The Impact of Balking Queues on n-Redundancy in Computer Systems

by

Maryam Sadeghi

B.Sc., Mathematics, Shiraz University, Iran, 2009
M.Sc., Mathematics, AmirKabir University, Iran, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE COLLEGE OF GRADUATE STUDIES

(Mathematics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Okanagan)

May 2018

The undersigned certify that they have read, and recommend to the College of Graduate Studies for acceptance, a thesis entitled: The Impact of Balking Queues on **n**-Redundancy in Computer Systems submitted by Maryam Sadeghi in partial fulfillment of the requirements of the degree of Master of Science

Dr. Javad Tavakoli, Department of Mathematics

Supervisor, Professor

Dr. Wayne Broughton, Department of Mathematics

Supervisory Committee Member, Professor

Dr. Paramjit Gill, Department of Mathematics

Supervisory Committee Member, Professor

Dr. Chen Feng, School of Engineering

University Examiner, Professor

Abstract

We study the steady state queue length probabilities for a queueing system for file downloading in which files are stored with redundancy. In particular, we address how the ability of a download file request to leave the system after seeing the queue length but before joining the queue. While it has been thoroughly proven that redundancy improves the performance of queueing systems, previous work has failed to address the inclusion of request loss affects a queueing system with redundancy. We develop a model for a queueing system with redundancy and request loss based on the Supermarket Model. Using that model, we solve for steady state queue length distribution and provide numerical results comparing systems with and without request loss. These numerical simulations show that in a queueing system with redundancy and request loss, the probability of having a particular number of requests in the system decreases as the number increases.

Lay Summary

After computer systems have been running for a long time, it is highly likely that file download requests will begin to be lost. As we know, losing requests can directly affect the cost and performance of such systems as well as increase user frustration. Thus, we are interested in understanding the behavior of such systems. We provide a model in which each file is stored in several servers, and in order to download a desired file, the download request should be served by one of these servers. We consider the case in which each download request enters the shortest queue in accordance with the number of requests waiting in that queue. Under these assumptions, we observe that request loss significantly affects the number of requests in a queue.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would first like to thank my thesis advisor Dr. Javad Tavakoli for his guidance in completing this thesis and providing me with the opportunity to work in queueing system. He consistently steered me in the right direction whenever he thought I needed it.

I would also like to thank my honorable committee members, Dr. Wayne Broughton and Dr. Paramjit Gill, for their precious comments and suggestions in preparing this thesis.

I would also like to acknowledge Dr. Chen Feng as the University examiner of this thesis, and I am gratefully indebted to him for very valuable comments on this thesis.

Finally, I would like to thank my family for their support which enabled me to pursue my education and finish my M.Sc. degree.

Dedication

To my parents

The reason of what I become today.

Thanks for your great support and continous care.

To my husband

who makes the impossible possible!

Thanks for being there for me.

Chapter 1

Introduction

Queuing theory is the mathematical study of the congestion and delays of waiting in line. The goal of queueing theory has been the design of balanced systems serving customers quickly and efficiently. For this reason, we have tried to examine the potential effects of impatience factors on computer and telecommunication systems to build efficient and cost-effective systems. This chapter is divided into three parts: motivation, literature review, and the organization of this thesis.

1.1 Motivation and Contribution

The field of queueing theory began in 1909 with the study of telephone conversations. The Danish mathematician A.K. Erlang was the pioneer researcher in this area whose goal was to determine the number of telephone operators to manage a given volume of calls. In the 1930s, F. Pollaczek extended Erlang's work and defined an $M/G/1$ queueing system in which jobs arrive according to the Poisson process and have general service time distribution. Currently, queueing systems have various applications in our

daily life, such as waiting in line at banks and post offices, traffic flows, and downloading files. In 2016, Li et al. [LRS16] studied cloud storage systems under the assumption that new file download requests enter the system with a constant rate λ. They found the expected waiting time a request spends in a system when files are stored in servers under the Maximum Distance Separable (MDS) code (see subsection 3.1.2 for more details). Moreover, they compared their results with a simple $M/M/1$ queueing system where download requests are not able to choose servers. Much study has gone into the design of algorithms to reduce waiting time in cloud storage systems. For example, redundancy which referrers to storing a file in several servers is one of these algorithms. However, some facts about request loss in computer systems were ignored. For instance, if a computer system is old or has been used a lot in the past, requests are discouraged from joining the queue. Hence, requests can choose whether or not enter the system based on the number of requests waiting in the line, this is called balking. To the best of our knowledge, none of the previous works considered a combination of redundancy and request loss (balking).

In this thesis, our objective is to obtain steady state probabilities of queue length in computer systems. For this, we first assign a probability for either joining or balking the system which depends on the number of download requests in the queue. In order to protect files stored in servers, we use redundancy. In Chapter 3, we will determine the steady state probabilities of queue length in accordance with balking probability and redundancy in

storage. In addition, we can observe how redundancy and balking affect the queue length steady state probability.

1.2 Literature review

An $M/M/1$ queue with balking and reneging was considered for the first time in 1957 by Haight [Hai57]. Since then, many researchers have studied queueing systems with various balking probabilities. The main goal of this research has been to determine steady state probabilities of queue length and waiting times. Ancker and Gafarian studied a queueing system in which customers arrived as a Poisson process (see section 2.3 for more details), and balked with the probability $\frac{i}{N}$ where i was the number of customers waiting in a line and N was the capacity of the system [AJG63]. In [SR67], Subba Rao analyzed an $M/G/1$ queueing system in which customers could balk with a constant probability $1-\beta$. In [Nat75] and [vTvdV80], the authors analyzed a birth and death process in which λ is defined as the general arrival rate and the probability of entering the system is $\frac{1}{i+1}$, consequently, customers join with arrival rate $\frac{\lambda}{i+1}$ where i customers were waiting in the line. In the work of Liu et al. [LK08, LK06], a queueing system with balking was considered. They defined a fixed level at the time of arrival such that an arriving customer will be served if the system workload is below that level. In [BK10, KC09], the authors studied a finite capacity queueing system with balking. In [Kap11], Kapodistria considered the case that impatient customers were able to de-

part the system simultaneously in an $M/M/c$ queueing system. In 2010, Wang et al. published a review paper covering many possible assumptions about impatient customers [WLJ10]. In 2011, Lozano defined two different probabilities for impatient customers. He analyzed a queueing system with balking in which customers were able to balk with either a constant probability or increasing probability depending on the number of customers in the line [LM08]. According to [Saa61], the expected number of customers in an $M/M/1$ queue is defined as $E_n = \sum_{i=0}^{\infty} iq_i = \sum_{i=1}^{\infty} i(\frac{\lambda}{\mu})^i q_0 \prod_{r=0}^{i-1} p_r$ with the following notations

- $p_r = \dfrac{1}{1+r}$ is the probability of an arrival joining the queue,
- λ and μ are the arrival and service time rates,
- i is the number of customers waiting in the queue (not being served),
- q_i is the probability that there will be a queue of i waiting customers.

Furthermore, he determined the following formula for the expected waiting time:

$$E_w = \frac{1-q_0}{\mu} + \frac{\lambda}{\mu^2} \sum_{i=1}^{\infty} ip_i q_i$$

In 2002, Wang and Chang considered a finite capacity $M/M/R$ queueing system in which customers were able to either depart the system before receiving service (reneging) or to not joint the system at all (balking)[WC02].

In 2001, Mitzenmacher [Mit01] proved that if customers were able to randomly choose two servers from a total of L servers, their waiting time reduced

exponentially compared to an $M/M/1$ queueing system. In [JDPF05], the authors provided a comparison of the delay performance among different coding schemes. In [SLR16], the authors assumed that in a system with n servers, each batch is divided into r requests. However, any distinct set containing k requests ($k \leq r$) is enough to complete the service for the given batch. Moreover, the authors assumed requests arrive based on a Poisson process and service times are defined to be exponentially distributed. They proved that when each batch is sent to all n servers, i.e. when redundancy occurs, the average waiting time is minimized. In [GZD$^+$15], the authors studied different methods of analyzing redundancy and proved that the method of redundancy surpassed the method of customers always joining the shortest queue in reducing latency. In [GHBSW$^+$17], the authors found a formula to express the expected waiting time under redundancy and proved this expected value decreased when redundancy increased. As can be seen from these works, a great deal of study has gone into the topic of redundancy in queueing systems. However, none of these works considered balking in addition to redundancy, an omission which this thesis aims to correct.

1.3 Organization

The organization of this thesis can be summarized as follows:
In chapter 2, some basic definitions and concepts of queueing systems will be reviewed. Chapter 3 is allocated to finding steady state probabilities of queue length for a queueing system where files are stored in n servers and there is a probability of balking. Chapter 4 presents the numerical results and analysis of the system in various situations. Finally, chapter 5 provides a summary of the results and presents ideas for future work based on the work of this thesis.

Chapter 2

Preliminaries

In this chapter, we will review some basic definitions and concepts in queueing systems. We will define different types of systems and give examples of each of them so as see applications of queueing systems in reality.

2.1 Markov Chain

In order to define a queueing system, it is best to begin with the concepts of the stochastic process and Markov chain. A stochastic process is the mathematical abstraction of an experiment which is developed by probabilistic laws [GSTH08].

Definition 2.1. A stochastic process is an infinite collection of random variables, usually indexed by integers (discrete) or real numbers (continuous), which are often interpreted as time. In mathematical notation, a stochastic process is the set $\{X(t) : t \in T\}$ where $X(t)$ is defined to be the state of the process at time t. Usually, $T = 0, 1, 2, \ldots$ or $T \in (0, \infty)$.

A stochastic process can be classified as a discrete or continuous time process. For discrete time, a stochastic process is a sequence of random

variables and the time series related to these random variables. However, in a continuous time stochastic process, the index variable takes a continues set of values. A Markov chain or Markov process for discrete or continuous random variables respectively, is a memoryless stochastic process and defined as follow.

Definition 2.2. In a Markov chain, the conditional distribution of any future state X_{n+1}, given the past states $X_0, X_1, ..., X_{n-1}$ and the present state X_n, is independent of the past states and depends only on the present state, i.e.,

$$\begin{aligned} P\{X_{n+1} = j | X_0 = i_0, X_2 = i_1, ..., X_n = i\} \\ = P\{X_{n+1} = j | X_n = i\} = p_{ij}. \end{aligned} \tag{2.1}$$

Therefore, the value p_{ij} is the probability of going from state i to state j. In accordance with the nature of the problem, a Markov chain can have a finite or infinite number of states. In what follows, we will provide some other concepts and definitions about states to classify Markov chains.

Definition 2.3. State j is said to be accessible from state i when there is a path starting from i going to j, i.e. the probability that a process in state i will be in state j after n additional transitions, $p_{ij}^n > 0$.

Hence, we can say two distinct states i and j communicate if i is accessible from j and j is accessible from i. In addition, the set of all states communicating with each other is called a class of states. If all of the states in a Markov chain are in only one class, this Markov chain is said to be irreducible.

Definition 2.4. State i is recurrent if starting from i, the process returns to state i at some point almost surely. Otherwise, state i is said to be transient.

Moreover, recurrent state i is positive recurrent if starting in i, the expected time until the process returns to state i is finite. It is possible to have a periodic or aperiodic state by allocating a specific number called period to each state. State i has a period d if $p_{ii}^n = 0$ whenever n is not divisible by d and d is the largest integer with this property [Ros14]. Thus, state i is aperiodic if $d = 1$. All states in the same class have the same period. A positive recurrent and aperiodic state is called an ergodic state. Our model in Chapter 3 is a positive recurrent and irreducible Markov chain.

2.2 Steady-state probability

Now, we need to define steady state probabilities and a transition matrix. The transition matrix P is defined using the probability of moving from one state to another. First, let us suppose $S = \{0, 1, 2, ...\}$ is a countable state space and p_{ij} is the probability that the process transits from state i to state j. Thus, it is possible to write a matrix in which each entry represents a transition probability. As mentioned in the previous section, $p_{ij}^n = p\{X_n = j | X_0 = i\}$ indicates the probability of going from state i to j in n transitions. If we assign $P = [p_{ij}]$ as a transition matrix, then $P^n = [p_{ij}^n]$ represents an n-step transition probabilities which is almost identical to P as n goes to infinity. In other words, p_{ij}^n is converging to a value which

is the same for all i (when $n \to \infty$). According to [Ros14], the limiting probability $\lim_{n\to\infty} p_{ij}^n$ exists for an irreducible ergodic Markov chain and it can be derived from the following equations:

$$\pi_j = \lim_{n\to\infty} p_{ij}^n, \quad j \in S \tag{2.2}$$

$$\sum_{j\in S} \pi_j = 1$$

thus, π_j can be interpreted as the probability of being in state j in the long term which is independent of the initial state i. According to the *Chapman-Kolmogorov* equation, $p_{ij}^{n+1} = \sum_k p_{ik}^n p_{kj}$, and by taking limit from both sides as $n \to \infty$, the limiting equation becomes $\pi_j = \sum \pi_k p_{kj}$. Consequently, the vector form of this equation is $\pi = \pi P$ where $\pi = (\pi_1, \pi_2, ..., \pi_L)$ and L is the total number of states in a finite state Markov chain or $\pi = \{\pi_i : i \geq 0\}$ in a countable Markov chain. Since finding the steady state probability can help analyze the behavior of a system, we will compute the steady state probability of queue length.

2.3 Poisson Process

A Poisson process is the most practical example of stochastic process and continuous time Markov chain. The Poisson process can help to model the time customers enter a system and describe the probability of an arrival at any time. It can be classified by two stochastic processes. The first one is the

interarrival times X_i for $i \geq 1$ and the second one is a counting process $N(t)$. In the following, we will introduce these two alternative processes completely. An arrival process is a sequence of positive increasing random variables $T_1 < T_2 < T_3 < ...$ that are independent and identically distributed. Let $T_{i+1} - T_i = X_i$ for $i > 1$. This is a positive random variable, called the interarrival interval. Interarrival intervals, X_i, are independent and identically distributed, and are exponentially distributed. That is,

$$f_X(x) = \lambda \exp(-\lambda x) \quad for\ some\ real\ \lambda > 0\ and\ x \geq 0. \tag{2.3}$$

In a Poisson process, the parameter λ is called the arrival rate of the process. Furthermore, since $T_i = \sum_{j=1}^{i} X_j$ for all $i \geq 1$, i.e. the sum of i $I.I.D.$ random variables, then the density function for each T_i, called the Erlang density, is given by

$$f_{T_i}(t) = \frac{\lambda^i t^{i-1} exp(-\lambda t)}{(i-1)!}. \tag{2.4}$$

Despite the interarrival times, the counting process contains positive, increasing, and integer-valued random variables $N(t)$, or the number of arrivals in the interval $(0, t]$. The arrival process and the counting process are related by

$$\{T_i \leq t\} = \{N(t) \geq i\} \tag{2.5}$$

which means that if the i^{th} customer enters the system by time t, the number of customers at time t is at least i [Gal96].

2.4 Queuing Theory

In a queueing system, customers arrive at random times. Having received service, they depart the system one by one or simultaneously, depending on the design of the system.

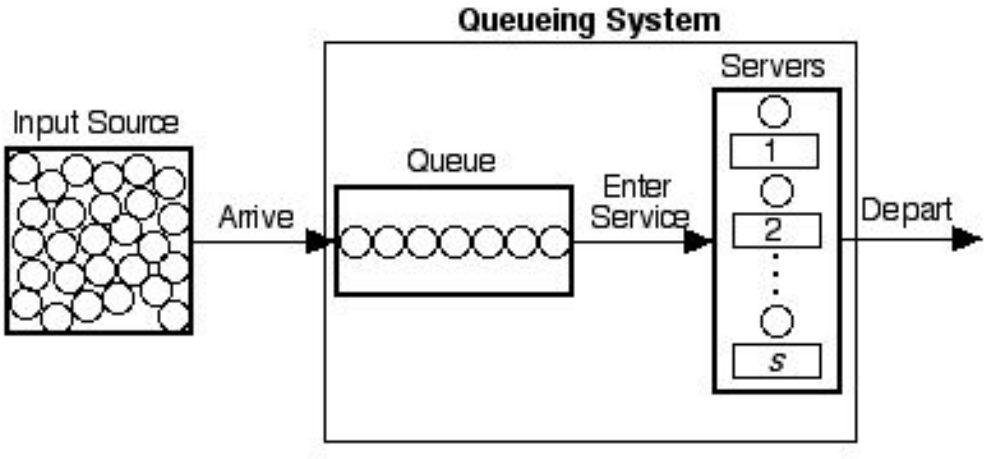


Figure 2.1: A queueing system with s servers

The classification of a queueing system is based on the input process, the service time distribution, and the queue discipline.

The input process is the probability distribution of customers arrival times. This probability is related to the pattern of customers arriving and joining the system. For example, customers can join the line either one by one or in group. Once customers join the system, a waiting line is formed. The number of customers in a queueing system can be finite or infinite. Therefore, if the system has a capacity, customers may not be able to join

the queue. Various distributions have been defined for the input process; the Poisson process is one of the most common. We can suppose customers arrive to the system with a constant rate or a rate depending on different factors such as the number of customers waiting in the line. Furthermore, in accordance with assumptions about the system, each customer may have the opportunity to decide whether or not to join the system. For instance, when many customers wait for service in a bank, a costumer enters, but then immediately decides the line is too long and leave.

The service time distribution is the probability distribution of the time spent serving a customer. One assumption about the service time distribution is that it is a non-negative, independent, and identically distributed random variable. An important measure to describe the service time pattern is the service rate, which is the reciprocal of the expected service time. Hence, each customer is served based on a defined rate called the service rate. Another important parameter in a queueing system is the number of servers, which can be either finite (e.g. in a supermarket) or infinite (e.g. in a cloud storage system). Increasing the number of servers can significantly reduce the waiting time a customer spends in a system.

The queueing discipline is the order in which customers are served. For example, it can be first come first serve (FCFS), which means customers will be served in order of their arrival, such as a line in a bank. Another example of queue discipline is last come first serve (LCFS) which means customers who entered the system most recently will be served first. Jobs arriving in a

production facility with one or more machining centers are often scheduled based on last come first serve since it is more convenient to have access to stored requests or meet due date requirements. Service in random order is another discipline for customers to be served. In this case, the service order for customers is chosen randomly.

In a queueing system, the waiting time is the sum of the time a customer waits in a line, and being served. Reducing the waiting time is one of the important issues in systems with queues. In every queueing system, customers enter the system with average rate λ and are served with average rate μ. The ratio of arrival rate and the service rate is called the traffic intensity $\rho = \frac{\lambda}{\mu}$, and is very significant in the performance of any queueing system. For example, if $\rho > 1$, the number of customers joining the system is greater than the number of customers departing the system, then the queue size will always increase, and no steady state exists. Thus, in order to have a stable queueing system, we always assume $\lambda < \mu$, so that the steady state distribution of queue length exists.

In queueing models, we mostly assume interarrival intervals are positive, independent, and identically distributed random variables and the format $A/B/C/c$ is used to describe the interarrival distribution, the service time distribution, the number of servers, and the capacity of the system respectively.

Example 1. An $M/M/R$ system for $R \geq 1$ is a queueing system in which customers arrive as a Poisson process, and service time is exponentially dis-

tributed, and there are R servers. This model can be finite or infinite. If $R = 1$ this system is called a single server queueing system. ■

In general, the number of customers in a queueing system at time t forms a birth and death process since arrivals can be considered as a birth process and departures as a death process.

Example 2. In an $M/GI/R$ model, arrivals form a Poisson process, service times are $I.I.D$ random variables with arbitrary distribution, and $R \geq 1$ is the number of servers. ■

Example 3. In the $GI/M/R$ model, arrivals have a general distribution, service times follow a Poisson process, and there are R servers. ■

Example 4. An $M/D/R$ model involves arrivals forming a Poisson process, a deterministic service time, and R servers. ■

Now, let us define a queueing system with impatience factors. In reality, each customer has the opportunity to decide whether or not enter the system. A probability distribution can be defined for this situation depending on either the waiting time a new customer must spend in the line, or the number of customers in the system. Since working with the number of customers in the system is less complicated than working with the waiting time, the majority of such probabilities are based on the number of customers. Consequently, the probability of balking, b_i, is the probability that the $(i+1)^{st}$ customer does not enter the system when i customers are already present.

Thus, $1 - b_i$ is the probability that $(i+1)^{st}$ customer joins the queue. In this thesis, we assume that a request will not enter the queue with probability depending on the number of existing requests in the system. In the next chapter, we will discuss this further.

Chapter 3

The Impact of Balking on Computer Systems

In this chapter, we will consider the case in which customers or requests may not join the queue due to the number of customers already in the queue. After defining our model and the load-balancing algorithm, we will derive the steady state queue length distribution in the case where arrival rates depend on the number of requests that exist in the queue. This work can be considered as an initial step toward applying redundancy in storing files when requests can balk the system. In addition to balking, sometimes, due to the number of customers waiting in a system, arriving customers may leave the system without receiving service. However, in this thesis, we assume that when a request joins the line, it is not able to leave until it has received service. Here, under both redundancy and balking, our objective is to find steady state probabilities for queue length.

In what follows, we will define our model and specify the load balancing scheme which is most appropriate to our work. Moreover, we will compute request loss to see how balking can influence the cost of the system.

3.1 Power of n Choices

In this section, we will first introduce the supermarket model [Mit01], then we move from the supermarket model to analysis of cloud storage systems [LRS16]. Since redundancy has been proven to reduce customer or request waiting time and improve the performance of the system, we focus mainly on this aspect of cloud storage systems.

3.1.1 Supermarket Model

In 2001, Mitzenmacher introduced a new queueing model called the supermarket model. In this model, each customer first chooses a subset of n servers from the total L servers, and then goes to the shortest queue among these choices. The n servers are chosen uniformly randomly, with replacement. Clearly, if a customer chooses an idle server, he will be served instantly and there is no waiting time. In this section, we will see the importance of the number of choices in reducing the waiting time. Under the supermarket model, customers arrive following a Poisson process with rate λ, where $\lambda < 1$. Since there are L servers in the system, the total arrival rate is $L\lambda$. Furthermore, service times are assumed to be exponentially distributed with rate $\mu = 1$.

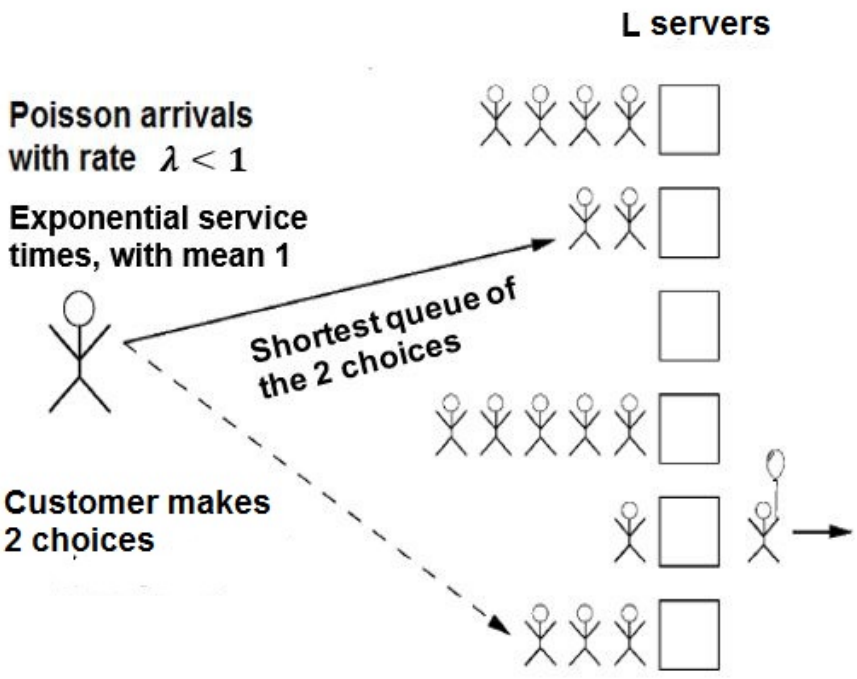


Figure 3.1: A super market model in which an arriving customer chooses two queues and joins the shortest one.

A differential equation based on arrivals and departures can be used to determine the steady state queue length probabilities. Let s_m be the probability that the customer ends up in a queue of at least m customers at a particular time t. Then, according to [Mit01], the differential equation

$$\begin{cases} \dfrac{ds_m}{dt} = \lambda(\ s_{m-1}^n - s_m^n)\ -(\ s_m - s_{m+1}) \quad for\ m \geq 1 \\ s_0 = 1. \end{cases} \tag{3.1}$$

expresses the change in the supermarket system. Since a fixed or a critical

point is a point p such that if $s(t) = p$ then $s(t^{'}) = p$ for all $t^{'} \geq t$, the sequence $\{s_m\}_{m\geq 0}$ converges to a fixed point if and only if $\frac{ds_m}{dt} = 0$ for all m. According to equation 3.1, it is proved that the s_m converges exponentially to a fixed point, denoted by $\lambda^{\frac{n^m-1}{n-1}}$. However, Mitzenmacher only considered a system in which an arriving customer will undoubtedly enter the system and join the shortest queue among his choices. As we know, if each customer has only one choice, i.e. $n = 1$, the model becomes an $M/M/1$ queueing system with the waiting time $T_1(\lambda) = \frac{1}{1-\lambda}$. However for $n \geq 2$, the supermarket model queueing system performs better than an $M/M/1$ queueing system. If $T_n(\lambda)$ is the waiting time when the customer makes n choices, then under the supermarket model, Mitzenmacher showed that $T_n(\lambda) < c_n(\log T_1(\lambda)) < T_1(\lambda)$ for some constanct c_n which depends only on n.

3.1.2 Mean Field Analysis of Coding in Cloud Storage System

In 2016, Li et al. proved that coding outperforms redundancy in file access delay [LRS16]. In their work, the Maximum Distance Separable (MDS) code is used to store files. Using an (n, k) MDS code, each file is divided into $k \leq n$ parts of equal size such that n combinations of k parts are stored in n distinct servers. In order to download a file, any set of k combination would be enough to complete a download request. For instance, let us assume file A is stored according to the MDS code $(2, 3)$ over the finite field $F_2 = \{0, 1\}$.

Thus, file A must be divided into 2 parts A_1 and A_2 and 3 combinations A_1, A_2 and $A_1 + A_2$ are stored in 3 different servers. Based on the property of MDS codes, downloading any 2 combinations A_1 and A_2, A_1 and $A_1 + A_2$, or A_2 and $A_1 + A_2$ can help to have the entire file A.

It is assumed that L servers exist in the system and that arrival requests form a Poisson process with mean λ, where $0 \leq \lambda \leq 1$. Let us assume that queues are organized based on their size. Hence, if Q_i represents the i^{th} shortest queue among n servers containing the file, we have

$$Q_1 \leq Q_2 \leq Q_3 \leq \cdots \leq Q_n. \tag{3.2}$$

Since any set of k parts can complete the request, it is assumed that the service times are exponentially distributed with the mean $\frac{1}{k}$. As mentioned before, in order to download a file, each request must be sent to k of the n servers that contain the file when queues are organized based on their sizes. Let π_m be the steady state probability that queue length is exactly m. In other words, π_m is the fraction of servers with queues of size m. If $L\pi_m$ is the average number of servers with queue length of m, then $L\pi_m\lambda\Delta$ is interpreted as the average number of these queues that become of size $m + 1$ as an arrival joins the system in a small time interval Δ. If the probability that i^{th} queue contains m requests is $Pr(Q_i = m)$, then $L\lambda\Delta \sum_{i=1}^{k} Pr(Q_i = m)$ also represents the average number of servers with $m+1$ requests. Consequently, $\pi_m = \sum_{i=1}^{k} Pr(Q_i = m)$.

Let us assume that $s_m = \sum_{i=m}^{\infty} \pi_i$ represents the steady state probability for queue of length at least m. Then equation 3.3 (below) illustrates the relation between steady state probabilities and the probability of having exactly m requests in the first k queues. Like the supermarket model, the steady state probability s_m is interpreted as the probability that a request will join a queue with at least m requests. According to [LRS16], we have

$$\sum_{i=1}^{k} Pr\{Q_i = m\} = f^{(n,k)}(s_m) - f^{(n,k)}(s_{m+1}) \tag{3.3}$$

where $f^{(n,k)}(x) = \sum_{l=1}^{k} \binom{n}{n-k+l}\binom{n-k+l-2}{l-1}(-1)^{l-1}x^{n-k+l}$.

From equation 3.3, we can see that a new request will join a queue of size m with probability $f^{(n,k)}(s_m) - f^{(n,k)}(s_{m+1})$. Similarly, a departure will occur with probability $s_{m+1} - s_{m+2}$. Due to having a stable system, for each arrival request there should be a departure. For this reason, there should be a balance equation between arrivals and departures, that is

$$\lambda(f^{(n,k)}(s_m) - f^{(n,k)}(s_{m+1})) = k(s_{m+1} - s_{m+2}) \ for \ m = 0, 1, 2, \ldots. \tag{3.4}$$

Under an (n, k) MDS code, the steady state queue length distribution s_m should satisfy equation 3.4. Thus, one of the solutions for equation 3.4 is

$$\begin{cases} s_{m+1} = \dfrac{\lambda}{k} f^{(n,k)}(s_m) \quad for \ m \geq 0 \\ s_0 = 1. \end{cases} \tag{3.5}$$

3.2 Our Model

To the best of our knowledge, all previous works on queueing systems for file download have focused on methods for serving requests as fast as possible after they arrive at servers. However, we do not know how the system will change if a request is prevented from entering the system. For this reason, we describe a queuing system in which the combination of choosing n server for redundancy and request loss is observable.

We consider a set of L independent servers, each of which stores a very large number of different types of files. We let the large number of servers L exists in the system. Under redundancy, a file is stored in n distinct servers, thus, we have $\binom{L}{n}$ choices for where to store each file. we assume requests arrive in the system as a Poisson process with rate $\lambda < 1$, though in practice it is hard to define a specific distribution for arrival requests and service times. Additionally, we assume a helper is located in the system to receive all requests. Based on information about storage, the helper will send a request to the shortest queue with probability $p_m = \dfrac{1}{m+1}$ where m is the number of requests in the queue. In other words, upon arrival, a request will either join a queue of size m with probability $p_m = \dfrac{1}{m+1}$, or balk (leave the system) with probability $b_m = \dfrac{m}{m+1}$, where $p_m + b_m = 1$. Hence, it turns out that requests join the system with rate $\lambda_m = \dfrac{\lambda}{m+1}$ and will be served based on first come first serve discipline.

Various load balancing techniques, such as Join the Shortest Queue (JSQ),

or Redundant Request with Killing (RRk), have been introduced to specify how requests are able to enter queues. In our model, we apply the Join the Shortest Queue (JSQ) model as our load balancing scheme. Each server includes a queue of requests to download a file stored in the server. Under the JSQ scheme, after entering the system each request is transmitted to the shortest queue containing the appropriate file.

According to n redundancy, n copies of each file are stored in n distinct servers, chosen uniformly at random. Each request will join the system with arrival rate $\lambda_m = \frac{\lambda}{m+1}$. For instance, when $m = 0$, a request will join the line with probability 1 and will be served instantly. Based on our storage method, each request can join the queue for one of the n servers in which the file is stored. Following the JSQ scheme, after joining the system the request will be sent to the shortest queue among these n servers. We have assumed that the processors are homogeneous, that is the service rates are the same at each server. Service times are considered to be exponentially distributed with mean μ for the system. Moreover, because each file is stored entirely in a single server, the time to download the file would be 1 unit. Consequently, it is assumed that the service rate $\mu = 1$. Based on our assumptions about the general arrival rate $\lambda < 1$ and the service time rate $\mu = 1$, the traffic intensity $\rho = \frac{\lambda}{\mu} < 1$. Thus the expected number of requests in the system remains finite which results in having a stable system. Since a new arrival will enter a queue in accordance with the queue length, all states depend on each other. Therefore as mentioned previously, our system is Markovian, as

the current state of the process depends only on the previous state. Because the combination of redundancy and request loss considered in this thesis is an initial step in the analysis of computer systems, we have assumed that a request takes no time to get information about the queue length and make the decision to either balk or join.

Our purpose in this chapter is to demonstrate that in a system where redundancy is used in file storage, the average request loss can significantly influence the entire system. In order to analyze such a system, we first need to find steady state probabilities of queue length.

3.3 Steady-state results

In this section, we will derive a formula for the steady state queue length distribution under redundancy and balking. It is clear that the system is easy to analyze and understand when a file is stored in only one server, that is $n = 1$. For this reason, it is impossible to send a request to the shortest queue since there will be only one long queue for all requests to join the system. Therefore, in order to analyze the system in all situations, this section will be divided into two parts. The first part is to find steady state probabilities for a simple $M/M/1$ queue with balking. In the second part, steady state probabilities for a system with balking and redundancy for $n \geq 2$ will be determined.

As mentioned previously, our model is a Markovian process, since an

arriving request will join the system based on the number of existing requests in the system. Under the assumption that the general arrival rate λ is less than 1, a request will join a queue of size m with the arrival rate $\lambda_m = \frac{\lambda}{m+1} < 1$ for $m \geq 0$. Furthermore, since the service time rate is $\mu = 1$, the traffic intensity is less than 1, that is $\rho_m = \frac{\lambda_m}{\mu} < 1$. This implies that the system is stable. Therefore, we are able to find steady state probabilities. If we define π_m as the probability that a queue is of size m, then $s_m = \sum_{i=m}^{\infty} \pi_i$ represents the probability that a queue contains at least m requests. Also, the sequence $\{s_m\}_{m=0}^{\infty}$ is interpreted as the steady state probabilities of queue length. Clearly, in an empty system, the probability that there is no request in the system is 1, i.e. $s_0 = \sum_{m=0}^{\infty} \pi_m = 1$ and the probability that a queue has at least m requests is $s_m = 0$ for $m \geq 1$. If we do not consider redundancy, for which $n = 1$, then each server works independently and we will have L $M/M/1$ queueing systems. Systems with redundancy $n > 1$ are much more difficult to completely analyze, as all servers depend on each other.

3.3.1 Steady-State for $n = 1$

Owing to the need to protect files from damage, each file will be stored in more than one server. Although it is not reliable to use a single server to store a file, our model will be easier to analyze in this case. Since arrival requests join the system with probability, $p_m = \frac{1}{m+1}$, depending on the number of requests in the queue, and form a Poisson process, we will have

an $M/M/1$ queue with balking probability $b_m = \dfrac{m}{m+1}$.

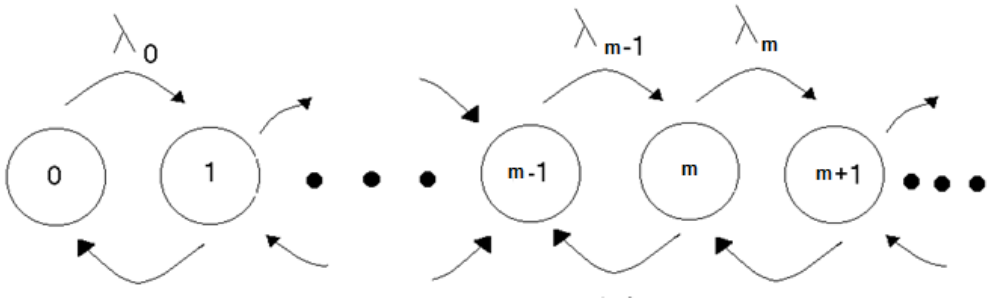


Figure 3.2: An M/M/1 queueing system with balking probability where the service rate is $\mu = 1$.

According to the general arrival rate λ and balking probability $b_m = \dfrac{m}{m+1}$, requests enter the queue with arrival rate $\lambda_m = \lambda(1 - b_m) = \dfrac{\lambda}{m+1}$, where m is the number of requests in the queue. Additionally, they will be served with rate $\mu = 1$. From figure 3.2, the balance equation among states is as follows:

$$\begin{cases} \lambda_{m-1} s_{m-1} + s_{m+1} = (1 + \lambda_m) s_m \quad for\ m \geq 1 \\ s_0 = 1. \end{cases} \tag{3.6}$$

or equivalently,

$$\begin{cases} \dfrac{\lambda}{m} s_{m-1} + s_{m+1} = (\dfrac{\lambda}{m+1} + 1) s_m \quad for\ m \geq 1 \\ s_0 = 1. \end{cases} \tag{3.7}$$

From figure 3.2, the balance equation between states 0 and 1 is

$$\lambda s_0 = s_1. \tag{3.8}$$

Since $s_0 = 1$, then $s_1 = \lambda$ and by substituting s_0 and s_1 in equation 3.7 we find the steady state probabilities for $m \geq 1$ to be:

$$\begin{cases} s_m = \lambda_{m-1} s_{m-1} \quad for\ m \geq 1 \\ s_0 = 1. \end{cases} \tag{3.9}$$

Since we have $s_0 = 1$ for an empty system, the remaining steady state probabilities can be derived from equation 3.9 by substitution. Hence, $s_m = \frac{\lambda^m}{m!}$ is the steady state probability of queue length for an $M/M/1$ queueing system with balking probability. Based on the definition of s_m, the sequence of steady state probabilities is decreasing which implies that $s_0 \geq s_1 \geq s_2 \geq \ldots$ and $\sum_{m=0}^{\infty} s_m < \infty$. The following figures indicate how the general arrival rate λ can affect the probability of one to four requests in the system.

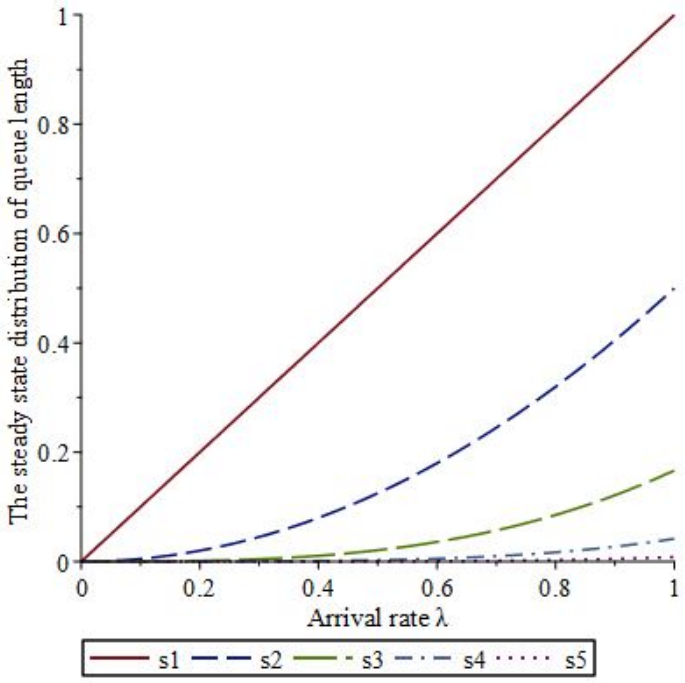


Figure 3.3: The impact of arrival rate on s_1, s_2, s_3, s_4 and s_5.

In figure 3.3, we see that the probability of having a particular number of requests in the system decreases as the number increases, even in high traffic periods, when λ approached 1. Furthermore, in light traffic areas, for which the arrival rate λ is close to zero, the probability of having more requests is close to zero. In such a case, since the queue length goes to zero, this probability decreases nearly to zero. For this reason, the time each request spends in the queue is its service time. In other words, the waiting time for each request is $\mu = 1$.

In figure 3.4, we can observe how the probability of the number of requests in the system changes during a heavy traffic period. As illustrated in this

figure, due to balking probability and requests loss, the probability of having m requests in the queue decreases when m increases.

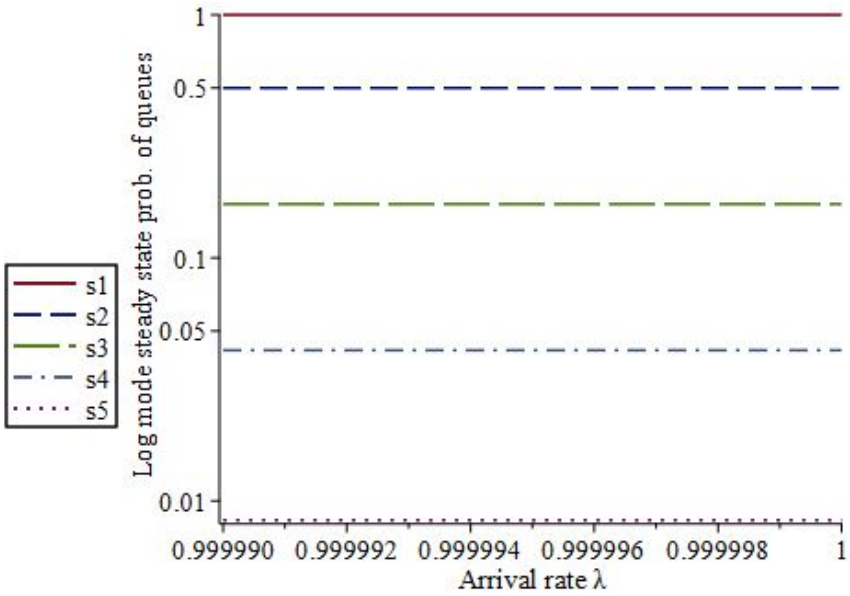


Figure 3.4: Heavy traffic period vs. the steady state distribution of queue length

3.3.2 Steady-states for $n \geq 2$

Redundancy has been shown to improve the waiting time in a queueing system, though it is not clear how many servers are needed to store files and to improve the performance of computer systems. In this section, it has been assumed that each file is stored in $n \geq 2$ servers. Our objective is to determine the steady state distribution of queue length when files are stored with redundancy and requests are able to leave without being served (balking).

A new request will join a queue of size m with probability $p_m = \dfrac{1}{m+1}$ and becomes the $(m+1)^{st}$ request in that queue. Let us assign π_m to be the probability that a queue has m requests. Under the assumptions we have made, our system is memoryless, because the future state depends only on the current state and not the past states. Furthermore, since states communicate with each other, our system is an irreducible Markov chain.

The load balancing policy is to route a request entering the system whichever of the n servers containing the file has the shortest queue length. We ignore reneging in the system. Thus, after joining the system (with the arrival rate $\dfrac{\lambda}{m+1}$), the request will wait until it is served.
Let us again assign s_m to be the probability that a queue has at least m requests. That is $s_m = \sum_{i=m}^{\infty} \pi_i$. Therefore, a new request will join the queue of exactly size m and becomes the $(m+1)^{st}$ request with probability $s_m^n - s_{m+1}^n$. Similarly, a departure will occur from a server of size $m+1$ with probability $s_{m+1} - s_{m+2}$. The following example illustrates how a queue of size m changes due to an arrival in a small queueing system. Because our model has various applications in real life, we will consider a call center with 3 operators.

Example 5. Suppose there are three servers in the system with queues of length 0, 1, or 2.

If a new customer goes to empty server, the customer will be served instantly and depart the system. Suppose now that the customer first makes two choices at random, then goes to the shortest queue. There are nine

options for the number of customers in the two queues that the new customer chooses: $(0,0)$, $(0,1)$, $(1,0)$, $(0,2)$, $(2,0)$, $(1,1)$, $(1,2)$, $(2,1)$, and $(2,2)$ with probabilities π_0^2, $\pi_0\pi_1$, $\pi_1\pi_0$, $\pi_0\pi_2$, $\pi_2\pi_0$, π_1^2, $\pi_1\pi_2$, $\pi_2\pi_1$, and π_2^2, respectively. The probability that the customer will join a queue of length at least 1 is the sum of these probabilities, for the choices with no empty servers:

$$\pi_1^2 + 2\pi_1\pi_2 + \pi_2^2 = (\pi_1 + \pi_2)^2 = s_1^2. \tag{3.10}$$

The probability that the customer will join a queue of length at least 2 is

$$\pi_2^2 = s_2^2. \tag{3.11}$$

Hence, the customer goes to a queue of length exactly 1 with probability

$$\pi_1^2 + 2\pi_1\pi_2 = s_1^2 - s_2^2. \tag{3.12}$$

■

From[LRS16], the summation of probabilities that the l^{th} shortest queue has exactly m requests is expressed as follows

$$\sum_{l=1}^{k} \binom{n}{n-k+l} \binom{n-k+l-2}{l-1} (-1)^{l-1} (s_m^{n-k+l} - s_{m+1}^{n-k+l}), \tag{3.13}$$

where each file is divided into n chunks and downloading any distinct set of k chunks would be sufficient to complete the service. However, in this thesis,

we have only considered redundancy for which $k = 1$. Therefore, having stored a file in n servers, through joining the shortest queue among these n queues, a request can be served and depart the system.

The steady state queue length distribution of a server under redundancy and balking is

$$\begin{cases} s_m = \lambda_{m-1} s_{m-1}^n \quad for\ m \geq 1 \\ s_0 = 1. \end{cases} \tag{3.14}$$

Let us explain the reason behind the equation 3.14. As we know, a new request will enter one of n queues of size m and become the $(m+1)^{st}$ waiting request with the probability $s_m^n - s_{m+1}^n$. Thus, if we add the probability of balking for this new arrival, we will have $\lambda_m s_m^n - \lambda_{m+1} s_{m+1}^n$. Since the service rate μ is 1, a departure from a queue of size $m+1$ will occur with probability $s_{m+1} - s_{m+2}$. Due to having a stable system, whenever there is an arrival, there should also be a departure. As a result, $\lambda_m s_m^n - \lambda_{m+1} s_{m+1}^n = s_{m+1} - s_{m+2}$ for $m = 0, 1, 2, \ldots$. Clearly, if we assign $s_{m+1} = \lambda_m s_m^n$, then equation 3.14 holds. In fact, one can obtain this equation by using 3.5 and taking $k = 1$ for arrival rate λ_m.

According to equation 3.14, the steady state probabilities for $n \geq 2$ is

$$\begin{cases} s_m = \dfrac{\lambda^{\frac{n^m - 1}{n-1}}}{\prod_{j=1}^{m} j^{n^{m-j}}} \quad for\ m \geq 1 \\ s_0 = 1 \end{cases} \tag{3.15}$$

where m is the number of request in the queue and n is the number of choices.

Proof. Since we know the initial point $s_0 = 1$, we can determine the steady state distribution by induction on m. From equation 3.14,

$$s_1 = \lambda_0 s_0 = \lambda.$$

$$s_2 = \lambda_1 s_1^n = \frac{\lambda}{2}(\lambda)^n = \frac{\lambda^{n+1}}{2} = \frac{\lambda^{\frac{n^2-1}{n-1}}}{2}.$$

$$s_3 = \lambda_2 s_2^n = \frac{\lambda}{3}\big(\frac{\lambda^{n+1}}{2}\big)^n = \frac{\lambda^{n^2+n+1}}{2^n.3} = \frac{\lambda^{\frac{n^3-1}{n-1}}}{2^n.3}.$$

Let us assume that $s_m = \dfrac{\lambda^{\frac{n^m-1}{n-1}}}{\prod_{j=1}^{m} j^{n^{m-j}}}$. We will prove that $s_{m+1} = \dfrac{\lambda^{\frac{n^{m+1}-1}{n-1}}}{\prod_{j=1}^{m+1} j^{n^{m+1-j}}}$. According to equation 3.14, we have $s_{m+1} = \lambda_m s_m^n$. Thus, by substituting s_m in the equation we get

$$s_{m+1} = \frac{\lambda}{m+1}\big(\frac{\lambda^{\frac{n^m-1}{n-1}}}{\prod_{j=1}^{m} j^{n^{m-j}}}\big)^n = \frac{\lambda}{m+1}\big(\frac{\lambda^{\frac{n^{m+1}-n}{n-1}}}{m^n.(m-1)^{n^2}\dots 3^{n^{m-2}}2^{n^{m-1}}}\big)$$

$$= \frac{\lambda^{\frac{n^{m+1}-n}{n-1}+1}}{(m+1).m^n.(m-1)^{n^2}\dots 3^{n^{m-2}}2^{n^{m-1}}}$$

Consequently, the steady state probability s_{m+1} is

$$s_{m+1} = \frac{\lambda^{\frac{n^{m+1}-1}{n-1}}}{\prod_{j=1}^{m+1} j^{n^{m+1-j}}}.$$

□

Equation 3.15 is the steady state queue length distribution when redundancy and balking occur in a large queueing system.

3.4 The Average Request Loss

As we know, impatience factors such as balking and reneging probabilities can significantly affect queueing systems. In order to control the cost of the system, we need to determine the factors that influence the cost, such as the average request loss. The average request loss is the sum of average balking rate and the average reneging rate. When reneging is ignored, the average request loss is defined to be the average balking rate.

A new request arrives with the rate λ and decides not to enter the queue with the balking probability $b_m = \frac{m}{m+1}$, where m is the number of waiting requests in the queue. Since our system behaves differently when redundancy is applied, we have determined the average request loss for the two cases $n = 1$ and $n \geq 2$. Equation 3.16 illustrates the average balking rate for a computer system in which only one server is used to store each file. We proved that the steady state probability of queue length where m requests wait in the queue is $s_m = \frac{\lambda^m}{m!}$. The average balking rate is given by

$$B_{n=1} = \sum_{m=1}^{\infty} \lambda b_m s_m = \sum_{m=1}^{\infty} \frac{m\lambda^{m+1}}{(m+1)!}. \tag{3.16}$$

It is clear that the sequence of balking probabilities $\{b_m\}_{m\geq 0}$ is increasing while the sequence of steady state probabilities $\{s_m\}_{m\geq 0}$ is decreasing. With a simple calculation to find the sum of the series we have

$$B_{n=1} = \sum_{m=0}^{\infty} \frac{\lambda^{m+1}(m+1)}{(m+1)!} - \sum_{m=0}^{\infty} \frac{\lambda^{m+1}}{(m+1)!}$$
$$= 1 + e^{\lambda}(\lambda - 1). \tag{3.17}$$

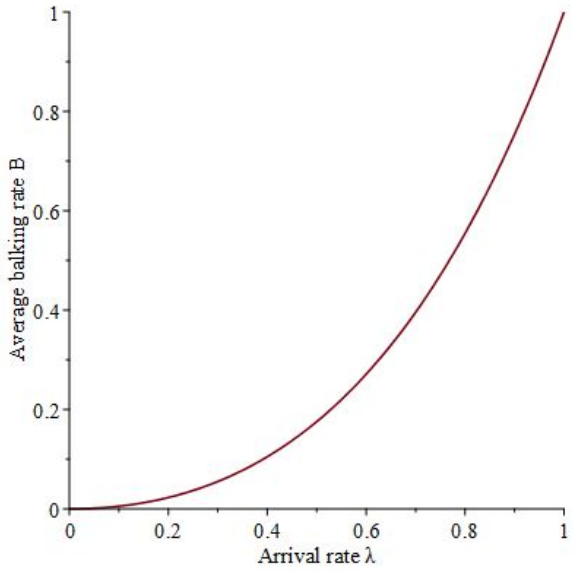


Figure 3.5: The impact of arrival rate on the average balking rate

As we know, the balking probability depends on the number of requests in the queue, which means that this probability increases when the number of requests waiting in the queue increases. Therefore, when the arrival rate λ is close to 1, the average balking rate increases exponentially. When the arrival rate λ is close to 0, the average balking rate or request loss is less than 0.2. From figure 3.5, we can observe that the results match our claim

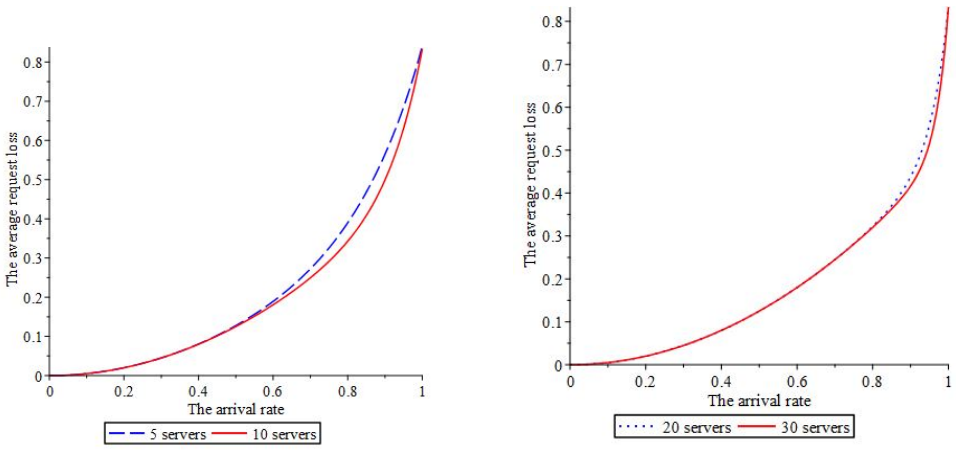
which is the negative impact of balking probability on the average request loss as well as the cost.

Now, we apply n redundancy in a system with balking probability, to determine the average request loss when $n \geq 2$. According to equation 3.15, the steady state probability of queue length would be $s_m = \frac{\lambda^{\frac{n^m-1}{n-1}}}{\prod_{j=1}^{m} j^{n^{m-j}}}$ for $n \geq 2$ and $m \geq 1$. Consequently, the average balking rate for this case is given by:

$$B_{n\geq 2} = \sum_{m=1}^{\infty} \lambda b_m s_m = \sum_{m=1}^{\infty} (\frac{m\lambda}{m+1}) \frac{\lambda^{\frac{n^m-1}{n-1}}}{\prod_{j=1}^{m} j^{n^{m-j}}}$$

$$= \sum_{m=0}^{\infty} \frac{m\lambda^{\frac{n^m+n-2}{n-1}}}{(m+1)\prod_{j=1}^{m} j^{n^{m-j}}}. \tag{3.18}$$

Clearly, a system including both redundancy and balking is more complicated to study. Hence, instead of theoretical results, we will analyze our system through numerical comparison. As we know, when we increase the storage space, download requests will be distributed among more servers. Thus, the average balking rate, or in general the average request loss, will decrease. This is shown in figure 3.6.



(a) The impact of the arrival rate on the average request loss where files are stored in 5 and 10 servers

(b) The impact of the arrival rate on the average request loss where files are stored in 20 and 30 servers

Figure 3.6: The impact of arrival rate on the average request loss

Figure 3.6 displays the average request loss for different queueing systems. As can be seen in this figure, the average request loss varies from 0 to approximately 0.86 when the arrival rate λ changes from 0 to 1. Furthermore, it turns out that when more servers are used to store a file the average request loss is smaller even though the impact of the balking probability is still considerable.

3.5 Chapter Summary

We applied the Join the Shortest Queue (JSQ) scheme in a queueing system where files were stored in n servers for $n \geq 1$. It has been proved that

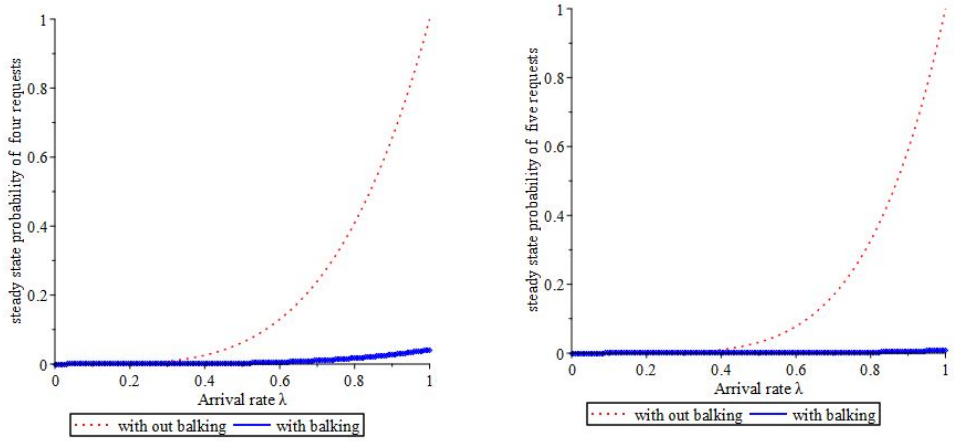
after a computer system has been in use for a long time, request loss will begin to occur. For this reason, we added the probability of balking b_m which depends on the number of requests waiting in a queue. In order to analyze the cost and the performance of our system, the steady state probabilities of queue length and the average request loss were determined. According to our model, it turned out that balking probability significantly affect computer systems.

Chapter 4

Numerical Results

In this chapter, we provide numerical results comparing the performance of a queueing system with redundancy with that of $M/M/1$ queueing system. In addition, since impatience factors affect the system, we will evaluate the impact of including a balking probability on a queueing system with redundancy and an $M/M/1$ queueing system.

4.1 The Performance of an $M/M/1$ Queueing System with Balking



(a) The impact of the general arrival rate on s_4 in a system with balking (solid) or without balking (dotted)

(b) The impact of the general arrival rate on s_5 in a system with balking (solid) or without balking (dotted)

Figure 4.1: The impact of the general arrival rate on s_4 and s_5 in an $M/M/1$ queueing system with or without balking

Figure 4.1 illustrates the comparison between two $M/M/1$ systems. In the first system, requests are not able to balk while in the second one, balking is possible. In figure 4.1(a), we see that when balking is not possible (dotted line), the steady state probability s_4 rises to 1 in high traffic periods (greater λ). However, when balking is possible (solid line), the steady state probability s_4 only reaches 0.05. The steady state probability s_5 (figure 4.1(b)) behaves similarly, but increases to 1 faster in the no balking case and remains smaller

in the balking case. In an $M/M/1$ system with an impatience factor, since the size of a queue can affect new arrivals, the steady state probabilities of queue length are small values.

Table 4.1: An $M/M/1$ queueing system with balking and arrival rate $\lambda = 0.1$

m	s_m
1	0.1
2	0.005
3	0.001666666667
4	$4.166666667 \times 10^{-6}$
5	$8.333333333 \times 10^{-8}$
6	$1.388888889 \times 10^{-9}$
7	$1.984126984 \times 10^{-11}$
8	$2.480158730 \times 10^{-13}$
9	$2.755731922 \times 10^{-15}$
10	$2.755731922 \times 10^{-17}$

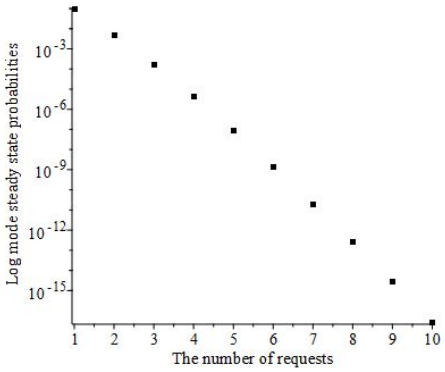


Figure 4.2: An $M/M/1$ queueing system in low traffic periods with balking and arrival rate $\lambda = 0.1$

The steady state probabilities give the information about the number of queues with m requests. In table 4.1, we have assume that each file is stored in only 1 server and the demand for joining the system is very low, i.e. the

arrival rate is close to zero. As we can observe, the probability of having m requests in the queue decreases when the number of requests increases. For example, for $\lambda = 0.1$, $s_6 = 1.388888889 \times 10^{-9}$ means the probability that a new request will join the queue of size 6 is $1.388888889 \times 10^{-9}$, however, in a system without balking, $s_6 = 10^{-6}$. Therefore, in low traffic periods, the probability that an arrival enters the system is highly influenced by balking probability.

Table 4.2: An $M/M/1$ queueing system with balking and arrival rate $\lambda = 0.1$

m	s_m
1	0.99
2	0.4900500000
3	0.1617165000
4	0.04002483375
5	0.007924917082
6	0.001307611319
7	0.0001849336008
8	$0.2288553310 \times 10^{-4}$
9	$2.517408641 \times 10^{-6}$
10	$2.492234554 \times 10^{-7}$

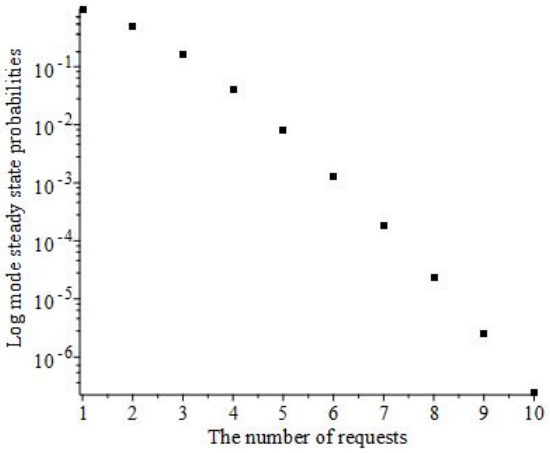
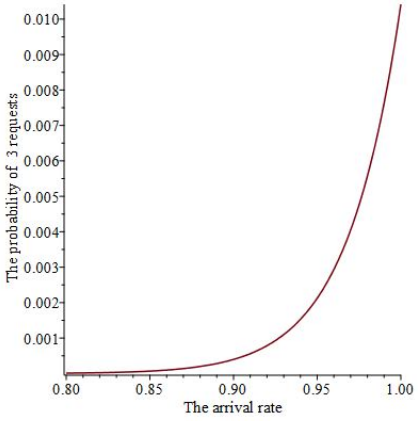


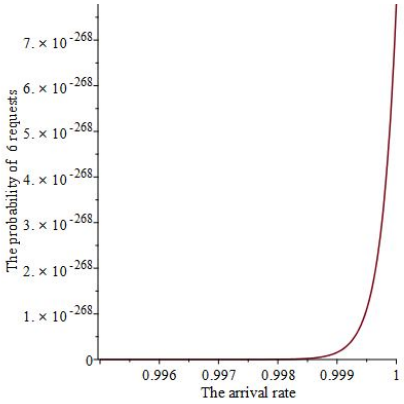
Figure 4.3: An $M/M/1$ queueing system with balking in a high traffic period

Despite of table 4.1, in table 4.2, we evaluate steady state probabilities for $\lambda = 0.99$. A comparison between tables 4.1 and 4.2 shows in high traffic periods the probability of having a queue of size m increases by the arrival rate λ. For instance, the probability of having a queue of size 6 is 0.001307611319 which is significantly higher than s_6 in low traffic periods.

4.2 The Performance of a Queueing System with n Redundancy and Balking



(a) The impact of the general arrival rate on s_3 in a system with redundancy



(b) The impact of the general arrival rate on s_6 in a system with redundancy

Figure 4.4: The impact of heavy traffic periods on s_3 and s_6 in a queueing system with redundancy $n = 5$

Figure 4.4 demonstrates how the steady state probability of having a queue of size 3 or 6 changes in a system where each file is stored in 5 servers. To be more accurate, from Chapter 3, for the steady state probability of queue length we have $s_m = \frac{\lambda^{\frac{n^m-1}{n-1}}}{\prod_{j=1}^{m} j^{n^{m-j}}}$. Thus, it is clear that when there is a small change in the number of either servers or download requests there is a significant change in the queue length steady state probability. According to figure 4.4, the numerical results validate our claim. It is clear that when

we have a busy system, the number of requests increases. However, due to the opportunity for a request not to enter the system, the probability of having requests in a queue of size 3 is very small when λ is close to zero. When the arrival rate reaches 1, the probability of having three requests increases to 0.01. As we can see, the probability of having 6 requests is very small since the number of requests in the queue has negative impact on this probability. Consequently, only when the arrival rate is very close to 1, does the probability of 6 requests in the queue increase. However, it only goes up to 7×10^{-268} which is still a very small value.

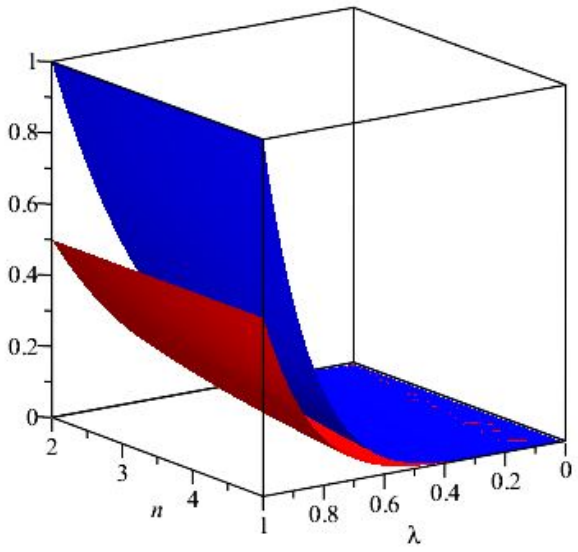


Figure 4.5: The probability of requests in the queue where λ varies from zero to one and storage space from two to five

Let us compare a queueing system consisting of both redundancy and

balking with a system that only includes redundancy. From our theoretical and numerical results, we can conclude that the performance of a system with redundancy and balking is much weaker since the number of requests decreases. Figure 4.5 is an example of such a system. In this example, the number of servers used for storage varies from 2 to 5 and the arrival rate λ varies from 0 to 1. According to this figure, the probability of having a queue of size 3 for $n = 2, 3, 4, 5$ and $0 \leq \lambda \leq 1$ with balking probability is smaller than the system without balking.

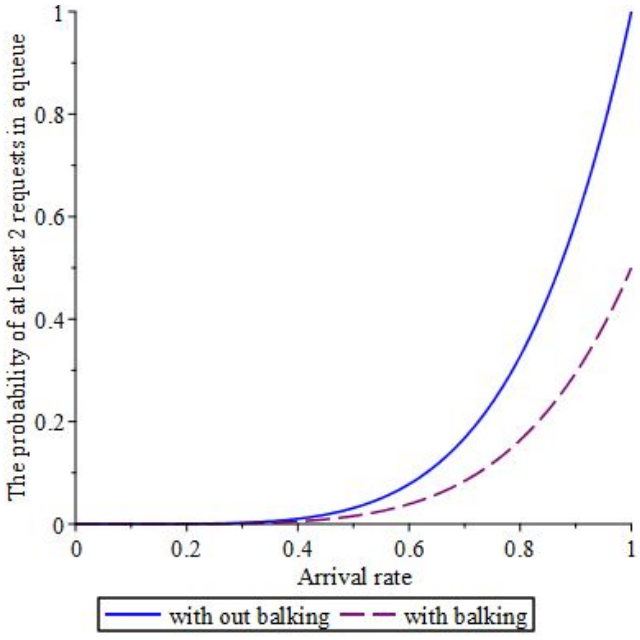


Figure 4.6: The probability of two requests without and with balking in a system that two servers used for storage.

Figure 4.6 is an example of having two waiting requests in a system in which each request has only two choices to download their intended file. In

the case with no balking (solid line), the probability of having at least two requests in a queue is higher than the case with balking (dashed line).

To obtain visual understanding of the impact of balking and redundancy on a computer system, the steady state probabilities s_m in low and high traffic periods for $n = 2$ is provided in tables 4.3 and 4.4.

Table 4.3: A queueing system with 2 redundancy and balking probability where the arrival rate $\lambda = 0.1$

m	s_m
1	0.1
2	0.5×10^{-3}
3	$8.333333333 \times 10^{-9}$
4	$1.736111111 \times 10^{-18}$
5	$6.028163580 \times 10^{-38}$
6	$6.056459357 \times 10^{-77}$
7	$5.240099993 \times 10^{-155}$
8	$3.432330993 \times 10^{-311}$
9	$1.308988450 \times 10^{-623}$
10	$1.713450762 \times 10^{-1248}$

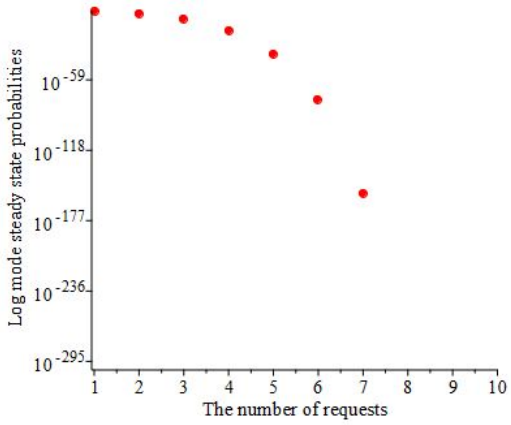


Figure 4.7: A queueing system with 2 redundancy and balking probability in a low traffic period

Table 4.4: A queueing system with 2-redundancy and balking where the arrival rate $\lambda = 0.99$

m	s_m
1	0.99
2	0.4851495
3	0.07767211232
4	$0.1493156866 \times 10^{-2}$
5	$4.414444503 \times 10^{-7}$
6	$3.215407844 \times 10^{-14}$
7	$1.462208447 \times 10^{-28}$
8	$2.645841258 \times 10^{-57}$
9	$7.700523559 \times 10^{-115}$
10	$5.870508245 \times 10^{-230}$

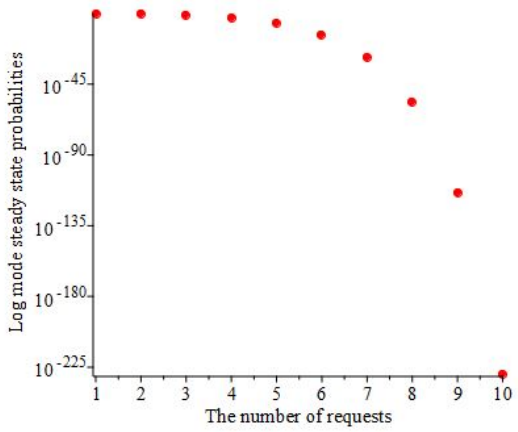


Figure 4.8: An $M/M/1$ queue with balking probability in a high traffic period

4.3 Chapter Summary

In this chapter, we analyzed the performance of a queueing system in which files are stored in various servers and download requests can balk. According to our observations, the request loss decreases the probability of having m requests in a queue. In addition to the balking probability, the number of servers used for storage also causes changes in the queue length. Even though redundancy itself can improve the performance of a queueing system, impatience factors such as balking strongly reduce the performance.

Chapter 5

Conclusion

We studied queueing systems under the case that a very large number of files are stored redundantly in a very large number of servers, and in which there is a probability that a request might not enter the system. That probability depends on the number of requests waiting in a queue. We considered Join the Shortest Queue as our load balancing scheme and found the steady state probabilities relevant to our model. In order to evaluate the cost and performance of the system, we determined the average request loss and proved that redundancy improves the performance of a system. Therefore, our objective was to analyze a queueing system in which the probability of balking was added. Using our balking probability and redundancy, we generated numerical results to illustrate how the number of requests in the system, or the number of servers used for storage, can affect the steady state probabilities of queue length. As a consequence, when a file download request is lost in a computer system, it can directly influence the cost. In order to reduce the cost and improve the performance, we need to change our computer systems so that balking probability has less of an influence. Since this thesis was the initial step in considering request loss, much more work can

be done in this area. For instance, moving from redundancy to coding, or applying another load balancing method, may help to both control the cost and decrease the waiting time. In particular, further studies can be carried out for the case in which files can stored according to Maximum Distance Separable codes.

Bibliography

[AJG63] CJ Ancker Jr and AV Gafarian. Some queuing problems with balking and reneging. i. *Operations Research*, 11(1):88–100, 1963. → pages 3

[AP09] Jesus R Artalejo and Vicent Pla. On the impact of customer balking, impatience and retrials in telecommunication systems. *Computers & Mathematics with Applications*, 57(2):217–229, 2009. → pages

[BK10] Jongho Bae and Sunggon Kim. The stationary workload of the g/m/1 queue with impatient customers. *Queueing systems*, 64(3):253–265, 2010. → pages 3

[Gal96] Robert G. Gallager. *Random Walks and Martingales*, pages 223–263. Springer US, Boston, MA, 1996. → pages 11

[GHBSW+17] Kristen Gardner, Mor Harchol-Balter, Alan Scheller-Wolf, Mark Velednitsky, and Samuel Zbarsky. Redundancy-d: The power of d choices for redundancy. *Operations Research*, 65(4):1078–1094, 2017. → pages 5

[GSTH08] Donald Gross, John F. Shortle, James M. Thompson, and Carl M. Harris. *Fundamentals of Queueing Theory*. Wiley-Interscience, New York, NY, USA, 4th edition, 2008. → pages 7

[GZD+15] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, and Esa Hyytia. Reducing latency via redundant requests: Exact analysis. *ACM SIGMETRICS Performance Evaluation Review*, 43(1):347–360, 2015. → pages 5

[Hai57] Frank A Haight. Queueing with balking. *Biometrika*, 44(3/4):360–369, 1957. → pages 3

[JDPF05] Sushant Jain, Michael Demmer, Rabin Patra, and Kevin Fall. Using redundancy to cope with failures in a delay tolerant network. *SIGCOMM Comput. Commun. Rev.*, 35(4):109–120, August 2005. → pages 5

[Kap11] Stella Kapodistria. The m/m/1 queue with synchronized abandonments. *Queueing Systems*, 68(1):79–109, May 2011. → pages 3

[KC09] Jau-Chuan Ke and Fu-Min Chang. Modified vacation policy for m/g/1 retrial queue with balking and feedback. *Computers & Industrial Engineering*, 57(1):433–443, 2009. → pages 3

[LK06] Liqiang Liu and Vidyadhar G Kulkarni. Explicit solutions for the steady state distributions in m/ph/1 queues with workload dependent balking. *Queueing Systems*, 52(4):251–260, 2006. → pages 3

[LK08] Liqiang Liu and Vidyadhar G. Kulkarni. Busy period analysis for m/ph/1 queues with workload dependent balking. *Queueing Systems*, 59(1):37, Jun 2008. → pages 3

[LM08] Macarena Lozano and Pilar Moreno. A discrete time single-server queue with balking: economic applications. *Applied Economics*, 40(6):735–748, 2008. → pages 4

[LRS16] B. Li, A. Ramamoorthy, and R. Srikant. Mean-field-analysis of coding versus replication in cloud storage systems. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016. → pages 2, 18, 20, 22, 32

[Mit01] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, Oct 2001. → pages 4, 18, 19

[Nat75] Bent Natvig. On a queuing model where potential customers

are discouraged by queue length. *Scandinavian Journal of Statistics*, 2(1):34–42, 1975. → pages 3

[Ros14] Sheldon Ross. - queueing theory. In Sheldon Ross, editor, *Introduction to Probability Models (Eleventh Edition)*, pages 481 – 558. Academic Press, Boston, eleventh edition edition, 2014. → pages 9, 10

[Saa61] Thomas L Saaty. *Elements of queueing theory: with applications*. McGraw-Hill New York, 1961. → pages 4

[SK82] Hari M Srivastava and BRK Kashyap. Special functions in queuing theory and related stochastic processes. *ACADEMIC PRESS, 111 FIFTH AVE., NEW YORK, NY 10003*, 1982. → pages

[SLR16] N. B. Shah, K. Lee, and K. Ramchandran. When do redundant requests reduce latency? *IEEE Transactions on Communications*, 64(2):715–722, Feb 2016. → pages 5

[SR67] S. Subba Rao. Queuing with balking and reneging in m—g—1 systems. *Metrika*, 12(1):173–188, Dec 1967. → pages 3

[VLDIK96] Nikita Vvedenskaya, R L. Dobrushin, and F I. Karpelevich. Queueing system with selection of the shortest of two queues: An asymptotic approach. 32, 01 1996. → pages

[vTvdV80] Martien van Tits and Henk van der Veeken. Simulation of a queueing problem with balking. *SIMULATION*, 35(3):88–93, 1980. → pages 3

[WC02] Kuo-Hsiung Wang and Ying-Chung Chang. Cost analysis of a finite m/m/r queueing system with balking, reneging, and server breakdowns. *Mathematical Methods of Operations Research*, 56(2):169–180, Nov 2002. → pages 4

[WLJ10] K. Wang, N. Li, and Z. Jiang. Queueing system with impatient customers: A review. In *Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics*, pages 82–87, July 2010. → pages 4

[YSK17] Lei Ying, R Srikant, and Xiaohan Kang. The power of slightly more than one sample in randomized load balancing. *Mathematics of Operations Research*, 2017. → pages