\
\ldots,\\
\bar{\pi}_J(w,(\mathbf{0},1,N_{\eta+1},\ldots,N_K))), & \text{for } w=0, J\neq\emptyset,\\
(\bar{\pi}_J(w,(0,\ldots,1,0,\mathbf{0})),\ldots,\bar{\pi}_J(w,(N_1,\ldots,N_{\eta-1},0,\mathbf{0})),\\
\bar{\pi}_J(w,(\mathbf{0},1,\mathbf{0})),\bar{\pi}_J(w,(\mathbf{0},1,0,\ldots,1)),\ldots,\\
\bar{\pi}_J(w,(\mathbf{0},1,N_{\eta+1},\ldots,N_K)),\bar{\pi}_J(w,(\mathbf{0},0,\mathbf{0})),\\
\ldots,\\
\bar{\pi}_J(w,(\mathbf{0},0,N_{\eta+1},\ldots,N_K))), & \text{for } w=0, J=\emptyset,
\end{cases}$$

$$\bar{\pi}_J = \begin{cases}
(\bar{\pi}_J(-),\bar{\pi}_J(0)), & \text{for } J=\emptyset,\\
(\bar{\pi}_J(+),\bar{\pi}_J(-),\bar{\pi}_J(0)), & \text{for } J\neq\emptyset.
\end{cases}$$

The vectors $\bar{\pi}_J$ can be computed from $\bar{\pi}_\emptyset$ using the relations

$$\bar{\pi}_k = \bar{\pi}_\emptyset R_{\emptyset,k} \tag{5.12}$$

$$\bar{\pi}_{J+k} = \bar{\pi}_J R_k, \text{ for } J\neq\emptyset, \tag{5.13}$$

for $k=\eta,\ldots,K$ and $\bar{\pi}_\emptyset$ is found by solving the boundary condition

$$\bar{\pi}_\emptyset = \bar{\pi}_\emptyset\left(\sum_{k=\eta}^{K} R_{\emptyset,k}\bar{D}_{\emptyset,k} + \bar{F}\right)$$

with the normalizing restriction that $\bar{\pi}_\emptyset e + \bar{\pi}_\emptyset R_\emptyset (I-R)^{-1}e = 1$. The matrices $R_{\emptyset,k}$ are given by $R_{\emptyset,k} = \bar{U}_{\emptyset,k}(I-V)^{-1}$, with

$$V = \bar{B} + \sum_{s=\eta}^{K} \bar{U}_s G_s, \tag{5.14}$$

$$G_k = (I-V)^{-1}\bar{D}_k, \tag{5.15}$$

and $R_\emptyset = R_{\emptyset,\eta} + \ldots + R_{\emptyset,K}$. Recall that we added some states to end up with this tree-like process. Consequently, the steady state probabilities of the original process can be obtained by applying an appropriate normalization, that is, by taking into account only those states with $W(t)=0$.

5.5.1 Characterization of the Tree-Like Process

Let us now have a closer look at the different matrices characterizing the tree-like process presented in Section 5.4. As mentioned before, at each step the chain can only make a transition to its parent (i.e., $X_{t+1} = X_t - f(X_t, 1)$, for $X_t \neq \emptyset$), to itself ($X_{t+1} = X_t$), or to one of its own children ($X_{t+1} = X_t + s$ for some $\eta \leq s \leq K$). Moreover, the chain's state at time $t+1$ is determined as follows:

$$P[(X_{t+1}, N_{t+1}) = (J', j) | (X_t, N_t) = (J, i)] =$$
$$\begin{cases} \bar{f}^{i,j} & J' = J = \emptyset, \\ \bar{b}^{i,j} & J' = J \neq \emptyset, \\ \bar{d}_k^{i,j} & J \neq \emptyset, f(J,1) = k, J' = J - f(J,1), \\ \bar{u}_s^{i,j} & J' = J + s, s = \eta, \ldots, K, \\ 0 & otherwise. \end{cases}$$

First, let us define the following numbers:

$$\mu_j = \prod_{k=j}^{K} (N_k + 1), \tag{5.16}$$

$$\kappa_j = \prod_{k=j}^{\eta-1} (N_k + 1), \tag{5.17}$$

$$\nu_j = N_j \kappa_{j+1}, \tag{5.18}$$

$$\nu^* = \sum_{k=1}^{\eta-1} \nu_k m_k \tag{5.19}$$

Recall that by definition $\prod_{k=i}^{j} x_k = 1$ if $i > j$. Then, define the matrices $\bar{D}_s$, $\bar{B}$ and $\bar{U}_s$ of dimension $m_a(\mu_{\eta+1}(m_\eta + 2) + \nu^*)$ with respective (i,j)-th elements given by $\bar{d}_s^{i,j}$, $\bar{b}^{i,j}$ and $\bar{u}_s^{i,j}$. These matrices that characterize the tree-like process can be divided into 4 levels, i.e., the first level corresponds to states with $W(t) = +$, the second level to states with $W(t) = -$, the third and fourth level to states with $W(t) = 0$ and $S(Q(t)) = k$, where $k < \eta$ (high priority) and $k = \eta$ (average priority), respectively.

Within these levels the inner variable denotes the state of the arrival process, the second one holds the phase of the service process (if necessary), the third variable denotes the number of customers of the lowest priority present in the Q-area, the fourth variable corresponds to the one but lowest priority, etc., until the outermost variable which denotes the number of customers of the highest priority present in the Q-area. Hence, if we order the variables from the outside to the inside, the priorities for the states with $W(t) = +, -$, or 0 with $S(Q(t)) = \eta$, resp. $W(t) = 0$ and $S(Q(t)) < \eta$, are ordered as $\eta + 1, \ldots, K$, resp. as $1, 2, \ldots, \eta - 1$.

The matrices $\bar{D}_s$, for $s = \eta + 1, \ldots, K$ contain the transition probabilities for transitions of the form $(J + s, -, (\mathbf{0}, 0, q_{\eta+1}, \ldots, q_K) - e^{(s)}) \to (J, -, (\mathbf{0}, 0, q_{\eta+1}, \ldots, q_K))$ or of the form $(J + s, -, (\mathbf{0}, 0, q_{\eta+1}, \ldots, q_K)) \to (J, -, (\mathbf{0}, 0, q_{\eta+1}, \ldots, q_K))$ if $q_s = N_s$. Since the probability on such a transition equals one, these matrices are given by

$$\bar{D}_s = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \bar{D}_s^{--} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{ where } \bar{D}_s^{--} = I_{\frac{\mu_{\eta+1}}{\mu_s}} \otimes I_{N_s+1}^{+} \otimes I_{\mu_{s+1}} \otimes I_{m_a} \tag{5.20}$$

is a matrix of dimension $\mu_{\eta+1}m_a$. The matrix $\bar{D}_\eta$ covers the transitions from a state $(J'+\eta,-,(\mathbf{0},0,q_{\eta+1},\ldots,q_K))$ to state $(J',0,(\mathbf{0},1,q_{\eta+1},\ldots,q_K))$ and can be written as

$$\bar{D}_\eta = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \bar{D}_\eta^{-a} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{ with } \bar{D}_\eta^{-a} = I_{\mu_{\eta+1}} \otimes \alpha_\eta \otimes I_{m_a}. \tag{5.21}$$

The $\bar{U}_s$ matrices describe the transitions from a node to one of its children and are given by

$$\bar{U}_s = \begin{bmatrix} \bar{U}_s^{++} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{U}_s^{hh} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \bar{U}_\eta = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{U}_\eta^{hh} & 0 \\ \bar{U}_\eta^{a+} & 0 & 0 & \bar{U}_\eta^{aa} \end{bmatrix}, \quad \text{for } s > \eta. \tag{5.22}$$

In these equations the $\mu_{\eta+1}m_a \times \mu_{\eta+1}m_a$ matrix $\bar{U}_s^{++}$ is given by

$$\bar{U}_s^{++} = \begin{bmatrix} I_{N_s+1}^{-} \otimes I_{\mu_{s+1}} \otimes I_{m_a} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{5.23}$$

and the $\nu^* m_a \times \nu^* m_a$ matrix $\bar{U}_s^{hh}$ by

$$\bar{U}_s^{hh} = I_{\nu^*} \otimes D_s/\lambda. \tag{5.24}$$

Analogously, one obtains

$$\bar{U}_\eta^{hh} = I_{\nu^*} \otimes D_\eta/\lambda, \tag{5.25}$$

$$\bar{U}_\eta^{a+} = I_{\mu_{\eta+1}} \otimes e \otimes \Delta, \tag{5.26}$$

$$\bar{U}_\eta^{aa} = I_{\mu_{\eta+1}m_\eta} \otimes D_\eta/\lambda, \tag{5.27}$$

where Δ is given by equation (5.11). The matrix $\bar{B}$ that holds the probabilities that the tree-like process remains in the same node can be written as

$$\bar{B} = \begin{bmatrix} 0 & 0 & \bar{B}^{+h} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \bar{B}^{h-} & \bar{B}^{hh} & 0 \\ 0 & \bar{B}^{a-} & 0 & \bar{B}^{aa} \end{bmatrix}.$$

The matrix $\bar{B}^{+h}$ covering the transitions out of a state with $W(t) = +$ has only nonzero entries on the first block row (of size m_a). Moreover, only the first blocks of the block vectors on this row, i.e., the blocks that correspond to a state of the form $(J,0,e^{(k)})$, differ from zero, hence,

$$\bar{B}^{+h} = \begin{bmatrix} \bar{B}_{\eta-1}^{+h} & \bar{B}_{\eta-2}^{+h} & \cdots & \bar{B}_1^{+h} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{5.28}$$

where $\bar{B}_k^{+h}$ represents the $m_a \times \nu_k m_k m_a$ matrix

$$\bar{B}_k^{+h} = \begin{bmatrix} \alpha_k \otimes \Delta^{-1} D_k & 0 & \cdots & 0 \end{bmatrix}. \tag{5.29}$$

The matrices $\bar{B}^{a-}$ and $\bar{B}^{h-}$ hold the probabilities corresponding to transitions to a state with $W(t) = -$. Such a transition takes place if the customer that is currently in service leaves the system. If this customer is of type η, a transition to a state with $W(t) = -$ is made while keeping the Q-area contents unaltered, hence

$$B^{a-} = I_{\mu_{\eta+1}} \otimes (-T_\eta)e/\lambda \otimes I_{m_a}. \tag{5.30}$$

If a high priority customer completes his service, a transition to a state $(J, -, \mathbf{0})$ only occurs when there is no other customer present in the Q-area, i.e.,

$$\bar{B}^{h-} = \begin{bmatrix} \bar{B}_{\eta-1}^{h-} & 0 & \cdots & 0 \\ \bar{B}_{\eta-2}^{h-} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{B}_1^{h-} & 0 & \cdots & 0 \end{bmatrix}, \text{ with } \bar{B}_k^{h-} = \begin{bmatrix} (-T_k)e/\lambda \otimes I_{m_a} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.31}$$

a $\nu_k m_k m_a \times m_a$ block column vector. The probability of having a transition from a state with $S(Q(t)) = \eta$ to a state with $S(Q(t)) = \eta$ can be found in

$$\begin{aligned} \bar{B}^{aa} &= I_{\mu_{\eta+1}} \otimes \left(I_{m_\eta m_a} + T_\eta/\lambda \oplus D_0/\lambda\right) \\ &+ \sum_{k=\eta+1}^{K} I_{\frac{\mu_{\eta+1}}{\mu_k}} \otimes I_{N_k+1}^{+} \otimes I_{\mu_{k+1}} \otimes \left(I_{m_\eta} \otimes D_k/\lambda\right). \end{aligned} \tag{5.32}$$

The last term in this equation describes the transitions in which a lower priority customer arrives while a customer of type η occupies the server. Finally, $\bar{B}^{hh}$ is given by

$$\bar{B}^{hh} = \begin{bmatrix} \bar{B}_{\eta-1,\eta-1}^{hh} & \bar{B}_{\eta-1,\eta-2}^{hh} & \cdots & \bar{B}_{\eta-1,1}^{hh} \\ \bar{B}_{\eta-2,\eta-1}^{hh} & \bar{B}_{\eta-2,\eta-2}^{hh} & \cdots & \bar{B}_{\eta-2,1}^{hh} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{B}_{1,\eta-1}^{hh} & \bar{B}_{1,\eta-2}^{hh} & \cdots & \bar{B}_{1,1}^{hh} \end{bmatrix},$$

with $\bar{B}_{k,l}^{hh}$ a $\nu_k m_k m_a \times \nu_l m_l m_a$ matrix that can be expressed by one of the following equations:

$$\begin{aligned} \bar{B}_{k,k}^{hh} &= I_{\nu_k} \otimes (I_{m_k m_a} + T_k/\lambda \oplus D_0/\lambda) + \sum_{i=k+1}^{\eta-1} I_{\nu_k/\kappa_i} \otimes I_{N_i+1}^{+} \otimes I_{\kappa_{i+1}} \otimes I_{m_k} \otimes D_i/\lambda \\ &+ I_{N_k}^{+} \otimes I_{\kappa_{k+1}} \otimes I_{m_k} \otimes D_k/\lambda + I_{N_k}^{-} \otimes I_{\kappa_{k+1}} \otimes (-T_k)e/\lambda\alpha_k \otimes I_{m_a}, \end{aligned} \tag{5.33}$$

$$\bar{B}_{k,l}^{hh} = \begin{bmatrix} 0 & I_{\nu_k} \otimes e\alpha_l \otimes D_l/\lambda & 0 \end{bmatrix}, \text{ for } k > l, \tag{5.34}$$

$$\bar{B}_{k,l}^{hh} = \begin{bmatrix} 0 \\ I_{\nu_l} \otimes (-T_k)e/\lambda\alpha_l \otimes I_{m_a} \\ 0 \end{bmatrix}, \text{ for } k < l. \tag{5.35}$$

Because equation (5.34) describes the probabilities of a new arrival of a customer of type l when a customer of type $k > l$ is being served, the first $\kappa_{k+1} m_l m_a$ columns will contain only zero entries. That is, these columns correspond to exactly those states with $q_l = 1$ and $q_i = 0$, for $k \geq i \neq l$. The nonzero entries then correspond to the states with $q_l = 1$, $q_k > 0$ and $q_i = 0$ $(k > i \neq l)$. Similarly, the number of leading zero rows in equation (5.35) equals $\kappa_{l+1} m_k m_a$.

5.5.2 Computing the V-Matrix

From the matrices $\bar{D}_k$, $\bar{B}$ and $\bar{U}_s$, the matrix V can be found using equation (2.9). Taking into account the zero entries of the transition matrices we obtain the following structure for this matrix V:

$$V = \begin{bmatrix} 0 & V^{+-} & V^{+h} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & V^{h-} & V^{hh} & V^{ha} \\ 0 & V^{a-} & 0 & V^{aa} \end{bmatrix}.$$

Keeping in mind its probabilistic interpretation, the zero entries of V can be understood as follows.

- There are no paths possible between a state of node $J+k$ with $W(t) = -$ and another arbitrary state of node $J+k$ under taboo of node J since $W(t) = -$ invokes a transition from a node to its parent node.
- In an analogous manner, transitions to a state with $W(t) = +$ are transitions from a parent to one of its children. Hence, there is no path from an arbitrary state of node $J+k$ to a state of the same node with $W(t) = +$ without visiting node J.
- Starting from a state of node J with $W(t) = +$, there are no paths that visit only node J and its descendants and end in a state of node J with $S(Q(t)) = \eta$. Because $W(t) = +$, the chain will first move from node J to a child $J+k$, with $k > \eta$ (except when the current state is of the form $(J, +, \mathbf{0})$, resulting in a transition to a state $(J, 0, e^{(j)})$ with $j < \eta$). When the chain eventually returns to node J from node $J+k$ through a state with $W(t) = -$, k will be removed from the stack and $W(t) = -$ remains unchanged since $k > \eta$, causing a transition to the parent of node J.
- Finally, a customer of type η is only served if there are no customers present of type $1, \ldots, \eta - 1$. If a customer with a higher priority arrives, the type-η customer will be pushed on the stack, inducing a transition from node J to node $J + \eta$. The chain will only return to node J after all high priority customers have received their service, causing the zero probabilities for paths between a state with $S(Q(t)) = \eta$ and a state of the same node with $S(Q(t)) < \eta$.

The matrix V can be computed by a standard fixed point iteration [15]. That is, the matrix V is obtained as $\lim_{N\to\infty} V_N$ from the recursion

$$V_{N+1} = \bar{B} + \sum_{s=\eta}^{K} \bar{U}_s (I - V_N)^{-1} \bar{D}_s, \tag{5.36}$$

where $V_0 = \bar{B}$. We can however benefit from the structural properties of the $\bar{B}$, $\bar{U}_s$ and $\bar{D}_s$ matrices involved to make this recursion faster. Moreover, the reasoning put forward in the third and the fourth bullet above implies that

$$V^{+h} = \bar{B}^{+h}, \tag{5.37}$$

$$V^{a-} = \bar{B}^{a-}. \tag{5.38}$$

Starting from a state of node J with $W(t) = 0$ and $S(Q(t)) < \eta$, there are no paths that visit at least one of the descendants of J, none of its ancestors and that end in a state of the same node J with $W(t) = 0$ and $S(Q(t)) < \eta$. Upon returning from some node $J + k$ to node J, we have $W(t) = -$, followed by a transition to the parent node of J or $W(t) = 0$ and $S(Q(t)) = \eta$. Therefore, the block V^{hh} can also be extracted immediately from the matrix $\bar{B}$, namely,

$$V^{hh} = \bar{B}^{hh}. \tag{5.39}$$

Define W_N as $W_N = (I - V_N)^{-1} = \sum_{i=0}^{\infty} V_N^i$, which can be written as

$$W_N = \begin{bmatrix} I & W_N^{+-} & W_N^{+h} & W_N^{+a} \\ 0 & I & 0 & 0 \\ 0 & W_N^{h-} & W_N^{hh} & W_N^{ha} \\ 0 & W_N^{a-} & 0 & W_N^{aa} \end{bmatrix}.$$

The different blocks of this matrix can be found using

$$W_N^{aa} = (I - V_N^{aa})^{-1} \tag{5.40}$$
$$W_N^{a-} = W_N^{aa} V_N^{a-} \tag{5.41}$$
$$W_N^{hh} = (I - V_N^{hh})^{-1} \tag{5.42}$$
$$W_N^{ha} = W_N^{hh} V_N^{ha} W_N^{aa} \tag{5.43}$$
$$W_N^{+a} = V_N^{+h} W_N^{ha} \tag{5.44}$$
$$W_N^{h-} = W_N^{hh} V_N^{h-} + W_N^{ha} V_N^{a-} \tag{5.45}$$
$$W_N^{+h} = V_N^{+h} W_N^{hh} \tag{5.46}$$
$$W_N^{+-} = V_N^{+-} + V_N^{+h} W_N^{h-} \tag{5.47}$$

Since V_N^{+h} and V_N^{hh} are independent of N, W_N^{+h} and W_N^{hh} need to be computed only once. Additionally, we can benefit from the structure of V^{+-} and V^{aa}. Define

$$S_k^* = \sum_{i=k}^{\infty} S_i, \tag{5.48}$$

then V^{+-} and V^{aa} have the following upper triangular Toeplitz-like structure (see page 93 for a detailed explanation):

$$S = \begin{bmatrix} S_0 & S_1 & \cdots & S_{N_{\eta+1}}^* \\ 0 & S_0 & \cdots & S_{N_{\eta+1}-1}^* \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & S_0^* \end{bmatrix}. \tag{5.49}$$

In this equation, the dimension of the square blocks S_i equals $C \prod_{l=\eta+2}^{K} (N_l + 1)$, where C denotes the dimension of the matrices $S_{i_1,\ldots,i_{K-\eta-1},i_{K-\eta}}$ (see further). The blocks S_i can be further partitioned into smaller blocks $S_{i,k}$, i.e.,

$$S_i = \begin{bmatrix} S_{i,0} & S_{i,1} & \cdots & S_{i,N_{\eta+2}}^* \\ 0 & S_{i,0} & \cdots & S_{i,N_{\eta+2}-1}^* \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & S_{i,0}^* \end{bmatrix}.$$

This structure repeats itself until

$$S_{i_1,\ldots,i_{K-\eta-1}} = \begin{bmatrix} S_{i_1,\ldots,i_{K-\eta-1},0} & S_{i_1,\ldots,i_{K-\eta-1},1} & \cdots & S^*_{i_1,\ldots,i_{K-\eta-1},N_K} \\ 0 & S_{i_1,\ldots,i_{K-\eta-1},0} & \cdots & S^*_{i_1,\ldots,i_{K-\eta-1},N_K-1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & S^*_{i_1,\ldots,i_{K-\eta-1},0} \end{bmatrix}.$$

Note that such a matrix S is fully characterized by the first block rows of all matrices $S_{i_1,\ldots,i_{K-\eta-1}}$, i.e., the first block row of the matrix V^{+-}, resp. V^{aa}, determines the complete matrix V^{+-}, resp. the matrix V^{aa}. Since this particular structure is preserved by multiplication, we can exploit it in the calculations, as shown in Algorithm 5.1 and explained in the following paragraphs.

For a recursion depth equal to one, i.e., when the blocks S_i cannot be subdivided into smaller blocks with a structure as in (5.49), Algorithm 5.2 can be employed to multiply the matrices S and T, both displaying the (5.49)-structure. Recall, C denotes the size of the inner blocks, here given by S_i. In this algorithm with a time complexity, resp. memory complexity, of $O(N_K^2C^3)$, resp. $O(N_KC^2)$, the last line in the for loop corresponds to the additional terms introduced by the last block column of the matrix T. That is, this last block column is composed of matrices $T_i^* = \sum_{k=i}^{\infty} T_k$, however, we only store the first block row of the matrix T, i.e., the blocks T_i, for $i = 0, \ldots, N_K$. Note, in Algorithm 5.2 the transposed operator is applied to the blocks in the block vector and not to the scalars within these blocks. Since the matrices S and T exhibit a btbT structure, except for the last block column, it is even possible to speed up the multiplication using fast Fourier transforms [111].

This algorithm can easily be generalized to multiply two (5.49)-structured matrices S and T, with a recursion depth equal to two, resulting in Algorithm 5.3 with a time complexity, resp. memory complexity, of $O(N_K^2N_{K-1}^2C^3)$, resp. $O(N_KN_{K-1}C^2)$. In this algorithm, the matrix H is used to compute the additional terms introduced by the last block column of the T_i matrices ($0 \leq i \leq N_{k-1}$). The line $m = \min(j+k, N_{K-1})$ deals with the additional terms originating from the last block column of the matrix T, i.e., whenever $j + k \geq N_{K-1}$, we add the result to $U_{N_{K-1},i}$.

The generalization of this approach to the multiplication of two matrices S and T with a recursion depth larger than two is somewhat less apparent. This multiplication is described in Algorithm 5.4, the time complexity, resp. memory complexity of which equals $O(\prod_{k=\eta+1}^{K} N_k^2C^3)$, resp. $O(\prod_{k=\eta+1}^{K} N_kC^2)$. The idea of this algorithm is to group the entries of the first block row of the matrices S and T into different matrices S'_i, resp. T'_i, for $1 \leq i \leq \sigma_{\eta+1}$ (see step 4 of Algorithm 5.1). Subsequently, Algorithm 5.4 determines the different combinations of two of these matrices required to obtain the first block row of the matrix $U = ST$. Although this algorithm is quite complex, we will briefly sketch the analogy with the algorithms for a recursion depth equal to one and two.

The generalization from recursion depth one to recursion depth two caused us to take into account two kinds of additional terms instead of one, those originating from the last block columns of the inner blocks and those caused by the last block column of the outer blocks. In the general case, each new dimension, i.e., each recursive step in (5.49), contributes to a number of additional terms. The terms introduced by the two innermost dimensions are covered by the use of the multiplication presented in Algorithm 5.3, denoted by the $\odot$ operator. For each of the remaining dimensions, a distinction between the additional and the nonadditional blocks is made using the variables M_i, $\eta + 1 \leq i \leq K - 2$ (cont. on Page 92).

Algorithm 5.1 Efficient multiplication of two matrices S and T, structured as in (5.49)

1. Define the matrix $U = ST$ with the same structure as the matrices S and T.

2. For the simple case in which the structure of S and T is not repeated in the blocks S_i and T_i, i.e., $K - \eta - 1 = 0$, we refer to Algorithm 5.2.

3. If the blocks S_j and T_j $(0 \leq j \leq N_{K-1})$ of the matrices S and T themselves exhibit the structure as shown by (5.49), the technique used in Algorithm 5.2 can be generalized as presented in Algorithm 5.3.

4. For the more general case in which the structure repeats itself more than two times, we can make use of this last algorithm. Denote the U matrix of Algorithm 5.3 as

   $$U = S \odot T$$

   and define $\sigma_i = \prod_{k=i}^{K-2}(N_k + 1)$, for $i = \eta + 1, \ldots, K - 2$, $\sigma_{K-1} = 1$, and $\tau_i = \sigma_{\eta+1}\sigma_i$, for $i = \eta + 1, \ldots, K - 1$. We will restructure the matrices S and T using only their first block row as follows

$$S'_1 = \begin{bmatrix} S_{0,\ldots,0,0,N_K} & \cdots & S_{0,\ldots,0,0,1} & S_{0,\ldots,0,0,0} \\ S_{0,\ldots,0,1,N_K} & \cdots & S_{0,\ldots,0,1,1} & S_{0,\ldots,0,1,0} \\ \vdots & \ddots & \vdots & \vdots \\ S_{0,\ldots,0,N_{K-1},N_K} & \cdots & S_{0,\ldots,0,N_{K-1},1} & S_{0,\ldots,0,N_{K-1},0} \end{bmatrix},$$

$$S'_2 = \begin{bmatrix} S_{0,\ldots,1,0,N_K} & \cdots & S_{0,\ldots,1,0,1} & S_{0,\ldots,1,0,0} \\ S_{0,\ldots,1,1,N_K} & \cdots & S_{0,\ldots,1,1,1} & S_{0,\ldots,1,1,0} \\ \vdots & \ddots & \vdots & \vdots \\ S_{0,\ldots,1,N_{K-1},N_K} & \cdots & S_{0,\ldots,1,N_{K-1},1} & S_{0,\ldots,1,N_{K-1},0} \end{bmatrix}, \cdots,$$

$$S'_{\sigma_{\eta+1}} = \begin{bmatrix} S_{N_{\eta+1},\ldots,N_{K-2},0,N_K} & \cdots & S_{N_{\eta+1},\ldots,N_{K-2},0,1} & S_{N_{\eta+1},\ldots,N_{K-2},0,0} \\ S_{N_{\eta+1},\ldots,N_{K-2},1,N_K} & \cdots & S_{N_{\eta+1},\ldots,N_{K-2},1,1} & S_{N_{\eta+1},\ldots,N_{K-2},1,0} \\ \vdots & \ddots & \vdots & \vdots \\ S_{N_{\eta+1},\ldots,N_{K-2},N_{K-1},N_K} & \cdots & S_{N_{\eta+1},\ldots,N_{K-2},N_{K-1},1} & S_{N_{\eta+1},\ldots,N_{K-2},N_{K-1},0} \end{bmatrix},$$

$$T'_1 = \begin{bmatrix} T_{0,\ldots,0,0,0} & T_{0,\ldots,0,1,0} & \cdots & T_{0,\ldots,0,N_{K-1},0} \\ T_{0,\ldots,0,0,1} & T_{0,\ldots,0,1,1} & \cdots & T_{0,\ldots,0,N_{K-1},1} \\ \vdots & \vdots & \ddots & \vdots \\ T_{0,\ldots,0,0,N_K} & T_{0,\ldots,0,1,N_K} & \cdots & T_{0,\ldots,0,N_{K-1},N_K} \end{bmatrix}, \cdots,$$

$$T'_{\sigma_{\eta+1}} = \begin{bmatrix} T_{N_{\eta+1},\ldots,N_{K-2},0,0} & T_{N_{\eta+1},\ldots,N_{K-2},1,0} & \cdots & T_{N_{\eta+1},\ldots,N_{K-2},N_{K-1},0} \\ T_{N_{\eta+1},\ldots,N_{K-2},0,1} & T_{N_{\eta+1},\ldots,N_{K-2},1,1} & \cdots & T_{N_{\eta+1},\ldots,N_{K-2},N_{K-1},1} \\ \vdots & \vdots & \ddots & \vdots \\ T_{N_{\eta+1},\ldots,N_{K-2},0,N_K} & T_{N_{\eta+1},\ldots,N_{K-2},1,N_K} & \cdots & T_{N_{\eta+1},\ldots,N_{K-2},N_{K-1},N_K} \end{bmatrix}.$$

5. Apply Algorithm 5.4 using these S'_k and T'_k matrices to compute the matrix U.

Algorithm 5.2 Multiplication of (5.49)-structured matrices with recursion depth one

1. If the recursion depth equals one, the matrix S, resp. T, can be represented by its first block row $[S_0\ S_1\ \cdots\ S_{N_K}]$, resp. $[T_0\ T_1\ \cdots\ T_{N_K}]$.

2. The matrix $U = ST$ can be computed as follows:

set $U_{N_K} = 0$
for $i = 0$ to $N_K - 1$ do
    $U_i = [S_i\ S_{i-1}\ \cdots\ S_0][T_0\ T_1\ \cdots\ T_i]^t$
    $U_{N_K} = U_{N_K} + [S_{N_K}\ S_{N_K-1}\ \cdots\ S_{N_K-i}][T_{N_K-i}\ T_{N_K-i+1}\ \cdots\ T_{N_K}]^t$
end
$U_{N_K} = U_{N_K} + [S_{N_K}\ S_{N_K-1}\ \cdots\ S_0][T_0\ T_1\ \cdots\ T_{N_K}]^t$

Algorithm 5.3 Multiplication of (5.49)-structured matrices with recursion depth two

1. In this situation, not only the matrices S and T exhibit the structure as shown by (5.49), but also the blocks S_j and T_j $(0 \le j \le N_{K-1})$ occurring in these matrices have the same structure. We can therefore represent S_j and T_j by their first block row $[S_{j,0}\ S_{j,1}\ \cdots\ S_{j,N_K}]$, resp. $[T_{j,0}\ T_{j,1}\ \cdots\ T_{j,N_K}]$.

2. The matrix U (which we will write as $U = S \odot T$ for further use) can be computed as described below:

for $i = 0$ to N_K do

$$G = \begin{bmatrix} S_{0,i} & S_{0,i-1} & \cdots & S_{0,0} \\ S_{1,i} & S_{1,i-1} & \cdots & S_{1,0} \\ \vdots & \vdots & \ddots & \vdots \\ S_{N_{K-1},i} & S_{N_{K-1},i-1} & \cdots & S_{N_{K-1},0} \end{bmatrix} \begin{bmatrix} T_{0,0} & T_{1,0} & \cdots & T_{N_{K-1},0} \\ T_{0,1} & T_{1,1} & \cdots & T_{N_{K-1},1} \\ \vdots & \vdots & \ddots & \vdots \\ T_{0,i} & T_{1,i} & \cdots & T_{N_{K-1},i} \end{bmatrix}$$

$$H = \begin{bmatrix} S_{0,N_K} & \cdots & S_{0,N_K-i} \\ S_{1,N_K} & \cdots & S_{1,N_K-i} \\ \vdots & \ddots & \vdots \\ S_{N_{K-1},N_K} & \cdots & S_{N_{K-1},N_K-i} \end{bmatrix} \begin{bmatrix} T_{0,N_K-i} & \cdots & T_{N_{K-1},N_K-i} \\ T_{0,N_K-i+1} & \cdots & T_{N_{K-1},N_K-i+1} \\ \vdots & \ddots & \vdots \\ T_{0,N_K} & \cdots & T_{N_{K-1},N_K} \end{bmatrix}$$

    for $j = 0$ to N_{K-1} do
        for $k = 0$ to N_{K-1} do
            $m = \min(j + k, N_{K-1})$
            $U_{m,i} = U_{m,i} + G_{j,k}$
            if $i < N_K$ then $U_{m,N_K} = U_{m,N_K} + H_{j,k}$
        end
    end
end

Algorithm 5.4 Multiplication of (5.49)-structured with general recursion depth

1. Initialize $M_i = 0$, for $\eta + 1 \leq i \leq K - 2$.

2. The blocks of U are then found by this algorithm:

$I_1 = 0$
for $n = 1$ to $\tau_{\eta+1}$ do
  $I_2 = 1$
  $i = K - 2$
  while $i > \eta$ do
    if $\mathrm{mod}(n-1, \tau_{i+1}) = 0$
      $M_i = M_i + 1$ and $M_j = 1$, for every $j > i$
    end
    $i = i - 1$
  end
  $i = K - 2$
  while $i > \eta$ do
    if $\mathrm{mod}(n-1, \sigma_i) \geq \sigma_{i+1} M_i$
      $k_i = N_i$ and $I_2 = I_2 + \sigma_i - \sigma_{i+1} M_i$
    else
      $k_i = M_i - 1$ and $I_2 = I_2 + (M_i - 1)\sigma_{i+1}$
      if $M_i > 1$
        $I_2 = I_2 - \mathrm{floor}(\mathrm{mod}(n-1, \sigma_i)/\sigma_{i+1})\sigma_{i+1}$
      end
    end
    $i = i - 1$
  end
  $I_1 = \mathrm{mod}(I_1, \sigma_{\eta+1}) + 1$
  $U_{k_{\eta+1},\ldots,k_{K-2}} = U_{k_{\eta+1},\ldots,k_{K-2}} + S'_{I_1} \odot T'_{I_2}$
end

We initialize $M_i = 1$, for $i = \eta + 1, \ldots, K - 2$. Then, we start with the matrix S'_1, i.e., $I_1 = 1$, and determine with which matrix T'_{I_2} this matrix has to be combined using the $\odot$ multiplication. Next, we do the same for the matrices $S'_2, \ldots, S'_{\sigma_{\eta+1}}$. Afterward, the values of M_i are updated as follows:

if $M_{K-2} < N_{K-2}$
  $M_{K-2} = M_{K-2} + 1$
else
  $M_{K-2} = 0$
  if $M_{K-3} < N_{K-3}$
    $M_{K-3} = M_{K-3} + 1$
  else
    $M_{K-3} = 0$
    ...
  end
end

With these updated values, we start again with the matrix $S'_{I_1} = S'_1$ and determine the associated matrix T'_{I_2}. This procedure is repeated $\sigma_{\eta+1}$ times. Each time, the number of additional $C(N_K+1)(N_{K-1}+1) \times C(N_K+1)(N_{K-1}+1)$ block terms needed for dimension i is given by $(N_i - M_i + 1)\prod_{k=i+1}^{K-2}(N_k+1)$. Hence, in the algorithm, if $\mathrm{mod}(n-1, \sigma_i) \geq \sigma_{i+1}M_i$, we compute the additional block terms caused by the last block column in the matrices $T_{j_1,\ldots,j_{i-\eta-1}}$, where the notation T_{j_1,j_0} denotes the matrix T. This gives an idea on how the techniques illustrated in Algorithm 5.2 and Algorithm 5.3 can be generalized to larger recursion depths. Let us now return to the discussion on the computation of the matrix V.

The observation that the matrix V^{+-} has this particular structure can be understood as follows. Since $W(t) = +$, all low priority customers that are present in the Q-area will be added to the current node J. The chain can only return to node J after all these low priority customers have been moved back into the Q-area. This explains the zeros below the main block diagonal. If new low priority customers arrived in the meantime, they will also be in the Q-area when the chain eventually returns to node J, causing the nonzero blocks above the diagonal. However, the number of new arrivals does not depend on the current Q-area contents, resulting in the structure presented in (5.49). In a similar manner one can easily check that also the structure of the matrix V^{aa} matches the one of (5.49).

Remark 9. The matrix V^{+-} only exhibits the (5.49)-structure starting from the second block row. The first block row is entirely filled with zeros since it contains those states where $W(t) = +$ and no customers are present in the Q-area. In that case, a transition will be made to a state of the same node with $W(t) = 0$, the probabilities of which are enclosed in V^{+h}. A mathematical argument, based on the expression for V^{+-}, will be presented further in this section. Nevertheless, we will compute V^{+-} as it would have the structure shown in (5.49) since this allows us to benefit from the efficiency of Algorithm 5.1. Only after the V-matrix has been computed, the entries of the first block row of V^{+-} will be replaced by zeros.

As a consequence, we can reduce the amount of memory needed to store the matrices V^{+-} and V^{aa} significantly by storing only their first block rows which can then be used for the computation of the new block rows in the next iteration step. From Eqn. (5.40) - (5.47) and the block diagonal structure of $V^{a-} = \bar{B}^{a-}$, we learn that the matrices W_N^{a-} and W_N^{aa} can also be stored using only their first block row. Note that the particular structure is built up during the recursion given by equation (5.36) since the matrix V_N includes only those paths that start and end in the same node with a path length $\leq N$. For example, if we consider the matrix V_1^{+-}, only paths with length zero and one are included. If there were no customers present in the Q-area, those paths cover at most one new arriving customer. However, if one customer was present in the Q-area, no new arrivals are counted in this iteration step, since these would cause a path length ≥ 2. Nevertheless, we can assume that during the N-th iteration this structure is already present in the matrices (by storing and computing only their first block row). This way, we will include more paths than only those with a maximum length of N, without changing the convergence. This can be explained by the fact that all rows $\bar{B} \leq X_N \leq V$ converge to V using the iteration (5.36) [103]. Moreover, because in each step we include more paths compared to the original scheme, the number of iteration steps will typically be smaller. That is, taking a larger matrix X_N that is still smaller than V will speed up the convergence of the iteration.

Having obtained the matrix W_N, we can compute V_{N+1} using equation (5.36). The entries in the last block column of V_{N+1} is obtained from the matrix product $\bar{U}_\eta W_N \bar{D}_\eta$, more specifically,

$$V_{N+1}^{ha} = \bar{U}_\eta^{hh} W_N^{h-} \bar{D}_\eta^{-a}, \tag{5.50}$$

$$V_{N+1}^{aa} = \bar{B}^{aa} + (\bar{U}_\eta^{a+} W_N^{+-} + \bar{U}_\eta^{aa} W_N^{a-}) \bar{D}_\eta^{-a}. \tag{5.51}$$

The second block column of V_{N+1} is obtained from

$$\sum_{s=\eta+1}^{K} \bar{U}_s W_N \bar{D}_s$$

as follows:

$$V_{N+1}^{+-} = \sum_{s=\eta+1}^{K} \bar{U}_s^{++} W_N^{+-} \bar{D}_s^{--}, \tag{5.52}$$

$$V_{N+1}^{h-} = \bar{B}^{h-} + \sum_{s=\eta+1}^{K} \bar{U}_s^{hh} W_N^{h-} \bar{D}_s^{--}. \tag{5.53}$$

There is no need to compute and store the matrices W_N^{+h} and W_N^{+a} since they are not used during the recursion. They are only needed to determine the R_k matrices and therefore, we postpone their computation until V is entirely identified, that is, until the difference between V_N and V_{N+1} is sufficiently small. As mentioned before, the matrices V_N^{+-}, V_N^{aa}, W_N^{a-} and W_N^{aa} are stored using only their first block row and the matrices V_N^{+h}, V_N^{hh} and V_N^{a-} are not stored since they can be extracted immediately from the matrix $\bar{B}$. With regard to the other matrices, one can benefit from their structure to reduce the amount of time and memory needed by the algorithm. That is, it suffices to store the first block row of the matrix $\bar{B}^{aa}$ and for the matrices $\bar{D}_\eta^{-a}$, $\bar{U}_s^{hh}$, $\bar{U}_\eta^{a+}$, $\bar{U}_\eta^{aa}$ and $\bar{B}^{a-}$ we only need the block in the upper left corner.

Remark 10. Let us return for a moment to the computation of V_{N+1}^{+-}. Recall that the matrix V^{+-} exhibits the (5.49)-structure, except for its first block row. This row is entirely filled with zeros since this is also the case for the first block row of every $\bar{U}_s^{++}$ matrix, for $s = \eta+1, \ldots, K$. For this reason, we decided to compute the matrix V_{N+1}^{+-} as it would have the (5.49)-structure (including its first block row), without changing the convergence of the iterative scheme given by (5.36). Therefore, we only need to compute the first block row of the matrix V_{N+1}^{+-}.

Denote Y as the result obtained from premultiplying a matrix X with U_s^{++}. From Eqn. (5.23) it can be observed that Y contains the first $(\mu_{\eta+1} - \mu_{s+1})m_a$ rows of X, only shifted downward over $\mu_{s+1}m_a$ rows, hence, there is no need to store the matrices U_s^{++}. Combining this with Eqn. (5.52) learns us that the second block row of V_{N+1}^{+-} is given by $W_N^{+-} D_K^{--}$. However, to be able to reconstruct the entire matrix V^{+-} we need the first block row of the (5.49)-structured matrix V_{N+1}^{+-}. From Eqn. (5.20) it can be observed that this block row is given by the first block row of W_N^{+-}. As a consequence, it also suffices to store only the first block of W_N^{+-}, given by $\bar{B}^{+h} W_N^{h-}$. At the end of the iteration, the matrix V^{+-} can be constructed by introducing the specific structure and replacing the entire first block row by zeros.

5.5.3 Solving the Boundary Condition

Having obtained the matrix V, we only need the vector $\bar{\pi}_\emptyset$ to compute all the stationary vectors $\bar{\pi}_J$. $\bar{\pi}_\emptyset$ is found as a solution to

$$\bar{\pi}_\emptyset = \bar{\pi}_\emptyset \left(\sum_k R_{\emptyset,k} \bar{D}_{\emptyset,k} + \bar{F} \right).$$

The matrix $\bar{F}$ contains the probability that the chain remains in the root node and, as we shall see below, it can be written as

$$\bar{F} = \begin{bmatrix} 0 & 0 & 0 & \bar{F}^{-l} \\ 0 & \bar{F}^{hh} & 0 & \bar{F}^{hl} \\ 0 & 0 & \bar{F}^{aa} & \bar{F}^{al} \\ 0 & \bar{F}^{lh} & \bar{F}^{la} & \bar{F}^{ll} \end{bmatrix}.$$

We again recognize 4 levels in this matrix, however, the states represented by these levels differ from those enclosed in the matrices $\bar{B}$, $\bar{D}_s$ and $\bar{U}_s$. That is, the first level corresponds to states with $W(t) = -$ since it is not possible to have $W(t) = +$ if $J = \emptyset$. The second and the third level correspond to states with $W(t) = 0$ and $S(Q(t)) = k$, where $k < n$, resp. $k = n$. The fourth level contains the idle state and the states with $W(t) = 0$ and $S(Q(t)) > \eta$, i.e., a low priority customer is being served, which is only possible if $X(t) = \emptyset$.

Since the matrix $\bar{F}$ covers the transitions between two states of the root node, in which there are no customers on the stack, the entries in the first block column, that correspond to transitions to a state with $W(t) = -$ are all equal to zero. From a state with $W(t) = -$, meaning that the last customer has been removed from the stack, there can only be a transition to a state that corresponds to a low priority customer being served. The two remaining zero blocks in the matrix $\bar{F}$ can be explained by noticing that an arrival of a high priority customer during the service of a customer of type η, will cause a transfer of this last customer to the stack. Also, the service of a high priority customer can only be followed by a service of a type η customer, without passing through the idle state, after this type η customer has been removed from the stack.
We define the numbers $\bar{\kappa}_j$, $\bar{\nu}_j$, $\bar{\nu}^*$ with regard to the low priority customers ($j > \eta$) as

$$\bar{\kappa}_j = \prod_{k=j}^{K} (N_k + 1), \tag{5.54}$$

$$\bar{\nu}_j = N_j \bar{\kappa}_{j+1}, \tag{5.55}$$

$$\bar{\nu}^* = \sum_{k=\eta+1}^{K} \nu_k m_k. \tag{5.56}$$

Although $\bar{\kappa}_j = \mu_j$, we introduced this variable here to increase the analogy between the discussion for the transitions from and to the root node and that for the other nodes. If in some node $J \neq \emptyset$ a customer of type $k \leq \eta$ finishes his service and there are no high priority customers present, customers will be removed from J until a type-η customer is found. If there is no type-η customer in J, the chain will eventually enter the root node with $W(t) = -$. From there a transition will be made to a state with $W(t) = 0$ and $S(Q(t)) > \eta$ as given by

the $\mu_{\eta+1}m_a \times (\bar{\nu}^* + 1)m_a$ matrix

$$\bar{F}^{-l} = diag \begin{bmatrix} 0 \\ I_{\bar{\nu}_K} \otimes \alpha_K \\ I_{\bar{\nu}_{K-1}} \otimes \alpha_{K-1} \\ \vdots \\ I_{\bar{\nu}_{\eta+1}} \otimes \alpha_{\eta+1} \end{bmatrix} \otimes I_{m_a}.$$

The zero in the first row of this expression can be explained by noticing that it is not possible to end up in a state of the root node with $W(t) = -$, while the Q-area is empty, therefore, the states of the form $(\emptyset, -, \mathbf{0})$ are transient states. The matrices $\bar{F}^{hh}$ and $\bar{F}^{aa}$ can be extracted immediately from the matrix $\bar{B}$ as

$$\bar{F}^{hh} = \bar{B}^{hh}, \tag{5.57}$$

$$\bar{F}^{aa} = \bar{B}^{aa}. \tag{5.58}$$

The $(\bar{\nu}^* + 1)m_a \times (\bar{\nu}^* + 1)m_a$ matrix $\bar{F}^{ll}$ has a structure, similar to the one of $\bar{F}^{hh}$, namely,

$$\bar{F}^{ll} = \begin{bmatrix} \bar{F}^{ll}_{0,0} & \bar{F}^{ll}_{0,K} & \bar{F}^{ll}_{0,K-1} & \cdots & \bar{F}^{ll}_{0,\eta+1} \\ \bar{F}^{ll}_{K,0} & \bar{F}^{ll}_{K,K} & \bar{F}^{ll}_{K,K-1} & \cdots & \bar{F}^{ll}_{K,\eta+1} \\ \bar{F}^{ll}_{K-1,0} & \bar{F}^{ll}_{K-1,K} & \bar{F}^{ll}_{K-1,K-1} & \cdots & \bar{F}^{ll}_{K-1,\eta+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{F}^{ll}_{\eta+1,0} & \bar{F}^{ll}_{\eta+1,K} & \bar{F}^{ll}_{\eta+1,K-1} & \cdots & \bar{F}^{ll}_{\eta+1,\eta+1} \end{bmatrix},$$

with

$$\bar{F}^{ll}_{0,0} = I_{ma} + D_0/\lambda, \tag{5.59}$$

$$\bar{F}^{ll}_{0,k} = [\alpha_k \otimes D_k/\lambda \; 0 \; \cdots \; 0], \tag{5.60}$$

$$\bar{F}^{ll}_{k,0} = \begin{bmatrix} (-T_k)e/\lambda \otimes I_{m_a} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{5.61}$$

$$\begin{aligned} \bar{F}^{ll}_{k,k} &= I_{\bar{\nu}_k} \otimes (I_{m_k m_a} + T_k/\lambda \oplus D_0/\lambda) + \sum_{i=k+1}^{K} I_{\bar{\nu}_k/\bar{\kappa}_i} \otimes I^+_{N_i+1} \otimes I_{\bar{\kappa}_{i+1}} \otimes I_{m_k} \otimes D_i/\lambda \\ &+ I^+_{N_k} \otimes I_{\bar{\kappa}_{k+1}} \otimes I_{m_k} \otimes D_k/\lambda + I^-_{N_k} \otimes I_{\bar{\kappa}_{k+1}} \otimes (-T_k)e/\lambda\alpha_k \otimes I_{m_a}, \end{aligned} \tag{5.62}$$

$$\bar{F}^{ll}_{k,l} = [0 \; I_{\bar{\nu}_k} \otimes e\alpha_l \otimes D_l/\lambda \; 0], \text{ for } k > l, \tag{5.63}$$

$$\bar{F}^{ll}_{k,l} = \begin{bmatrix} 0 \\ I_{\bar{\nu}_l} \otimes (-T_k)e/\lambda\alpha_l \otimes I_{m_a} \\ 0 \end{bmatrix}, \text{ for } k < l. \tag{5.64}$$

The entries in the first block column and the first block row correspond to the state in which the system is idle. Due to arguments, similar to those used for equations (5.34) and (5.35), the number of leading zero columns, resp. rows, in equations (5.63), resp. (5.64), equals $\bar{\kappa}_{k+1}m_l m_a$, resp. $\bar{\kappa}_{l+1}m_k m_a$.

When a customer of type η finishes his service, two different transitions are possible. If there are still low priority customers present, the customer with the highest priority will be served. Otherwise, the system will become idle until a new customer arrives. Both type of transitions are enclosed in

$$\bar{F}^{al} = diag \begin{bmatrix} (-T_\eta)e/\lambda \\ I_{\bar{\nu}_K} \otimes (-T_\eta)e/\lambda\alpha_K \\ I_{\bar{\nu}_{K-1}} \otimes (-T_\eta)e/\lambda\alpha_{K-1} \\ \vdots \\ I_{\bar{\nu}_{\eta+1}} \otimes (-T_\eta)e/\lambda\alpha_{\eta+1} \end{bmatrix} \otimes I_{m_a}.$$

The matrix $\bar{F}^{lh}$ that holds the probabilities of an arrival of a high priority customer in an idle system. If there is at least one low priority customer present, a high priority arrival will cause a transfer of all low priority customers from the Q-area to the stack. Therefore, $\bar{F}^{lh}$ has only its first block row different from zero, i.e., this block row equals $[\bar{F}^{lh}_{\eta-1}\ \bar{F}^{lh}_{\eta-2}\ \cdots\ \bar{F}^{lh}_{1}]$, with

$$\bar{F}^{lh}_k = \begin{bmatrix} \alpha_k \otimes D_k/\lambda & 0 & \cdots & 0 \end{bmatrix}, \tag{5.65}$$

a $m_a \times \nu_k m_k m_a$ matrix. On the other hand, transitions from a state with a high priority customer in service to an idle system can be found in

$$\bar{F}^{hl} = \begin{bmatrix} \bar{F}^{hl}_{\eta-1} & 0 & \cdots & 0 \\ \bar{F}^{hl}_{\eta-2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{F}^{hl}_{1} & 0 & \cdots & 0 \end{bmatrix}, \text{ with } \bar{F}^{hl}_k = \begin{bmatrix} (-T_k)e/\lambda \otimes I_{m_a} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

since such a transition can only occur if there are no other customers in the Q-area. Finally, we give an expression for the $(\bar{\nu}^* + 1)m_a \times \mu_{\eta+1} m_\eta m_a$ matrix $\bar{F}^{la}$ that covers transitions from a state with $S(Q(t)) > \eta$ to a state with $S(Q(t)) = \eta$. Such a transition takes place if a type-η customer arrives while a low priority customer occupies the server, that is,

$$\bar{F}^{la} = diag \begin{bmatrix} \alpha_\eta \\ I_{\bar{\nu}_K} \otimes e\alpha_\eta \\ I_{\bar{\nu}_{K-1}} \otimes e\alpha_\eta \\ \vdots \\ I_{\bar{\nu}_{\eta+1}} \otimes e\alpha_\eta \end{bmatrix} \otimes D_\eta/\lambda.$$

Note that for transitions from and to the root node, there are no entries for the states with $W(t) = +$. However, additional entries corresponding to the states in which $S(Q(t)) > \eta$ are required for the matrices $\bar{D}_{\emptyset,s}$ and $\bar{U}_{\emptyset,s}$ that are needed to solve the equation

$$\bar{\pi}_\emptyset = \bar{\pi}_\emptyset \left(\sum_{k=\eta}^{K} R_{\emptyset,k} \bar{D}_{\emptyset,k} + \bar{F} \right).$$

For the $\bar{D}_{\emptyset,s}$ matrices these entries are all equal to zero since transitions from some node s to the root node always go through a state of $\emptyset$ with $W(t) = -$, or through a state with

$q_\eta(t) = 1$ (if $s = \eta$), that is,

$$\bar{D}_{\emptyset,s} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \bar{D}_s^{--} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \bar{D}_{\emptyset,\eta} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{D}_\eta^{-a} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{5.66}$$

for $s > \eta$. Recall, there is no block row corresponding to states with $W(t) = +$ since the MC shall never reside in one of these states if $J = \emptyset$. With respect to the matrices $\bar{U}_{\emptyset,s}$ $(s > \eta)$ we need an extra block row corresponding to transitions from a state with $S(Q(t)) > \eta$, namely

$$\bar{U}_{\emptyset,s} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{U}_s^{hh} & 0 \\ 0 & 0 & 0 & 0 \\ \bar{U}_s^{l+} & 0 & 0 & 0 \end{bmatrix}, \tag{5.67}$$

with $\bar{U}_s^{l+}$ a $(\bar{\nu}^* + 1)m_a \times \mu_{\eta+1} m_a$ matrix given by

$$\bar{U}_s^{l+} = \begin{bmatrix} I_{N_s+1}^- \otimes I_{\mu_{s+1}} \otimes e \otimes \Delta & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{5.68}$$

If a low priority customer is being processed and a type-η customer arrives, the Markov chain remains in the root node, therefore,

$$\bar{U}_{\emptyset,\eta} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{U}_\eta^{hh} & 0 \\ \bar{U}_\eta^{a+} & 0 & 0 & \bar{U}_\eta^{aa} \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.69}$$

This allows us to write

$$\sum_{k=\eta}^{K} R_{\emptyset,k} \bar{D}_{\emptyset,k} + \bar{F} = \begin{bmatrix} 0 & V^{+-} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \bar{F}^{-l} \\ 0 & V^{h-} - \bar{A}^{h-} & \bar{A}^{hh} & V^{ha} & \bar{F}^{hl} \\ 0 & 0 & 0 & V^{aa} & \bar{F}^{al} \\ 0 & \sum_{s=\eta+1}^{K} \bar{U}_s^{l+} W^{+-} \bar{D}_s^{--} & \bar{F}^{lh} & \bar{F}^{la} & \bar{F}^{ll} \end{bmatrix}.$$

Before describing the computation of the vector $\bar{\pi}_\emptyset$, let us first introduce the following notations:

$$\begin{aligned} \bar{\pi}_{\emptyset,h}(0) &= (\bar{\pi}_\emptyset(0,(0,\ldots,1,0,\mathbf{0})),\ldots,\bar{\pi}_\emptyset(0,(N_1,\ldots,N_{\eta-1},0,\mathbf{0}))), \\ \bar{\pi}_{\emptyset,a}(0) &= (\bar{\pi}_\emptyset(0,(\mathbf{0},1,\mathbf{0})),\bar{\pi}_\emptyset(0,(\mathbf{0},1,0,\ldots,1)),\ldots,\bar{\pi}_\emptyset(0,(\mathbf{0},1,N_{\eta+1},\ldots,N_K))), \\ \bar{\pi}_{\emptyset,l}(0) &= (\bar{\pi}_\emptyset(0,(\mathbf{0},0,\mathbf{0})),\bar{\pi}_\emptyset(0,(\mathbf{0},0,0,\ldots,1)),\ldots,\bar{\pi}_\emptyset(0,(\mathbf{0},0,N_{\eta+1},\ldots,N_K))). \end{aligned}$$

To increase the readability of the equations that can be used to obtain $\bar{\pi}_\emptyset$, we also introduce some auxiliary matrices,

$$\begin{aligned} M_1 &= \bar{F}^{lh}(I - \bar{B}^{hh})^{-1}, \\ M_2 &= (M_1 \bar{V}^{ha} + \bar{F}^{la})(I - \bar{V}^{aa})^{-1}, \\ M_3 &= M_1(\bar{V}^{h-} - \bar{B}^{h-}) + \sum_{s=\eta+1}^{K} \bar{U}_s^{l+} W^{+-} \bar{D}_s^{--}. \end{aligned}$$

Finally, the vector $\bar{\pi}_\emptyset = (\bar{\pi}_\emptyset(-), \bar{\pi}_{\emptyset,h}(0), \bar{\pi}_{\emptyset,a}(0), \bar{\pi}_{\emptyset,l}(0))$ can be computed by

$$\begin{aligned}
\bar{\pi}_\emptyset(-) &= \bar{\pi}_{\emptyset,l}(0)M_3, && (5.70)\\
\bar{\pi}_{\emptyset,h}(0) &= \bar{\pi}_{\emptyset,l}(0)M_1, && (5.71)\\
\bar{\pi}_{\emptyset,a}(0) &= \bar{\pi}_{\emptyset,l}(0)M_2, , && (5.72)\\
\bar{\pi}_{\emptyset,l}(0) &= \bar{\pi}_{\emptyset,l}(0)(M_3\bar{F}^{-l} + M_1\bar{F}^{hl} + M_2F^{al} + \bar{F}^{ll}). && (5.73)
\end{aligned}$$

Hence, one after the vector $\bar{\pi}_{\emptyset,l}(0)$ is obtained from Eqn. (5.73), it can be plugged in the other equations to find the entire $\bar{\pi}_\emptyset$ vector. Recall, this vector is normalized using

$$\bar{\pi}_\emptyset e + \bar{\pi}_\emptyset R_\emptyset (I-R)^{-1}e = 1.$$

The steady state probabilities $\bar{\pi}_J$, for $J \neq \emptyset$, can now be computed by

$$\begin{aligned}
\bar{\pi}_k &= \bar{\pi}_\emptyset R_{\emptyset,k}, && (5.74)\\
\bar{\pi}_{J+k} &= \bar{\pi}_J R_k, && (5.75)
\end{aligned}$$

for $k = \eta, \ldots, K$. Of course, the resulting vector still contains additional entries corresponding to the states with $W(t) \in \{+,-\}$. The steady state vectors π_J of the original Markov chain are given by

$$\pi_J = \frac{\bar{\pi}_J(0)}{1-c}, \tag{5.76}$$

with the normalization constant

$$c = \bar{\pi}_\emptyset \begin{bmatrix} e_{\mu_{\eta+1}m_a} \\ \mathbf{0} \end{bmatrix} + \bar{\pi}_\emptyset R_\emptyset (I-R)^{-1} \begin{bmatrix} 2e_{\mu_{\eta+1}m_a} \\ \mathbf{0} \end{bmatrix}. \tag{5.77}$$

5.6 Loss Rate Computation

As discussed in Chapter 4, the low priority loss probability is often one of the most important performance measures in a priority system. In this situation, the loss probability of a type-k ($k > \eta$) customer can be computed as one minus the output rate of the type-k customers divided by the class-k input rate, or

$$p_{k,loss} = 1 - \frac{\pi_{\emptyset,l,k}(0)\ (e_{\bar{\nu}_k} \otimes (-T_k e) \otimes e_{m_a})}{\theta D_k e}, \tag{5.78}$$

where $\pi_{\emptyset,l,k}$ denotes the vector containing exactly those states where a type-k customer is being served, i.e.,

$$\pi_{\emptyset,l,k}(0) = \frac{(\bar{\pi}_\emptyset(0,(\mathbf{0},1,\mathbf{0})), \ldots, \bar{\pi}_\emptyset(0,(\mathbf{0},N_k,\ldots,N_K)))}{1-c}. \tag{5.79}$$

Here, the one in $\bar{\pi}_\emptyset(0,(\mathbf{0},1,\mathbf{0}))$ is positioned at the k-th entry and θ is the solution to the equations $\theta D = 0$ and $\theta e = 1$, where D is given by $D = \sum_{i=0}^{K} D_i$.

In Section 4.2.2, it was mentioned that high priority buffers are usually dimensioned such that hardly any losses occur. Of course, initially one should have an idea how to make sure that this is in fact the case, without wasting too much buffer space. This can be done by computing the high priority loss probability and dimensioning the accompanying buffer such

that this probability remains under a certain predefined threshold. To compute the high priority loss probability, we first determine the vector

$$\hat{\pi} = \frac{(\mathbf{0}, \bar{\pi}_\emptyset(-), \bar{\pi}_{\emptyset,h}(0), \bar{\pi}_{\emptyset,a}(0)) + \bar{\pi}_\emptyset R_\emptyset (I - R)^{-1}}{1 - c}. \tag{5.80}$$

The loss probability of a high priority class $k < \eta$ can then be calculated as

$$p_{k,loss} = 1 - \frac{\hat{\pi}_{h,k} \ (e_{\nu_k} \otimes (-T_k e) \otimes e_{m_a})}{\theta D_k e}. \tag{5.81}$$

Similar to equation (5.79), $\hat{\pi}_{h,k}$ corresponds to the part of the vector $\hat{\pi}$ where a type-k customer occupies the server. This means that at least one customer of type k is present and there are no type-j customers, for every $j < k$. Remark, since $N_\eta = \infty$ there are no losses for $k = \eta$.

Let us illustrate these formulas with some numerical examples where $K = 5$ priority classes are distinguished and η is set to 3. The MMAP[5] arrival process is given by the following matrices:

$$D_0 = \begin{bmatrix} \beta_1 & 0.5 \\ 0.5 & \beta_2 \end{bmatrix}, \qquad D_1 = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix},$$

$$D_2 = \begin{bmatrix} 0.03 & 0 \\ 0 & 0.015 \end{bmatrix}, \qquad D_3 = \begin{bmatrix} 0.002 & 0 \\ 0 & 0.001 \end{bmatrix},$$

$$D_4 = \begin{bmatrix} \gamma_{4,1} & 0 \\ 0 & \gamma_{4,2} \end{bmatrix}, \qquad D_5 = \begin{bmatrix} \gamma_{5,1} & 0 \\ 0 & \gamma_{5,2} \end{bmatrix}.$$

In each example, the values of β_1 and β_2 are adapted such that $De = 0$. The service times of the type-1, type-4 and type-5 customers are exponentially distributed with a mean equal to 0.2, $1/\mu_4$, and $1/\mu_5$, respectively. The class-2 and class-3 service times are given by the following PH distributions:

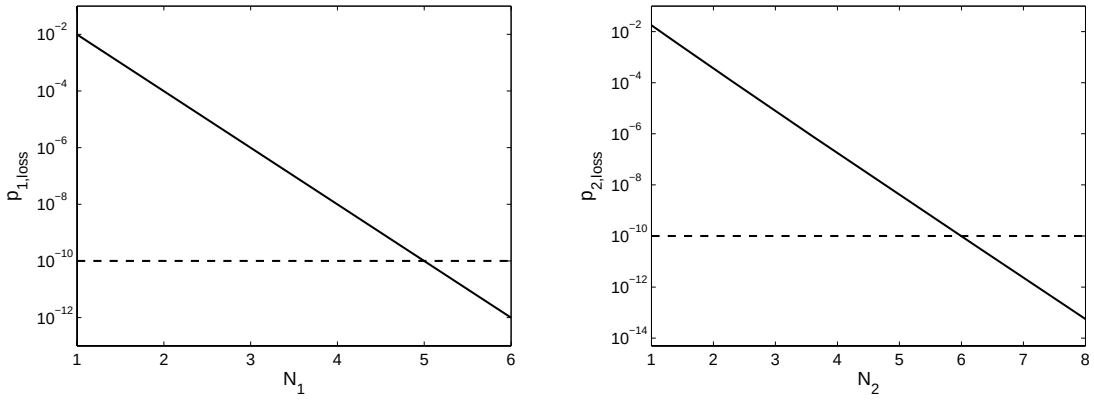
$$\alpha_2 = (0.5, 0.5), \quad T_2 = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix},$$

$$\alpha_3 = (1, 0, 0), \qquad T_3 = \begin{bmatrix} -0.5 & 0.45 & 0 \\ 0 & -0.8 & 0.1 \\ 0 & 0 & -0.75 \end{bmatrix}.$$

We start with dimensioning the high priority buffers such that, for $k = 1, 2$, the loss probability $p_{k,loss} < 10^{-10}$. Since we consider a preemptive scheduling discipline, the traffic of lower priority classes does not have an influence on the high priority customers. Figure 5.5 illustrates the loss probabilities of the type 1 (Fig. 5.5(a)) and the type 2 (Fig. 5.5(b)) customers for different values of N_1, resp. N_2. The horizontal line on these plots indicates the predefined threshold 10^{-10}. The results show that setting $N_1 = 5$ and $N_2 = 6$ (after N_1 was set to 5) suffices to keep the loss probabilities below this threshold. We will use these values for N_1 and N_2 to examine the influence of the arrival and service rates on the loss probability of the low priority traffic.

Moreover, set

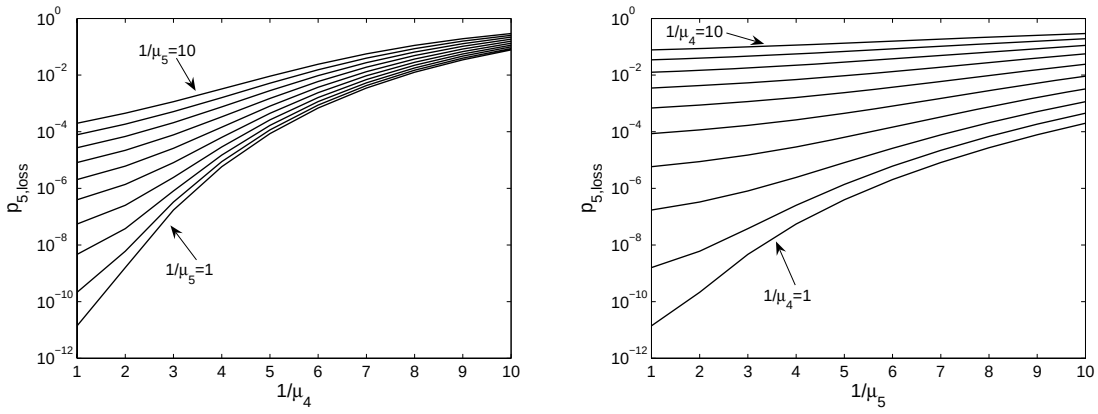
$$N_4 = 10, \ N_5 = 10, \ \gamma_{4,1} = 0.1, \ \gamma_{4,2} = 0.03, \ \gamma_{5,1} = 0.06, \ \gamma_{5,2} = 0.02, \tag{5.82}$$



(a) Priority class 1 (b) Priority class 2

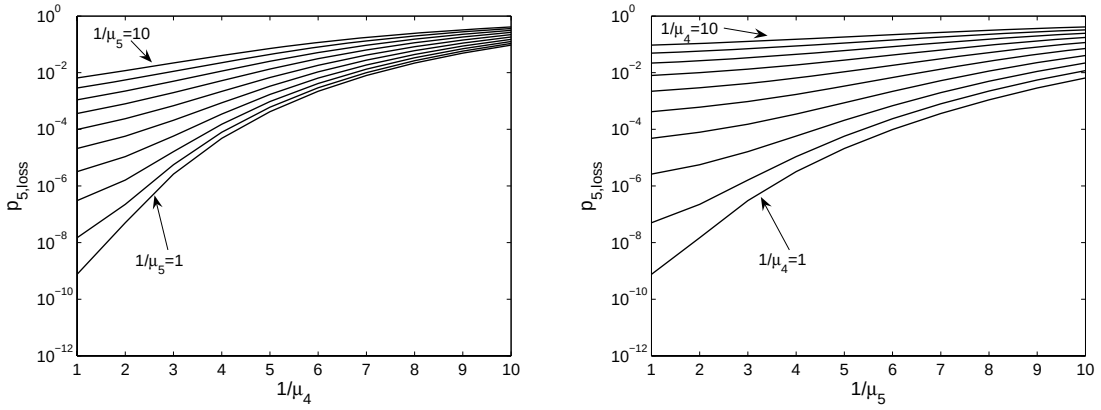
Figure 5.5: Dimensioning the high priority buffers

and let $1/\mu_i$ take values equal to $1, 2, \ldots, 10$, for $i = 4, 5$. For different combinations of $1/\mu_4$ and $1/\mu_5$, the class-5 loss probability is presented in Figure 5.6. In Figure 5.6(a) each line corresponds to a specific value of $1/\mu_5$, the lowest line denotes the loss probabilities for $1/\mu_5 = 1$, the second line from the bottom presents the loss probabilities for $1/\mu_5 = 2$, etc. In Figure 5.6(b) we do the same for different values of $1/\mu_4$. An increase in the mean service time of the type-5 customers will increase the preemption probability of these customers, whereas a larger service time of the type-4 customers will lead to longer waiting times for the type-5 traffic, since this can only be served when there are no customers with a higher priority in the queue. We now want to investigate which service rate has the largest influence on the loss probabilities.



(a) Variable type 4 service rate (b) Variable type 5 service rate

Figure 5.6: Influence of the low priority service rates on the type-5 loss probabilities



(a) Variable type 4 service rate

(b) Variable type 5 service rate

Figure 5.7: Influence of the low priority service rates on the type-5 loss probabilities under equal low priority arrival rates

It can be observed that an increase in the mean service time of the type-4 customers has a stronger impact on the type-5 loss probability compared to an increase in the mean service time of the type-5 customers. For a high type-4 service rate, reasonable loss probabilities can be obtained for lower type-5 service rates, whereas the combination of a high type-5 service rate and a low type-4 service rate leads to high loss probabilities. For example, if we take $(1/\mu_4, 1/\mu_5) = (1, 8)$, resp. $(1/\mu_4, 1/\mu_5) = (8, 1)$, the probability of a type-5 loss equals 2.72e-5, resp. 1.26e-2.

One might think that this could be explained by considering the arrival rates, i.e., the mean type-4 arrival rate is given by $\theta D_4 e = 0.065$, whereas the mean type-5 arrival rate is only $\theta D_5 e = 0.040$. To verify this hypotheses, we will perform the same analysis with

$$D_4 = D_5 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.02 \end{bmatrix},$$

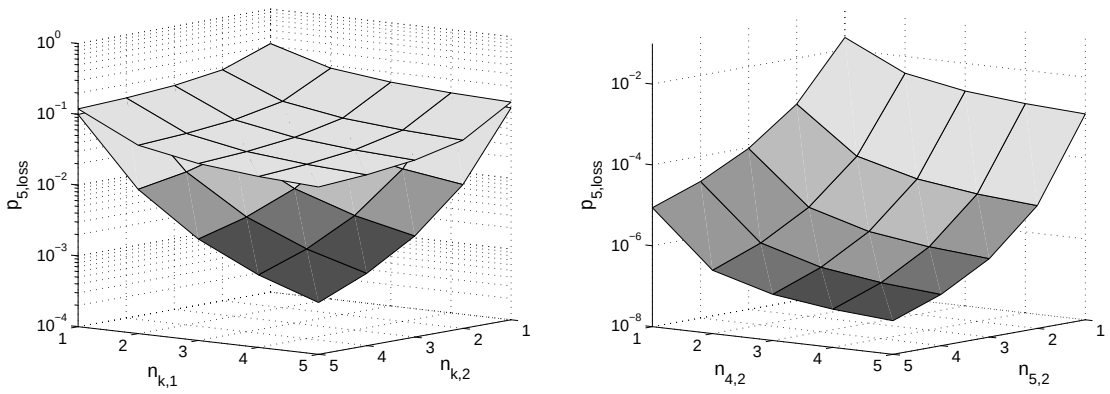
resulting in the loss probabilities presented in Figure 5.7. Although the type-4 and the type-5 traffic now has the same mean arrival rate, it can be observed that the influence of the mean service time of the customers of type 4 is still significantly larger compared to the influence of the mean service time of the type-5 customers. If we compare Figures 5.6 and 5.7, we notice that a small part of the difference in loss probability behavior for various service rates can be explained by the different arrival rates. Nevertheless, the figures show that the major influence needs to be attributed to the nature of the priority system, where an increase in the service time of the type-4 traffic causes the type-5 customers to get less access to the server.

After varying the service rates of the low priority customers, we will now take a look at the influence their arrival rates have on the loss probabilities. To do so, we set

$$1/\mu_4 = 1,\ 1/\mu_5 = 4/3$$

and we vary $\gamma_{k,i}$, with $k = 4, 5$ and $i = 1, 2$. More specifically, we let

$$\gamma_{k,i} = \frac{1}{2n_{k,i}},\ \text{with } n_{k,i} \in \{1, \ldots, 5\}.$$

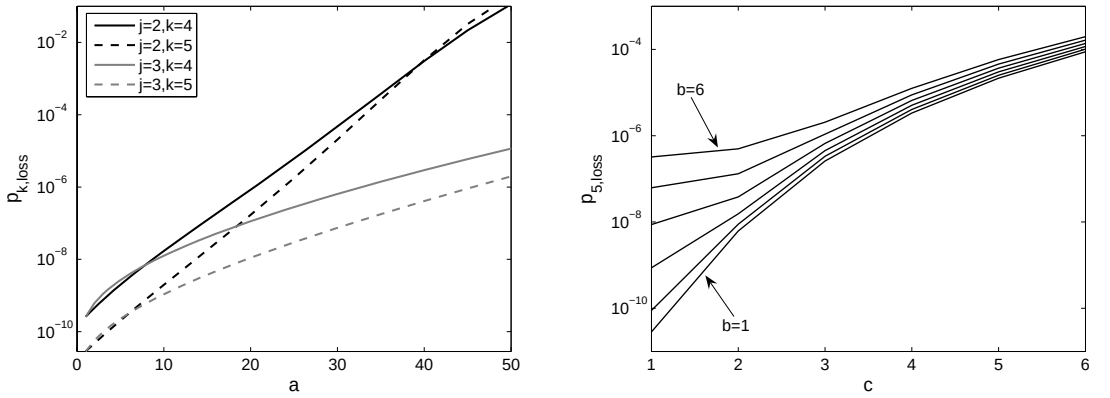


(a) Variable rates for one type

(b) Variable rates for one state

Figure 5.8: Influence of the low priority arrival rates on the type-5 loss probability

Figure 5.8(a) shows the loss probabilities of the type-5 traffic for different combinations of $n_{4,i}$ (top plane) and $n_{5,i}$ (bottom plane). To compute the plane that results from varying $n_{4,i}$ (resp. $n_{5,i}$), the values of $n_{5,i}$ (resp. $n_{4,i}$) are set to one. These results show that the arrival rate of the type-5 traffic has a larger influence on the probability that a type-5 customer is lost, compared to the arrival rate of the type-4 traffic. The same observation can be made based on the results shown in Figure 5.8(b), where we keep the rates in state one of the arrival process constant (i.e., we set $n_{4,1} = n_{5,1} = 5$), while varying the rates in state two. Similar to the previous example, a small part of this effect can be explained by the difference in service rates, however if we set $1/\mu_4 = 1/\mu_5 = 1$ we still obtain $p_{5,loss} = 4.37\text{e-}6$, resp. $p_{5,loss} = 2.74\text{e-}4$, for $(n_{4,2}; n_{5,2}) = (1, 5)$, resp. $(n_{4,2}; n_{5,2}) = (5, 1)$.



(a) Variable arrival rates

(b) Variable service rates

Figure 5.9: Influence of the high priority traffic on the low priority loss rates

Finally, we would like to examine the influence of the higher priority traffic on the low priority loss rates. In Figure 5.9(a), we vary either the arrival rate of the type-2 traffic or the

one of the type-3 traffic. Therefore, we set $1/\mu_4 = 1$, $1/\mu_5 = 4/3$, $D_j^* = aD_j$, with $j = 2$, resp. $j = 3$, and $D_k^* = D_k$, for $k \neq j$. Here, D_j denotes the original MMAP matrix and D_j^* represents the matrix containing the arrival rates of type-j customers used in this example. The values for $\gamma_{k,i}$ $(k = 4, 5)$ are set as in Eqn. (5.82). Except in the initial part, the type-2 arrival rate obviously has a larger influence on the loss probability of type-4 and type-5 customers. On the one hand, this is caused by higher arrival rates enclosed in D_2, compared to the rates in D_3 and on the other hand there is the influence of the priority scheduling discipline. A higher type-2 arrival rate will have an influence on the customers of type 2, 3, 4 and 5, whereas the type-3 traffic only influences customers of type 3, 4 and 5.

Remark 11. To make sure that not too much type-2 customers are dropped due to buffer overflow, we set $N_2 = 30$ in the cases with $i = 2$. For values of $a < 30$, this still results in a small value of $p_{2,loss}$. However, if $a \geq 30$, we obtain larger loss probabilities for the type-2 traffic. Even when $N_2 = 50$, we end up with $p_{2,loss} = 1.97\text{e-}4$.

The influence of the service rates of the type-2 and the type-3 customers is presented in Figure 5.9(b). Note that for these results, the original D_i matrices were used. The service times of the type-2 and type-3 traffic are given by the PH distributions $(2, \alpha_2, T_2/b)$, resp. $(3, \alpha_3, T_3/c)$. From Figure 5.9(b) it can be observed that a multiplicative increase in the mean service time of the type-3 traffic has the largest impact on the loss probability of a type-5 customer. The same observations can be made with relation to the type-4 loss probabilities, not included in this figure. These results meet our expectations since the mean service time of type-3 customers is much larger than the one of a customer of type 2. That is, for $b = c = 1$, the mean service time of a type-2, resp. a type-3 customer, equals 0.75, resp. 3.275.

6

Multilevel Feedback Queues

Priority scheduling as discussed in Chapters 4 and 5 can be applied to any kind of systems where the importance of a customer can be determined upon arrival. That is, one decides to give a certain type of traffic priority over the other traffic. This kind of priority system is a static scheduling discipline since the type of a packet is known upon arrival and remains unchanged during the entire sojourn time of this packet. For certain applications we would rather prefer a dynamic way of assigning priorities. Consider for example a system in which there is a large distinction between the required service times. In this situation it would be preferable to give priority to customers with a shorter service time to minimize the mean delay in the system. Some examples of scheduling disciplines that pursue such goals are shortest-process-next and shortest-remaining-time-next. A typical application of this type of priority mechanism is the scheduling of jobs by an operating system. Therefore, we will often refer to the service time using the term *job duration*. A problem that arises when implementing a discipline that takes into account the duration of a job is that this duration is not known a priori, meaning that it is not possible to determine the priority level of a job upon arrival in the system.

In a multilevel feedback queue the service time attained so far is used to determine the priority level of a job. At each moment, the job that has received the least service time so far will be processed. As opposed to the priority systems in Chapters 4 and 5, the priority of a job in a multilevel feedback queue changes over time. The priority level of a job decreases by one every time it has received one quantum of service. This way, shorter jobs will always complete their service before jobs requiring a longer processing time. A detailed description of multilevel feedback queues is presented in Section 6.5. To analyze a multilevel feedback queue where jobs arrive in batches and have generally distributed service times with finite support, we introduce a new matrix analytic paradigm in Section 6.3, termed Quasi-Birth-Death Markov chains on binomial-like trees. A necessary and sufficient stability condition is established in Section 6.4 and the stationary vector is shown to have a matrix geometric form, provided that it exists. We also indicate how to develop a quadratically converging algorithm to compute the steady state vector. Finally, in Section 6.5 we indicate how to model a multilevel feedback queue as a QBD Markov chain on a binomial-like tree.

6.1 Related Work

Multilevel feedback queues were originally introduced in the late 1960s [93, 27, 55] and appear nowadays in many modern textbooks on operating systems, e.g., [99]. Also, many contemporary operating systems support some form of multilevel feedback queues (e.g., Solaris). Multilevel feedback queues process jobs in an RR order (see Section 1.2.2), but whenever a job completes its quantum, its level is increased by one (all jobs start as level 0 jobs). Ongoing jobs are distributed over various FIFO queues, depending on their current level and a strict priority mechanism is used to visit these queues (where queue 0 has the highest priority). We refer to Section 6.5 for a more detailed description.

Feedback queues are inspired by the optimality of scheduling orders such as shortest-process-next (SPN) or shortest-remaining-time-next (SRTN), which minimize the mean delay by favoring short jobs, but do not rely on a priori knowledge of the job durations. Instead the service time attained so far is used to determine the priority of a job. To avoid starvation effects for large jobs, the quantum received by a level i job typically increases as a function of i. Other queueing disciplines that are very closely related to multilevel feedback queues include least-attained-service, foreground-background and shortest elapsed time. Many papers have been devoted to the analysis of multilevel feedback queues, a detailed survey on this topic was written by Nuyens and Wierman [79]. As can be seen from this survey, most of the work focuses on the limiting case where the number of levels is infinite and the quantum q approaches zero.

6.2 Introduction

In Section 2.1.3 we introduced the notion of a tree-like process as a discrete-time bivariate Markov chain $\{(X_t, N_t), t \geq 0\}$, in which the values of X_t are the nodes of a d-ary tree. The process we will use to model a multilevel feedback queue has a different type of state space, termed a binomial-like tree (due to its similarity with binomial trees). A binomial-like tree consists of an infinite number of nodes, where the root node has d children. Also, every node that is the i-th child of its parent node, has i children of its own. Thus, when considering a discrete-time Markov chain $\{(X_t, N_t), t \geq 0\}$ on such a state space, the value of X_t is represented as an *decreasingly ordered* string of integers (between 1 and d).

A QBD type Markov chain on such a state space is obtained by restricting the transitions in a manner similar to a tree-like process. However, to improve the flexibility of the framework, the range of the variable N_t may depend on the rightmost integer of X_t. Apart from discussing the ergodicity condition of such a Markov chain, we also develop an algorithm with quadratic convergence to determine its steady state vector, by showing that the matrices needed to characterize this vector can be obtained by solving a series of d quadratic matrix equations, for which various algorithms and software tools exist (e.g., [14]).

We will demonstrate that the behavior of a multilevel feedback queue can be captured by a QBD type Markov chain on a binomial-like tree, when jobs arrive in batches, have generally distributed service times with finite support and the number of levels is large enough such that any job starting service in the last level completes service before its quantum expires. Notice, due to the finite support assumption, this corresponds to having a finite number of levels. The model presented in this chapter can be easily extended to the case where jobs receive different service quanta depending on their current level.

6.3 QBDs on Binomial-Like Trees

A *binomial-like* tree of order d consisting of an infinite set of nodes Ω is constructed as follows. Let $\emptyset \in \Omega$ be the root node of the tree, having d children (i.e., neighbors) labeled 1 to d. Each i-th child node with a label of the form $J+i$, has exactly i children of its own, labeled $J+i+1$ to $J+i+i$. Thus, the set of all nodes is given by

$$\Omega = \{\emptyset\} \cup \{J | J = j_1\ j_2\ \ldots\ j_k, k \geq 1, 1 \leq j_l \leq d, j_l \geq j_{l'}, 1 \leq l \leq l' \leq k\},$$

that is, Ω includes all strings J of integers between 1 and d that are ordered descendingly. A binomial-like tree differs from the well-known binomial tree of order d [29] in the sense that the i-th child of a node has i children, instead of $i-1$. Therefore, a binomial-like tree has an infinite number of nodes, whereas a binomial tree of order d has exactly 2^d nodes.

Define a discrete-time bivariate Markov chain $\{(X_t, N_t), t \geq 0\}$ in which the values of X_t are represented by nodes of a binomial-like tree of order d, for $d \geq 1$, and where the range of N_t is defined as $\{1, \ldots, h_{f(X_t,1)}\}$, where $f(X_t, 1)$ represents the rightmost integer of the string X_t (and as $\{1, \ldots, h_\emptyset\}$ if $X_t = \emptyset$). We will refer to X_t as the *node* and to N_t as the *auxiliary* variable of the Markov chain (MC) at time t. Notice, the number of states belonging to a node may depend on the rightmost integer of the node label, meaning sibling nodes may contain a different number of states, but two nodes that are the i-th child of some nodes J_1 and J_2 both hold exactly h_i states. As for the tree-like processes, we use the '+' to denote the concatenation on the right and '-' to represent the deletion from the right. Furthermore, $f(J, k)$, for $J \neq \emptyset$, denotes the k rightmost elements of the string J, thus $J - f(J, 1)$ represents the parent node of J.

The following restrictions need to apply for an MC $\{(X_t, N_t), t \geq 0\}$ to be a QBD on a binomial-like tree. At each step the chain can only make a transition to its parent (i.e., $X_{t+1} = X_t - f(X_t, 1)$, for $X_t \neq \emptyset$), to itself ($X_{t+1} = X_t$), or to one of its own children ($X_{t+1} = X_t + s$ for some $1 \leq s \leq f(X_t, 1)$ or $1 \leq s \leq d$ if $X_t = \emptyset$). Moreover, the chain's state at time $t+1$ is determined as follows:

$$P[(X_{t+1}, N_{t+1}) = (J', j) | (X_t, N_t) = (J, i)] =$$
$$\begin{cases} f^{i,j} & J' = J = \emptyset, \\ b_k^{i,j} & J' = J \neq \emptyset, f(J,1) = k, \\ d_{k,l}^{i,j} & J \neq \emptyset, f(J,2) = lk, J' = J - f(J,1) = J - k, \\ d_k^{i,j} & J = k, J' = \emptyset, \\ u_{k,s}^{i,j} & J \neq \emptyset, f(J,1) = k, J' = J + s, \\ u_s^{i,j} & J = \emptyset, J' = s, \\ 0 & otherwise. \end{cases}$$

Notice, the transition probability between two nodes depends only on the rightmost element of their labels and not on their specific values. We can now define the $h_\emptyset \times h_\emptyset$ matrix F, the $h_k \times h_k$ matrix B_k, the $h_k \times h_l$ matrix $D_{k,l}$, the $h_k \times h_\emptyset$ matrix D_k, the $h_k \times h_s$ matrix $U_{k,s}$ and the $h_\emptyset \times h_s$ matrix U_s with their (i,j)-th elements given by $f^{i,j}$, $b_k^{i,j}$, $d_{k,l}^{i,j}$, $d_k^{i,j}$, $u_{k,s}^{i,j}$ and $u_s^{i,j}$, respectively. Notice, a QBD on a binomial-like tree is fully characterized by the matrices F, B_k, $D_{k,l}$, D_k, $U_{k,s}$ and U_s, for $1 \leq s \leq k \leq l \leq d$, meaning in general by $1 + d(d+4)$ matrices (see Figure 6.1).

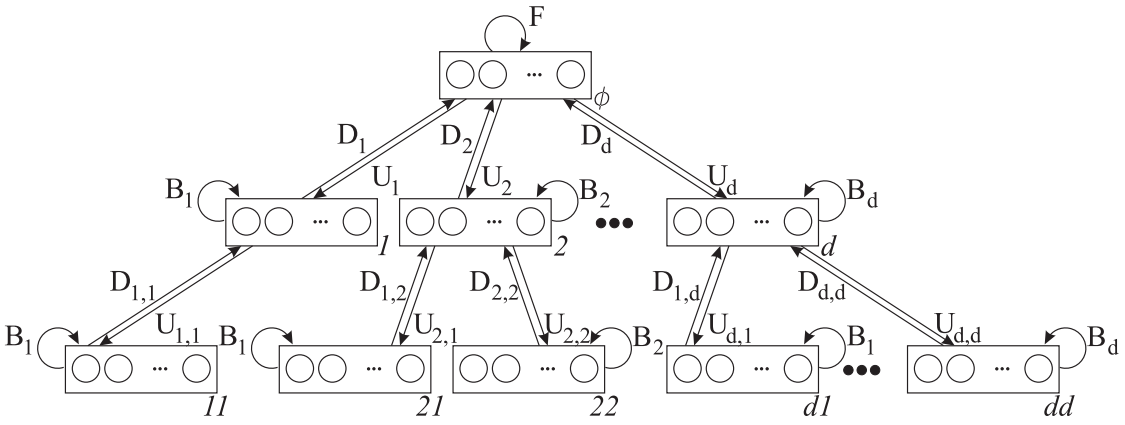


Figure 6.1: Structure of the first three levels of a binomial-like tree Markov chain and the matrices characterizing its transitions

Clearly, we have

$$B_k e + D_{k,l} e + \sum_{s=1}^{k} U_{k,s} e = e, \tag{6.1}$$

$$B_k e + D_k e + \sum_{s=1}^{k} U_{k,s} e = e, \tag{6.2}$$

$$F e + \sum_{s=1}^{d} U_s e = e, \tag{6.3}$$

for $1 \leq k \leq l$. This implies that $D_k e = D_{k,l} e$ for all $k \leq l$. Further notice, setting $d = 1$ reduces the binomial-like tree to an infinite strip and the associated QBD to the classic QBD Markov chains where the number of states at level 0 may differ from the number of states on level $i > 0$.

Next, we introduce a number of matrices that play a crucial role when studying the stability and stationary behavior of a QBD on a binomial-like tree, analogous to the matrices introduced for the tree-like processes. Let V_k denote the $h_k \times h_k$ matrix whose (i,j)-th element is the taboo probability that starting from a state of the form $(J+k, i)$, the process eventually returns to node $J+k$ by visiting $(J+k, j)$, under the taboo of node J. These taboo probabilities only depend on k as $D_k e = D_{k,l} e$, for all $k \leq l$. One readily establishes that the following relationships hold:

$$V_k = B_k + \sum_{s=1}^{k} U_{k,s} (I - V_s)^{-1} D_{s,k}. \tag{6.4}$$

Let the (i,j)-th element of the $h_k \times h_s$ matrix $R_{k,s}$ denote the expected number of visits to state $(J+ks, j)$ before visiting node $J+k$ again, given that $(X_0, N_0) = (J+k, i)$ and let R_s hold the expected number of visits to state (s,j), under taboo of node $\emptyset$, given $(X_0, N_0) = (\emptyset, i)$. These matrices obey the following equations:

$$R_{k,s} = U_{k,s} (I - V_s)^{-1}, \tag{6.5}$$

$$R_s = U_s (I - V_s)^{-1}. \tag{6.6}$$

Finally, define the (i,j)-th entry of the matrix $h_k \times h_l$ $G_{k,l}$ as the probability that, starting from a state of the form $(J+lk,i)$, we eventually visit node $J+l$ by visiting state $(J+l,j)$. Similarly, let the $h_k \times h_\emptyset$ matrix G_k hold the probabilities that the root node is visited through state $(\emptyset,j)$ starting from state (k,i). The G-matrices can be expressed in terms of the V_k matrices by observing that:

$$G_{k,l} = (I-V_k)^{-1}D_{k,l}, \tag{6.7}$$
$$G_k = (I-V_k)^{-1}D_k. \tag{6.8}$$

6.4 Steady State Analysis

Throughout this section, we assume that the QBD process $\{(X_t,N_t), t \geq 0\}$ defined in the previous section is irreducible. In practice some transient states are allowed as their corresponding entries in the steady state vector will automatically become zero when applying the proposed algorithm. Clearly, the MC $\{(X_t,N_t), t \geq 0\}$ is recurrent if and only if

$$D_k e + V_k e = e,$$

for all $1 \leq k \leq d$, which can be reformulated as $G_{k,l}e = e$, for all $k \leq l$, by rewriting it as $(I-V_k)^{-1}D_k e = e$ and using $D_k e = D_{k,l}e$, as well as equation (6.7). An alternate stability condition will be given further on.

Define

$$\pi_i(J) = \lim_{t\to\infty} P[X_t = J, N_t = i], \tag{6.9}$$
$$\pi(\emptyset) = (\pi_1(\emptyset), \pi_2(\emptyset), \ldots, \pi_{h_\emptyset}(\emptyset)), \tag{6.10}$$
$$\pi(J') = (\pi_1(J'), \pi_2(J'), \ldots, \pi_{h_{f(J',1)}}(J')), \tag{6.11}$$

for $J' \neq \emptyset$. As a direct consequence of [62, Theorem 5.2.1] the following matrix-geometric form can be revealed for the components $\pi(J)$ of the steady state vector:

$$\pi(s) = \pi(\emptyset)R_s, \tag{6.12}$$
$$\pi(J+ks) = \pi(J+k)R_{k,s}. \tag{6.13}$$

The first set of balance equations can be rewritten as

$$\pi(\emptyset) = \pi(\emptyset)(F + \sum_{s=1}^{d} U_s(I-V_s)^{-1}D_s), \tag{6.14}$$

to provide us with the vector $\pi(\emptyset)$. Using (6.12) and (6.13), the normalization condition $\sum_J \pi(J)e = 1$ can be expressed as

$$\pi(\emptyset)\left(e + \sum_{s=1}^{d} R_s\Phi_s\right) = 1, \tag{6.15}$$

with

$$\Phi_1 = (I-R_{1,1})^{-1}e, \tag{6.16}$$
$$\Phi_k = (I-R_{k,k})^{-1}\left(e + \sum_{s=1}^{k-1} R_{k,s}\Phi_s\right), \tag{6.17}$$

for $1 \leq k \leq d$. If the MC $\{(X_t, N_t), t \geq 0\}$ is ergodic, then the series $\sum_J \pi(J)e$ must converge, meaning $\Phi_1, \Phi_2, \ldots, \Phi_d$ must converge, and this implies that the spectral radius of the matrices $R_{k,k}$, for $1 \leq k \leq d$, is less than one and $(I - R_{k,k})^{-1}$ exists.

The steady state vectors $\pi(J)$ can therefore be determined from the matrices V_k, for $1 \leq k \leq d$. Equation (6.4), for $k = 1$, equals

$$V_1 = B_1 + U_{1,1}(I - V_1)^{-1}D_{1,1},$$

which has an identical form as the matrix equation for the U-matrix of a classic QBD Markov chain [62]. Moreover, $D_{1,1}e + B_1e + U_{1,1}e = e$, therefore V_1 is the U-matrix of a QBD Markov chain characterized by $(D_{1,1}, B_1, U_{1,1})$. Thus, we can compute V_1 by solving the quadratic matrix equation

$$Y_1 = D_{1,1} + B_1Y_1 + U_{1,1}Y_1^2, \tag{6.18}$$

and setting $V_1 = B_1 + U_{1,1}Y_1$. Any algorithm, including the logarithmic or cyclic reduction algorithm, to solve this type of equation can be used to obtain Y_1 [14]. Rewriting equation (6.4) for V_k, for $k = 2, \ldots, d$ gives

$$V_k = \left(B_k + \sum_{s=1}^{k-1} U_{k,s}(I - V_s)^{-1}D_{s,k} \right) + U_{k,k}(I - V_k)^{-1}D_{k,k}. \tag{6.19}$$

Having obtained the matrices V_1 to V_{k-1}, V_k is also a U-matrix of a classic QBD characterized by $(D_{k,k}, \bar{B}_k, U_{k,k})$, where $\bar{B}_k$ is defined as the expression between brackets. It is readily checked that

$$D_{k,k}e + \bar{B}_ke + U_{k,k}e = e, \tag{6.20}$$

whenever $V_ke + D_ke = e$. Hence,

$$V_k = \bar{B}_k + U_{k,k}Y_k, \tag{6.21}$$

where Y_k solves the quadratic equation

$$Y_k = D_{k,k} + \bar{B}_kY_k + U_{k,k}Y_k^2. \tag{6.22}$$

In conclusion, to obtain the matrices V_k, it suffices to solve d quadratic matrix equations, each of these can be solved using an algorithm with quadratic convergence.

From the probabilistic interpretation of the Y_k matrices, one finds that $Y_k = G_{k,k}$. Also, if the $G_{k,k}$ matrices are stochastic, then so are all $G_{k,l}$ matrices, for $k < l$, because of (6.7) and $D_{k,k}e = D_{k,l}e$. Therefore, using the classic QBD stability condition [62], we find that the MC $\{(X_t, N_t), t \geq 0\}$ is stable if and only if

$$\theta_kD_{k,k}e < \theta_kU_{k,k}e,$$

where θ_k is the stochastic invariant vector of the stochastic matrix $D_{k,k} + \bar{B}_k + U_{k,k}$. Notice, this condition is not readily checked as the matrices V_k need to be determined first for $k = 1, \ldots, d-1$ in order to compute the matrices $\bar{B}_2$ to $\bar{B}_d$.

We end this section by introducing a simple procedure for some quantities that are useful when deriving performance measures. Let Π_k be the sum of all $\pi(J)$ vectors for which $f(J,1) = k$, then

$$\Pi_d = \pi(\emptyset)R_d(I - R_{d,d})^{-1}, \tag{6.23}$$

$$\Pi_k = \left(\pi(\emptyset)R_k + \sum_{s=k+1}^{d} \Pi_s R_{s,k}\right)(I - R_{k,k})^{-1}, \tag{6.24}$$

for $k = 1, \ldots, d-1$.

Theorem 7. *The probability of having t ($t > 0$) integers k on the string can be expressed as*

$$P[N(J,k) = t] = \Pi_k(I - R_{k,k})R_{k,k}^{t-1}(I - R_{k,k})\Phi_k.$$

Proof. One easily notices that

$$\Pi_k(I - R_{k,k}) = \pi(\emptyset)R_k + \sum_{s=k+1}^{d} \Pi_s R_{s,k}$$

corresponds to the sum of all $\pi(J)$ vectors for which $f(J,1) = k$, but $f(J,2) \neq kk$. As a consequence, $\Pi_k(I - R_{k,k})R_{k,k}^{t-1}$ equals the sum of all $\pi(J)$ vectors for which $f(J,t) = k \ldots k$, but $f(J,t+1) \neq kk \ldots k$. Multiplying this result with

$$(I - R_{k,k})\Phi_k = \left(e + \sum_{s=1}^{k-1} R_{k,s}\Phi_s\right)$$

is therefore exactly the probability of having t integers k on the string. □

6.5 The Multilevel Feedback Queue

The multilevel feedback queue is a scheduling discipline introduced in the late 1960s that has become an important CPU scheduling principle found in many operating systems textbooks. The goal of a multilevel feedback queue scheduler is to fairly and efficiently schedule a mix of processes with a variety of execution characteristics (e.g., I/O-bound or CPU-bound). By introducing different priority levels and by controlling how a process moves between priority levels, processes with different characteristics can be scheduled as appropriate. The scheduler attempts to provide a compromise between several desirable metrics, such as response time for interactive jobs, throughput for compute-intensive jobs, and fair allocations for all jobs. A multilevel feedback queue gives preference to short and I/O-bound jobs and quickly establishes the nature of a process and schedules it accordingly.

In its pure form, the multilevel feedback queue uses a set of priority queues and operates as follows. All new job arrivals are termed class-0 jobs and are placed in the priority 0 FIFO queue. The scheduler processes these class-0 jobs in a time sharing manner, that is, each class-0 job is processed for a maximum of q_0 time units. If a job is not completed before the quantum q_0 expires, it becomes a class-1 job and is placed at the back of the priority 1 FIFO queue, where it is set to wait until the priority 0 queue is empty. When the scheduler finds the priority 0 queue empty, it continues serving class-1 jobs. A class-1 job receives at most q_1 time units of uninterrupted service before becoming a class-2 job, etc. In general,

- A strict priority discipline is used between all FIFO queues.
- A class-k job can only receive service when there are no class-l jobs with $l < k$ present.
- When a class-k job enters the CPU, it is allowed to occupy the processor for a maximum of q_k time units.
- If a class-k job does not end within its quantum of length q_k, it becomes a class-$(k+1)$ job and is placed at the back of the priority $k+1$ FIFO queue.
- New jobs arriving while a class-k job, with $k > 0$, is in service will not preempt the class-k job before it has received q_k time units of service, unless the class-k job completes.

Thus, a multilevel feedback queue will favor short jobs, without having a priori knowledge about the job lengths. Starvation of large jobs can be counteracted by assigning longer quanta q_k to the lower priority classes.

Most work in this area has focused on systems with an infinite number of classes and on the limiting case where $q_k = q \to 0$ for all k. Using the framework introduced in the previous sections, we demonstrate how one can analyze the multilevel feedback queue under the following assumptions:

1. Time is slotted and all quanta q_k are expressed as multiples of the time unit.
2. The quanta $q_k = 1$ for all k. This assumption can be relaxed without great difficulty, we restrict ourselves to $q_k = 1$ to simplify the Markov chain description.
3. A (possibly empty) batch of jobs arrives at each slot boundary. The batch size in consecutive slots is assumed independent and identically distributed. We denote b_i as the probability of having a size i batch, for $i \geq 0$. Without much effort, one can also incorporate a batch Markovian arrival process (D-BMAP), if needed.
4. The processing time S of all jobs is independent and identically distributed. Denote $s_i = P[S = i]$ as the probability that a job requires i time units of processing. This general distribution is assumed to have a finite support, i.e., there exists some d such that $P[S > d+1] = 0$.
5. The number of classes is (greater than or) equal to $d+1$ and are labeled class-0 to class-d. Notice, a class-d job always ends after one time unit of processing.
6. All class-k queues have a finite buffer size equal to N.

6.5.1 Markov Model

Denote c_t^i as the number of class-i jobs present in the system at time t. When observing the system just prior to the slot boundaries, a Markov chain $(C_t)_{t\geq 0}$ is obtained by keeping track of the queue contents c_t^i of all $d+1$ queues at the end of time slot t. Let $m(t)$ be the smallest index i such that $c_t^i > 0$ (or $d+1$ if no such index exists). This means that during time slot t a class-$m(t)$ job will be processed. We represent the tuple $(c_t^{m(t)+1}, \ldots, c_t^d)$ as an ordered string of integers $J_t = \psi(c_t^{m(t)+1}, \ldots, c_t^d)$ between 1 and d as follows:

$$\psi(c_t^{m(t)+1}, \ldots, c_t^d) \stackrel{def.}{=} \overbrace{d \ldots d}^{c_t^d} \overbrace{(d-1) \ldots (d-1)}^{c_t^{d-1}} \ \ldots \ \overbrace{(m(t)+1) \ldots (m(t)+1)}^{c_t^{m(t)+1}}. \qquad (6.25)$$

Thus, the tuple $(J_t, (m(t), c_t^{m(t)}))$ fully characterizes the system at time t and provides us with a Markov chain having a binomial-like tree of order d as its state space. Next, let us consider the different transitions that might occur in such a Markov chain, for now assume all queues are of infinite size ($N = \infty$). In all cases, the job in progress at time t will finish with probability

$$p_{m(t)+1} = P[S = m(t) + 1 | S > m(t)]. \tag{6.26}$$

Assume there are no new job arrivals at time t.

(A1) If $c_t^{m(t)} > 1$ and the job in progress finishes, $m(t+1) = m(t)$, $c_{t+1}^{m(t+1)} = c_t^{m(t)} - 1$ and $J_{t+1} = J_t$. If the job in progress did not complete, $m(t) + 1$ needs to be added to the string J_t to get J_{t+1}.

(A2) On the other hand, if $c_t^{m(t)} = 1$ and the job completes at time t, we will delete all the integers equal to $m(t+1) = f(J_t, 1)$ from the string J_t to obtain J_{t+1} and set $c_{t+1}^{m(t+1)}$ equal to number of integers removed. Provided that the string $J \neq \emptyset$, otherwise the string remains identical to $\emptyset$. If the job does not complete, $m(t+1) = m(t) + 1$ and we remove all (if any) integers equal to $m(t) + 1$ from the string J_t, while setting $c_{t+1}^{m(t+1)}$ equal to one plus the number of removed $m(t)+1$ values. Clearly, some of these transitions are not allowed in a QBD MC, as multiple integers are potentially deleted from the string J_t at once. Further in this section, we will pay attention to this kind of transitions.

If one or multiple new jobs do arrive, we distinguish between having (B1) $m(t) = 0$ and (B2) $m(t) > 0$, in both cases $m(t+1) = 0$.

(B1) In this particular case we simply increase the value of $c_t^{m(t)}$ taking into account the new arrivals and add a 1 to the string J_t if the job in progress did not complete.

(B2) If there were no class-0 jobs present, we first add a single $m(t) + 1$ (if there is no job completion) followed by a series of $c_t^{m(t)} - 1$ integers $m(t)$. Also, c_{t+1}^0 will hold the number of new arrivals. Again, the QBD does not allow us to add a series of integers at once to the string J_t.

To reduce the Markov chain $(J_t, (m(t), c_t^{m(t)}))$ to a QBD having a binomial-like tree as its state space, we will slightly expand the state space of the chain. Currently, we have $J_t \in \Omega$ and

$$\begin{aligned} (m(t), c_t^{m(t)}) &\in \{(k,n) | k = 0, \ldots, f(J_t, 1) - 1, n = 1, \ldots, N\}, \quad && J_t \neq \emptyset, \\ (m(t), c_t^{m(t)}) &\in \{(k,n) | k = 0, \ldots, d, n = 0, \ldots, N\}, && J_t = \emptyset. \end{aligned}$$

Remark that the number of states of a node depends on the rightmost integer of the string identifying the node. By limiting the range of the variable $c_t^{m(t)}$ to N, we automatically enforce that all queues have a finite capacity equal to N. Although the string J_t might contain more than N identical integers i, the ones that should have been deleted earlier (due to a buffer overflow), are deleted as soon as service is provided to the class-i queue (because all these integers need to be stored into the $c_t^{m(t)}$ variable first).

The transitions described in step (A2) can be avoided by adding a set of states

$$\{(-,n)|0 \leq n \leq N\}$$

to each node variable J_t. Whenever we need to remove a series of $k > 1$ identical integers $m(t+1)$ from the string J_t, we can remove them one at a time. First, we enter the state $(-,0)$ after which the first $m(t+1)$ is removed from J_t and a transition to state $(-,1)$ is made, subsequently we visit the states $(-,2),(-,3)$ to $(-,k-1)$, each time removing one $m(t+1)$ from the string. While removing the last $m(t+1)$ we enter state $(m(t+1),k)$. Remark that there is no real need to include state $(-,0)$, however it allows us to give a somewhat easier description of the transition matrices. Notice, such transitions can be captured by a QBD as a transition to a parent node may depend on the rightmost integer of both the node and its parent, thus we can stay in states of the form $(-,x)$ as long as both these integers remain identical. As a consequence, there is no need to store the class $m(t+1)$ in the auxiliary variable when removing a series of identical integers, which is beneficial for the block size of the matrices involved.

The transitions of step (B2) are slightly more complicated in the sense that we add a list of k identical integers $m(t)$ possibly preceded by a single $m(t)+1$. To distinguish between these two scenario's we add two sets of additional states:

$$\{(+,n)|1 \leq n \leq N\} \cup \{(+1,n)|1 \leq n \leq N\}.$$

For such transitions, the variable $c_{t+1}^{m(t+1)}$ will hold the nonzero size of the batch arrival. If needed, we start by adding $m(t)+1$ to the string J_t and setting the auxiliary variable equal to $(+1,k)$, to indicate that k identical integers need to be added to the string. The value of these integers can be deduced by subtracting one from the rightmost element of the string (hence, the 1 in the notation). Subsequently, we visit states $(+,k-1)$ to $(+,1)$ and each time add a single integer to the string (the value of which is identical to the current rightmost element of the string, except for the first step where we might need to subtract one). Finally, we enter state $(0,x)$, where x is drawn from the batch size distribution, knowing it concerns a nonempty batch. Thus, we postpone determining the batch size until we add the last element to the string. We can support all these transitions by the QBD framework as a transition to a child node is allowed to depend on the rightmost element of the current node. The matrices B_k, $D_{k,l}$, $U_{k,s}$, F, D_k and U_s, for $1 \leq s \leq k \leq l \leq d$, describing the QBD on the binomial-like tree are presented in Section 6.5.2. Having obtained the steady state vectors $\pi(J)$ of this expanded (QBD) Markov chain, we can deduce the steady state of the chain $(J_t,(m(t),c_t^{m(t)}))$ by applying a censoring argument. Actually, it suffices to replace the vector e in (6.15) by a vector that contains only ones on the entries corresponding to the original states.

6.5.2 Transition Matrices

Before presenting some numerical examples, we will briefly describe the transition matrices of the expanded QBD Markov chain $(J_t,(m(t),c_t^{m(t)}))$. Since three types of additional states were added, we have $h_k = (3+k)N+1$, for $k = 1,\ldots,d$. The first block row corresponds to a transition from an additional state of type $(-,n)$, with $0 \leq n \leq N$, the second and the third block row to a transition from a state of type $(+1,n)$, respectively $(+,n)$, with $1 \leq n \leq N$. The first block row will therefore consist of $N+1$ rows, whereas the other block rows have

only N rows. To reduce the notational complexity, we will use the short notations presented on Page xv for the matrices with a special structure. Besides, we introduce the following matrices of which the appropriate dimension depends on the context:

- $M^{(i,j)}$ is a zero matrix with a one in row i, column j.
- $b^1 = (b_1, b_2, \ldots, b_N)$, where $b_j = 0$ if j exceeds the maximum batch size.
- B_T^1 denotes an upper triangular Toeplitz-like matrix with its first row equal to b^1 that is given by

$$B_T^1 = \begin{bmatrix} b_1 & b_2 & \cdots & b_N \\ 0 & b_1 & \cdots & b_{N-1}^* \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & b_1^* \end{bmatrix},$$

with $b_i^* = \sum_{k=i}^{N} b_k$. The $h_k \times h_k$ matrix B_k covers the transitions where $J_{t+1} = J_t$ and $f(J_t, 1) = k$. Such a transition occurs when $c_t^{m(t)} = 1$, with $m(t) \in \{0, \ldots, k-1\}$, and no new jobs arrive. The resulting state then equals $(-, 0)$. If there is at least one new arrival, we can skip the states of the form $(-, n)$ by moving immediately to state with $m(t+1) = 0$. The transitions that take place if more than one job was present while the one being processed finishes its service are described in (A1). This results in

$$B_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ p_1 b_0 M^{(1,1)} & 0 & 0 & p_1(B_T^1 + b_0 I^-) & 0 & 0 & \cdots & 0 \\ p_2 b_0 M^{(1,1)} & 0 & 0 & p_2 e_1 b^1 & p_2 b_0 I^- & 0 & \cdots & 0 \\ p_3 b_0 M^{(1,1)} & 0 & 0 & p_3 e_1 b^1 & 0 & p_3 b_0 I^- & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_k b_0 M^{(1,1)} & 0 & 0 & p_k e_1 b^1 & 0 & 0 & \cdots & p_k b_0 I^- \end{bmatrix}. \tag{6.27}$$

The $h_k \times h_l$ matrices $D_{k,l}$ contain the transition probabilities between state $(-, i)$ and state $(-, i+1)$, where $f(J_t, 1) = k$ and $f(J_{t+1}, 1) = l$. As long as $k = l$, we remove the next k from the string. Whenever the last k has been removed, the system moves to a state where $m(t+1) = k$. Hence,

$$D_{k,k} = \begin{bmatrix} I^+ & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6.28}$$

$$D_{k,l} = \begin{bmatrix} 0 & \cdots & 0 & I^= + M^{(N+1,N)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6.29}$$

for $l > k$. The nonzero entries of the matrix $D_{k,l}$ are positioned on the $(k+4)$-th block column of the first block row since these correspond to states where type-k jobs are processed. The

$h_k \times h_s$ matrices $U_{k,s}$ represent the situations in which an integer is added to the string J_t and are given by

$$U_{k,1} = \Delta_{k,1} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} \\ b_0 M^{(1,1)} & 0 & 0 & B_T^1 + b_0 I^- \\ 0 & 0 & (1-b_0) I^{--} & e_2 b^1 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.30}$$

$$U_{k,s} = \Delta_{k,s} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} & 0 & \cdots & 0 \\ 0 & 0 & I^- & \frac{e_1 b^1}{1-b_0} & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_0 M^{(1,1)} & (1-b_0) I^- & 0 & e_1 b^1 & 0 & \cdots & b_0 I^- \\ 0 & 0 & (1-b_0) I^{--} & e_2 b^1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6.31}$$

(6.32)

with $s \neq 1$ and $\Delta_{k,s}$ a $h_k \times h_k$ diagonal matrix represented by

$$\Delta_{k,s} = diag[(e^t, 1_{\{k=s+1\}} e^t, 1_{\{k=s\}} e^t, e^t, \ldots, (1-p_s) e^t, p_{s+1} e^t, e^t, \ldots, e^t)],$$

In the matrix $U_{k,s}$ the nonzero (and in $\Delta_{k,s}$ the non-one) entries are located on the block rows 2, 3, $s+3$, and $s+4$. In these matrices we can distinguish different situations. The second and third block row correspond to the states $(+1, n)$, resp. $(+, n)$. If the Markov chain resides in one of these states, we concatenate an s to the string and upon the concatenation of the last s, the chain makes a transition to a state that corresponds to a class-0 job being processed. The nonzero entry in the first column covers the case in which the last class-$(s-1)$ customer with a service time of at least $s+1$ slots is being processed. This customer will be added to the string as a class-s customer after which all customers of this class will be removed from the string, given that there are no new arrivals. It might seem a little laborious to add a customer to the string that will be immediately removed again, however, this extra transition simplifies the presentation of the matrices.

If new customers arrive while class-$(s-1)$ customers are being processed, a transition to a state $(+1, n)$ or $(+, n)$ will follow depending on the possible termination of the actual service (see block entries $s+3$ and $s+4$ in $\Delta_{k,s}$). Note the occurrence of the matrix I^{--} in the third block column due to the combination of a service completion and a customer being added to the string. In the fourth block column we encounter expressions comparable to those in the matrices B_k since we can also skip the transition to $(-, 0)$ here. Finally, the last block column contains nonzero entries for the states without arrivals and where the job in process

does not complete its service. Considering the root node, the following equations describe the matrices F, U_s and D_k (for $k, s = 1, \ldots, d$),

$$F = \begin{bmatrix} b_0 & b^1 & 0 & 0 & \cdots & 0 & 0 \\ p_1 b_0 e_1 & p_1(B_T^1 + b_0 I^-) & 0 & 0 & \cdots & 0 & 0 \\ p_2 b_0 e_1 & p_2 e_1 b^1 & p_2 b_0 I^- & 0 & \cdots & 0 & 0 \\ p_3 b_0 e_1 & p_3 e_1 b^1 & 0 & p_3 b_0 I^- & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ p_d b_0 e_1 & p_d e_1 b^1 & 0 & 0 & \cdots & p_d b_0 I^- & 0 \\ b_0 e_1 & e_1 b^1 & 0 & 0 & \cdots & 0 & b_0 I^- \end{bmatrix}, \tag{6.33}$$

$$D_k = \begin{bmatrix} 0 & \cdots & 0 & I^= + M^{(N+1,N)} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6.34}$$

with the nonzero block on the $(k+2)$-th block column,

$$U_1 = \Delta_1 \begin{bmatrix} 0 & 0 & 0 & 0 \\ b_0 M^{(1,1)} & 0 & 0 & B_T^1 + b_0 I^- \\ 0 & 0 & (1-b_0) I^{--} & e_2 b^1 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{6.35}$$

$$U_s = \Delta_s \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ b_0 M^{(1,1)} & (1-b_0) I^- & 0 & e_1 b^1 & 0 & \cdots & b_0 I^- \\ 0 & 0 & (1-b_0) I^{--} & e_2 b^1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{6.36}$$

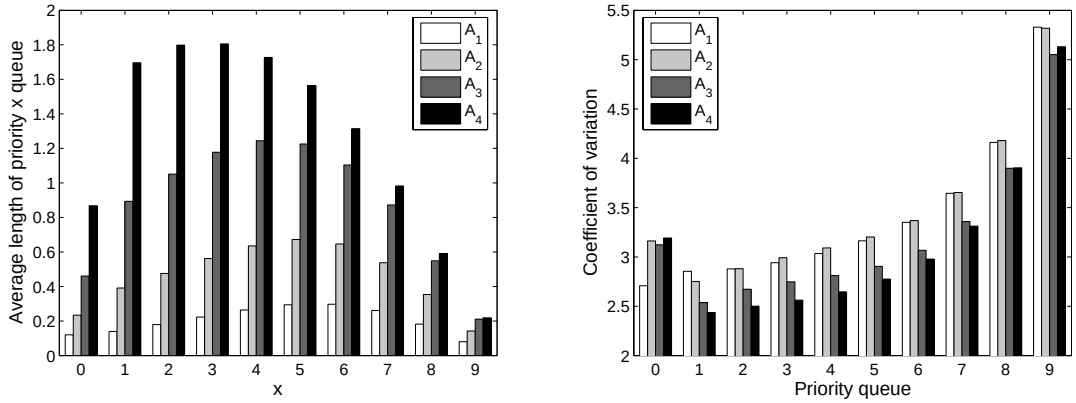
for $s \neq 1$ and where

$$\Delta_s = diag[(e^t, \ldots, e^t, (1-p_s)e^t, p_{s+1}e^t, e^t, \ldots, e^t)].$$

These equations can be understood using arguments similar to those presented for transitions between two nonroot nodes. Remark, there are no additional states in the root node, hence $h_\phi = (d+1)N + 1$. The first row of the matrices F and U_s corresponds to a transition from an idle system, that is, the situation where $m(t) = d + 1$.

6.5.3 Performance Measures

To conclude this chapter, we present some illustrative numerical examples. In these examples, we take a closer look at the influence the batch size and the processing time distribution have on the queue length of each queue individually and on the average buffer occupancy of the entire system. The average response times (per job length) are also discussed. Consider a multilevel feedback queue with 10 priority classes where each buffer can store at most $N = 30$ jobs.



(a) Average queue length

(b) Dispersion of the queue length distribution

Figure 6.2: Average queue lengths in a multilevel feedback queue with 10 priorities and different arrival characteristics

The processing time of a job follows a discrete uniform distribution with $P[S = i] = 0.1$, for $i = 1, \ldots, 10$. The mean arrival rate equals 0.12 jobs per slot, resulting in a total load of $\rho = 0.66$. We consider 4 different arrival processes, given by

$A_1 :\ b_0 = 0.88$, $b_1 = 0.12$, and, $b_i = 0$, for $i > 1$,

$A_2 :\ b_0 = 0.94$, $b_1 = 0.03$, $b_i = 0.01$, for $2 \leq i \leq 4$, and, $b_i = 0$, for $i > 4$,

$A_3 :\ b_0 = 0.98$, $b_6 = 0.02$, and, $b_i = 0$, for $i \notin \{0, 6\}$,

$A_4 :\ b_0 = 0.99$, $b_{12} = 0.01$, and, $b_i = 0$, for $i \notin \{0, 12\}$.

Notice, $A_1 \leq_{cx} A_2 \leq_{cx} A_3 \leq_{cx} A_4$ in the convex ordering sense (see Def. 2). Figure 6.2(a) shows the average queue length for each priority class. As the batch size distribution becomes larger, the average queue length increases for each queue. Furthermore, the number of the priority queue attaining the highest average occupancy decreases with the burstiness of the arrival process. From Figure 6.2(b) it can be observed that the coefficient of variation, i.e., the ratio of the standard deviation to the mean, is larger for queues with a lower priority. Furthermore, for a burstier arrival process, the coefficient of variation of the queue length is slightly smaller.

In the following example, we keep the arrival process fixed, while varying the job processing times. New jobs enter the multilevel feedback queue according to the second arrival process

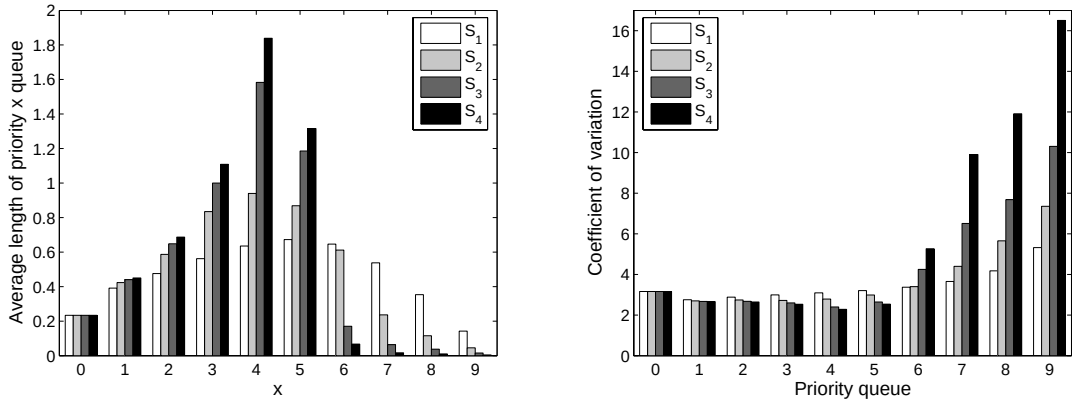
discussed in the previous example, i.e., $b_0 = 0.94$, $b_1 = 0.03$, $b_i = 0.01$, for $2 \leq i \leq 4$, and $b_i = 0$ for $i > 4$. Furthermore, we consider four different job processing time distributions given by:

$S_1 : P[S = i] = 0.1$ for $1 \leq i \leq 10$,

$S_2 : P[S = i] = 0.05$ for $1 \leq i \leq 3$, $8 \leq i \leq 10$, $P[S = i] = 0.175$ for $4 \leq i \leq 7$,

$S_3 : P[S = i] = 0.025$ for $1 \leq i \leq 4$, $7 \leq i \leq 10$, $P[S = i] = 0.4$ for $i \in \{5, 6\}$,

$S_4 : P[S = i] = 0.01$ for $1 \leq i \leq 4$, $7 \leq i \leq 10$, $P[S = i] = 0.46$ for $i \in \{5, 6\}$.



(a) Average queue length

(b) Dispersion of the queue length distribution

Figure 6.3: Average queue lengths in a multilevel feedback queue with 10 priorities under varying processing time distributions

Each of these distributions has an average, resp. maximum, processing time of 5.5, resp. 10, slots. Also, $S_1 \geq_{cx} S_2 \geq_{cx} S_3 \geq_{cx} S_4$ in the convex ordering sense. The queue length results obtained for these scenarios are presented in Figure 6.3. It can be observed that for the uniform processing time distribution, the average queue length is more spread out over the different queues compared to a processing time distribution where the probability mass is more centered around the mean. Considering the coefficient of variation, we observe larger values for the queues with a lower priority similar to the previous example.

In Figure 6.4 we present the average queue lengths over all 10 priority queues for each of the discussed scenarios (as well as the total loss rate ν due to the finite nature of the queues). How this loss rate can be computed, will be discussed later on. As mentioned earlier, the average queue length increases together with the burst size of newly arriving jobs. Considering the processing time distribution, we observe that the average queue length is minimal for the uniform distribution and increases for distributions where the probability mass is less spread out.

From the average queue length we can also deduce the average response time for the different jobs in the system. Denote jobs with a required processing time of $i + 1$ slots as type-i jobs, i.e., a type-i job has to pass through priority queues $0, \ldots, i$ before leaving the system. Using Little's law we can compute the average response time of a type-i job who

scenario	avg queue length	loss rate	scenario	avg queue length	loss rate
A_1	2.041 jobs	1.93e-7	S_1	4.650 jobs	1.38e-4
A_2	4.650 jobs	1.38e-4	S_2	4.896 jobs	2.55e-4
A_3	8.789 jobs	2.08e-3	S_3	5.378 jobs	9.72e-4
A_4	12.566 jobs	8.90e-3	S_4	5.732 jobs	1.48e-3

Figure 6.4: Average queue length in the 10-class multilevel feedback queue

was allowed to enter the first queue. We start by determining the input rate of the different buffers, taking into account jobs that leave the system after having received their required processing time as well as jobs that are dropped due to buffer overflow. For priority queue 0, the input rate equals the average rate at which a job arrives in the system, that is,

$$\lambda_0 = \sum_{i=1}^{\infty} i b_i. \tag{6.37}$$

The input rate λ_i of queue i ($i > 0$) can be obtained from the output rate μ_{i-1} of queue $i-1$, which can be computed by

$$\mu_{i-1} = \pi_{i-1}(\emptyset)e + \sum_{j=i}^{d} \Pi_{j,i-1}e, \tag{6.38}$$

with

$$\begin{aligned} \pi_{k,l}(J) &= \lim_{t\to\infty} P[X_t = J, (m(t), c_t^{m(t)}) = (k,l)], \\ \pi_k(J) &= (\pi_{k,1}(J), \ldots, \pi_{k,N}(J)), \\ \pi(J) &= (\pi_0(J), \ldots, \pi_d(J)), \end{aligned}$$

and $\Pi_{k,l}$ is defined as the sum off all vectors $\pi_l(J)$ for which $f(J,1) = k$, analogous to equations (6.23) and (6.24). That is, in (6.38) we sum all probabilities corresponding to states where queue $i-1$ is active. The arrival rate λ_i is then given by

$$\lambda_i = (1 - p_i)\mu_{i-1}. \tag{6.39}$$

Applying Little's law to each of these $d+1$ queues, we see that the mean time spent in queue i of all jobs passing through queue i equals N_i/μ_i (notice, as the queue is finite, we have to use μ_i and not λ_i). As the type of a type-j job with $j \geq i$ is irrelevant for the time it spends in queue i, this mean value is correct for all type-j jobs passing through queue i. We denote the average response time of a type-i job that entered our system as RT_i. Thus, if a type-i job is only partially processed, it also contributes to RT_i. As all the type-0 jobs entering the system get processed by queue 0, we have

$$RT_0 = \frac{N_0}{\mu_0}. \tag{6.40}$$

For the type-1 jobs, there are two components: N_0/μ_0 for the mean time a class-1 customer spends in queue 0, plus the mean time it spends in queue 1, provided that it enters queue 1. The probability that a type-1 job enters this queue is μ_1/λ_1. In conclusion,

$$RT_1 = \frac{N_0}{\mu_0} + \frac{N_1}{\lambda_1}. \tag{6.41}$$

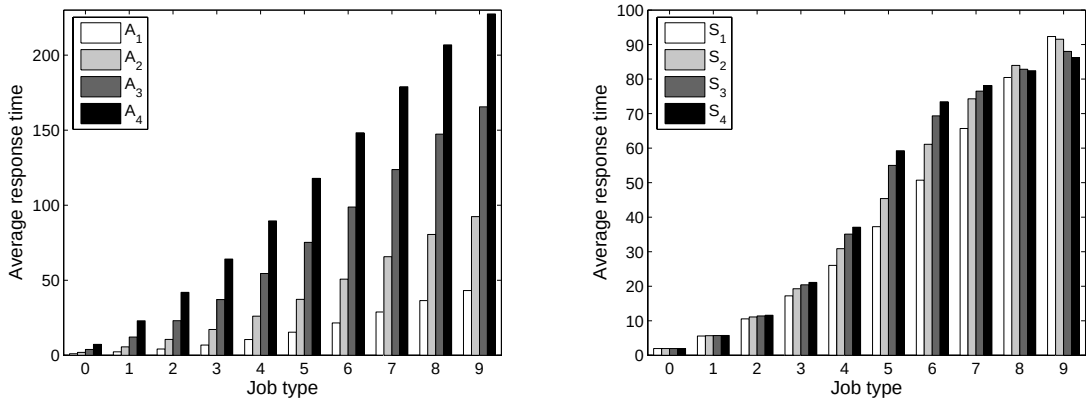


Figure 6.5: Average response time of type t customers in a multilevel feedback queue under varying arrival characteristics and with different processing time distributions

Similarly, one establishes that

$$RT_i = \frac{N_0}{\mu_0} + \sum_{k=1}^{i} \frac{N_k}{\lambda_k}.$$

Figure 6.5 presents the average response time of the different types of jobs in the 8 scenarios. As expected, the burstiness of the arriving jobs has a large influence on this average response time. For the processing time distributions we observe that the distribution that leads to the smallest average response time depends on the type of customer. For example, for a customer with a required processing time of 6 slots, a larger distribution in the convex ordering sense leads to a smaller average response time, whereas for a customer of type 9, the opposite observation can be made.

From the input and output rate of each queue we can also obtain the different loss rates, i.e., the loss rate ν_i $(i > 0)$ of queue i is given by

$$\nu_i = \lambda_i - \mu_i. \tag{6.42}$$

The total loss rate of the system then equals

$$\nu = \sum_{k=0}^{d} \nu_k. \tag{6.43}$$

In Figure 6.4 this total loss rate is presented for the different scenarios under consideration.

Part III

Transient Performance Measures

7

Quasi-Birth-Death Markov Chains

Part II focused on the steady state performance of different queueing systems. Intuitively, this can be seen as the behavior the system will exhibit after a considerable long time. In this part we are interested in the transient behavior of the system, that is, we want to know how the system evolves in its initial stage. This is especially interesting when studying systems that do not have a steady state, but also for other systems it can be useful to know something about their initial behavior since it might take a considerable amount of time before the steady state is eventually reached.

In this first chapter we present an algorithm to compute transient performance measures of a Quasi-Birth-Death Markov chain and in Chapter 8 we extend this method to the class of tree-like processes. In contrast to the results presented in Part II, we will not concentrate on a particular system here. Rather, we present a general framework that can be applied to any system that can be modeled as a QBD or a tree-like process. Although the work is presented in a discrete-time setting, it can be easily adapted for continuous-time systems.

The results presented in this chapter make use of the framework termed QBDs with marked time epochs [111] that transforms the transient problem into a stationary one by applying a discrete Erlangization and constructing a reset Markov chain. In contrast to a steady state analysis, the initial state of the process has an influence on the transient behavior of the system. To identify this influence, there is a need for an efficient manner of comparing the results for different initial states. A possible solution might be to compute the performance measures of interest for one specific initial state, then change the initial state and redo all computations. Obviously, this not the most optimal strategy and in this chapter we will show how the same result can be obtained with significantly less computations.

To avoid the need to repeat all computations for each initial configuration, we present a level based recursive algorithm that uses intermediate results obtained for initial states belonging to levels $0, \ldots, r-1$ to compute the transient measure when the initial state is part of level r. Also, the computations for all states belonging to level r are performed simultaneously. A key property of our approach lies in the exploitation of the internal structure of the block matrices involved, avoiding any need to store large matrices.

7.1 Related Work

Transient performance measures have long been recognized as being complementary to steady state measures [66, 24, 80, 84]. Either because there exist a need to understand the initial behavior of a system, or simply because the system has no steady state. Transient studies are moreover motivated as a means to investigate the impact of the initial system state in a simulation environment. Also the study of QBD Markov chains has a long tradition dating back to the early work by Neuts in the 1960s. Today this topic is still very actively pursued by many researchers. Early contributions to the transient analysis of QBDs were made by Ramaswami [82] and, Zhang and Coyle [119], mostly relying on Laplace transforms. More recent works include those by Le Ny and Sericola [80] and Remke et al. [85, 86], which both combine uniformization techniques with a recursive approach to tackle the resulting discrete-time problem. Their strength lies mostly in solving transient problems over short time scales, as the computation times increase as a function of the time epoch of the event of interest.

A new approach that reduces a variety of transient QBD problems to stationary ones, was introduced in [110] for the special case of a D-MAP/PH/1 queue. The two main steps introduced were the use of a discrete Erlangization and the construction of a reset Markov chain. Although the reset Markov chain contained considerably more states, compared to the original QBD, computation times were limited by exploiting the internal structure of the block matrices that characterize its transition matrix.

The underlying ideas presented in [110] gave rise to the development of a new framework, termed QBDs with marked time epochs [111], to assess transient measures in a plug-and-play manner for any QBD, where a significant improvement over [110] was also achieved in terms of the required computational resources. However, [111] was limited to the case where the initial state of the Markov chain was part of the boundary level. That is, the i-th component of some vector α_{ini} gave the initial probability of being in the i-th boundary state at time 0. An approach to deal with more general initial conditions for the D-MAP/PH/1 queue was included in [110]. This approach can even be extended to include general (bounded) initial distributions, meaning, the set of all possible initial states does not need to be a subset of a specific level. Although useful, this approach is restrictive in the sense that all computations need to be redone when considering a different initial state (distribution). This is especially problematic when we wish to compute transient results for each possible initial state as is often a measure of interest in the area of model checking [84].

7.2 Introduction

In this chapter we provide a novel approach to deal with more general initial conditions. Actually, a level based recursive algorithm is presented that computes the transient performance measure for all possible initial states. That is, having performed the necessary computations when the initial state is part of the first $r-1$ levels, we demonstrate how to obtain the transient measure when the initial state is part of level r (for each of the states belonging to level r). During this step we heavily rely upon previously computed intermediate results, avoiding the need to redo all computations for each new initial state considered. Notice, the recursion is *not* on the time epoch, as in many other studies, but on the level of the initial state. If we know the transient measure for any single initial state (up to some level N), the results are readily available for any initial distribution (bounded to the first N levels).

In practice, we will mark certain time epochs based on the performance measure of interest. The system state at the n-th marked time epoch will be approximated by introducing a so-called reset Markov chain. This translates the transient problem into a stationary one. To reduce the computational complexity, the reset Markov chain is converted into a level dependent Markov chain with a generalized initial condition. First, we will discuss an approach for an initial level equal to zero. For the other initial levels, we will develop a recursive algorithm that makes use of the computations done for the previous initial levels and that exploits the special structure of the matrices occurring in the calculations. A flexible Matlab implementation of the algorithm, taking among others the six matrices that characterize the QBD with marked time epochs as its input, is available online.

To some extent, the contribution of this chapter is related to the following. Many researchers that need to compute the system state at time n for some discrete-time Markov chain, will often take the initial probability vector and repeatedly multiply this with the transition matrix. Such an approach is fruitful, but the computation needs to be repeated each time we consider a different initial distribution. This can, in principle, be avoided by computing the n-th power of the transition matrix, but in practice this is often too time consuming. Our approach, although completely different in methodology, can intuitively be understood as intermediate to these two approaches. We compute the state for some initial vectors and use some of the intermediate results, to drastically speed-up further computations.

We start by giving some background information on QBDs with marked time epochs and we introduce the reset Markov chain of interest (Section 7.3). In Section 7.4 we convert this reset process into a level dependent QBD with a generalized boundary condition. A key role for the computation of the steady state probabilities of this QBD is played by a set of R-matrices [61] that can be computed recursively as demonstrated in Section 7.4.1. Next, the steady state probabilities are obtained from these R-matrices (Section 7.4.2) and to summarize, an algorithmic overview is given in Section 7.4.3. In Section 7.5 we illustrate our algorithm with some numerical examples.

7.3 QBDs with Marked Time Epochs: a Review

We are interested in the transient behavior of a QBD MC characterized by the transition matrix

$$\bar{P} = \begin{bmatrix} \bar{B}_1 & \bar{B}_0 & 0 & 0 & \cdots \\ \bar{B}_2 & \bar{A}_1 & \bar{A}_0 & 0 & \ddots \\ 0 & \bar{A}_2 & \bar{A}_1 & \bar{A}_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}, \tag{7.1}$$

where $\bar{B}_i$ and $\bar{A}_i$ are square matrices of dimension h, for $i = 0, 1$ and 2. Recall that the state space of this infinite Markov chain is partitioned into an infinite number of sets, which we call levels. Transitions among the levels are described by the $\bar{B}_i$ and $\bar{A}_i$ matrices. To avoid the notations from becoming unnecessarily complex, we assume that level zero of the QBD has the same number of states as all other levels. However, the method described in this chapter does not rely on this assumption and therefore may be relaxed if needed. The Matlab tool we developed does not have this restriction either, as will be illustrated in Section 7.5.

Furthermore, we denote the states of this MC as $\langle i, j \rangle$, where $i \geq 0$ denotes the level and $1 \leq j \leq h$ identifies the state within the level.

To obtain various transient performance measures in a unified manner, QBDs with marked time epochs (QBD^m) were introduced in [111]. Such a QBD^m is fully characterized by two sets of nonnegative matrices: a set with superscript ^{m} and one where all matrices have superscript ^{u}. These matrices have the additional property that the matrices defined as

$$\bar{A}_i = \bar{A}_i^u + \bar{A}_i^m, \tag{7.2}$$

$$\bar{B}_i = \bar{B}_i^u + \bar{B}_i^m, \tag{7.3}$$

for $i = 0, 1$ and 2, characterize a QBD Markov chain, and $\bar{B}_2^m e = \bar{A}_2^m e$. The probabilistic interpretation is as follows. The (j, j')-th entry of the matrix $\bar{A}_i^m$ contains the probability that at time t a transition occurs from state j of level s to state j' of level $s - i + 1$ and time epoch t is *marked*. The probabilities of the corresponding events without marking time t are given by the matrix $\bar{A}_i^u$. The same holds for the matrices $\bar{B}_i^m$ and $\bar{B}_i^u$.

Based on the transient performance measure we want to study, we mark part of the time epochs. For example, to obtain the system state at time n, we simply mark all time epochs. To compute the waiting time of the n-th customer in some queueing system that can be modeled as a QBD, we mark each time epoch in which an arrival occurs. To compute the work left behind by the n-th customer, we mark each time epoch in which a departure occurs, etc. For more examples, see [110] and [111]. Any transient problem that can be formulated in terms of the n-th marking of a QBD^m, can be solved in a plug-and-play manner. The technique used to obtain the system state at the n-th marking consists of two steps.

Step 1: Discrete Erlangization. Denote $\pi^m(n)$ as the probability vector associated with the n-th marked time epoch, i.e.,

$$\pi^m(n) = (\pi_0^m(n), \pi_1^m(n), \ldots),$$

where $\pi_i^m(n)$ is of size h for $i \geq 0$. To speed-up the computations, the system state at the n-th marking $t^m(n)$ is approximated by considering the system state at the $Z_{k,n}$-th marked time epoch $t^m(Z_{k,n})$, where $Z_{k,n}$ is a negative binomially distributed (NBD) random variable with k phases and mean n, for k ($\leq n$) sufficiently large. The larger k, the lower the variation of $Z_{k,n}$ and the better the approximation becomes. Setting $k = n$ provides us with exact results, however, k cannot always be set equal to n as the reset MC might become periodic (see Step 2). The idea of applying an Erlangization to assess transient results dates back to Ross [90], who applied this technique to approximate the system state at time t of a finite continuous-time Markov chain.

To compute the system state at time $t^m(Z_{k,n})$, an expanded Markov chain, called a reset Markov chain, was introduced. The key feature of such a Markov chain is that it reformulates the transient problem of computing the state at time $t^m(Z_{k,n})$ into a steady state analysis. Using the NBD as a reset time implies, among others, that the transition blocks of this reset MC have a special structure that can be exploited when computing its steady state probabilities.

Step 2: Reset Markov chains. Consider the stochastic process that evolves according to the transition matrix $\bar{P}$, but that is repeatedly reset when leaving the $Z_{k,n}$-th marked time epoch. A reset event corresponds to a transition from the current state to the initial state

of the Markov chain. Denote the initial level as r and the initial state within this level as l. A reset event therefore corresponds to a transition from the current state to state $\langle r, l\rangle$. If we perform a Bernoulli trial with parameter $p = k/n$ each time we have a transition out of a marked time epoch, the system is reset whenever k successes have occurred. We define the reset counter as being the number of pending successes before the next reset event. It is clear that this reset counter takes values in the range $\{1, 2, \ldots, k\}$. We will add the reset counter as an auxiliary variable to the Markov chain characterized by $\bar{P}$ and label its states as (c, j) with $1 \leq c \leq k$ and $1 \leq j \leq h$. As explained further on, if we succeed in computing the stationary behavior of the expanded MC, we can readily compute the system state just prior to a reset event, which is exactly the system state at time $t^m(Z_{k,n})$.

The work presented in [111] was restricted to QBDs whose initial state was part of level zero. Although the algorithm in [110], for more general initial conditions, is easily extended to the QBD^m framework, it requires that all computations need to be redone when considering a different initial state. In this chapter we therefore present an efficient algorithm to obtain transient performance measures for every possible initial configuration, without the need to repeat all computations for each configuration. We assume that at time zero the QBD resides in state l ($1 \leq l \leq h$) of level r. The reset process is characterized by the transition matrix $P_{k,n}$:

$$P_{k,n} = \mathcal{Q}_{k,n} + \mathcal{C}_{k,n}, \tag{7.4}$$

with

$$\mathcal{Q}_{k,n} = \begin{bmatrix} B_1^{k,n} & B_0^{k,n} & 0 & 0 & \cdots \\ B_2^{k,n} & A_1^{k,n} & A_0^{k,n} & 0 & \ddots \\ 0 & A_2^{k,n} & A_1^{k,n} & A_0^{k,n} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}, \tag{7.5}$$

$$\mathcal{C}_{k,n} = \begin{bmatrix} 0 & \cdots & 0 & C_0^{k,n} & 0 & \cdots \\ 0 & \cdots & 0 & C_1^{k,n} & 0 & \cdots \\ 0 & \cdots & 0 & C_1^{k,n} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{7.6}$$

where the matrices $C_0^{k,n}$ and $C_1^{k,n}$ appear on the $(r+1)$-th block column and where

$$A_i^{k,n} = (I_k \otimes (\bar{A}_i^u + (1-p)\bar{A}_i^m)) + (I^- \otimes p\bar{A}_i^m) \tag{7.7}$$

$$B_i^{k,n} = (I_k \otimes (\bar{B}_i^u + (1-p)\bar{B}_i^m)) + (I^- \otimes p\bar{B}_i^m) \tag{7.8}$$

$$C_0^{k,n} = M_1^k \otimes (p(\bar{B}_0^m e + \bar{B}_1^m e)\alpha_l) \tag{7.9}$$

$$C_1^{k,n} = M_1^k \otimes (p\bar{A}^m e\alpha_l), \tag{7.10}$$

for $i = 0$, 1 or 2 and $\bar{A}^m = \bar{A}_0^m + \bar{A}_1^m + \bar{A}_2^m$. The $k \times k$ matrix M_1^k has only one entry differing from zero being its last entry on the first row, which equals one. Further, $\alpha_l = e^{(l)}$, i.e., α_l is a $1 \times h$ row vector with all entries equal to zero, except for the l-th entry which equals one. Let

$$\pi^{k,n} = (\pi_0^{k,n}, \pi_1^{k,n}, \pi_2^{k,n}, \ldots)$$

be the steady state vector of $P_{k,n}$. Moreover, let $\pi_i^{k,n} = (\pi_{i,1}^{k,n}, \ldots, \pi_{i,k}^{k,n})$, partitioned in the obvious manner. Provided that the vector $\pi^{k,n}$ exists, the system state $\pi^m(Z_{k,n})$ at time $t^m(Z_{k,n})$, used as an approximation to $\pi^m(n)$, is the stochastic vector proportional to

$$(\pi_{0,1}^{k,n}.\phi_0, \pi_{1,1}^{k,n}.\phi_1, \pi_{2,1}^{k,n}.\phi_1, \ldots)p.$$

Here, ϕ_0 and ϕ_1 are the transposed vectors of $\left(\bar{B}_0^m e + \bar{B}_1^m e\right)$ and $\bar{A}^m e$, respectively and '.' denotes the point-wise vector product. That is, a reset will occur if the value of the reset counter equals one and the Bernoulli trial with parameter p results in a success. To compute the steady state vector of $P_{k,n}$ we shall construct a level dependent QBD Markov chain with a generalized initial condition, the transition matrix of which is denoted by P_{QBD}. It is in this respect that the current work differs from the approach taken in [110], where a different QBD reduction method was used to compute $\pi^{k,n}$. The novel construction allows us to simplify the calculation of the steady state vectors significantly when computing $\pi^{k,n}$ for every initial setting $\langle r, l\rangle$ with $r \geq 0$ and $1 \leq l \leq h$.

7.4 QBD Reduction

In this section, we show how to convert the reset MC, introduced in Section 7.3, into a level dependent QBD with a generalized boundary condition. For this purpose, we introduce h additional states to each of the levels $0, \ldots, r$ and we refer to these states as artificial states. Meaning, we increase the number of states for each of the first $r+1$ levels by a factor $(k+1)/k$. When a reset event occurs, the reset Markov chain characterized by $P_{k,n}$ makes a transition to state $\langle r, (k, l)\rangle$. In the reduced QBD approach, we split such a transition into $r + 2$ steps:

- **Step 1:** A transition to artificial state l of level zero takes place.
- **Step 2 to r+1:** Next, transitions between artificial state l of level i and artificial state l of level $i + 1$ will follow, for $i = 0, \ldots, r - 1$.
- **Step r+2:** Finally, when the state equals artificial state l of level r, a transition to the nonartificial state (k, l) of level r is made.

Before we discuss the transition matrix, let us reflect a bit on the reduction choices made. The key property of this reduction is that every reset event causes the MC to jump to level zero. This is different from the technique that was used in [110] where a reset event from some level i caused $|i - r|$ transitions in the direction of level r (after a transition to the artificial state of level i). That is, if a reset occurred when the current state was part of level $i < r$, first a transition was made to the single artificial state of level i after which $r - i$ transitions between artificial levels followed, each increasing the level by one. If the current level equals $i > r$, the transitions at a reset event are analogous, except that the level will now be decreased by one at every transition between two artificial states. Thus, as opposed to this technique, even if the reset event occurs at some level s, with $0 < s < r$, the path to state $\langle r, (k, l)\rangle$ now goes through level zero. Therefore, the MC becomes level independent starting from level $r + 1$ and requires no artificial states on the levels $s > r$.

The steady state vector of a QBD that becomes level independent at some level can be determined by a finite set of R-matrices [61]. By immediately selecting state l while visiting level zero, a single set of R-matrices suffices to compute the transient performance measures

for all h initial states of the form $\langle r, l\rangle$, with $1 \leq l \leq h$. Furthermore, these R-matrices can be reused when we look at initial states of the form $\langle r', l\rangle$, for $r' > r$. More specifically, when we increase r by one, the proposed reduction method only requires us to compute one additional R-matrix. This is a significant improvement over [110], where the entire set of R-matrices had to be recomputed for each new level. Finally, we shall see that these R-matrices have a useful block-structure and are identical to one another, except for the last block row.

Remark, it is possible to shorten the reduction procedure to $r+1$ steps by removing the h artificial states from level r (and by returning to a nonartificial state when going from level $r-1$ to level r). This causes no true performance gain (even though we need one R-matrix less), however, for uniformity reasons we decided to include step $r+2$.

The state space of the constructed QBD MC is organized such that whenever we visit an artificial state, we temporarily set the reset counter equal to $k+1$. This gives rise to the following transition matrix:

$$P_{QBD} = \begin{bmatrix} B_1^r + C_0^r & B_0^r & 0 & \cdots & 0 & 0 & 0 & \cdots \\ B_2^r + C_1^{\leq r} & A_1^{<r} & A_0^{<r} & \cdots & 0 & 0 & 0 & \cdots \\ C_1^{\leq r} & A_2^{\leq r} & A_1^{<r} & \ddots & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots \\ C_1^{\leq r} & 0 & 0 & \ddots & A_1^{<r} & A_0^{<r} & 0 & \cdots \\ C_1^{\leq r} & 0 & 0 & \ddots & A_2^{\leq r} & A_1^r & A_0^r & \cdots \\ C_1^{>r} & 0 & 0 & \ddots & 0 & A_2^{r+1} & A_1^{>r} & \cdots \\ C_1^{>r} & 0 & 0 & \ddots & 0 & 0 & A_2^{>r} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{7.11}$$

In this equation $A_i^{>r} = A_i^{k,n}$, for $i = 0, 1, 2$,

$$B_0^r = \begin{bmatrix} B_0^{k,n} & 0 \\ 0 & I_h \end{bmatrix}, \quad A_0^{<r} = \begin{bmatrix} A_0^{k,n} & 0 \\ 0 & I_h \end{bmatrix},$$

$$A_0^r = \begin{bmatrix} A_0^{k,n} \\ 0 \end{bmatrix}, \quad B_1^r = \begin{bmatrix} B_1^{k,n} & 0 \\ 0 & 0 \end{bmatrix},$$

$$A_1^{<r} = \begin{bmatrix} A_1^{k,n} & 0 \\ 0 & 0 \end{bmatrix}, \quad A_1^r = \begin{bmatrix} A_1^{k,n} & 0 \\ J & 0 \end{bmatrix},$$

$$B_2^r = \begin{bmatrix} B_2^{k,n} & 0 \\ 0 & 0 \end{bmatrix}, \quad A_2^{\leq r} = \begin{bmatrix} A_2^{k,n} & 0 \\ 0 & 0 \end{bmatrix},$$

and

$$\begin{aligned} A_2^{r+1} &= \begin{bmatrix} A_2^{k,n} & 0 \end{bmatrix}, \\ C_0^r &= M_1^{k+1} \otimes (p(\bar{B}_0^m e + \bar{B}_1^m e)\alpha_l), \\ C_1^{\leq r} &= M_1^{k+1} \otimes (p\bar{A}^m e\alpha_l), \\ C_1^{>r} &= M_2^{k+1} \otimes (p\bar{A}^m e\alpha_l). \end{aligned}$$

In these equations J is an $h \times hk$ matrix given by

$$J = \begin{bmatrix} 0 & \cdots & 0 & I_h \end{bmatrix}. \tag{7.12}$$

The matrix M_2^{k+1} is a $k \times (k+1)$ matrix with all entries equal to zero, except for the last entry on the first row which equals one.

A key role in the computation of the invariant vector $\hat{\pi}$ of P_{QBD} is played by a set of $(k+1)h \times (k+1)h$ R-matrices [61], which we denote as

$$R_0\{r\},\ R_1\{r\},\ \ldots,\ R_r\{r\},\ R_{>r}\{r\}.$$

For details about the computation of $\hat{\pi}$, we refer to Section 7.4.2. We briefly outline the case in which the Markov chain starts in state l of level zero, as this case was already discussed in detail in [111].

7.4.1 Computing the R-Matrices

7.4.1.1 Reset to Level Zero

Recall that there is no need to add step $r+2$ in the reduction process, which explains why there are no artificial states in [111], where the reset level equals zero. As a consequence, the dimension of the R-matrices for this level equals kh instead of $(k+1)h$. As described in [111, Section 5], we can first use Algorithm 7.1 to solve the equation

$$R = A_0^{>r} + RA_1^{>r} + R^2 A_2^{>r}. \tag{7.13}$$

This algorithm reduces the above quadratic matrix equation involving large matrices (size kh) to a single quadratic matrix equation and a set of Sylvester matrix equations of a much smaller dimension (size h). The Cyclic Reduction (CR) algorithm [72, 14] can be used to solve the quadratic matrix equation, whereas a solution to the Sylvester equations can be found using a Hessenberg algorithm [46]. This reduction can be applied because the matrices $A_i^{>r}$, for $i = 0, 1$ and 2, have a block triangular block Toeplitz (btbT) structure. A btbT matrix X is fully characterized by its first block column as follows:

$$X = \begin{bmatrix} X_1 & 0 & \cdots & 0 & 0 \\ X_2 & X_1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ X_{k-1} & X_{k-2} & \ddots & X_1 & 0 \\ X_k & X_{k-1} & \cdots & X_2 & X_1 \end{bmatrix}. \tag{7.19}$$

Actually, only X_1 and X_2 differ from zero in case of the $A_i^{>r}$ matrices. Since the btbT structure is preserved by the matrix multiplication, R also has a btbT structure, which allows us to compute the matrix R in a time and memory complexity of $O(h^3k^2)$ and $O(h^2k)$, respectively. Next, the matrix $R_0\{0\}$ defined as

$$R_0\{0\} = B_0^{k,n}(I - A_1^{>r} - RA_2^{>r})^{-1}, \tag{7.20}$$

can be computed efficiently by exploiting the btbT structure.

Algorithm 7.1 Computation of the btbT R-matrix for a reset to level zero

1. Due to the addition of the reset counter, the matrices $A_i^{>r}$ $(i = 0, 1, 2)$ exhibit the following structure with only two block diagonals different from zero,

$$A_i^{>r} = \begin{bmatrix} E_i & 0 & 0 & \cdots & 0 \\ F_i & E_i & 0 & \ddots & 0 \\ 0 & F_i & E_i & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & F_i & E_i \end{bmatrix}, \tag{7.14}$$

   where $E_i = \bar{A}_i^u + (1-p)\bar{A}_i^m$ and $F_i = p\bar{A}_i^m$.

2. Since R is the smallest nonnegative solution to equation (7.13), this matrix will have a btbT structure and can therefore be characterized by its first block column (see Eqn. (7.19)). Let $R_1, \ldots, R_k$ denote the blocks characterizing the btbT matrix R. Exploiting the structural properties of (7.13) one finds that the R_s matrices obey the following set of relations:

$$R_1 = E_0 + R_1 E_1 + R_1^2 E_2, \tag{7.15}$$
$$K_{s,4} = K_{s,1} R_s K_{s,2} + R_s K_{s,3}, \tag{7.16}$$

   where, for $s = 2, \ldots, k$,

$$\begin{aligned} K_{s,1} &= R_1, \\ K_{s,2} &= E_2, \\ K_{s,3} &= E_1 + R_1 E_2 - I, \\ -K_{s,4} &= R_{s-1} F_1 + \left(\sum_{j=2}^{s-1} R_j R_{s-j+1} \right) E_2 + \left(\sum_{j=1}^{s-1} R_j R_{s-j} \right) F_2 + 1_{\{k=2\}} F_0. \end{aligned}$$

   The first block R_1 can be found as a solution to equation (7.15). From this we can determine the consecutive blocks $R_2, \ldots, R_k$. That is, to compute the block R_s $(s = 2, \ldots, k)$, we need expressions for $K_{s,j}$ $(j = 1, \ldots, 4)$ which only depend on the matrices E_i, F_i and $R_1, \ldots, R_{s-1}$.

3. Equation (7.15) can be solved using the Cyclic Reduction (CR) algorithm [72]. Note that we still end up with a quadratic matrix equation, but the dimension of the matrices involved is significantly smaller compared to the original equation (7.13).

4. The $k-1$ Sylvester equations (7.16) can be solved using a Hessenberg algorithm [46] as presented in Algorithm 7.2. Note, the form of the equation that is solved in Algorithm 7.2 can be obtained from equation (7.16) by applying the transpose operator.

Algorithm 7.2 Solving a Sylvester equation of the form $AXB + CX = D$

1. Perform the following Hessenberg-triangular decomposition of (A, C):

$$\begin{aligned} WAV &= \bar{L}, \\ WCV &= \bar{N}, \end{aligned}$$

where W and V are unitary ($WW^* = VV^* = I$, with Z^* the complex conjugate of Z), $\bar{L}$ is a Hessenberg matrix and $\bar{N}$ a triangular matrix.

2. Let $U^*BU = T$ be a complex Schur decomposition of B, with U unitary and T a triangular matrix.

3. Next, premultiply $AXB + CX = D$ by W and postmultiply it by U. This results in

$$\bar{L}YT + \bar{N}Y = F, \tag{7.17}$$

with $Y = V^*XU$ and $F = WDU$.

4. Denote z_i and z_{ij} as the i-th column, resp. (i, j)-th entry of Z. Extracting column i from (7.17) leads to

$$(\bar{N} + t_{ii}\bar{L})y_i = f_i - \sum_{j=1}^{i-1} t_{ji}\bar{L}y_j. \tag{7.18}$$

Hence, to compute X, we need to solve d Hessenberg systems, where d denotes the dimension of the square matrices A, B, C and D and where each system can be solved in $O(d^2)$, resulting in a time complexity of $O(d^3)$.

7.4.1.2 Reset to an Arbitrary Level $r > 0$

The fundamental idea of our algorithm is to compute the steady state probabilities for the other initial configurations recursively. To achieve this, we establish a simple recursion on r among the R-matrices $R_0\{r\}, R_1\{r\}, \ldots, R_r\{r\}$ and $R_{>r}\{r\}$ needed to obtain the steady state vector of interest. Thus, after obtaining the two R-matrices R and $R_0\{0\}$ for level zero, we continue with the R-matrices needed when the initial level equals one, two, etc. The recursion can be understood by looking at the probabilistic interpretation of the R-matrices involved. More specifically, the (u, v)-th entry of the matrix $R_i\{r\}$ denotes the expected number of visits to state v of level $i + 1$, starting from state u of level i under taboo of the levels $0, \ldots, i$ and under the assumption that the reset level of the Markov chain equals r.

As explained below the following equalities hold:

$$R_i\{r\} = \begin{bmatrix} R & 0 \\ JG^{r-i-1}(I_{kh} - U)^{-1} & I_h \end{bmatrix}, \tag{7.21}$$

$$R_r\{r\} = \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{7.22}$$

$$R_{>r}\{r\} = R, \tag{7.23}$$

for $r \geq 1$ and $0 < i < r$. The btbT matrices U and G are given by

$$U = A_1^{k,n} + RA_2^{k,n}, \tag{7.24}$$

$$G = (I_{kh} - U)^{-1} A_2^{k,n}. \tag{7.25}$$

The expression for $R_{>r}\{r\}$ is immediate, as the expected number of visits to a state of level $i+1$ under taboo of the levels $0, \ldots, i$ does not depend on the actual value of $i > r$ since the transition matrix P_{QBD} is level independent from level $r+1$ onwards.

The equality for $R_i\{r\}$, for $i = 1, \ldots, r$, can be justified in two steps. First, consider the situation in which the starting state u is a nonartificial state of level i. All sample paths that avoid levels 0 to i do not contain a reset event, as all reset events result in a visit to level zero. Hence, all these paths evolve according to the matrices $A_0^{k,n}$, $A_1^{k,n}$ and $A_2^{k,n}$. Therefore, there are no visits to any of the artificial states of level $i+1$, while the expected number of visits to level $i+1$ is determined by R.

Second, if the starting state u is one of the h artificial states of level i, the next $r-i+1$ transitions are fixed. The first $r-i$ transitions are between the two corresponding artificial states of level j and $j+1$, for $i \leq j < r$, and the last transition between an artificial state l and its corresponding nonartificial state (k,l) both of level r. The expected number of visits to level $i+1$ will therefore solely depend on the number of levels between i and r. The (u,v)-th entry of G gives us the probability that the first visit to level i is a visit to state v, provided that we started in state u of level $i+1$, without the occurrence of a reset event.

Therefore, the first visit to level $i+1$ under taboo of the levels 0 to i, starting from a state of level r is determined by the matrix G^{r-i-1}. The (w,w')-th entry of the matrix $(I_{kh}-U)^{-1}$ holds the expected number of visits to state w' of level $i+1$, starting from state w of level $i+1$, under taboo of level 0 to i. Furthermore, starting from an artificial state of level $i < r$ implies that we visit exactly one artificial state of level $i+1$ before returning to level zero (hence, the I_h matrix in the lower right corner).

The matrix that remains to be determined is $R_0\{r\}$. In the case where the starting state u of level zero is not an artificial state, all sample paths that avoid level zero evolve according to the matrices $A_0^{k,n}$, $A_1^{k,n}$ and $A_2^{k,n}$, except for the first transition which is characterized by the matrix B_0^r. As a result, the expected number of visits to level 1 are determined by $R_0\{0\}$. If, on the other hand, the starting state is an artificial start, we may rely on the same arguments as for $i > 0$. Hence,

$$R_0\{r\} = \begin{bmatrix} R_0\{0\} & 0 \\ JG^{r-1}(I_{kh}-U)^{-1} & I_h \end{bmatrix}. \tag{7.26}$$

Equations (7.24) and (7.25) naturally lead to a simple recursion as

$$JG^{(r+1)-i-1}(I_{kh}-U)^{-1} = JG^{r-i-1}(I_{kh}-U)^{-1}\left(A_2^{k,n}(I_{kh}-U)^{-1}\right). \tag{7.27}$$

In conclusion, to compute all the R-matrices needed to assess the transient performance for all initial states $\langle r', l\rangle$ with $r' \leq r$, it suffices to compute

1. The first block column of the btbT matrices R and $R_0\{0\}$.

2. The last block row of the r matrices $R_0\{r\}, \ldots, R_{r-1}\{r\}$, which can be found recursively using a single btbT product (see (7.27)). From the probabilistic interpretation we see that $R_i\{r'\}$ equals $R_{i+r-r'}\{r\}$ for $r' \leq r$.

7.4.2 Computing the Invariant Probabilities

7.4.2.1 Reset to Level Zero

This section describes how to obtain the steady state probability vector $\hat{\pi}\{0,l\}$ of P_{QBD} when the initial state $\langle r,l\rangle = \langle 0,l\rangle$, for $1 \leq l \leq h$, using the matrices R and $R_0\{0\}$:

$$\hat{\pi}_0\{0,l\} = \hat{\pi}_0\{0,l\}(B_1^{k,n} + C_0^{k,n} + R_0\{0\}(B_2^{k,n} + (I-R)^{-1}C_1^{k,n})), \tag{7.28}$$

$$\hat{\pi}_1\{0,l\} = \hat{\pi}_0\{0,l\}R_0\{0\}, \tag{7.29}$$

$$\hat{\pi}_i\{0,l\} = \hat{\pi}_{i-1}\{0,l\}R, \tag{7.30}$$

for $i > 1$, while $\hat{\pi}_0\{0,l\}$ and $\hat{\pi}_1\{0,l\}$ are normalized as

$$\hat{\pi}_0\{0,l\}e + \hat{\pi}_1\{0,l\}(I-R)^{-1}e = 1.$$

Denote P_0 as the matrix $B_1^{k,n} + C_0^{k,n} + R_0\{0\}(B_2^{k,n} + (I-R)^{-1}C_1^{k,n})$ in Eqn. (7.28) of which $\hat{\pi}_0\{0,l\}$ is an eigenvector with eigenvalue one. The matrix P_0 is the transition matrix of the MC, characterized by P, when censored on the states corresponding to π_0. It can be observed that P_0 is the sum of a btbT matrix and a matrix with all entries equal to zero, except for the last block column. In order to compute $\hat{\pi}_0\{0,l\}$ efficiently, we can therefore rely on the algorithm presented in [111, Section 5.2], the time and space complexity of which equal $O(h^3k^2)$ and $O(h^2k)$, respectively. For reasons of completeness we will briefly present this algorithm in Algorithm 7.3.

Algorithm 7.3 Solving the boundary condition for a reset to level zero

1. Let $\hat{\pi}_0\{0,l\} = (\hat{\pi}_{0,1}\{0,l\},\ldots,\hat{\pi}_{0,k}\{0,l\})$ and denote the j-th entry of $\hat{\pi}_{0,i}\{0,l\}$ as $\langle i,j\rangle$.

2. Define G_i such that its (j,j')-th entry equals the expected number of visits to state $\langle k-i+1,j'\rangle$ starting from $\langle k,j\rangle$ until the first return to some state $\langle k,s\rangle$. This allows us to write

$$\hat{\pi}_{0,k-i+1}\{0,l\} = \hat{\pi}_{0,k}\{0,l\}G_i, \text{ for } i = 2,\ldots,k. \tag{7.31}$$

3. The following algorithm can be used to compute $\hat{\pi}_{0,k}\{0,l\}$ and the matrices G_i:

   (a) Let $G_1 = Y_1 + Z_k$ and set $G_i = Y_i$ for $i = 2,\ldots,k$

   (b) for $i = 3$ to $k+1$ do
   $G_{i-1} = G_{i-1}(I - Y_1)^{-1}$
   for $j = i$ to k do
   $G_j = G_j + G_{i-1}Y_{j-(i-1)+1}$
   end
   $G_1 = G_1 + G_{i-1}Z_{k-(i-1)+1}$
   end

   (c) $\hat{\pi}_{0,k}\{0,l\}$ is the solution to $\hat{\pi}_{0,k}\{0,l\} = \hat{\pi}_{0,k}\{0,l\}G_1$ and $\hat{\pi}_{0,k}\{0,l\}e = 1$.

   The variables Y_i and Z_i used in this algorithm are the blocks of the two terms in the expression of the matrix P_0, as defined in (7.32).

This algorithm solves a linear system of hk equations of the following form:

$$\hat{\pi}_0\{0,l\} = \hat{\pi}_0\{0,l\} \begin{bmatrix} Y_1 & 0 & \cdots & 0 & Z_1 \\ Y_2 & Y_1 & \ddots & 0 & Z_2 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ Y_{k-1} & Y_{k-2} & \cdots & Y_1 & Z_{k-1} \\ Y_k & Y_{k-1} & \cdots & Y_2 & Y_1 + Z_k \end{bmatrix}. \tag{7.32}$$

The computation of the set of matrices $G_1, \ldots, G_k$ is by far the most expensive part of the algorithm. However, instead of plugging in the h different α_l vectors in this algorithm (recall, the vector α_l has a one on the l-th position and zeros elsewhere), we can compute the h vectors $\hat{\pi}_0\{0,l\}$ simultaneously because changing the initial state $\langle 0,l\rangle$ does not alter the matrices $G_2, \ldots, G_k$, but only affects G_1. Moreover, keeping in mind that G_1 has to be stochastic, one can show that

$$G_1 = Y_1 + \alpha_l \otimes (e - Y_1 e)\,, \tag{7.33}$$

where Y_1 does not depend on l. Notice that also the matrices R and $R_0\{0\}$ do not depend on the initial state l and need to be computed only once.

7.4.2.2 Reset to an Arbitrary Level $r > 0$

Having obtained the matrices $R_0\{r\}, \ldots, R_r\{r\}$ and R, we immediately have the following expression for the steady state probability vector $\hat{\pi}$ of P_{QBD} (for the system where the reset level equals $r > 0$), due to [61]:

$$\hat{\pi}_i\{r,l\} = \begin{cases} \hat{\pi}_{i-1}\{r,l\}R_{i-1}\{r\} & \text{for } i = 1, \ldots, r+1, \\ \hat{\pi}_{i-1}\{r,l\}R, & \text{for } i > r+1. \end{cases} \tag{7.34}$$

Thus, it suffices to determine $\hat{\pi}_0\{r,l\}$, which obeys the following equation:

$$\hat{\pi}_0\{r,l\} = \hat{\pi}_0\{r,l\}N, \tag{7.35}$$

with

$$N = B_1^r + C_0^r + R_0\{r\}B_2^r + R_S^r C_1^{\leq r} + R_P^r (I - R)^{-1} C_1^{>r}, \tag{7.36}$$

and

$$R_S^r = \sum_{k=0}^{r-1} \left(\prod_{i=0}^{k} R_i\{r\} \right), \; R_P^r = \left(\prod_{i=0}^{r} R_i\{r\} \right). \tag{7.37}$$

To speed-up the computation of $\hat{\pi}_0\{r,l\}$, for $1 \leq l \leq h$, we exploit the structure of the square N matrix, the dimension of which equals $h(k+1)$. It might seem odd that we are considering a system of $h(k+1)$ linear equations, instead of just $hk+1$ as only one of the last h entries of $\hat{\pi}_0\{r,l\}$ will differ from zero (being the l-th). However, considering this somewhat larger N matrix, enables us to compute the h different $\hat{\pi}_0\{r,l\}$ vectors simultaneously.
Write the $h(k+1) \times h(k+1)$ matrix N as

$$N = \begin{bmatrix} N^B & N^C \end{bmatrix}, \text{with } N^B = \begin{bmatrix} N^T \\ N_r^R \end{bmatrix},$$

where N^T is an $hk \times hk$ matrix corresponding to transitions between nonartificial states, N_r^R an $h \times hk$ matrix and N^C an $h(k+1) \times h$ matrix.

Only the last block column of $C_0^r, C_1^{\leq r}$ and $C_1^{>r}$ differs from zero; therefore, N^T is a btbT matrix given by

$$N^T = B_1^{k,n} + R_0\{0\}B_2^{k,n}, \tag{7.38}$$

meaning N^T does not depend on the initial level r (and state l) of the QBD and needs to be computed only once. The block row N_r^R does depend on the initial level r and can be computed as

$$N_r^R = JG^{r-1}(I_{kh} - U)^{-1}B_2^{k,n}. \tag{7.39}$$

As we increase the initial level r, this matrix changes. However, we can easily compute it recursively by relying on Equation (7.27) with $i = 0$. The matrix N^C will only contain one nonzero column, which does not need to be computed to determine $\hat{\pi}_0\{r, l\}$.

Let us now discuss how the vectors $\hat{\pi}_0\{r, l\}$, for $1 \leq l \leq h$, can be obtained simultaneously from the matrices N^T and N_r^R. Therefore, we introduce the following notations:

$$N^T = \begin{bmatrix} N_1^T & 0 & \cdots & 0 \\ N_2^T & N_1^T & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ N_k^T & N_{k-1}^T & \cdots & N_1^T \end{bmatrix} \quad \text{and} \quad N_r^R = \begin{bmatrix} N_{r,1}^R & \cdots & N_{r,k}^R \end{bmatrix}.$$

The computation of the vectors $\hat{\pi}_0\{r, l\}$ is contained in Algorithm 7.4, which is a variant Algorithm 7.3 used to compute the invariant vector of a matrix of the form seen in Section 7.4.2.1. N has a slightly different structure, causing some minor modifications to the algorithm, however, as in [111, 60] it stems from a repeated censoring argument.

The algorithm discussed in this section allows us to compute the steady state vectors $\pi^{k,n}\{r', l\}$, for $r' \leq r$ and $1 \leq l \leq h$, from the matrices $R_0\{r\}, \ldots, R_r\{r\}$ and R. As we are looking at a transient event, choosing r sufficiently large, guarantees that $\pi_{0,1}^{k,n}\{r, l\}$ decreases to zero, because the probability of reaching level zero before a reset occurs decreases to zero. Thus, for r sufficiently large, $\pi^{k,n}\{r+1, l\}$ can be obtained from $\pi^{k,n}\{r, l\}$ by shifting its subvectors by one position. To determine the transient behavior for all initial states, we dynamically increase r until $\pi_{0,1}^{k,n}\{r, l\}$ is negligible or until a predefined upper bound on r is reached.

7.4.3 Algorithmic Overview

To conclude this section, we will present a short algorithmic overview of the discussed results. To obtain the transient performance measures of interest for every initial setting $\langle r, l\rangle$, implement the steps as described in Algorithm 7.5. Since we started from an arbitrary QBD Markov chain, this framework can be applied to any system that can be modeled as such. In the next section, we will consider the simple example of a D-MAP/PH/1 queue since this allows us to compare the efficiency of our algorithm with the algorithm presented in [110].

Algorithm 7.4 Computing the steady state probabilities for a reset to level r

1. First, we compute the k matrices G_2 until G_{k+1}:

   (a) Set $G_i = N^R_{r,k-i+2}$, for $i = 2, \ldots, k+1$,

   (b) for $i = 3$ to $k+2$ do
   $G_{i-1} = G_{i-1}(I_h - N_1^T)^{-1}$
   for $j = i$ to $k+1$ do
   $G_j = G_j + G_{i-1}N^T_{j-(i-1)+1}$
   end
   end

   The time and memory complexity of this step equals $O(h^3k^2)$ and $O(h^2k)$ respectively.

2. The vectors

$$\hat{\pi}_0\{r,l\} = (\hat{\pi}_{0,1}\{r,l\}, \ldots, \hat{\pi}_{0,k+1}\{r,l\}) \tag{7.40}$$

   can be retrieved from these matrices as follows:

   (a) $\hat{\pi}_{0,k+1}\{r,l\} = \alpha_l$,

   (b) for $i = 2$ to $k+1$
   $\hat{\pi}_{0,k-i+2}\{r,l\} = \alpha_l G_i$
   end

   Recall, α_l is a $1 \times h$ vector with a one in position l and zeros elsewhere. Thus, $\alpha_l G_i$ corresponds to the l-th row of the matrix G_i.

3. Due to Equation (7.34), the vector $\hat{\pi}_0\{r,l\}$ is normalized as

$$\hat{\pi}_0\{r,l\}e + \hat{\pi}_0\{r,l\}\left(R^r_S e + R^r_P(I-R)^{-1}e\right) = 1. \tag{7.41}$$

   Remark that for the btbT part of the matrices appearing in this equation, we only store the first block column.

4. Having obtained the steady state vector $\hat{\pi}\{r,l\}$ of P_{QBD}, we can derive $\pi^{k,n}\{r,l\}$, the invariant vector of $P_{k,n}$ as follows:

   (a) Write $\hat{\pi}_j\{r,l\}$, for $j = 0, \ldots, r$, as $(\hat{\pi}^f_j\{r,l\}, \hat{\pi}^{art}_j\{r,l\})$, where $\hat{\pi}^{art}_j\{r,l\}$ is a $1 \times h$ vector. Notice, there are no artificial states from level $r+1$ onwards, consequently we may write $\hat{\pi}_j\{r,l\} = \hat{\pi}^f_j\{r,l\}$, for $j > r$.

   (b) When censored on the nonartificial states, P_{QBD} coincides with $P_{k,n}$, hence,

$$\pi^{k,n}_j\{r,l\} = \hat{\pi}^f_j\{r,l\}/(1-c), \tag{7.42}$$

   where $j \geq 0$ and $c = \sum_{j=0}^{r} \hat{\pi}^{art}_j\{r,l\}e$.

5. An approximation for $\pi_m(n)$, the system state at the n-th marking can now be computed as described in Step 2 of Section 7.3.

Algorithm 7.5 Obtaining the transient performance measures for every initial configuration

1. Mark time epochs in correspondence with the desired performance measure and determine the matrices introduced in (7.11), using the equations (7.7) – (7.10). Note that for every matrix with a btbT structure only the first block column has to be stored.

2. Compute the matrices R and $R_0\{0\}$ from

$$\begin{aligned} R &= A_0^{>r} + RA_1^{>r} + R^2 A_2^{>r}, \\ R_0\{0\} &= B_0^{k,n}(I - A_1^{>r} - RA_2^{>r})^{-1}, \end{aligned}$$

   as discussed in Algorithm 7.1.

3. Use (7.28) – (7.30) to compute the steady state vector $\hat{\pi}\{0,l\}$. This can be done simultaneously for $l = 1, \ldots, h$ as described in Section 7.4.2.1.

4. Compute

$$N^T = B_1^{k,n} + R_0\{0\}B_2^{k,n} \quad \text{and} \quad U = A_1^{k,n} + RA_2^{k,n}.$$

5. Set $R_0^*\{1\} = J(I_{kh} - U)^{-1}$.

6. For each initial level $r > 0$ do

   - If $r > 1$ calculate

$$R_0^*\{r\} = R_0^*\{r-1\}\left(A_2^{k,n}(I_{kh} - U)^{-1}\right).$$

   - Set $N_r^R = R_0^*\{r\}B_2^{k,n}$.
   - Use the algorithm described in Section 7.4.2.2 to obtain $\hat{\pi}_0\{r,l\}$ from N^T and N_r^R simultaneously for $1 \leq l \leq h$.
   - Normalize the vector $\hat{\pi}_0\{r,l\}$ using Eqn. (7.41), where R_S^r and R_P^r can be computed recursively from R_S^{r-1} and R_P^{r-1} as indicated by Eqn. (7.37).
   - Compute

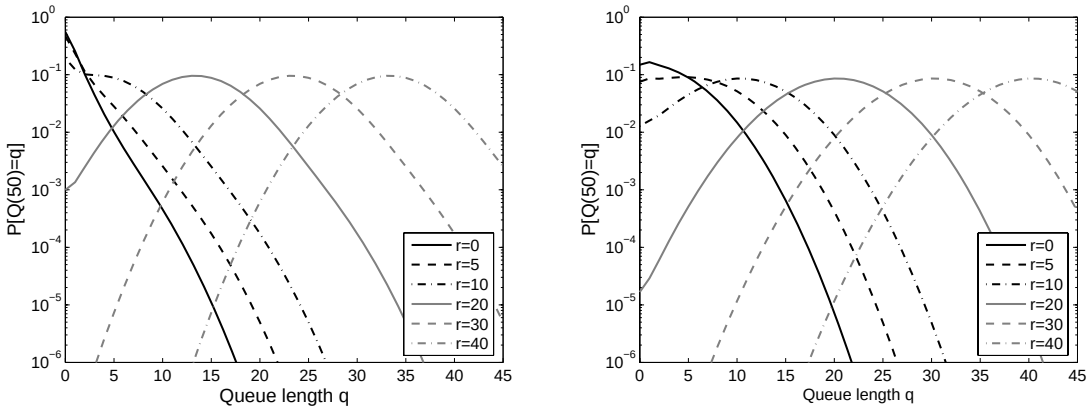
$$\begin{aligned} \hat{\pi}_i\{r,l\} &= \hat{\pi}_{i-1}\{r,l\}R_{i-1}\{r\}, \text{ for } 0 < i \leq r+1, \\ \hat{\pi}_i\{r,l\} &= \hat{\pi}_{i-1}\{r,l\}R \text{ for } i > r+1, \end{aligned}$$

     where the R-matrices are described in Section 7.4.1.2. Remark, the rows of $R_i\{r\}$ corresponding to the artificial states are given by $[R_0^*\{r-i\}\ I_h]$. Also, the product between a vector and a btbT matrix can be implemented using fast Fourier transforms, see [111, Section 5.3], resulting in a significant gain in time.
   - Finally, $\pi^{k,n}\{r\}$ can be found from Eqn. (7.42).

   until $\pi_{0,1}^{k,n}\{r,l\}$ is negligible or until a predefined upper bound on r is reached.

7.5 Numerical Examples

In this section we will demonstrate our approach with some numerical examples and compare the efficiency with the algorithm presented in [110, 111]. Therefore, we will compute the queue length distribution of a D-MAP/PH/1 queue (see Section 2.2) at some time t as well as the waiting time distribution of the n-th customer in this queue.



(a) $\alpha = 1$ (b) $\alpha = 2$

Figure 7.1: Queue length distributions of the D-MAP/PH/1 queue at time $t = 50$

In this example we consider a 2-state D-MAP (i.e., $m_a = 2$) that generates an arrival with probability 0.1 while in state 1 and with probability 0.25 when the state of the underlying Markov chain equals 2. The average sojourn time is 500 slots in state 1 and 1000 slots in state 2, resulting in an arrival rate $\lambda = 0.2$. Hence,

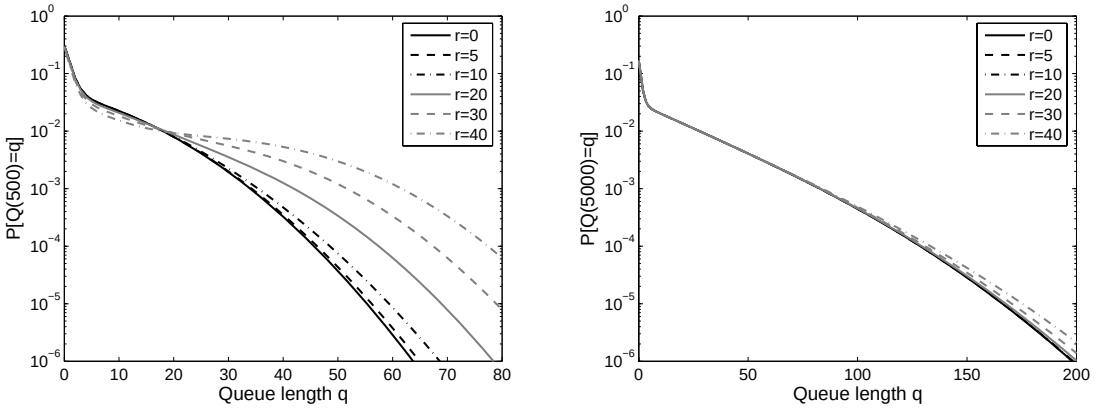
$$D_0 = \begin{bmatrix} .89820 & .00180 \\ .00075 & .74925 \end{bmatrix}, \; D_1 = \begin{bmatrix} .09980 & .00020 \\ .00025 & .24975 \end{bmatrix}.$$

Denote the initial state of the D-MAP process by α. The service time of a customer follows a discrete-time PH distribution with a matrix representation (m_s,β,T), where m_s is a positive integer. For this example we take $m_s = 3$,

$$\beta = (3/4, 1/8, 1/8), \tag{7.43}$$

$$T = \begin{bmatrix} 4/5 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix}. \tag{7.44}$$

Remark, the mean service time equals $E[S] = \beta(I - T)^{-1}e = 4.1666$ slots, resulting in a total system load $\rho = 0.8333$. This gives rise to a QBD Markov chain, characterized by the transition matrix $\bar{P}$ (as in Equation (7.1)), as described in Section 2.2.3. Note that for this example the variable $h = 6$. Since we are interested in the system state at some time t, we mark every time instant, meaning $\bar{A}_i^m = \bar{A}_i$ and $\bar{B}_i^m = \bar{B}_i$, for $i = 0, 1, 2$. Notice, in this example, level 0 has fewer states than level $r > 0$, demonstrating that our algorithm is easily generalized to such cases.



(a) $t = 500$, $\alpha = 1$ (b) $t = 5000$, $\alpha = 1$

Figure 7.2: Queue length distributions at different time epochs

Take $\alpha = 1$, then Figure 7.1(a) shows the queue length distribution for respectively $0, 5, 10, 20, 30$ and 40 customers in the system at time 0. If the system is initially nonempty, it is assumed that the first customer starts his service at time 0. Thus, for $r > 0$, all curves plotted for $\alpha = 1$ or 2 are actually the weighted sum of three distributions, where the initial service phase equals 1, 2 and 3 and where the weights are given by β.

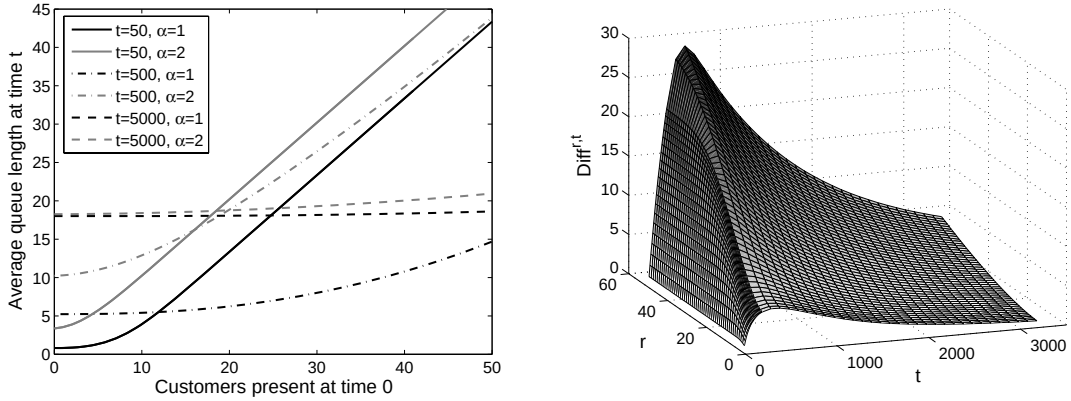
We observe a large influence of the number of customers originally present on the queue length distribution. As could be expected this influence becomes smaller for larger values of t as can be observed from Figures 7.2(a) and 7.2(b), for $t = 500$, resp. $t = 5000$, and $k = 100$. Remark that after choosing the values for t and k, we need to execute our recursive algorithm only once to obtain the results for every possible combination of r and α.

Figure 7.1(b) shows, for $t = 50$, the queue length distributions for the same number of customers in the system at time 0, but with $\alpha = 2$. Since the arrival rate in state 2 of the D-MAP is larger than the arrival rate in state 1, we observe higher probabilities for the larger queue lengths in this example. From Figure 7.3(a) it can be observed that for a small value of t the average queue length increases almost linearly with the number of customers present in the queueing system at time 0. In addition, the influence of the number of customers initially present decreases when the time t at which we evaluate the queue length increases.

A different behavior can be observed with relation to the initial state of D-MAP arrival process. Denote by $q_1^{r,t}$, resp. $q_2^{r,t}$, the average queue length at time t, given that r customers were present at time 0 and the initial state of the D-MAP process equals state 1, resp. state 2. Next, let us define

$$\mathit{Diff}^{r,t} = q_2^{r,t} - q_1^{r,t}. \tag{7.45}$$

From Figure 7.3(a) we notice for example that $\mathit{Diff}^{r,50} < \mathit{Diff}^{r,500}$, while $\mathit{Diff}^{r,500} > \mathit{Diff}^{r,5000}$, for $1 \leq r \leq 50$. A better insight in the behavior of $\mathit{Diff}^{r,t}$ for different values of r and t can be obtained by observing Figure 7.3(b). For $1 \leq r \leq 50$ we can see that the difference between $q_2^{r,t}$ and $q_1^{r,t}$ increases for small values of t and decreases later on. This effect can be explained by looking at the arrival rates in both states of the D-MAP process. Recall, while in state 1, resp. state 2, there is an arrival in an arbitrary time slot with probability 0.1, resp. 0.25.



(a) Average queue lengths $q_i^{r,t}$ $(i = 1, 2)$ (b) $\mathit{Diff}^{r,t} = q_2^{r,t} - q_1^{r,t}$

Figure 7.3: Average queue length of the D-MAP/PH/1 queue

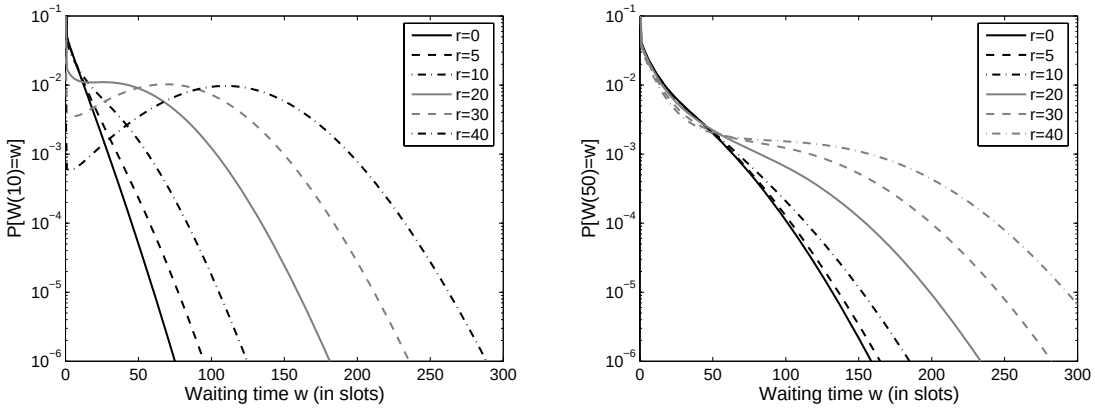
For values of t close to zero, the difference $\mathit{Diff}^{r,t}$ will be relatively small since the average number of customers present at time t, is mainly influenced by r, the number of customers initially present. As long as there is no transition of the D-MAP arrival process, the difference between the average queue lengths $q_2^{r,t}$ and $q_1^{r,t}$ increases with t, due to the different arrival rates in the corresponding states. After a state transition of the D-MAP process, $\mathit{Diff}^{r,t}$ will start to decrease and eventually, this difference will diminish to zero when t approaches ∞ as the system has a steady state for $\rho = 0.8333$.

In Figure 7.4 the time needed to compute the queue length distribution for every initial configuration is compared with the execution time of the original algorithm [110]. The measurements were made using the Matlab Profiler on an Intel Pentium M 1.70GHz processor with 1GB of memory. We computed the queue length distribution at time $t = 50$, 500 and 5000, with a predefined upper bound of $r = 50$ on the initial level. For the new algorithm, the time needed to compute the R-matrices is presented as well as the total execution time. For the original algorithm we show the time needed to compute the queue length distributions for all initial configurations, for every state $1 \leq l \leq 6$ of level 50 and for a single initial configuration $\langle 50, l \rangle$.

time	new algorithm		original algorithm		
t	R's	total	all states	level 50	$\langle 50, l \rangle$
50	0.46	9	2310	121	20
500	1.94	53	31595	1052	175
5000	1.95	133	33499	1080	180

Figure 7.4: Execution times (s) to get the transient queue length distribution for $r \leq 50$

As before, we set $k = t-1$ for $t = 50$, for both $t = 500$ and $t = 5000$, k is set to 100. In the first scenario where $t = 50$, the algorithm stops after the initial level reached $r = 39$ because $\pi_{0,1}^{k,n}\{r, l\}$ becomes negligible for larger values of r. For $t > 50$ the predefined maximum of $r = 50$ was reached. The same stopping criterion was used by the original algorithm.



(a) $n = 10$ (b) $n = 50$

Figure 7.5: Waiting time distribution of the n-th customer in a D-MAP/PH/1 queue

These results show that the computation time is reduced considerably by the new algorithm. The computation time of the level based recursion to compute the queue length distribution for every possible initial setting is even below the time the original algorithm needs for a single initial state. With the new algorithm, the dominant part of the computation lies in obtaining the $\hat{\pi}\{r, l\}$ vectors. This also explains the smaller difference between the total computation time of the new and the single state computation time of the old algorithm in the $t = 5000$ result, since a larger value of u is required such that

$$1 - \sum_{i=0}^{u} \hat{\pi}_i\{r, l\} < \epsilon.$$

Of course, if we were interested in just one single initial state $\langle 50, l\rangle$, the time consuming iteration for the $\hat{\pi}\{r, l\}$ vectors is performed only once (as opposed to the 302 times for all states up to level $r = 50$).

In the next example, we consider the same D-MAP/PH/1 queue, however we are now interested in the waiting time distribution of the n-th customer, which can also be obtained using the presented algorithm. To compute this waiting time it suffices to know the system state at the n-th arrival epoch. Therefore, we mark only those time epochs at which an arrival occurs (instead of marking all time epochs as needed to obtain the queue length distribution at some time t). We have

$$\begin{array}{lcllcl} \bar{A}_0^m & = & \bar{A}_0, & \bar{B}_0^m & = & \bar{B}_0, \\ \bar{A}_1^m & = & D_1 \otimes T^* \beta, & \bar{B}_1^m & = & 0, \\ \bar{A}_2^m & = & 0, & \bar{B}_2^m & = & 0. \end{array}$$

Executing our algorithm with these matrices gives us an approximation of the system state at the n-th arrival. From this system state we can obtain the waiting time distribution of the n-th customer by considering the number of customers present at this n-th arrival, together with the sum of their (residual) service times. More precisely, from the queue length distribution X_n at the n-th arrival, we can deduce the waiting time distribution of the n-th

customer as follows:

$$P[W_n = 0] = \sum_j (X_n)_{\langle 0,j \rangle} + \sum_{s,j} (X_n)_{\langle 1,j,s \rangle} t_s, \tag{7.46}$$

$$P[W_n = w] = \sum_{q \geq 1} \sum_{s,j} (X_n)_{\langle q,j,s \rangle} P[S^{(q-1)*} + R(s) = w]. \tag{7.47}$$

In the superscript $\langle i, j, s \rangle$ of X_n, i denotes the number of customers, j the state of the arrival process and s the phase of the service process. That is, the waiting time of the arriving customer equals zero time slots if there are no customers present or if the single customer that is present completes his service. Similar, we can obtain equation (7.47), where $S^{(q-1)*}$ denotes the $(q-1)$-fold convolution of the service time distribution S and $R(s)$ denotes the residual service time, provided that the phase of the service is s, i.e., $P[R(s) = r] = (T^r t)_s$. More details about the calculation of this waiting time distribution can be found in [110, Section 4].

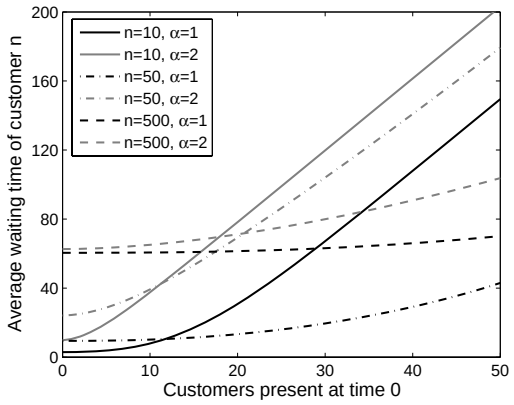


Figure 7.6: Average waiting time of the n-th customer in a D-MAP/PH/1 queue

Figure 7.5(a) shows the waiting time distribution of the 10-th customer with $\alpha = 1$ for different values of r. If we compare these results with the waiting time distributions of the 50-th customer, presented in Figure 7.5(b), we observe that r has a larger influence in the first case, as one could expect. Also, for the average waiting times, presented in Figure 7.6, we can draw similar conclusions as for the average queue length. That is, as n increases, the influence of the number of customers initially present on the average waiting time of the n-th customer decreases. Moreover, for small n, the difference that is caused by the initial D-MAP state increases together with n and it decreases hereafter.

8

Tree-Like Processes

In Section 2.1.3 we presented the notion of a tree-like process as a subclass of the tree-structured process. The theory of tree-like processes generalizes the well-known paradigm of Quasi-Birth-Death Markov chains and has various applications. These processes allow us to build analytical models for systems where the state space of the accompanying stochastic process can be represented as a tree. Typical applications of tree-like processes are queueing systems with a last-in-first-out scheduling discipline and random access systems with an underlying stack structure. Chapter 5 also discusses how tree-like processes can be used to study priority queues.

In this chapter, we discuss how the construction of a framework to perform a transient analysis of a QBD Markov chain can be generalized such that it applies to tree-like processes. We introduce the concept of tree-like processes with marked time epochs and show how the transient problem can be transformed into a steady state analysis, analogous to the line of reasoning put forward in Chapter 7. The algorithm we propose is not limited to a specific system, it provides a general methodology to deal with any application that can be modeled as a tree-like process. Since in [107] it was shown that the classes of tree-like processes and tree-structured QBD MCs are equivalent, this last class is, in principle, also covered by our method.

To show the generality of our work, at the end of this chapter, we present an application somewhat outside the queueing theory domain. We demonstrate our approach on the Capetanakis-Tsybakov-Mikhailov (CTM) random access protocol yielding new insights on its initial behavior both in normal and overload conditions. Random access protocols are designed to control the transmission of packets on a link shared among multiple users. When two or more users transmit a packet at the same time, a collision occurs, causing all packets to get lost. Consequently, these packets have to be sent again. If all users would send their packet again immediately after they found out about the collision, a new collision will occur. Random access mechanisms try to solve this by introducing specific rules describing which users are allowed to send at what time, comparable to a scheduling discipline in queueing theory. The CTM protocol does this by recursively splitting the users involved in the collision into two groups, one that can send again in the next time slot and one that has to wait.

8.1 Related Work

The set of tree-like processes [15] was originally introduced as a subclass of the tree-structured QBD Markov chains. Meanwhile, it has been shown that any tree-structured QBD can be reduced to a tree-like process [107]. Typical applications of tree-like processes include single server queues with a LIFO service discipline [118, 103, 43, 42]. In [106, 112, 108] tree-like processes were used to analyze a set of well-known random access algorithms, called tree or splitting algorithms. In [113] a tree-like process was used to model a discrete-time priority queueing system with three priority classes. In Chapter 5 this idea was generalized to continuous-time queueing systems with an arbitrary number of priority classes. Various numerical methods exist to examine the stability of a tree-like process and to determine its stationary behavior, e.g., [117, 15]. In this chapter, however, we focus on the transient behavior of a tree-like process.

After developing a methodology to perform a transient analysis of a tree-like process, we demonstrate our approach by analyzing the transient behavior of the celebrated binary tree algorithm by Capetanakis, Tsybakov and Mikhailov (CTM) [23, 105, 31, 33] for random access systems. The key characteristic of the CTM algorithm is its underlying tree (or stack) structure. An abundance of papers has been devoted to evaluate its performance (see [75] for an extensive overview), to the best of our knowledge very few results have been devoted to its transient behavior. Tree algorithms have received considerable attention over the last three decades and are still analyzed within the context of various networking areas. The use of tree algorithms in cable-TV networks (i.e., Hybrid-Fiber Coaxial networks) has been proposed by various authors including [52, 51, 20], where the focus was either on the frame structure, large round-trip times or the finite population. A lot of this work was motivated by the activities of the IEEE 802.14 working group on tree algorithms [36]. However, the 802.14 standardization process was prematurely terminated by the introduction of the DOCSIS standard, which does not rely on these tree algorithms. Tree algorithms as a means to solve collision resolution problems have been addressed in the context of Ad-Hoc networks [92], where the residual battery energy affects the splitting process, and W-CDMA systems [3] with a priority random-access protocol. A unified probabilistic treatment to assess the asymptotic behavior of tree algorithms has also appeared more recently [88, 74], avoiding the need to resort to complex analysis techniques.

8.2 Introduction

In this chapter we focus on the transient behavior of tree-like processes. The contribution is both of theoretical and practical nature. In Section 8.3.1 we introduce a new framework called *tree-like processes with marked time epochs* to assess transient performance measures in a unified manner. Similar to the QBD with marked time epochs, such a process distinguishes itself from the common tree-like process in the sense that as it evolves, it also marks part of the time epochs. Any transient problem that can be linked to the n-th marking, for some n, of some tree-like process with marked time epochs, can be dealt with using our methodology.

Second, to analyze this new process (that is, to generate numerical results) we develop an efficient algorithm in Section 8.3.2 that uses a discrete Erlangization to approximate the system state at the n-th marking. The main building block of this algorithm is a reset Markov chain that allows us to reformulate the transient problem at hand into a stationary problem

of an expanded Markov chain. When performing this stationary analysis we further exploit the btbT structure (see Page 132 for a definition) of the matrices involved. As mentioned in Section 2.1.3, a key role in obtaining the stationary behavior of a tree-like process, including the reset MC, is played by a matrix V, for which several algorithms are presented in Section 8.3.3. After having obtained V, we demonstrate how to derive the transient measures of interest from this matrix (Section 8.3.4).

To illustrate our algorithm we will study the Capetanakis, Tsybakov and Mikhailov (CTM) algorithm for random access in Section 8.4. By considering the transient regime of tree algorithms we can get an understanding of the initial protocol dynamics even when the system is overloaded (that is, the overall input rate λ is above the maximum stable throughput). We apply our framework to assess a number of transient performance measures associated with the CTM protocol. To the best of our knowledge, it is the first time that analytical results of this type are provided for the CTM protocol. The link between the CTM algorithm and tree-like processes necessary to apply our new methodology was established in [106]. The transient behavior of other protocol variations can be investigated in a similar manner by relying on the results presented in [112].

Earlier in Part III of this thesis, we mentioned that transient measures for discrete-time Markov chains are often obtained through an iterative or recursive approach, e.g., one starts with the initial probability vector and subsequently multiplies this vector with the transition matrix P (while exploiting its structure if possible). For QBD Markov chains, we noted that such an approach can be fruitful when analyzing the system over short time scales, but that the complexity rapidly increases with the time epoch of interest. When studying tree-like processes this problem is even more important. Even if the event of interest occurs rather early in time, an iterative or recursive approach causes extra difficulties as the number of states that can be visited by the chain at time t grows exponentially fast. Problems like this are circumvented by our approach as we perform a single steady state analysis of an expanded Markov chain.

8.3 Tree-Like Process with Marked Time Epochs

Recall from Section 2.1.3 that a tree-like processes is discrete-time bivariate MC $\{(X_t, N_t), t \geq 0\}$ in which the values of X_t are represented by nodes of a $(d+1)$-ary tree, for $d \geq 0$, and where N_t takes integer values between 1 and h. Here, X_t is referred to as the node and N_t as the auxiliary variable of the MC at time t. Notice that, in this chapter, we will assume that the nodes of X_t form a $(d+1)$-ary tree instead of a d-ary tree as used for the definition in Section 2.1.3. This way, we can include zero in the set of integers used to represent the nodes of the tree. This choice has of course no influence on our analysis, nevertheless, it allows us to slightly simplify the notation.

8.3.1 Process Definition

In order to develop a novel framework to derive transient performance measures in a unified manner, we introduce the tree-like process with marked time epochs. As such a tree-like process evolves, each time epoch is either marked or not, where the marking rules typically depend on the performance measure of interest. For examples about which time epochs should be marked in a specific setting, we refer to Sections 7.3 and 8.4. In Section 8.3.3 and 8.3.4 we will present a number of algorithms that allow us to compute (i) the system state at the

n-th marked time epoch and (ii) the average time epoch at which the n-th marking occurs. Although this was not stated explicitly, the latter performance measure can also be obtained for QBDs with marked time epochs.

Consider an MC $\{(X_t, N_t, M_t), t \geq 0\}$, where the values of X_t are nodes of a $(d+1)$-ary tree, N_t takes values that range from 1 to h and $M_t = m$ or u. $\{(X_t, N_t, M_t), t \geq 0\}$ is a tree-like process with marked time epochs if the transition probabilities are of the following form, for $y = m$ and u:

$$P[(X_{t+1}, N_{t+1}, M_{t+1}) = (J', j, y) | (X_t, N_t) = (J, i)] = \begin{cases} \bar{f}^{i,j,y} & J' = J = \emptyset, \\ \bar{b}^{i,j,y} & J' = J \neq \emptyset, \\ \bar{d}_k^{i,j,y} & J \neq \emptyset, f(J,1) = k, J' = J - f(J,1), \\ \bar{u}_s^{i,j,y} & J' = J + s, s = 0, \ldots, d, \\ 0 & otherwise. \end{cases}$$

Notice, the value of M_{t+1} is affected by (X_t, N_t) and $(X_{t+1},\ N_{t+1})$, but not by M_t (given (X_t, N_t)); hence, the value of M_0 is irrelevant. Next, define the $h \times h$ matrices $\bar{D}_k^y$, $\bar{B}^y$, $\bar{F}^y$ and $\bar{U}_s^y$, for $y = m$ and u, in the obvious manner. Furthermore, let

$$\begin{aligned} \bar{D}_k &= \bar{D}_k^m + \bar{D}_k^u, \\ \bar{B} &= \bar{B}^m + \bar{B}^u, \\ \bar{U}_s &= \bar{U}_s^m + \bar{U}_s^u, \\ \bar{F} &= \bar{F}^m + \bar{F}^u, \end{aligned}$$

for $0 \leq k, s \leq d$. Remark that $\{(X_t, N_t), t \geq 0\}$ is a tree-like process characterized by the matrices $\bar{D}_k$, $\bar{B}$, $\bar{F}$ and $\bar{U}_s$. We state that the t-th time epoch of this tree-like process is marked if and only if $M_{t+1} = m$. Therefore, the nonnegative matrix $\bar{D}_k^m$ contains the probabilities that $(X_t, N_t) = (J + k, i)$, while $(X_{t+1}, N_{t+1}) = (J, j)$ and time epoch t gets marked. $\bar{D}_k^u$ contains the probabilities for the same event, but without marking time t. A similar interpretation can be given for the other matrices. Whether a time epoch t is marked therefore depends on the transition from time t to time $t+1$. We refer to the initial time epoch as time $t = 0$ and when performing the transient analysis we limit ourselves to the case where the tree-like process is in the root node at time $t = 0$, that is, the probability that we start in state $(\emptyset, i)$ is given by the i-th entry of the stochastic vector α_{ini}.

Apart from the introduction of this new framework, one of the main contributions in this chapter lies in the development of efficient algorithms to compute the system state, either in an exact or approximated manner, of the MC $\{(X_t, N_t, M_t), t \geq 0\}$ at the n-th marked time epoch. As a byproduct we will also derive the average time at which this n-th marking occurs, which is often an important performance measure on its own. For this purpose, we will generalize the concepts of a discrete Erlangization and a reset Markov chain to the class of tree-like processes.

8.3.2 Discrete Erlangization and Reset Markov Chains

The idea of applying a discrete Erlangization and constructing a reset Markov chain was discussed in Chapter 7, where the transient behavior of a QBD MC was analyzed. Nevertheless, to increase the readability of this chapter on its own, we will include a general discussion of these concepts, not purely focusing on the differences of the implementation in the context of

tree-like processes. Denote the i-th entry of the $1 \times h$ vector $\pi_n^m(J)$ as the probability that the Markov chain $\{(X_t, N_t), t \geq 0\}$ is in state (J, i) at the n-th marked time epoch. The key property of a discrete Erlangization is that we can approximate the system state at the n-th marking $t^m(n)$ by considering the system state at the $Z_{l,n}$-th marked time epoch $t^m(Z_{l,n})$, where $Z_{l,n}$ is a negative binomially distributed (NBD) random variable with l phases and a mean n, for $l(\leq n)$ sufficiently large. Recall, setting $l = n$ would result in exact results, however, l cannot always be set to n as the reset MC might become periodic. Using the NBD as a reset time implies, among others, that the transition blocks of the reset MC presented below have a special structure that can be exploited when computing the MC's steady state probabilities. Remark, in this chapter we will use the variable l to refer to the number of phases of the NBD random variable instead of the classical notation k, since k is typically used as an index for the different matrices when referring to the k-th child of a certain node.

The transient problem of computing the system state at time $t^m(Z_{l,n})$ can be reformulated into a steady state analysis by constructing a reset Markov chain. Consider the stochastic process that evolves according to $\{(X_t, N_t, M_t), t \geq 0\}$, but that is repeatedly reset when leaving the $Z_{l,n}$-th marked time epoch (meaning a transition occurs to state $(X_0 = \emptyset, i, m)$ with probability $(\alpha_{ini})_i$). Hence, if we perform a Bernoulli trial with parameter $p = l/n$ each time we have a transition out of a marked time epoch, the system is reset whenever l successes have occurred.

We define the *reset counter* as being the number of pending successes before the next reset event. It is clear that this reset counter takes values in the range $\{1, 2, \ldots, l\}$. When we add this reset counter as an additional auxiliary variable to the tree-like process $\{(X_t, N_t), t \geq 0\}$, we obtain a Markov chain $\{(\mathcal{X}_t, \mathcal{N}_t), t \geq 0\}$, where $\mathcal{N}_t$ has the set $\{(i, j) | 1 \leq i \leq l, 1 \leq j \leq h\}$ as its range. Remark, if we succeed in computing the stationary behavior of the expanded MC $(\mathcal{X}_t, \mathcal{N}_t)$, we can easily find the system state just prior to a reset event, which is exactly the system state at time $t^m(Z_{l,n})$. The transition probabilities of the reset MC can be written as

$$P[(\mathcal{X}_{t+1}, \mathcal{N}_{t+1}) = (J', (i', j')) | (\mathcal{X}_t, \mathcal{N}_t) = (J, (i, j))] =$$
$$\begin{cases} f^{i,j,i',j'} & J' = J = \emptyset, \\ b^{i,j,i',j'} & J' = J \neq \emptyset, \\ d_k^{i,j,i',j'} & |J| > 1, f(J,1) = k, J' = J - f(J,1), \\ c_k^{i,j,i',j'} & |J| > 1, f(J,1) = k, J' = \emptyset, \\ d_k^{i,j,i',j'} + c_k^{i,j,i',j'} & J = k, J' = \emptyset, \\ u_s^{i,j,i',j'} & J' = J + s, s = 0, \ldots, d, \\ 0 & otherwise, \end{cases}$$

where $|J|$ represents the length of the string J. Let $d_k^{i,j,i',j'}$, $c_k^{i,j,i',j'}$, $b^{i,j,i',j'}$, $f^{i,j,i',j'}$ and $u_s^{i,j,i',j'}$ be the $((i-1)h + j, (i'-1)h + j')$-th entry of the $lh \times lh$ matrix D_k, C_k, B, F and U_s, respectively. One readily establishes that these matrices are given by:

$$\begin{aligned} D_k &= (I \otimes (\bar{D}_k^u + (1-p)\bar{D}_k^m)) + (I^- \otimes p\bar{D}_k^m), \\ B &= (I \otimes (\bar{B}^u + (1-p)\bar{B}^m)) + (I^- \otimes p\bar{B}^m), \\ U_s &= (I \otimes (\bar{U}_s^u + (1-p)\bar{U}_s^m)) + (I^- \otimes p\bar{U}_s^m), \\ C_k &= M_1^l \otimes p\bar{T}_k^m e\alpha_{ini}, \end{aligned} \tag{8.1}$$

where

$$\bar{T}_k^m = \bar{D}_k^m + \bar{B}^m + \sum_{s=0}^{d} \bar{U}_s^m. \tag{8.2}$$

Recall from Chapter 7, the $l \times l$ matrix M_1^l has only one entry different from zero, being the last entry on the first row, which equals one. The matrix F that governs the transitions from the root node to itself can be written as

$$F = (I \otimes (\bar{F}^u + (1-p)\bar{F}^m)) + (I^- \otimes p\bar{F}^m) + \left(M_1^l \otimes p \left(\bar{F}^m e + \sum_{s=0}^{d} \bar{U}_s^m e \right) \alpha_{ini} \right). \tag{8.3}$$

Consider the reset MC $\{(\mathcal{X}_t, \mathcal{N}_t), t \geq 0\}$ and define its transition matrix $P^{l,n}$ of infinite (countable) size such that the strings are ordered lexicographically – that is, $J < J'$ if either $|J| < |J'|$ or there exists an $k > 0$ such that the $J - f(J,k) = J' - f(J',k)$ and the $(|J|-k+1)$-th entry of J is smaller than that of J'. Except for the first balance equation, the block matrix $P^{l,n}$ has the same structure as that of any tree-like process (see [15]), therefore, the components $\pi^{l,n}(J)$, defined as

$$\begin{aligned}
\pi_{(i,j)}^{l,n}(J) &= \lim_{t\to\infty} P[\mathcal{X}_t = J, \mathcal{N}_t = (i,j)], \\
\pi_i^{l,n}(J) &= (\pi_{(i,1)}^{l,n}(J), \pi_{(i,2)}^{l,n}(J), \ldots, \pi_{(i,h)}^{l,n}(J)), \\
\pi^{l,n}(J) &= (\pi_1^{l,n}(J), \pi_2^{l,n}(J), \ldots, \pi_l^{l,n}(J)),
\end{aligned}$$

of its stationary probability vector have the same form as those of a tree-like process, i.e.,

$$\pi^{l,n}(J+k) = \pi^{l,n}(J) R_k, \tag{8.4}$$

except for $\pi^{l,n}(\emptyset)$ (see Section 2.1.3). In this equation $R_k = U_k(I-V)^{-1}$, where the $lh \times lh$ matrix V is the smallest nonnegative solution to

$$V = B + \sum_{s=1}^{d} U_s (I-V)^{-1} D_s. \tag{8.5}$$

Hence, we can easily compute the necessary components of

$$\pi^{l,n} = (\pi^{l,n}(\emptyset), \pi^{l,n}(1), \pi^{l,n}(2), \ldots)$$

once V, and the vector $\pi^{l,n}(\emptyset)$ are obtained. As lh can be considerably large, we will speed up their computation by exploiting the structure of the D_k, B, U_s and C_k matrices. Three such algorithms to determine V are presented in the next section. A sufficient condition for the existence of $\pi^{l,n}$ is that the average reset time of $\{(\mathcal{X}_t, \mathcal{N}_t), t \geq 0\}$ is finite, irrespective of whether the MC $\{(X_t, N_t), t \geq 0\}$ is ergodic. The finiteness of the average reset time is often obvious from an operational point of view. If not, it can be verified rigorously by checking whether the special radius of $R = R_0 + \ldots + R_d$ is less than one (the G_k matrices of $\{(\mathcal{X}_t, \mathcal{N}_t), t \geq 0\}$ need not to be stochastic due to the reset events).

8.3.3 Computing the V-Matrix

8.3.3.1 A First Approach

The $lh \times lh$ matrix V is the smallest nonnegative solution to Eqn. (8.5) and can be computed by a standard fixed point iteration [15]. That is, the matrix V is obtained as $\lim_{N\to\infty} V[N]$ from the recursion

$$V[N+1] = B + \sum_{s=0}^{d} U_s (I - V[N])^{-1} D_s, \tag{8.6}$$

where

$$V[0] = B.$$

We can however benefit from the structural properties of the B, U_s and D_s matrices involved to make this recursion faster. The matrices B, U_s and D_s, for $0 \le s \le d$, are all lower block bidiagonal matrices (see Eqn. (8.1)). Let us denote the blocks on the main diagonal of a lower block bidiagonal matrix X as $X^{(0)}$ and the blocks below the main diagonal as $X^{(1)}$. Hence,

$$\begin{array}{rclcrcl} B^{(0)} &=& \bar{B}^u + (1-p)\bar{B}^m, & & B^{(1)} &=& p\bar{B}^m, \\ U_s^{(0)} &=& \bar{U}_s^u + (1-p)\bar{U}_s^m, & & U_s^{(1)} &=& p\bar{U}_s^m, \\ D_s^{(0)} &=& \bar{D}_s^u + (1-p)\bar{D}_s^m, & & D_s^{(1)} &=& p\bar{D}_s^m. \end{array}$$

The lower block bidiagonal structure implies that V has a btbT structure. Since, in this section a lot of matrices with subscripts have been (and will be) used, we will represent the blocks of a btbT matrix Y as given by

$$Y = \begin{bmatrix} Y^{(0)} & 0 & 0 & \cdots & 0 \\ Y^{(1)} & Y^{(0)} & 0 & \cdots & 0 \\ Y^{(2)} & Y^{(1)} & Y^{(0)} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ Y^{(l-1)} & Y^{(l-2)} & \dots & Y^{(1)} & Y^{(0)} \end{bmatrix}.$$

As a consequence, it suffices to determine the first block column of V in order to know the entire matrix. Notice also that the btbT structure is preserved by every operation used in Eqn. (8.6), meaning all $V[N], G_k$ and R_k matrices are also btbT. Making use of Eqn. (2.6–2.9), we can calculate matrix $V[N+1]$ from matrix $V[N]$ using block column vectors only as presented in Algorithm 8.1. Hence, the amount of memory needed by this algorithm to compute the matrix V is only linear in l as opposed to square.

8.3.3.2 Recursive Computation of the Blocks in V

In each iteration step of the previous algorithm, we computed the V-matrix entirely, i.e., we calculated its entire first block column as this fully determines V. A second approach is to compute the blocks $V^{(0)}, \dots, V^{(l-1)}$ that characterize the btbT matrix V recursively as discussed in Algorithm 8.2. In this algorithm we start with the computation of $V^{(0)}$ and afterward $V^{(i)}$ can be deduced from $V^{(0)}, \dots, V^{(i-1)}$ using one of the two iterative approaches presented in step 4 of Algorithm 8.2.

Algorithm 8.1 Computing the matrix $V[N+1]$ from $V[N]$

1. Define the btbT matrix

$$W[N] = (I - V[N])^{-1},$$

which can be calculated in a the time and memory complexity of $O(l^2h^3)$ and $O(lh^2)$, respectively. The time complexity can be further reduced to $O(lh^3 + l\log(l)h^2)$ by making use of fast Fourier transforms (see [71, Chapter 2]). The block $W^{(i)}[N]$ can be expressed in terms of the blocks $V^{(k)}[N]$, $0 \leq k \leq i$, and the blocks $W^{(k)}[N]$, $0 \leq k < i$, as follows:

$$W^{(0)}[N] = (I - V^{(0)}[N])^{-1}, \tag{8.7}$$

$$W^{(k)}[N] = \sum_{i=1}^{k} (W^{(0)}[N])(V^{(i)}[N])(W^{(k-i)}[N]), \text{ for } k > 0. \tag{8.8}$$

2. Using this definition, set

$$R_k^{(i)}[N+1] = \begin{cases} U_k^{(0)} W^{(0)}[N] & i = 0, \\ U_k^{(0)} W^{(i)}[N] + U_k^{(1)} W^{(i-1)}[N] & i > 0. \end{cases}$$

3. Now, compute $V[N+1]$ as

$$V^{(i)}[N+1] = \begin{cases} B^{(0)} + \sum_{k=0}^{d} R_k^{(0)}[N+1] D_k^{(0)} & i = 0, \\ \sum_{k=0}^{d} (R_k^{(i)}[N+1] D_k^{(0)} + R_k^{(i-1)}[N+1] D_k^{(1)}) + 1_{\{i=1\}} B^{(1)} & i > 0. \end{cases}$$

The second iterative procedure requires the solution to a Sylvester matrix equation of the form $X + AXB = C$, for which we can use Algorithm 7.2. This algorithm has a time complexity of $O(h^3)$. It is easy to implement as for instance each of the decompositions required is a Matlab built-in function. Moreover, the '\' Matlab command to solve the h linear systems recognizes the Hessenberg structure and solves each of the systems in $O(h^2)$ time. Notice from Eqn. (8.14) that we only need to perform the Hessenberg and the Schur decomposition once because $A = U_0^{(0)} W^{(0)}$ and $B = W^{(0)} D_0^{(0)}$ are both independent of i and N.

8.3.3.3 Comparison of the Two Approaches

When we compare the two iterative approaches presented in step 4 of Algorithm 8.2, the Sylvester scheme will typically result in fewer iterations, however, more time is needed to perform a single iteration, making the first approach often the fastest. Compared to the algorithm presented in Section 8.3.3.1 both schemes discussed in 8.3.3.2 tend to be significantly faster. However, as reported in [15], the convergence of (8.13) and (8.14) is not guaranteed, as opposed to the convergence of Algorithm 8.1. For the application discussed in Section 8.4 we experienced no convergence problems for both (8.13) and (8.14).

Algorithm 8.2 Computing the blocks of the V-matrix recursively

1. If we consider only the first block row of Eqn. (2.9), we get

$$V^{(0)} = B^{(0)} + \sum_{s=0}^{d} U_s^{(0)} (I - V^{(0)})^{-1} D_s^{(0)}. \tag{8.9}$$

   As $V^{(0)}$ has size $h \times h$ only, we can rely on the standard fixed point iteration presented in Eqn. (8.6) for its computation.

2. Define the btbT matrix W as $W = (I - V)^{-1}$. Recall from step 1 of Algorithm 8.1 that $W^{(i)}$ can be expressed in terms of $V^{(k)}$, $0 \leq k \leq i$, and $W^{(k)}$, $0 \leq k < i$. Then, compute

$$\begin{aligned} K_i \;=\; 1_{\{i=1\}} B^{(1)} \;+\; & \sum_{s=0}^{d} \left[U_s^{(1)} W^{(i-1)} D_s^{(0)} + U_s^{(0)} W^{(0)} \left(\sum_{k=1}^{i-1} V^{(k)} W^{(i-k)} \right) D_s^{(0)} \right. \\ & \left. + \; \left(U_s^{(1)} W^{(i-2)} 1_{\{i>1\}} + U_s^{(0)} W^{(i-1)} \right) D_s^{(1)} \right]. \end{aligned} \tag{8.10}$$

   Notice, having computed $V^{(0)}, \ldots, V^{(i-1)}$ allows us to compute K_i as these suffice to compute the $W^{(k)}$ matrices, for $k < i$, that characterize the btbT matrix $(I - V)^{-1}$.

3. Due to (2.9) we can establish the following equation for $V^{(i)}$, for $i > 0$:

$$V^{(i)} \;=\; K_i + \sum_{s=0}^{d} U_s^{(0)} W^{(0)} V^{(i)} W^{(0)} D_s^{(0)}. \tag{8.11}$$

4. Two iterative approaches [15] can be followed to retrieve $V^{(i)}$ from Eqn. (8.11). Both start by setting

$$V^{(i)}[0] = K_i. \tag{8.12}$$

   (a) A first iteration consists in generating the sequence $V^{(i)}[N]$ as

$$V^{(i)}[N+1] = K_i + \sum_{s=0}^{d} U_s^{(0)} W^{(0)} V^{(i)}[N] W^{(0)} D_s^{(0)}. \tag{8.13}$$

   (b) The second one generates a sequence by solving the Sylvester matrix equation

$$V^{(i)}[N+1] - U_0^{(0)} W^{(0)} V^{(i)}[N+1] W^{(0)} D_0^{(0)} = K_i + \sum_{s=1}^{d} U_s^{(0)} W^{(0)} V^{(i)}[N] W^{(0)} D_s^{(0)}. \tag{8.14}$$

8.3.4 Calculating the Probability Vectors $\pi_n^m(J)$

Having obtained the $lh \times lh$ matrix V, we are now in a position to compute the stationary measure $\pi^{l,n}$ of the reset MC $(\mathcal{X}_t, \mathcal{N}_t)$. Consider the Markov chain obtained by censoring $(\mathcal{X}_t, \mathcal{N}_t)$ on the states of the root node $(\emptyset, (i,j))$. Its transition matrix $P_\emptyset^{l,n}$ can be written as

$$P_\emptyset^{l,n} = (F + V - B) + (e - (F + V - B)e)(e^{(l)} \otimes \alpha_{ini}). \tag{8.15}$$

The matrix F covers the single step transitions, whereas $V - B$ holds all the paths from the root node to itself that start with a transition to one of its child nodes without the occurrence of a reset event as Eqn. (2.8) implies that

$$\sum_{s=0}^{d} U_s (I - V)^{-1} D_s = V - B.$$

The return state (i,j) of all the remaining paths from $\emptyset$ to itself is of the form (l,j) and is determined by α_{ini} as the final transition of these paths is a reset event.

The vector $\pi^{l,n}(\emptyset)$ corresponding to the root node $\emptyset$, is the invariant vector of $P_\emptyset^{l,n}$, normalized as

$$\pi^{l,n}(\emptyset)(I - R)^{-1} e = 1.$$

The matrix $P_\emptyset^{l,n}$ has a special structure, which we can exploit when computing $\pi^{l,n}(\emptyset)$. More precisely, it is the sum of a btbT matrix $(F + V - B)$ and an lbC matrix, where an lbC matrix is a matrix with its nonzero entries all positioned in the last block column. In order to compute $\pi^{l,n}(\emptyset)$ efficiently, we can therefore rely on Algorithm 7.3, the time and space complexity of which equal $O(h^3 l^2)$ and $O(h^2 l)$, respectively. The remaining components of the stationary measure $\pi^{l,n}$ for the nodes of the form $J + k$ are given by

$$\pi^{l,n}(J + k) = \pi^{l,n}(J) R_k,$$

where $R_k = U_k (I - V)^{-1}$ (with $0 \leq k \leq d$), as expressed in Eqn. (2.7). Fast Fourier transforms can be used to speed up the computation of the components $\pi^{l,n}(J + k)$, because of the btbT structure of R_k (see [111, Section 5] for details).

We now indicate how the probability vector $\pi_{Z_{l,n}}^m(J)$, used to approximate the system state $\pi_n^m(J)$ at the n-th marked time epoch, can be obtained from $\pi^{l,n}(J)$. The j-th entry of the vector $\pi_{Z_{l,n}}^m(J)$ holds the probability that the MC $(\mathcal{X}_t, \mathcal{N}_t)$ is in state $(J, (1,j))$ provided that a reset event occurs, hence,

$$\begin{aligned} \pi_{Z_{l,n}}^m(\emptyset) &= \frac{(\pi_1^{l,n}(\emptyset).\phi_0)p}{c}, \\ \pi_{Z_{l,n}}^m(J + k) &= \frac{(\pi_1^{l,n}(J + k).\phi_{1,k})p}{c}. \end{aligned}$$

Here, ϕ_0 and $\phi_{1,k}$ are the transposed vectors of $\bar{F}^m e + \sum_{s=0}^{d} \bar{U}_s^m e$ and $\bar{T}_k^m e$ respectively and '.' denotes the point-wise matrix product. The normalization constant c, which guarantees that

$$\sum_J \pi_{Z_{l,n}}^m(J) e = 1,$$

is identical to the probability that a reset event occurs at an arbitrary time instant. It can be verified, using the structural properties of $\pi^{l,n}$, that $c = \gamma e$, where the $1 \times h$ vector γ is given by

$$\gamma/p = \pi_1^{l,n}(\emptyset).\phi_0 + \pi^{l,n}(\emptyset)(I - R)^{-1}\left(\sum_{k=0}^{d} R_k^{(*)}.(e_{hl}\phi_{1,k})\right), \tag{8.16}$$

where $R_k^{(*)}$ represents the first block column of the btbT matrix R_k. Notice, in order to calculate the normalization constant c, it suffices to know the probability vector corresponding to the root node of the reset MC.

Remark 12. The average time after which the $Z_{l,n}$-th reset event takes place

$$\bar{t}_n = 1/c,$$

which may be regarded as an approximation to the average time at which the n-th marking occurs, is thus found as a byproduct and is often of particular interest on its own. Exact results can be generated provided that setting $l = n$ does not cause periodicity in the Markov chain $(\mathcal{X}_t, \mathcal{N}_t)$.

8.4 Transient Analysis of the CTM Protocol with Free Access

In this section we make use of our framework *tree-like processes with marked time epochs* to analyze the transient behavior of the celebrated Capetanakis-Tsybakov-Mikhailov protocol for random multiple access communication. We will restrict ourselves to the basic binary CTM algorithm with free access. Other protocol variations are also within reach of our framework as their behavior can often be captured by a tree-like process (see [112]).

8.4.1 Protocol Definition

The following standard assumptions are made [11, 75]. A single channel is shared among an infinite population of users that transmit packetized messages. A user is said to becomes active as soon as he has a message ready for transmission. Users refrain from generating new messages while being active. The time is slotted and each slot has a fixed duration, equal to the length of a packet. Transmissions of a packet are assumed to occur at the beginning of a time slot and each transmission is within the reception range of every user. Binary feedback (collision/no collision) is immediately available at the end of each time slot. Simultaneous transmissions are destructive, in the sense that all packets involved become corrupt. However, a discrete-time batch Markovian arrival process (D-BMAP) is used to model the user activity, which greatly relaxes the standard Poisson assumption. Some of the other standard assumptions can be further relaxed if necessary.

The D-BMAP process is more general compared to the D-MAP in the sense that it allows batch arrivals (see Section 2.2.1). To use an unambiguous notation, in this chapter we will denote the sequence of $h^* \times h^*$ matrices characterizing the D-BMAP process as $\mathcal{D}_n$, for $n \geq 0$. These matrices have the following probabilistic interpretation. If the D-BMAP is in state i $(1 \leq i \leq h^*)$ at the start of slot t, n users become active and the state of the D-BMAP equals j $(1 \leq j \leq h^*)$ at the beginning of slot $t+1$ with a probability given by the (i,j)-th entry of

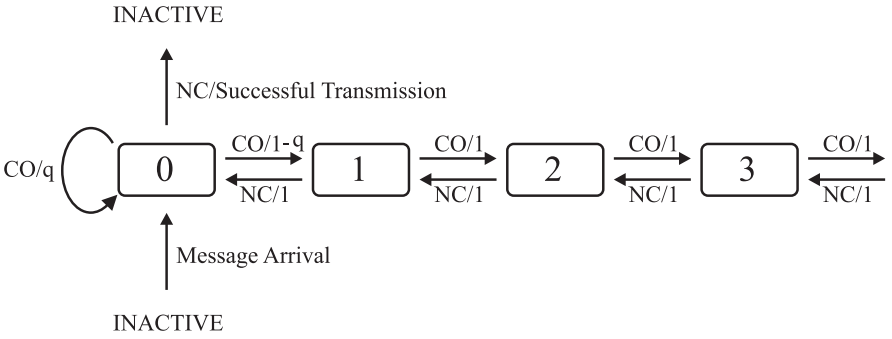


Figure 8.1: The CTM protocol for random multiple access communication

$\mathcal{D}_n$. The arrivals are assumed to occur on the boundary of time slot $t-1$ and t, meaning the n users that became active can transmit their packet as early as time slot t. How one checks the stability of the CTM protocol under D-BMAP arrivals is explained in [106].

The CTM protocol is a collision resolution algorithm in which users continue to retransmit a packet until it is correctly received. Apart from the immediate feedback, users do not exchange any information about their activity on the channel, as a consequence contention has to be resolved without any additional information. In order to resolve contention the CTM protocol divides the users involved in a collision into two groups $GR(1)$ and $GR(2)$. The users of the first group $GR(1)$ retransmit their packet in the next time slot, while users of the second group $GR(2)$ wait until the first group is resolved (meaning all messages belonging to $GR(1)$ have to be transmitted successfully and the users of $GR(2)$ need to be aware of this). The algorithm has a recursive nature, in the sense that if $GR(1)$ causes another collision, $GR(1)$ is split into two groups $GR(1a)$ and $GR(1b)$, and so on. Notice, in this case the users of $GR(2)$ need postpone any retransmission attempts until both $GR(1a)$ and $GR(1b)$ have been resolved. The channel access protocol is assumed to be the free access protocol. This means that users that become active immediately – that is, at the start of the next slot boundary – access the channel. The operation of the CTM protocol is illustrated in Figure 8.1. In this figure, CO denotes a collision and NC denotes no collision. By allowing the immediate participation of new arrivals in the contention tree, this problem can be regarded as a dynamic tree algorithm. The asymptotic behavior of tree algorithms that do not support this dynamic feature were studied in [88, 74]. Some transient statistics on these (nondynamic) tree algorithms can also be derived from our setting as will be illustrated in Section 8.4.3.

In the CTM protocol each active user maintains an integer value, referred to as the current stack level, as indicated in the boxes of Figure 8.1. A user that became active during slot $t-1$ initializes his current stack level for slot t at zero. A user is allowed to send his packet during the current time slot if his current stack level equals zero. The current stack level of a user is updated as follows. If slot t does not hold a collision, all users with a current stack level for slot t equal to $i > 0$ set their current stack level for slot $t+1$ to $i-1$. If slot t holds a collision, the current stack level for slot $t+1$ is set to $i+1$, while users with a current stack level for slot t equal to zero are divided into two groups. A user joins the first (resp. the second) group with a probability q (resp. $1-q$) and sets his current stack level for time $t+1$ to zero (resp. one). Remark, the current stack level maintained by a user may be regarded as the number of pending groups that require resolution before the user is allowed to retransmit.

8.4.2 Analytical Model

Let us now briefly indicate how to model the CTM protocol as a tree-like process (for more details see [106]). Consider the Markov chain $\{(\tilde{X}_t, \tilde{N}_t), t \geq 0\}$. Let $\tilde{X}_t$ be the string holding the current stack level for slot t for all backlogged stations, i.e., stations with a current stack level (for slot t) larger than zero. The sample space of the random variable $\tilde{X}_t$ is

$$\Omega_1 = \{\emptyset\} \cup \{J : J = s_k \dots s_1, s_j \geq 0, 1 \leq j \leq k, k \geq 1\}. \tag{8.17}$$

For instance, if $\tilde{X}_t = s_k \dots s_1$ there are $\sum_{i=1}^{k} s_i$ backlogged stations and the current stack level equals i for s_i of the backlogged stations. The auxiliary variable $\tilde{N}_t$ holds both the number of active stations with a current stack level for slot t equal to zero and the state of the D-BMAP at the start of slot $t+1$. Its sample space is given by

$$\Omega_2 = \{(n, j) : n \geq 0, 1 \leq j \leq h^*\}. \tag{8.18}$$

The Markov chain $\{(\tilde{X}_t, \tilde{N}_t), t \geq 0\}$ (with state space $\Omega_1 \times \Omega_2$) has a tree structure, but each node has an infinite number of children. To reduce this MC to a tree-like process one can approximate $\{(\tilde{X}_t, \tilde{N}_t), t \geq 0\}$ by a bivariate Markov chain $\{(\tilde{X}_t^d, \tilde{N}_t^d), t \geq 0\}$, obtained by setting a maximum value d on the number of stations that can have a same current stack level for slot t. In this way, each node of $\tilde{X}_t^d$ has only $(d+1)$ children and the variable $\tilde{N}_t^d$ has a finite range of size $h = h^*(d+1)$. The introduction of the value d as a restriction on the number of stations having the same current stack level, implies that some packets get dropped whenever a situation occurs in which more than d packets attain the same current stack level value. The CTM protocol does not drop packets, thus, d has to be chosen large enough such that number of dropped packets can be neglected (e.g., $< 10^{-10}$). In practice, setting $d = 10$ or 15 often suffices.

For further use, we will now quickly reintroduce the matrices $\bar{D}_k$, $\bar{U}_s$ and $\bar{F}$ that fully characterize the tree-like process $\{(\tilde{X}_t^d, \tilde{N}_t^d), t \geq 0\}$. The matrices $\bar{D}_k$ cover the transition probabilities from state $(J+k, (i,j))$ to the state $(J, (i', j'))$. This happens when slot t does not hold a collision, which implies

$$\bar{D}_k((i,j),(i',j')) = \begin{cases} (\mathcal{D}_{i'-k})_{j,j'} & i \leq 1, i' \geq k, i' < d, \\ \sum_{n \geq d-k} (\mathcal{D}_n)_{j,j'} & i \leq 1, i' \geq k, i' = d, \\ 0 & \textit{otherwise.} \end{cases} \tag{8.19}$$

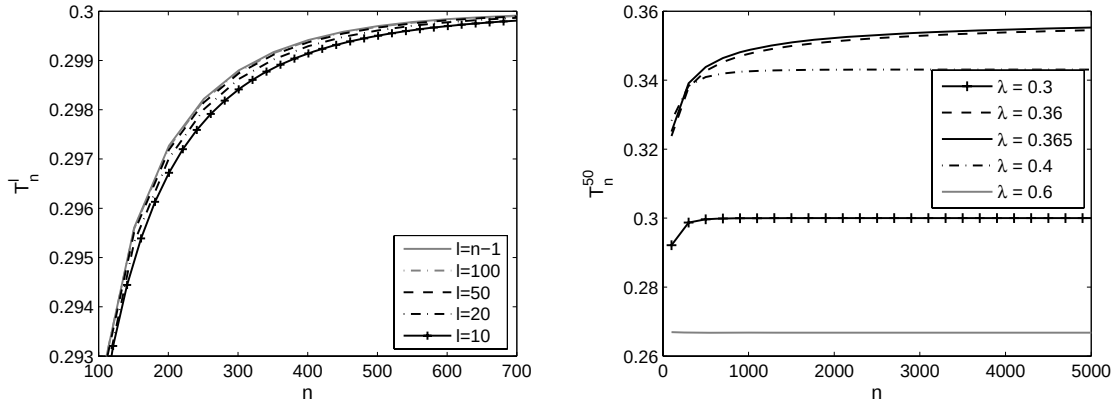
The matrices $\bar{U}_s$ hold the transition probabilities of going from state $(J+k, (i,j))$ to the state $(J+ks,\ (i', j'))$, which happens when a collision occurs in slot t. Hence,

$$\bar{U}_s((i,j),(i',j')) = \begin{cases} C_s^i q^{i-s} \tilde{q}^s (\mathcal{D}_{i'-(i-s)})_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' < d, \\ C_s^i q^{i-s} \tilde{q}^s \sum_{n \geq d-(i-s)} (\mathcal{D}_n)_{j,j'} & i > 1, i \geq s, i' \geq i-s, i' = d, \\ 0 & \textit{otherwise}, \end{cases} \tag{8.20}$$

where $\tilde{q} = 1 - q$ and C_s^i denotes the number of different possible combinations of s from i different items. Finally, the transition probabilities that the process goes from state $(\emptyset, (i,j))$ to the state $(\emptyset, (i', j'))$ are given by the matrix $\bar{F}$. This matrix is given by

$$\bar{F}((i,j),(i',j')) = \begin{cases} (\mathcal{D}_{i'})_{j,j'} & i \leq 1, i' < d, \\ \sum_{n \geq d} (\mathcal{D}_n)_{j,j'} & i \leq 1, i' = d, \\ 0 & \textit{otherwise.} \end{cases} \tag{8.21}$$

We are now in a position to apply our framework to assess some transient performance measures of the CTM protocol.



(a) Influence of l on the accuracy

(b) Various arrival rates λ

Figure 8.2: Throughput in slot n under Poisson arrivals

8.4.3 Performing a Transient Analysis

Let us first take a look at the transient throughput of the CTM algorithm. The throughput T_n in slot n equals the probability that a packet is successfully transmitted in slot n. This happens when there is exactly one station with his current stack level equal to zero for slot n. We can apply our framework by marking all time epochs, i.e.,

$$D_k^u = B^u = U_s^u = F^u = 0,$$

as this probability is readily available from the system state at time n. More specifically, we have

$$T_n = \sum_{j=1}^{h^*} \pi_n^m(1,j),$$

where

$$\pi_n^m(i,j) = \sum_J \pi_n^m(J,(i,j)),$$

and $\pi_n^m(J,(i,j))$ are the obvious $h = h^*(d+1)$ components of the vector $\pi_n^m(J)$. A simple expression for T_n can be found from the $1 \times h$ vector $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_d)$, defined by Eqn. (8.16), with $\gamma_i = (\gamma_i(1), \ldots, \gamma_i(h^*))$. That is, $\gamma_i(j)/c$ equals the probability that, at the n-th marking there are i stations with a current stack level equal to zero, while the state of the D-BMAP process is j. Hence,

$$T_n = \sum_{j=1}^{h^*} \frac{\gamma_1(j)}{c}.$$

We first present some results in case the user activity is modeled as a Poisson arrival process, that is, we set

$$\mathcal{D}_n = \frac{e^{-\lambda}\lambda^n}{n!}, \text{ for } n \geq 0.$$

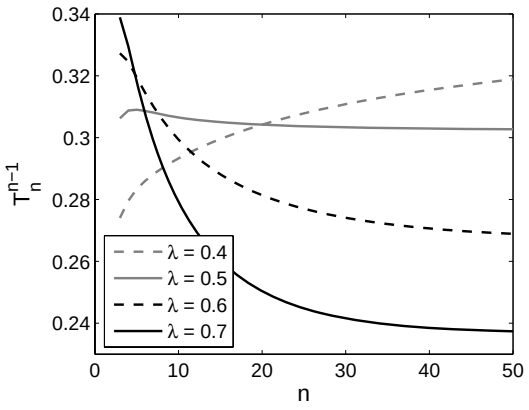
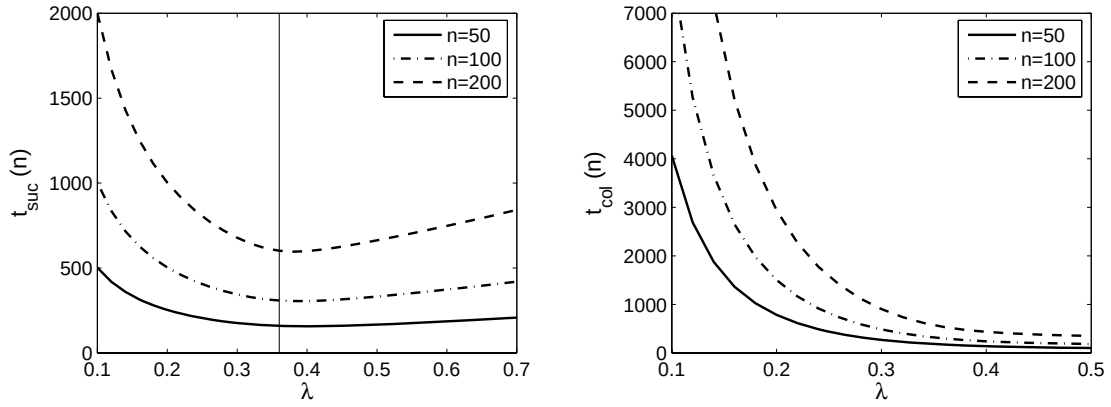


Figure 8.3: Initial throughput in the n-th slot under Poisson arrivals

Flajolet and Jacquet [33] have shown that the CTM algorithm with free access (with $q = 0.5$) is stable under a Poisson flow of arrivals if the arrival rate $\lambda < 0.360177$. We refer to this value as the maximum stable throughput (MST) of the random access algorithm. That is, in the standard information theoretical setting, the MST is defined as the highest possible (Poisson) input rate for which a packet has a finite delay with probability one.

Figure 8.2(a) depicts some approximations T_n^l for the transient throughput T_n for different values of l, the number of phases of the negative binomial distribution used to setup the reset process. T_n^l is computed in the same manner as T_n, except that the vectors $\pi_n^m(J)$ are approximated by $\pi_{Z_{l,n}}^m(J)$. The rate of the Poisson process $\lambda = 0.3$. As mentioned earlier, in some cases we cannot set l equal to n as this causes the MC $\{(\tilde{X}_t, \tilde{N}_t), t \geq 0\}$ to become periodic. Marking all time epochs will always cause periodicity, therefore the best approximation we can realize is by setting l equal to $n-1$. Figure 8.2(a) indicates that smaller values of l will result in a larger underestimation of the throughput T_n, however, for relatively small values of l we still obtain a quite accurate approximation of the throughput compared to letting $l = n - 1$. Working with a smaller l value implies that less time and memory is needed for the computation of the matrix V and the probability vectors $\pi_{Z_{l,n}}^m(J)$. For $l = 100$, the computation time, as given by the Matlab Profiler, is close to 30 seconds using an Intel Pentium M 1.70GHz processor with 1GB of memory. The Windows Task Manager indicated a peak memory usage of 80Mb for the entire Matlab session during the calculation of this figure.

In Figure 8.2(b) we compare the transient throughput for different arrival rates λ. For $\lambda = 0.3$, we notice that the transient throughput rapidly converges to the input rate λ. When we take the arrival rate λ closer to the MST (e.g., $\lambda = 0.36$), the throughput grows more slowly over time, but still seems to reach the input rate for n large enough. One can also observe that taking values for λ above the MST, does not seem to result in an output rate that is above the MST. Actually, for slot $n = 10^7$ we have $T_n^{100} = 0.35996$ and $T_n^{100} = 0.35812$ for $\lambda = 0.36$ and $\lambda = 0.365$, respectively. For such arrival rates a somewhat higher throughput is often achieved in the first few slots, as shown in Figure 8.3. After this initial phase during which several stations get backlogged, the number of collisions increases as the new arrivals need to compete with some of these backlogged stations. However, output rates above the MST can be achieved by some other random access mechanisms, e.g., FS-ALOHA [111].



(a) Avg time $\bar{t}_{suc}(n)$ until n-th success

(b) Avg time $\bar{t}_{col}(n)$ until n-th collision

Figure 8.4: Average time until the n-th event under Poisson input

Next, we will calculate the average time $\bar{t}_{suc}(n)$ at which the n-th successful transmission takes place. To apply our framework, it suffices to mark all time epochs that correspond to a successful transmission. The matrices $\bar{D}_k^m$, $\bar{U}_s^m$ and $\bar{F}^m$ that mark a time epoch are retrieved from the matrices $\bar{D}_k$, $\bar{U}_s$ and $\bar{F}$, given by Eqns. (8.19–8.21), as follows:

$$\bar{D}_k^m((i,j),(i',j')) = \begin{cases} \bar{D}_k((i,j),(i',j')) & \text{if } i = 1, \\ 0 & \textit{otherwise}, \end{cases}$$

for $k = 0, \ldots, d$ and

$$\bar{F}^m((i,j),(i',j')) = \begin{cases} \bar{F}((i,j),(i',j')) & \text{if } i = 1, \\ 0 & \textit{otherwise}. \end{cases}$$

The entries of the matrices $\bar{U}_s$ are equal to zero for $i = 1$, hence, $\bar{U}_s^m = 0$. When computing $\bar{t}_{suc}(n)$, we can set l equal to n to generate exact results. Recall,

$$\bar{t}_{suc}(n) = \bar{t}_n = 1/c,$$

where c is the normalization constant defined in Section 8.3.4.

Figure 8.4(a) shows the mean time $\bar{t}_{suc}(n)$ that passes until the n-th successful transmission occurs, for $n = 50, 100$ and 200. If the arrival rate λ is low, the number of slots needed to transmit n packets successfully is relatively high. When λ increases $\bar{t}_{suc}(n)$ decreases up to some point where a minimum is reached, afterward $\bar{t}_{suc}(n)$ increases again. This can be understood as follows. When λ is low, the total number of packets generated is small and therefore a considerable amount of time is required to generate n packets, most of these packets are successful during their first attempt. Higher values of λ reduce the time required to generate n packets, but at the same time also increase the mean number of transmission attempts needed. When the arrival rate becomes too large, the effect of the increased collision probability starts to dominate, which causes the minimum. Our results indicate that for large values of n, the rate λ for which $\bar{t}_{suc}(n)$ is minimal, can become larger than the MST.

Similarly, define $\bar{t}_{col}(n)$ as the average time at which the n-th collision takes place. To determine $\bar{t}_{col}(n)$ we can apply our framework with

$$\bar{D}_k^m = \bar{F}^m = 0 \text{ and } \bar{U}_s^m = \bar{U}_s.$$

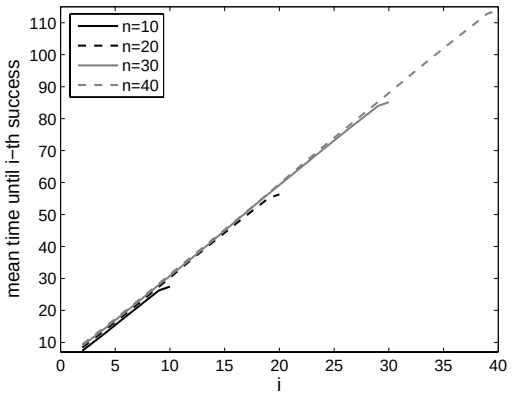


Figure 8.5: Mean time until the i-th success of a size n conflict

Remark, each time epoch where at least two stations have a current stack level equal to zero are marked, as these stations will transmit their packet in the next time slot causing a collision. The average time until the n-th collision is depicted in Figure 8.4(b), for $n = 50, 100, 200$. As one can expect, this average time decreases as the Poisson arrival rate increases.

Although we mainly focus on tree algorithms that allow new arrivals during the contention resolution period, we can also deduce some results for tree algorithms that do not support this feature. For instance, by assuming Poisson arrivals with $\lambda = 0$ and by considering a conflict of size n as the initial state, we can compute the mean time until the i-th success (for $i \leq n$) of a size n conflict. By repeatedly resetting the tree-like process at the i-th success, exact results can be realized for limited values of n. The results presented in Figure 8.5 show that one typically observes a linear growth as a function of i. Also, keeping i constant, a larger value of n results in a (slightly) larger mean time until the i-th success. Finally, one can observe that the end of the curves deviate from the linear trend since there will be no collision if there is only one active customer left on the channel.

Until now, we assumed that the users become active according to a Poisson process. As a final result, we will consider a two-state Markov modulated Poisson process (MMPP) as well. When in the first (the second) state, stations become active according to a Poisson process with rate λ_1 (λ_2). Transitions between the two states take place with a probability 0.001 at the end of each time slot, that is, the average sojourn time in both states is 1000 slots. We assume that the system is initialized in state 1. Figure 8.6 shows the transient throughput for different values of l for $\lambda_1 = 0.5$ and $\lambda_2 = 0.1$. Therefore the matrices $\mathcal{D}_n$ are given by

$$\mathcal{D}_n = \begin{bmatrix} 0.999e^{\lambda_1}\lambda_1^n/n! & 0.001e^{\lambda_1}\lambda_1^n/n! \\ 0.001e^{\lambda_2}\lambda_2^n/n! & 0.999e^{\lambda_2}\lambda_2^n/n! \end{bmatrix}.$$

As can be seen in Figure 8.6, during the first 1000 to 1500 slots the throughput decreases, after which it slowly starts to grow toward the arrival rate $\lambda = 0.3$. This is in line with the previous observations: initially we have an overloaded system as $\lambda_1 = 0.5$ which causes the initial decrease as explained before. On average, after about 1000 slots, the state of the D-BMAP will change to $\lambda_2 = 0.1$, allowing the backlogged stations to access the channel more often. As the CTM protocol is still stable under this arrival process, the throughput will eventually converge to $\lambda = 0.3$ as n goes to infinity.

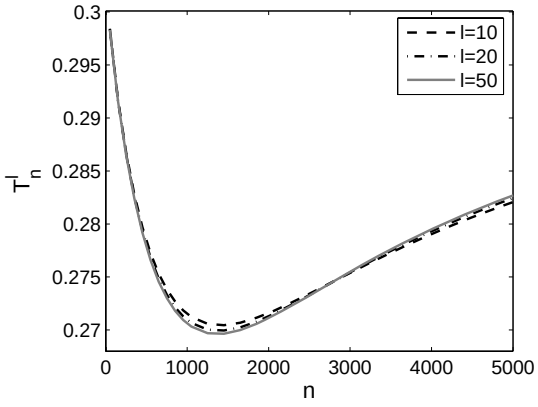


Figure 8.6: Throughput in the n-th slot under MMPP arrivals

This example concludes not only our study of the transient behavior of the CTM protocol, but also the last chapter of this thesis. Recall, our framework is not limited to random access mechanisms. It can be applied to any problem that can be modeled as a tree-like process.

Bibliography

[1] J. Abate and W. Whitt. The Fourier-series method for inverting transforms of probability distributions. *Queueing Systems*, 10:5–88, 1992.

[2] J. Abate and W. Whitt. Asymptotics for M/G/1 low-priority waiting-time tail probabilities. *Queueing Systems*, 25:173–233, 1997.

[3] K. Al Agha, P. Jacquet, and N. Vvedenskaya. Analysis of the priority stack random-access protocol in W-CDMA systems. *IEEE Transactions on Vehicular Technology*, 51(3):588–596, 2002.

[4] A.S. Alfa. Matrix-geometric solution of discrete time MAP/PH/1 priority queue. *Naval Research Logistics*, 45:23–50, 1998.

[5] A.S. Alfa. Discrete time queues and matrix-analytic methods. *TOP Sociedad de Estadística e Investigación Operativa*, 10(2):147–210, 2002.

[6] A.S. Alfa, B. Liu, and Q. He. Discrete-time analysis of MAP/PH/1 multiclass general preemptive priority queue. *Naval Research Logistics*, 50:662–682, 2003.

[7] A.S. Alfa, B. Sengupta, T. Takine, and J. Xue. A new algorithm for computing the rate matrix of GI/M/1 type Markov chains. In *Proceedings of the 4th International Conference on Matrix Analytic Methods*, pages 1–16, Adelaide, Australia. 2002.

[8] S. Asmussen, O. Nerman, and M. Olsson. Fitting phase-type distributions via the EM algorithm. *Scandinavian Journal of Statistics*, 23:419–441, 1996.

[9] F. Baccelli, P. Boyer, and G. Hebuterne. Single-server queues with impatient customers. *Advances in Applied Probability*, 16:887–905, 1984.

[10] D.Y. Barrar. Queueing with impatient customers and ordered services. *Operations Research*, 5:650–656, 1957.

[11] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall Int., Inc., 1992.

[12] P.P. Bhattacharya and A. Ephremides. Stochastic monotonicity properties of multi-server queues with impatient customers. *Advances in Applied Probability*, 28:673–682, 1991.

[13] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains*. Oxford University Press, 2005.

[14] D. A. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chains solver: algorithms. In *SMCtools Workshop*, Pisa, Italy. ACM Press, 2006.

[15] D.A. Bini, G. Latouche, and B. Meini. Solving nonlinear matrix equations arising in tree-like stochastic processes. *Linear Algebra and its Applications*, 366:39–64, 2003.

[16] C. Blondia. A discrete-time batch Markovian arrival process as B-ISDN traffic model. *Belgian Journal of Operations Research, Statistics and Computer Science*, 32(3,4):3–23, 1993.

[17] C. Blondia and O. Casals. Statistical multiplexing of VBR sources: A matrix-analytical approach. *Performance Evaluation*, 16:5–20, 1992.

[18] A. Bobbio, A. Horváth, M. Scarpa, and M. Telek. Acyclic discrete phase type distributions: Properties and a parameter estimation algorithm. *Performance Evaluation*, 54(1):1–32, 2003.

[19] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic Models*, 21:303–326, 2005.

[20] O. Boxma, D. Denteneer, and J. Resing. Delay models for contention resolution in closed populations. *Performance Evaluation*, 53:169–185, 2003.

[21] A. Brandt and M. Brandt. On the M(n)/M(n)/s queue with impatient calls. *Performance Evaluation*, 35:1–18, 1999.

[22] L. Breuer. An EM algorithm for batch Markovian arrival processes and its comparison to a simpler estimation procedure. *Annals of Operations Research*, 112:123–138, 2002.

[23] J.I. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25(5):319–329, 1979.

[24] R.M.L.R Carmo, E. de Souza e Silva, and R. Marie. Efficient solutions for an approximation technique for the transient analysis of Markovian models. Technical report, IRISA Publication Interne N 1067, 1996.

[25] G. Casale, E.Z. Zhang, and E. Smirni. Characterization of moments and autocorrelation in MAPs. *Performance Evaluation Review*, 35(2), 2007.

[26] E. Cinlar. *Introduction to Stochastic Processes*. Prentice-Hall, 1975.

[27] E. Coffman and L. Kleinrock. Feedback queueing models for time-shared systems. *Journal of the Association for Computing Machinery*, 15:549–576, 1968.

[28] M.B. Combé. Impatient customers in the MAP/G/1 queue. Technical Report BS-R9413, CWI, Amsterdam, April 1994.

[29] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press and McGraw-Hill, 1990.

[30] E. Falkenberg. On the asymptotic behaviour of the stationary distribution of Markov chains of M/G/1-type. *Stochastic Models*, 10(1):75–97, 1994.

[31] G. Fayolle, P. Flajolet, M. Hofri, and P. Jacquet. Analysis of a stack algorithm for random multiple-access communication. *IEEE Transactions on Information Theory*, IT-31(2):244–254, 1985.

[32] W. Feller. *An Introduction to Probability Theory and its Applications*, volume I. Wiley, New York, 1971.

[33] P. Flajolet and P. Jacquet. Analytic models for tree communication protocols. Technical Report 648, INRIA, 1987.

[34] O. Garnet, A. Mandelbaum, and M. Reiman. Designing a call center with impatient customers. *Manufacturing & Service Operations Management*, 4:208–227, 2002.

[35] D.P. Gaver, P.A. Jacobs, and G. Latouche. Finite Birth-and-Death models in randomly changing environments. *Advances in Applied Probability*, 16:715–731, 1984.

[36] N. Golmie, Y. Saintillan, and D.H. Su. A review of contention resolution algorithms for IEEE 802.14 networks. *IEEE Communication Surveys*, 2(1):2–12, 1999.

[37] D. Gross and C.M. Harris. *Fundamentals of Queueing Theory.* John Wiley and Sons, New York, 1974.

[38] A. György and T. Borsos. Estimates on the packet loss ratio via queue tail probabilities. In *IEEE Globecom*, pages 2407–2411, San Antonio, TX, USA. Nov 2001.

[39] S. Karlin H. M. Taylor. *An Introduction to Stochastic Modeling.* Academic Press, 1984.

[40] Q. He. Queues with marked customers. *Advances in Applied Probability*, 28:567–587, 1996.

[41] Q. He. The versatility of the MMAP[K] and the MMAP[K]/G[K]/1 queue. *Queueing Systems*, 38:397–418, 2001.

[42] Q. He and A.S. Alfa. The MMAP[K]/PH[K]/1 queues with a last-come-first-serve preemptive service discipline. *Queueing Systems*, 28:269–291, 1998.

[43] Q. He and A.S. Alfa. The discrete time MMAP[K]/PH[K]/1/LCFS-GPR queue and its variants. In *Proceedings of the 3rd International Conference on Matrix Analytic Methods*, pages 167–190, Leuven (Belgium). 2000.

[44] Q. He and M.F. Neuts. Markov chains with marked transitions. *Stochastic Processes and their Applications*, 74:37–52, 1998.

[45] Q. He and Hanqin Zhang. PH-invariant polytopes and coxian representations of phase type distributions. *Stochastic Models*, 22:383–410, 2006.

[46] N.J. Higham and H. Kim. Solving a quadratic matrix equation by Newton's method with exact line search. *SIAM Journal on Matrix Analysis and Applications*, 23:303–316, 2001.

[47] G. Horváth, P. Buchholz, and M. Telek. A MAP fitting approach with independent approximation of the inter-arrival time distribution and the lag correlation. In *Proceedings of QEST 2005*, pages 124–133. IEEE Computer Society, 2005.

[48] C.-L. Hwang and S. Q. Li. On the convergence of traffic measurement and queueing analysis: A Statistical-MAtch Queueing (SMAQ) tool. *Proceedings of Infocom'95*, pages 602–612, 1995.

[49] F. Ishizaki. Numerical method for discrete-time finite-buffer queues with some regenerative structure. *Stochastic Models*, 18(1):25–39, 2002.

[50] K.P. Sapna Isotupa and David A. Stanford. An infinite-phase quasi-birth-and-death model for the non-preemptive priority M/PH/1 queue. *Stochastic Models*, 18(3):387–424, 2002.

[51] P. Jacquet, P. Mühlethaler, and P. Robert. Performant implementations of tree collision resolution algorithms for CATV networks. Technical Report 4107, INRIA, 2001.

[52] A.J.E.M. Janssen and M.J.M. de Jong. Analysis of contention tree algorithms. *IEEE Transactions on Information Theory*, 46:2163–2172, 2000.

[53] R. Khayari, R. Sadre, and B.R. Haverkort. Fitting world-wide web request traces with the em-algorithm. *Performance Evaluation*, 52:175–191, 2003.

[54] N.K. Kim and M.L. Chaudry. Numerical inversion of generating functions: a computational experience. Manuscript, 2005.

[55] L. Kleinrock. A continuum of time-sharing algorithms. In *Proceedings of AFIPS SJCC*, volume 36, pages 453–548. AFIPS Press, 1970.

[56] L. Kleinrock. *Queueing Systems Vol. I.* Wiley, New York, 1975.

[57] V.G. Kulkarni. *Modeling and Analysis of Stochastic Systems.* Chapman & Hall, 1995.

[58] K. Laevens and H. Bruneel. Analysis of a single wavelength optical buffer. In *Proceedings of Infocom*, volume 3, pages 2262–2267, San Francisco. April 2003.

[59] A. Lang and J. L. Arthur. Parameter approximation for phase-type distributions. In *Matrix-Analytic Methods in Stochastic Models, (S. R. Chakravarthy and A. S. Alfa (Editors))*, pages 151–206, New York. Marcel-Dekker, Inc., 1996.

[60] G. Latouche, P.A. Jacobs, and D.P. Gaver. Finite Markov chain models skip-free in one direction. *Naval Research Logistics Quarterly*, 31:571–588, 1984.

[61] G. Latouche, C.E.M. Pearce, and P.G. Taylor. Invariant measure for quasi-birth-and-death processes. *Stochastic Models*, 14(1&2):443–460, 1998.

[62] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods and stochastic modeling.* SIAM, Philadelphia, 1999.

[63] G. Latouche and P.G. Taylor. Drift conditions for matrix-analytic models. *Mathematics of Operations Research*, 28(2):346–360, 2003.

[64] D. Liu and Y. Zhao. Determination of explicit solution for a general class of Markov processes. *Matrix-Analytic Methods in Stochastic Models*, pages 343–357, 1996.

[65] D.M. Lucantoni. New results on the single server queue with a batch Markovian arrival process. *Stochastic Models*, 7(1):1–46, 1991.

[66] D.M. Lucantoni, G.L. Choudhury, and W. Whitt. The transient BMAP/PH/1 queue. *Stochastic Models*, 10:461–478, 1994.

[67] D.M. Lucantoni, K.S. Meier-Hellstern, and M.F. Neuts. A single server queue with server vacations and a class of non-renewal arrival processes. *Advances in Applied Probability*, 22:676–705, 1990.

[68] F. Machihara. On the queue with PH-Markov renewal preemption. *Journal of the Operations Research Society of Japan*, 36:13–28, 1993.

[69] A. W. Marshall and I. Olkin. *Inequalities: theory of majorization and its applications.* Academic Press, New York, 1979.

[70] B. Meini. An improved FFT-based version of Ramaswami's formula. *Stochastic Models*, 13:223–238, 1997.

[71] B. Meini. *Fast algorithms for the numerical solution of structured Markov chains.* PhD thesis, University of Pisa, 1998.

[72] B. Meini. Solving QBD problems: the cyclic reduction algorithm versus the invariant subspace method. *Advances in Performance Analysis*, 1:215–225, 1998.

[73] D.G. Miller. Computation of steady-state probabilities for M/M/1 priority queues. *Operations Research*, 29(5):945–958, 1981.

[74] A. Mohamed and P. Robert. A probabilistic analysis of some tree algorithms. *Annals of Applied Probability*, 15(4):2445–2471, 2005.

[75] M. L. Molle and G.C. Polyzos. Conflict resolution algorithms and their performance analysis. Technical report, University of Toronto, CS93-300, 1993.

[76] R. Nelson. *Probability, Stochastic Processes, and Queueing Theory.* Springer-Verlag, 1995.

[77] M.F. Neuts. *Matrix-Geometric Solutions in Stochastic Models, An Algorithmic Approach.* John Hopkins University Press, 1981.

[78] M.F. Neuts. *Structured Stochastic Matrices of M/G/1 type and their applications.* Marcel Dekker, Inc., New York and Basel, 1989.

[79] M. Nuyens and A. Wierman. The foreground-background queue: a survey. *Performance Evaluation*, 2007. doi:10.1016/j.peva.2007.06.028.

[80] L.M. Le Ny and B. Sericola. Transient analysis of the BMAP/PH/1 queue. *International Journal of Simulation*, 3(3-4):4–14, 2003.

[81] C. Palm. Methods of judging the annoyance caused by congestion. *Tele. (English Edition)*, 2:1–20, 1953.

[82] V. Ramaswami. The busy period of queues which have a matrix-geometric steady state probability vector. *Operations Research*, 19(4):238–261, 1982.

[83] V. Ramaswami. Nonlinear matrix equations in applied probability - solution techniques and open problems. *SIAM review*, 30(2):256–263, June 1988.

[84] A. Remke, B.R. Haverkort, and L. Cloth. Model checking infinite-state Markov chains. In *TACAS 2005, LNCS 3440, Springer*, pages 237–252, Feb 2005.

[85] A. Remke, B.R. Haverkort, and L. Cloth. Uniformization with representatives: comprehensive transient analysis of infinite-state QBDs. In *Proceedings of SMCTools workshop*, Pisa, Italy. ACM, Oct 2006.

[86] A. Remke, B.R. Haverkort, and L. Cloth. CSL model checking algorithms for QBDs. *Theoretical Computer Science*, 382(1):24–41, 2007.

[87] A. Riska, V. Diev, and E. Smirni. An EM-based technique for approximating long-tailed data sets with PH distributions. *Performance Evaluation*, 55(1-2):147–164, 2004.

[88] P. Robert. On the asymptotic behavior of some algorithms. *Random Structures and Algorithms*, 27(2):235–250, 2005.

[89] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, Inc., 1972.

[90] S.M. Ross. Approximating transient probabilities and mean occupation times in continues-time Markov chains. *Probability in the Engineering and Informational Sciences*, 1:251–264, 1987.

[91] T. Ryden. An EM algorithm estimation in Markov-modulated Poisson processes. *Computational Statistics & Data Analysis*, 21(4):431–447, 1996.

[92] Y.E. Sagduyu and A. Ephremides. Energy-efficient collision resolution in wireless Ad-Hoc networks. In *IEEE Infocom*, volume 1, pages 492–502, San Francisco. 2003.

[93] L. Schrage. The queue M/G/1 with feedback to lower priorities. *Management Science*, 18:466–474, 1967.

[94] E. Seneta. *Non-negative Matrices and Markov Chains*. Springer-Verlang, New York, 1981.

[95] M. Shaked and J.G. Shanthikumar. *Stochastic Orders and their Applications*. Academic Press, New York, 1994.

[96] A. Sleptchenko, A. van Harten, and M.C. van der Heijden. An exact analysis of the multi-class M/M/k priority queue with partial blocking. *Stochastic Models*, 19(4):527–548, 2003.

[97] A. Sleptchenko, A. van Harten, and M.C. van der Heijden. An exact solution for the state probabilities of the multi-class, multi-server queue with preemptive priorities. *Queueing Systems*, 50(1):81–107, 2005.

[98] K. Spaey, B. Van Houdt, and C. Blondia. On the generality of binary tree-like markov chains. In *Proceedings of MAM 2006: Markov Aniversary Meeting*, pages 79–88, Charleston, USA. Boston Books, 2006.

[99] W. Stallings. *Operating Systems: Internals and design principles*. Prentice Hall, 2005.

[100] T. Takine. A nonpreemptive priority MAP/G/1 queue with two classes of customers. *Journal of the Operations Research Society of Japan*, 39(2):266–290, 1996.

[101] T. Takine and T. Hasegawa. The workload in a MAP/G/1 queue with state-dependent services: its applications to a queue with preemptive resume priority. *Stochastic Models*, 10(1):183–204, 1994.

[102] T. Takine and T. Hasegawa. The nonpreemptive priority MAP/G/1 queue. *Operations Research*, 47(6):917–927, 1999.

[103] T. Takine, B. Sengupta, and R.W. Yeung. A generalization of the matrix M/G/1 paradigm for Markov chains with a tree structure. *Stochastic Models*, 11(3):411–421, 1995.

[104] M. Telek and G. Horváth. A minimal representation of markov arrival processes and a moments matching method. *Performance Evaluation*, 64:1153–1168, 2007.

[105] B. S. Tsybakov and V.A. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Inform*, 14(4):32–59, 1978.

[106] B. Van Houdt and C. Blondia. Stability and performance of stack algorithms for random access communication modeled as a tree structured QBD Markov chain. *Stochastic Models*, 17(3):247–270, 2001.

[107] B. Van Houdt and C. Blondia. Tree structured QBD Markov chains and tree-like QBD processes. *Stochastic Models*, 19(4):467–482, 2003.

[108] B. Van Houdt and C. Blondia. Robustness of Q-ary collision resolution algorithms in random access systems. *Performance Evaluation*, 57:357–377, 2004.

[109] B. Van Houdt and C. Blondia. The waiting time distribution of a type k customer in a MMAP[K]/PH[K]/c (c=1,2) queue using QBDs. *Stochastic Models*, 20(1):55–69, 2004.

[110] B. Van Houdt and C. Blondia. Approximated transient queue length and waiting time distributions via steady state analysis. *Stochastic Models*, 21(2-3):725–744, 2005.

[111] B. Van Houdt and C. Blondia. QBDs with marked time epochs: a framework for transient performance measures. In *Proceedings of QEST 2005*, pages 210–219. IEEE Computer Society, 2005.

[112] B. Van Houdt and C. Blondia. Throughput of Q-ary splitting algorithms for contention resolution in communication networks. *Communications in information and systems*, 4(2):135–164, 2005.

[113] B. Van Houdt and C. Blondia. Analyzing priority queues with 3 classes using tree-like processes. *Queueing Systems*, 54:99–109, Oct 2006.

[114] B. Van Houdt, R.B. Lenin, and C. Blondia. Delay distribution of (im)patient customers in a discrete time D-MAP/PH/1 queue with age dependent service times. *Queueing Systems*, 45(1):59–73, 2003.

[115] J. Walraevens, B. Steyaert, and H. Bruneel. Performance analysis of a single-server ATM queue with a priority scheduling. *Computers & Operations Research*, 30(12):1807–1829, 2003.

[116] J. Walraevens, B. Steyaert, and H. Bruneel. A packet switch with a priority scheduling discipline: Performance analysis. *Telecommunication Systems*, 28(1):53–77, 2005.

[117] R.W. Yeung and A.S. Alfa. The quasi-birth-death type Markov chain with a tree structure. *Stochastic Models*, 15(4):639–659, 1999.

[118] R.W. Yeung and B. Sengupta. Matrix product-form solutions for Markov chains with a tree structure. *Advances in Applied Probability*, 26:965–987, 1994.

[119] J. Zhang and E.J. Coyle. Transient analysis of quasi-birth-death processes. *Stochastic Models*, 5(3):459–496, 1989.

[120] Y.Q. Zhao and A.S. Alfa. Performance analysis of a telephone system with both patient and impatient customers. *Telecommunication Systems*, 4:201–215, 1995.

Related Publications

The following publications summarize the published research that was carried out in function of this thesis:

- J. Van Velthoven, B. Van Houdt and C. Blondia. Response Time Distribution in a D-MAP/PH/1 Queue with General Customer Impatience. *Stochastic Models*, 21(2-3): 745-765, 2005.
- J. Van Velthoven, B. Van Houdt and C. Blondia. On the probability of abandonment in queues with limited sojourn and waiting times. *Operations and Research Letters*, 34(3): 333-338, 2006.
- J. Van Velthoven, B. Van Houdt and C. Blondia. The impact of buffer finiteness on the loss rate in a priority queueing system. *EPEW 2006, Lecture Notes in Computer Science*, LNCS 4054: 211-225, 2006.
- J. Van Velthoven, B. Van Houdt and C. Blondia. Transient analysis of tree-like processes and its application to random access systems. *ACM Sigmetrics Performance Evaluation Review*, 34(1): 181-190, 2006.
- J. Van Velthoven, B. Van Houdt and C. Blondia. Simultaneous Transient Analysis of QBD Markov Chains for all Initial Configurations using a Level Based Recursion. *Proceedings of QEST 2007, Edinburgh (UK), IEEE Computer Society*, 79-88, 2007.
- B. Van Houdt, J. Van Velthoven and C. Blondia. QBD Markov chains on binomial-like trees and its application to multilevel feedback queues. *Accepted for publication in the special issue of Annals of Operations Research in conjunction with The Sixth International Conference on Matrix-Analytic Methods in Stochastic Models (MAM6)*, 2008.

Nederlandse Samenvatting

Deze thesis beschrijft verschillende wachtlijnmodellen en bijbehorende algoritmes om ze te analyseren. De modellen stellen typische onderdelen van een telecommunicatiesysteem voor, waar klanten een bepaalde verwerking krijgen. Voor het opstellen van de modellen en de bijbehorende algoritmes maken we gebruik van matrix analytische methodes, een techniek die kort wordt besproken in Deel I. We zullen de structuur van de matrices die berekend moeten worden zoveel mogelijk benutten. Door rekening te houden met deze structuur, kunnen we namelijk aanzienlijk efficiëntere algoritmes ontwikkelen om de performantie van een systeem te analyseren. Deel I bevat ook een korte inleiding tot wachtrijtheorie. Intuïtief bestaat een wachtrijsysteem uit klanten die op willekeurige tijdstippen aankomen en worden verwerkt, waarna ze het systeem terug verlaten. Als klanten niet onmiddellijk kunnen worden verwerkt, worden ze in een wachtrij geplaatst, of dienen ze het systeem te verlaten als er geen plaats meer is in die wachtrij. Als een klant zijn bediening beëindigt, wordt bepaald welke wachtende klant als volgende bediend zal worden. Dit gebeurt aan de hand van een zogenaamd *scheduling* mechanisme. We bespreken kort enkele van deze scheduling mechanismen, alsook een aantal frequent gebruikte aankomst- en bedieningsprocessen. Tenslotte wordt er in dit inleidende stuk ook een definitie en enkele basisprincipes van Markov ketens gegeven.

Er wordt in de thesis een onderscheid gemaakt tussen twee grote gehelen. In Deel II wordt de *stationaire toestand* van bepaalde systemen besproken, terwijl onze aandacht in Deel III uitgaat naar het *transiënte gedrag*. In Hoofdstuk 3 wordt een methode voorgesteld om de verdeling van de responstijd van ongeduldige klanten in een D-MAP/PH/1 wachtrij te bepalen. Dit is de totale tijd die een klant in het wachtrijsysteem doorbrengt, vanaf het moment dat hij aankomt tot hij dit terug verlaat na volledig bediend te zijn. De methode maakt gebruik van een Markov keten, gebaseerd op de leeftijd van de klant die wordt bediend. Er worden drie types ongeduldige klanten beschouwd: klanten die altijd ongeduldig zijn, klanten die enkel ongeduldig zijn tijdens het wachten en klanten die hun bediening uitsluitend mogen aanvangen wanneer ze deze ook effectief kunnen beëindigen vooraleer hun geduld is verstreken. Buiten het berekenen van de responstijd, laat onze methode ook toe resultaten te bepalen in verband met de verdeling van het aantal klanten in de wachtrij. Omdat de uitvoeringstijd van dit algoritme een kwadratische functie is in de maximale hoeveelheid geduld van een klant, wordt er een reductie voorgesteld van deze Markov keten naar een *quasi-birth-death* proces, waardoor de tijdscomplexiteit van het algoritme drastisch gereduceerd kan worden ten koste van slechts een kleine hoeveelheid extra geheugen. Dankzij de tridiagonale structuur van de transitiematrix van de overeenkomstige Markov keten, heeft het resulterende algoritme een tijdscomplexiteit die slechts lineair is in de maximale hoeveelheid geduld. Dit wordt geïllustreerd aan de hand van enkele numerieke voorbeelden waarin we nagaan welke invloed de verdeling van het ongeduld heeft op de responstijd van klanten.

Deze voorbeelden suggereren een mogelijk verband tussen de variantie in de verdeling van de hoeveelheid geduld en het aantal klanten dat het systeem dient te verlaten ten gevolge van een volledig gevulde wachtrij. In Sectie 3.3 onderzoeken we of er inderdaad sprake is van een dergelijk verband. Om een algemene conclusie te kunnen trekken, veronderstellen we hier wel een eenvoudiger aankomsten- en bedieningsproces. Voor een wachtrijsysteem waarin zowel de aankomst- als de bedieningstijden een geometrische verdeling volgen, kunnen we bewijzen dat er minder klanten het systeem zullen verlaten zonder volledig verwerkt te worden als de verdeling van hun geduld kleiner is in de convexe ordening betekenis. Dit wordt zowel aangetoond voor klanten die geduldig zijn eens ze worden bediend als voor klanten die voortdurend ongeduldig blijven. Ook wordt er voor een wachtrijsysteem met geometrische bedieningstijden aangetoond dat er minder klanten gedwongen het systeem verlaten wanneer ze enkel ongeduldig zijn tijdens het wachten en niet meer tijdens hun bediening.

Hoofdstukken 4 en 5 handelen over prioriteitssystemen. In dergelijke systemen wordt er een onderscheid gemaakt tussen verschillende soorten verkeer, waarbij het belangrijke verkeer voorrang krijgt op het minder belangrijke verkeer. In Hoofdstuk 4 beschouwen we een systeem waarbij er twee verschillende klassen worden onderscheiden, één met hoge en één met lage prioriteit. In vele analyses worden de buffers in het systeem oneindig verondersteld, terwijl we in werkelijkheid natuurlijk altijd te maken hebben met eindige buffers. De reden voor deze keuze is dat het wiskundig vaak eenvoudiger is om te werken met oneindige buffers dan met eindige. Door uit te gaan van oneindige buffers en een veel voorkomende benadering te hanteren die gebruik maakt van het staartgedrag van de bezetting van de lage prioriteitsbuffer, kunnen er echter eigenaardige effecten optreden die we niet verwachten wanneer we werken met eindige buffers. In Hoofdstuk 4 wordt besproken hoe de kansen dat een klant verloren gaat omdat de buffer volledig bezet is, exact kunnen worden berekend door in de analyse gebruik te maken van eindige buffers. Tevens wordt er onderzocht hoe groot de invloed is als we één of beide buffers oneindig veronderstellen. Om de invloed van het al dan niet eindig nemen van de buffers duidelijk te kunnen identificeren, wordt er gewerkt met bedieningstijden die voor alle klanten gelijk zijn aan één tijdsslot. Voor de verschillende gevallen wordt er dan een algoritme voorgesteld om de verlieskansen van de lage prioriteitsklanten te bepalen. Een numeriek voorbeeld toont aan dat de benadering van deze verlieskansen die we krijgen door de buffers oneindig te veronderstellen niet altijd even accuraat is. Het blijkt dat we een overschatting van de verlieskansen voor de lage prioriteitsklanten krijgen wanneer we veronderstellen dat de overeenkomstige buffer oneindig is. Het al dan niet werken met een oneindige buffer voor de hoge prioriteit heeft echter een kleine invloed op de verlieskansen van lage prioriteit. Onder de veronderstelling van een eindige buffer voor de hoge prioriteit is er ook geen opmerkelijk verschil tussen de resultaten gebaseerd op het asymptotisch staartgedrag en degene die we bekomen, gebruik makende van de stationaire kansen. Foutieve resultaten kunnen wel worden bekomen wanneer beide buffers oneindig worden verondersteld en wanneer we gebruik maken van het asymptotisch staartgedrag. Deze fouten komen vooral voor in de omgeving van het zogenaamde transitiepunt dat de scheiding aangeeft tussen de gevallen die een geometrisch staartgedrag vertonen en degene waarbij dit niet het geval is.

In Hoofdstuk 5 wordt er een alternatieve wijze voorgesteld om een prioriteitssysteem te modelleren. In sommige gevallen kan het namelijk nuttig zijn om meer dan twee klassen verkeer te identificeren. In een netwerk kan er bijvoorbeeld een onderscheid worden gemaakt tussen spraak, video en ander verkeer zoals e-mail, websites, enz. De eerste twee zijn erg

gevoelig voor vertragingen en krijgen dus bij voorkeur prioriteit over het andere verkeer. Er is echter een groot verschil tussen videopakketten en spraakpakketten wat betreft de grootte. Als gevolg hiervan vergen de laatste beduidend minder verwerkingstijd en kan het dus interessant zijn om deze de hoogste prioriteit te geven, waardoor we reeds drie verschillende klassen hebben geïdentificeerd. We beschouwen in Hoofdstuk 5 een algemener aankomsten- en bedieningsproces dan in Hoofdstuk 4. Het aankomende verkeer wordt beschreven door een zogenaamd Markov aankomstenprocess met gemarkeerde jobs en de bediening van de klanten volgt een Phase-Type verdeling, die verschillend kan zijn voor de verschillende prioriteitsklassen. Het wachtrijsysteem bevat K prioriteitsklassen en wordt genoteerd als een MMAP[K]/PH[K]/1 wachtrij. Om een dergelijk systeem te modelleren, is een andere aanpak nodig dan diegene die werd gebruikt in Hoofdstuk 4. Het vraagt namelijk zeer veel geheugen als we voor elke klasse, het aantal klanten van deze klasse in de queue moeten bijhouden. De dimensies van de matrices die in het algoritme voorkomen, zouden te groot zijn om de gewenste eigenschappen te kunnen berekenen binnen een aanvaardbare tijd en met een redelijk gebruik van geheugen. Daarom wordt er in Hoofdstuk 5 gebruik gemaakt van een boomgestructureerde Markov keten. We onthouden slechts van bepaalde klassen het aantal klanten in de wachtrij, terwijl de andere klanten tijdelijk worden opgeslagen in een *stack*. In tegenstelling tot de andere hoofdstukken in deze thesis, beschouwen we dit systeem in continue tijd. Dit vereenvoudigt de analyse enigszins aangezien gelijktijdige gebeurtenissen die kunnen voorkomen wanneer we in discrete tijd werken, worden uitgesloten in continue tijd. Via een zogenaamde *uniformization* kan het continue-tijd systeem worden omgezet naar discrete tijd. Door een aantal extra toestanden toe te voegen, kan het algemene boomgestructureerd proces worden gereduceerd tot een *tree-like* proces. De extra toestanden zorgen ervoor dat er geen transities mogelijk zijn waarbij meer dan één klant op de stack wordt geplaatst, of hiervan wordt verwijderd. Deze toevoeging heeft als gevolg dat de verschillende matrices een interessante structuur krijgen, die we kunnen uitbuiten om de berekeningen te vereenvoudigen en te versnellen.

Bij prioriteitssystemen wordt de klasse van een klant bepaald bij aankomst. Dit is echter niet altijd mogelijk. Als we bijvoorbeeld verschillende taken beschouwen die ieder een bepaalde verwerkingstijd vergen, zouden we de voorkeur kunnen geven om korte taken eerst af te handelen en daarna pas de langere te verwerken. Op deze manier zouden we de gemiddelde responstijd van een taak kunnen minimaliseren. Helaas kennen we de duur van een bepaalde taak in het algemeen niet op voorhand en zullen we dus een *scheduling* moeten toepassen die op de één of andere manier voorrang tracht te geven aan kortere taken. In diverse besturingssystemen wordt hiervoor gebruik gemaakt van zogenaamde *multilevel feedback queues*. Deze bepalen de prioriteit van een taak aan de hand van de reeds verworven bedieningstijd. Hierdoor krijgen taken die reeds langer werden bediend een lagere prioriteit toebedeeld dan taken die tot op heden een kortere bedieningstijd kregen. In dit geval is prioriteit dus geen statisch gegeven, maar wordt de prioriteit van een job dynamisch aangepast op basis van de reeds verworven bedieningstijd. In Hoofdstuk 6 wordt een dergelijk scheduling mechanisme gemodelleerd aan de hand van een subklasse van de boomgestructureerde Markov ketens, namelijk de *quasi-birth-death processen op een binomial-like tree*. Een dergelijke boom heeft als eigenschap dat elk i-de kind van een knoop zelf i kinderen heeft. Na het invoeren van deze nieuwe klasse processen, tonen we hoe een multilevel feedback queue op die manier kan worden gemodelleerd. Op basis van dit model worden er methodes opgesteld om onder andere de gemiddelde responstijd van een taak en de verdeling van de lengte van de verschillende

wachtrijen te bepalen. We illustreren de methodes aan de hand van enkele numerieke voorbeelden waarin we de invloed onderzoeken van (i) de verdeling van de *batch* grootte en (ii) de verdeling van de verwerkingstijden op (a) de gemiddelde lengte van de wachtrij en (b) de variatiecoëfficiënt van de verdeling van de wachtrijlengte.

In Deel III bestuderen we het *transiënte gedrag* van twee soorten processen. We zijn met andere woorden geïnteresseerd in de toestand van het systeem op een relatief korte tijdsspanne. In verscheidene situaties kan een dergelijke analyse worden verkozen boven een stationaire studie. Bepaalde systemen hebben namelijk geen stationaire toestand, of indien deze er wel is, kan het zeer lang duren vooraleer hij wordt bereikt. In tegenstelling tot de modellen die in Deel II worden opgesteld, beperken we ons hier niet tot de analyse van één welbepaald systeem, maar stellen we algoritmes op die kunnen worden toegepast op elk systeem dat gemodelleerd kan worden als een *quasi-birth-death* Markov keten of als een *tree-like* proces. Hoofdstuk 7 beschrijft hoe het transiënte gedrag van quasi-birth-death processen geanalyseerd kan worden door gebruik te maken van een *quasi-birth-death proces met gemarkeerde tijdstippen*. De kracht van deze aanpak ligt in de mogelijkheid om transiënte problemen over een grote tijdschaal te bekijken, wat voor vele algoritmes onmogelijk is aangezien de tijdscomplexiteit hiervan stijgt in functie van het tijdstip waarin we zijn geïnteresseerd. In onze aanpak is dit niet het geval omdat we het transiënte probleem omzetten naar het bepalen van het stationair gedrag van een gerelateerde Markov keten. We bereiken dit door een discrete Erlangisatie toe te passen en een reset Markov keten op te stellen.

Daar de initiële toestand bij een transiënte studie een noemenswaardige invloed kan hebben op het bekomen resultaat, is er nood aan een manier om de resultaten voor verschillende initiële toestanden met elkaar te vergelijken. Het zou echter zeer inefficiënt zijn als voor elke starttoestand alle berekeningen opnieuw gemaakt zouden moeten worden. Daarom wordt er in Hoofdstuk 7 een algoritme geïntroduceerd dat gebruik maakt van een recursie per initiële level. Eerst worden de resultaten bepaald voor alle initiële toestanden die deel uitmaken van level nul, waarna deze worden gebruikt voor de berekeningen voor toestanden van level één, enz. Cruciaal in deze berekeningen is het bepalen van een verzameling R-matrices. Voor level nul moeten we hiervoor een kwadratische matrixvergelijking oplossen. Dit kan echter worden vereenvoudigd tot het oplossen van een kwadratische matrixvergelijking met een veel kleinere dimensie en een aantal Sylvester matrixvergelijkingen. In de recursieve stap moeten we slechts één nieuwe blokrij berekenen van één van deze matrices. De andere matrices kunnen we rechtstreeks uit de reeds bepaalde R-matrices halen. Dit kan worden ingezien aan de hand van de probabilistische interpretatie van deze matrices. De matrix R_i bevat namelijk het verwacht aantal bezoeken aan level $i+1$ vooraleer terug te keren naar level i. De reset Markov keten wordt door de toevoeging van een aantal artificiële toestanden herleid tot een quasi-birth-death vorm, waardoor elke reset een transitie naar level nul veroorzaakt. Als gevolg hiervan worden de transities vanaf een bepaalde level r onafhankelijk van de level waarop zij voorkomen en zijn alle R_i-matrices identiek voor $i > r$. Bovendien laat onze aanpak toe dat de berekeningen voor alle initiële toestanden van eenzelfde level simultaan kunnen gebeuren. Op het einde van Hoofdstuk 7 wordt de efficiëntie van onze recursie geïllustreerd aan de hand van een numeriek voorbeeld.

Tenslotte wordt in Hoofdstuk 8 het transiënte gedrag van een boomgestructureerde Markov keten onderzocht. Analoog aan de methode uit Hoofdstuk 7, voeren we het concept *tree-like*

proces met gemarkeerde tijdstippen in. Het struikelblok bij het oplossen van een boomgestructureerde Markov keten, is het bepalen van de matrix V. Hiervoor worden twee algoritmes besproken en met elkaar vergeleken op basis van convergentie en complexiteit. Eén van de mogelijke toepassingen van boomgestructureerde Markov ketens zijn de *multiple access algoritmes* die toelaten dat verschillende gebruikers eenzelfde link delen. Als twee of meer gebruikers op hetzelfde ogenblik data willen versturen, treedt er een botsing op en zal de data niet correct worden ontvangen. We bestuderen het zogenaamde Capetanakis-Tsybakov-Mikhailov protocol dat na een dergelijke botsing de gebruikers opdeelt in twee groepen en op basis hiervan beslist wanneer deze opnieuw hun data mogen sturen. Zo kunnen we onder andere bepalen hoe groot de kans is dat een pakket succesvol kan worden verstuurd in een bepaald slot, alsook de gemiddelde tijd tot het n-de succes of tot de n-de botsing.