34340

TELETRAFFIC ENGINEERING AND
NETWORK PLANNING

ERLANG – C FORMULA

VILLY BÆK IVERSEN, PH.D
IGOR CEKICEVSKI, M.SC.E.E

Lyngby
May 15th, 2004
Contents

Abstract ................................................................................................................................................................. 3

1. Introduction .................................................................................................................................................. 4

2. Queuing Theory and Erlang - C Formula .................................................................................................... 6
   2.1 Derivation of Erlang – C Formula ........................................................................................................... 7
   2.2. Calculation of the Carried and Offered Traffic ...................................................................................... 9
   2.3 Various Representations of Erlang – C .................................................................................................. 10
      2.3.1 Relation between Erlang – C and Erlang –B ................................................................................... 10
      2.3.2 Tabular Representation ................................................................................................................ 11
      2.3.3 Graphical Interpretation ................................................................................................................ 11
      2.3.4 MATLAB Code for Erlang – C ....................................................................................................... 12
      2.3.5 Recursive Erlang – C formula and its inversion ............................................................................. 13
      2.3.6 MATLAB Code for Inverse Erlang – C ......................................................................................... 14
      2.3.7 Sander’s Formula .......................................................................................................................... 15

3. Queue Length Distribution (QLD) .............................................................................................................. 16
   3.1 MATLAB Code for PDF and QLD – M/M/n ............................................................................................ 17
   3.2 Mean Queue Length .............................................................................................................................. 18
      3.2.1 Mean Queue Length at an arbitrary point of time ........................................................................... 18
      3.2.2 Mean Queue Length, if the queue is greater than zero .................................................................. 19

4. Waiting Time Distribution (WTD) - M/M/n .............................................................................................. 21
   4.1 Waiting Time Distribution for M/M/n, FCFS .......................................................................................... 21
   4.2 MATLAB Code for PDF and WTD – M/M/n .......................................................................................... 23
   4.3 Waiting Time Distributions for other Queuing Disciplines .................................................................. 24
   4.4 Mean Waiting Time .............................................................................................................................. 25
      4.4.1 Mean Waiting Time - All customers ............................................................................................... 25
      4.4.2 Mean Waiting Time - Delayed customers ...................................................................................... 25

5. Improvement Functions for M/M/n ............................................................................................................ 26
   5.1 MATLAB Code for Improvement Function of M/M/n ........................................................................... 28

6. Finite-buffer Queuing System ..................................................................................................................... 29
   6.1. General considerations ......................................................................................................................... 29
      6.1.1 Single-server System ...................................................................................................................... 29
         6.1.1.1 M/M/1/k .................................................................................................................................. 29
         6.1.1.1.1 MATLAB Code for Blocking Probability in M/M/1/k ........................................................... 30
         6.1.1.2 M/G/1/k .................................................................................................................................. 31
         6.1.1.3 M/D/1/k .................................................................................................................................. 32
      6.2. Multiple-server System ....................................................................................................................... 32
         6.2.1 Steady State Probabilities ............................................................................................................. 32
         6.2.2 Blocking Probability – M/M/n/k .................................................................................................... 33
            6.2.2.1 MATLAB Code for the Blocking Probability in M/M/n/k ....................................................... 33
         6.2.3 Queue Length Distribution – M/M/n/k ....................................................................................... 35
            6.2.3.1. MATLAB Code for PDF and QLD – M/M/n/k ................................................................. 35
6.2.3.2 Mean Queue Length – M/M/n/k................................................................. 37
6.2.4 Waiting Time Distribution – M/M/n/k......................................................... 38
6.2.4.1 Mean Waiting Time – M/M/n/k................................................................. 39

7. Cost Function........................................................................................................... 40

8. Priority of customers.............................................................................................. 41

8.1 Analysis of M/M/n/K Queues with Priorities....................................................... 42
8.1.1 M/M//-/- Queue with Pre-emptive Priority.................................................. 42
8.1.1.1 Mean Waiting Time for Pre-emptive-resume Queuing Discipline.............. 45
8.1.1.2 Mean Queue Length for Pre-emptive-resume Queuing Discipline.............. 46
8.1.2 M/M//-/- Queue with Non-pre-emptive Priority......................................... 46
8.1.2.1 Mean Waiting Time for Non-pre-emptive Queuing Discipline (M/M/n).... 48
8.1.2.2 Mean Queue Length for non-pre-emptive Queuing Discipline (M/M/n).... 49

8.2 Concrete example for a Non-pre-emptive Priority System.................................. 49
8.2.1 Disks in SAN................................................................................................... 50
8.2.1.1 Read and Write Requests......................................................................... 50
8.2.1.2 Disk Response Time.................................................................................. 51

8.3 Analysis Model..................................................................................................... 51
8.3.1 Basic Assumptions.......................................................................................... 51
8.3.2 Finite Queuing Model...................................................................................... 52
8.3.2.1 State Probabilities...................................................................................... 52
8.3.2.2 Performance Measures............................................................................. 52
8.3.2.3 Numerical Results...................................................................................... 53

9. Systems with Limited Waiting Time...................................................................... 54
9.1 Exponentially distributed Time-out Interval...................................................... 54
9.2 Systems with Constant Time-out Interval....................................................... 54
9.2.2 Queue Length Distribution – Systems with Constant Time-out Interval...... 55
9.2.2.1 Mean Queue Length - Systems with Constant Time-out Interval............ 56
9.2.3. Waiting Time Distribution - Systems with Constant Time-out Interval..... 56
9.2.3.1 MATLAB Code for PDF and WTD of Systems with Constant Time-out Interval.................................................. 58
9.2.3.2. Mean Waiting Time - Systems with Constant Time-out Interval........... 59

10. Concluding remarks.............................................................................................. 60

Appendix A – Gamma Function.................................................................................. 61

Appendix B – Error Function..................................................................................... 62

Glossary of Acronyms............................................................................................... 63

References.................................................................................................................. 64
Abstract

This paper deals with the queuing systems in different configurations. As the basic tool dealing with problems, related to these systems, we put our emphasis to Erlang - C formula. It denotes by what probability an arbitrary customer arriving to the queuing system would experience waiting i.e delay before service is performed. Since its pure form is not adequate in computer simulation processes we have derived it in terms of Erlang - B formula. There are also other forms that we have considered, such as Erlang - C recursive formula and Erlang - C inverse formula.

The remaining part of the paper concentrates on waiting time and length distributions in queuing systems with infinite and finite buffer. Finite queues are often hard or impossible to analyse in an exact analytical way. In such cases, the often-preferred solution is to approximate the loss probability in a finite queue by the probability that the corresponding infinite queue (with the same service time) exceeds the finite queue length. We confirm the theoretical results with the appropriate MATLAB codes, together with corresponding graphical representations.
1. Introduction

Queuing theory is the mathematical study of waiting. It is concerned with the mathematical modeling and analysis of systems that provide service to random demand. Queuing theory has played an important role in modeling communications networks. The generic procedure is to model a system as a stochastic queuing system or a random process. A (single) queuing system has three components: customers, one or multiple servers, and a single queue, and involves two processes i.e. the arrival process and the service process. Customers arrive at the system and get service from the server(s). If all the servers are busy in serving other customers, a newly arriving customer will wait in the queue until a server becomes idle, hence the name queuing systems. In this report, the queue refers to a queuing system as well as to a buffer. The key random processes used in queuing theory are Markov processes.

There are several everyday examples that can be described as queuing systems, such as bank-teller service, computer systems, manufacturing systems, maintenance systems, communications systems, and etc.

A queuing system is characterized by the following three components:

- **Arrival process**
- **Service mechanism**
- **Queue discipline**

**Arrival processes.** Arrival processes, such as telephone calls arriving to an exchange are described mathematically as stochastic point processes. We can define two basic arrival processes:

- **Point process.** It is the case when any two arrivals must be distinguished from each other. Information concerning the single arrival (e.g. service time, number of customers) is ignored. Such information can only be used to determine whether an arrival belongs to the process or not. For telephone calls this may be realized by choosing sufficient detailed time scale. We always assume Poisson arrival process.

- **Multiple arrivals.** It is the case when at the input of the server there are two or more arrivals at the same time. This is more complex situation then the previous one and therefore it is no considered in this report.

**Service mechanism.** There are many service mechanisms. In order to describe them we need to know the number of servers in the system and the average duration of the service time. The number of servers can vary from one to infinity. On the other hand the distribution of the service time can be divided into two main groups:

- **General**
- **Deterministic**
Queue Discipline. Discipline of a queuing system defines the rule that is used by a server when it chooses the next customer from the queue (if any) when the service of the current customer is completed. Commonly used queue disciplines are:

- **FIFO.** Customers are served on a *First – In – First - Out* basis
- **LIFO.** Customers are served in *Last – In – First - Out* manner
- **Priority.** Customers are served in order of their importance on the basis of their service requirements.

Erlang-C is most commonly used to calculate how long calls will have to wait before the connection to the destination is established. This model assumes that all blocked calls stay in the system until they are handled. Also, it can be applied to the design of call-center staffing arrangements where, if calls cannot be immediately answered, they enter a queue.

Erlang-C formula is only valid for Poisson arrival process and an exponential holding time distribution. It denotes what the possibility is that an arbitrary customer arriving to the queuing system would experience waiting i.e. delay before service is performed. Erlang–C is much similar to Erlang–B. However the major difference is that Erlang–B applies to loss systems.
2. Queuing Theory and Erlang - C formula

A queue is a waiting line (like customers waiting at a supermarket checkout counter) and queuing theory is the mathematical theory of waiting lines. More generally, queuing theory is concerned with the mathematical modelling and analysis of systems that provide service to random demands. A queuing model is an abstract description of such a system. Typically, a queuing model represents:

- The physical configuration of the system, by specifying the number and arrangement of the servers, that provide service to the customers
- The stochastic (i.e probabilistic or statistical) nature of the demands, by specifying the variability in the arrival process and in the service process.

For instance, in computer communications, a communication channel might be a server, and messages could represent customers. The (random) time at which messages request the use of the channel would be the arrival process, and the (random) duration of service time for the messages while being transmitted would constitute the service process.

Queuing theory was born in the early 1900s with the work of A. K. Erlang at the Copenhagen Telephone Company, who derived several important formulae for teletraffic engineering that today bear his name. The range of applications has grown and today they cover not only telecommunications and computer science, but also manufacturing, air traffic control, military logistics, and many other areas that involve service systems whose demands are random.

The essence of queuing theory is that it takes into account the randomness of the arrival process and the randomness of the service process. The most common assumption about the arrival process is that the customer arrival epochs follow a Poisson process. One of the most important queuing models is the Erlang loss model. It assumes that the arrivals follow a Poisson process, and that the blocked customers (those who find all servers busy) are cleared i.e. they are denied entry into the system and therefore the blocked customers are lost. The fraction of arriving customers who find all servers busy (the probability of blocking, or loss probability) is given by the famous Erlang loss (or Erlang - B) formula.

The Erlang delay model (also called M/M/n in queuing theory) is similar to the Erlang loss model, except that now it is assumed that the blocked customers will wait in a queue as long as it is necessary for a server to become available. In this model, the probability of blocking (the fraction of customers who will find all $n$ servers busy and must wait in the queue) is given by the famous Erlang delay (or Erlang C) formula.
2.1 Derivation of Erlang - C formula

In this section we consider a traffic that enters a system with \( n \) identical servers. Also we assume full accessibility and an infinite number of waiting positions in the buffer. If a customer arrives and sees all \( n \) servers busy then he joins a queue and waits until one server becomes idle. There is no option for a customer to be in the queue when a server is idle. This characteristic of the system is called full accessibility.

We shall consider the case of Poisson arrival process (an infinite number of sources) and exponentially distributed service times (PCT-I). This is the most important queuing system, called Erlang delay system. This system is denoted as \( M/M/n \) where the carried traffic is equal to the offered traffic since no customers are blocked.

Here, \( M \) stands for Poisson arrival process, whereas \( M \) for exponential service times and \( n \) designates the number of servers. Initially we shall assume a buffer with an infinite number of waiting positions. The probability of a positive waiting time, mean queue lengths, mean-waiting times, carried traffic per channel, and improvement functions will be dealt later in this report.

![Figure 2.1 State transition diagram of the M/M/n Delay System having \( n \) servers and an unlimited number of waiting positions.](image)

The state of the system is defined as the total number of customers in the system (either being served or waiting in queue). We are interested in the steady state probabilities of the system. The procedure begins with setting up the state transition diagram shown in Fig. 2.1. Assuming statistical equilibrium, the cut equations become:

\[
\lambda \cdot p(0) = \mu \cdot p(1) \quad (2.1)
\]
\[
\lambda \cdot p(0) = 2\mu \cdot p(1) \quad (2.2)
\]
\[
\vdots
\]
\[
\lambda \cdot p(i) = (i+1)\mu \cdot p(i+1) \quad (2.3)
\]
\[
\vdots
\]
\[
\lambda \cdot p(n-1) = n\mu \cdot p(n) \quad (2.4)
\]
\[
\lambda \cdot p(n) = n\mu \cdot p(n+1) \quad (2.5)
\]
\[
\vdots
\]
\[
\lambda \cdot p(n+j) = n\mu \cdot p(n+j+1) \quad (2.6)
\]

\( \lambda \) represents the arrival intensity and \( 1/\mu \) is the mean service time. As \( A = \lambda /\mu \) is the offered traffic, we get:

\[
p(i) = \begin{cases} 
  p(0) \left( \frac{A}{n} \right)^i, & 0 \leq i \leq n \\
  p(n) \left( \frac{A}{n} \right)^{i-n} = p(0) \cdot \frac{A^i}{n^i n^{r-n}}, & i \geq n
\end{cases} \quad (2.7)
\]

By normalization of the state probabilities we obtain \( p(0) \):

\[
\sum_{i=0}^{\infty} p(i) = 1 \quad (2.8)
\]

\[
p(0) \left[ 1 + \frac{A}{1} + \frac{A^2}{2!} + \ldots + \frac{A^n}{n!} \right] = 1 \quad (2.9)
\]

Realizing the geometric progression with quotient \( A/n \), the normalization condition can only be fulfilled for:

\[
A < n \quad (2.10)
\]

Statistical equilibrium is only obtained for \( A < n \). Otherwise, the queue will continue to increase towards infinity. We obtain:

\[
p(0) = \frac{1}{\sum_{i=0}^{n-1} \frac{A^i}{i!} + \frac{A^n}{n!} \frac{n}{n-A}}, \quad A \leq n \quad (2.11)
\]

Equations (2.7) and (2.11) produce the steady state probabilities.

\[
p(i) = \begin{cases} 
  \frac{A^i}{i!}, & 0 \leq i \leq n \\
  \frac{A^i}{n! n^{r-n}}, & i \geq n
\end{cases} \quad , A \leq n \quad (2.12)
\]

In the case when the Poisson arrival process is independent of the state of the system, the probability that an arbitrary arriving customer has to wait in the queue is equal to the proportion of
time all servers are occupied (PASTA–property: Poisson Arrivals See Time Average). The waiting time is a random variable denoted by \( W \). For an arbitrary arriving customer we have:

\[
E_{2,n}(A) = p[W > 0]
\]  

(2.13)

\[
E_{2,n}(A) = \frac{\sum_{i=0}^{n} \lambda p(i)}{\sum_{i=0}^{n} \lambda p(i)} = \sum_{i=0}^{n} p(i)
\]  

(2.14)

\[
E_{2,n}(A) = p(n) \frac{n}{n - A}
\]  

(2.15)

Finally, we can formulate the Erlang - C formula

\[
E_{2,n} = \frac{A^n \frac{n}{n!} \frac{n}{n - A}}{1 + A^{1} + A^{2} + \ldots + A^{n-1} + \frac{A^n}{(n-1)!} + \frac{A^n}{n!} \frac{n}{n - A}} , \quad A < n
\]  

(2.16)

This delay probability depends only upon \( A \), the product of \( \lambda \) and \( 1/\mu \), and not upon the parameters individually. According to the Erlang-C formula, the probability that a customer is served immediately upon arrival to the system is given by:

\[
S_n = 1 - E_{2,n}(A)
\]  

(2.17)

\[2.2\text{ Calculation of the Carried and the Offered traffic}\]

Since we are dealing with delay and lossless systems, it is intuitively that the carried and the offered traffic should coincide, as no customers are rejected and the arrival process is a Poisson process. The proof is given below. The carried traffic is given by

\[
Y = \sum_{i=1}^{n} i \cdot p(i) + \sum_{i=n+1}^{\infty} n \cdot p(i)
\]  

(2.18)

Using the cut equations, we can reformulate (2.18) to obtain:

\[
Y = \sum_{i=1}^{n} \frac{\lambda}{\mu} \cdot p(i - 1) + \sum_{i=n+1}^{\infty} \frac{\lambda}{\mu} \cdot p(i - 1) = \frac{\lambda}{\mu} = A
\]  

(2.19)
2.3 Various Representations of Erlang - C

Erlang – C formula could be expressed in a number of different ways. This is crucial since it the direct evaluation of (2.16) seems to be very difficult. Some of these representations are analyzed in more detail below.

2.3.1 Relation between Erlang – C and Erlang –B

Knowing that Erlang – B formula (denoted by $E_{1,n}(A)$) is given by

$$E_{1,n}(A) = \frac{A^n}{n!} \left(1 + \frac{A}{1} + \frac{A^2}{2!} + \ldots + \frac{A^{n-1}}{(n-1)!} + \frac{A^n}{n!}\right)$$  \hspace{1cm} (2.20)

we can derive Erlang - C formula in order so that it is expressed in terms of Erlang – B formula. The whole procedure is explained below.

$$E_{2,n}(A) = \frac{A^n}{n!} \left(\sum_{i=0}^{n-1} \frac{A^i}{i!} \frac{n}{n-A}\right) = \frac{A^n}{n!} \left(\frac{n}{n-A} \sum_{i=0}^{n-1} \frac{A^i}{i!} \frac{1}{n-A} - \frac{A^n}{n!} \sum_{i=0}^{n-1} \frac{A^i}{i!}\right)$$

$$E_{2,n}(A) = \frac{A^n}{n!} \left(\frac{1}{1 + A + \ldots + \frac{A^n}{n!}}\right) = \frac{n \cdot E_{1,n}(A)}{n-A(1-E_{1,n}(A))} = \frac{E_{1,n}(A)}{1-\frac{A\cdot[1-E_{1,n}(A)]}{n}}$$ \hspace{1cm} (2.21)

$$E_{2,n}(A) = \frac{n \cdot E_{1,n}(A)}{n-A[1-E_{1,n}(A)]} = \frac{E_{1,n}(A)}{1-\frac{A\cdot[1-E_{1,n}(A)]}{n}}$$, \hspace{0.5cm} A<n \hspace{1cm} (2.22)

It is easily noticeable that:

$$E_{2,n}(A) > E_{1,n}(A)$$ \hspace{1cm} (2.23)

as the term $A[1 - E_{1,n}(A)] / n$ is the average carried traffic per channel in the corresponding loss system. For $A \geq n$, we have $E_{2,n}(A) = 1$ since it is a probability and all customers are delayed.
2.3.2 Tabular Representation

Erlang - C formula has been tabulated in Moe’s Principle [2]. The Table 2.1 below denotes the value of the Erlang-C as a function of the offered traffic and the total number of servers.

<table>
<thead>
<tr>
<th>Number of Servers</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.2500</td>
<td>0.0278</td>
<td>0.0022</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.50</td>
<td>0.5000</td>
<td>0.1000</td>
<td>0.0152</td>
<td>0.0018</td>
<td>0.0002</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>0.75</td>
<td>0.7500</td>
<td>0.2045</td>
<td>0.0441</td>
<td>0.0077</td>
<td>0.0011</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.00</td>
<td>1.0000</td>
<td>0.3333</td>
<td>0.0089</td>
<td>0.0204</td>
<td>0.0038</td>
<td>0.0006</td>
<td>0.0001</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.25</td>
<td>1.2500</td>
<td>0.4808</td>
<td>0.1555</td>
<td>0.0422</td>
<td>0.0097</td>
<td>0.0019</td>
<td>0.0003</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.50</td>
<td>1.5000</td>
<td>0.6429</td>
<td>0.2368</td>
<td>0.0746</td>
<td>0.0201</td>
<td>0.0047</td>
<td>0.0010</td>
<td>0.0002</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>1.75</td>
<td>1.7500</td>
<td>0.8167</td>
<td>0.3337</td>
<td>0.1184</td>
<td>0.0364</td>
<td>0.0098</td>
<td>0.0023</td>
<td>0.0005</td>
<td>0.0001</td>
<td>0.0000</td>
</tr>
<tr>
<td>2.00</td>
<td>2.0000</td>
<td>1.0000</td>
<td>0.4444</td>
<td>0.1739</td>
<td>0.0597</td>
<td>0.0180</td>
<td>0.0048</td>
<td>0.0011</td>
<td>0.0002</td>
<td>0.0000</td>
</tr>
<tr>
<td>2.25</td>
<td>2.2500</td>
<td>1.0000</td>
<td>0.5678</td>
<td>0.2412</td>
<td>0.0998</td>
<td>0.0363</td>
<td>0.0090</td>
<td>0.0024</td>
<td>0.0006</td>
<td>0.0001</td>
</tr>
<tr>
<td>2.50</td>
<td>2.5000</td>
<td>1.0000</td>
<td>0.7022</td>
<td>0.3199</td>
<td>0.1304</td>
<td>0.0474</td>
<td>0.0154</td>
<td>0.0045</td>
<td>0.0012</td>
<td>0.0003</td>
</tr>
<tr>
<td>2.75</td>
<td>2.7500</td>
<td>1.0000</td>
<td>0.8467</td>
<td>0.4095</td>
<td>0.1788</td>
<td>0.0702</td>
<td>0.0248</td>
<td>0.0079</td>
<td>0.0023</td>
<td>0.0006</td>
</tr>
<tr>
<td>3.00</td>
<td>3.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.5094</td>
<td>0.2362</td>
<td>0.0991</td>
<td>0.0376</td>
<td>0.0129</td>
<td>0.0040</td>
<td>0.0012</td>
</tr>
<tr>
<td>3.25</td>
<td>3.2500</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.6191</td>
<td>0.3026</td>
<td>0.1348</td>
<td>0.0546</td>
<td>0.0201</td>
<td>0.0068</td>
<td>0.0021</td>
</tr>
<tr>
<td>3.50</td>
<td>3.5000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.7379</td>
<td>0.3778</td>
<td>0.1775</td>
<td>0.0702</td>
<td>0.0299</td>
<td>0.0107</td>
<td>0.0035</td>
</tr>
<tr>
<td>3.75</td>
<td>3.7500</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.8650</td>
<td>0.4618</td>
<td>0.2274</td>
<td>0.1029</td>
<td>0.0427</td>
<td>0.0163</td>
<td>0.0057</td>
</tr>
<tr>
<td>4.00</td>
<td>4.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.5541</td>
<td>0.2848</td>
<td>0.1351</td>
<td>0.0590</td>
<td>0.0238</td>
<td>0.0088</td>
</tr>
<tr>
<td>4.25</td>
<td>4.2500</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.6545</td>
<td>0.3495</td>
<td>0.1731</td>
<td>0.0793</td>
<td>0.0336</td>
<td>0.0131</td>
</tr>
<tr>
<td>4.50</td>
<td>4.5000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.7025</td>
<td>0.4217</td>
<td>0.2172</td>
<td>0.1039</td>
<td>0.0460</td>
</tr>
<tr>
<td>4.75</td>
<td>4.7500</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.8778</td>
<td>0.5010</td>
<td>0.2675</td>
<td>0.1331</td>
<td>0.0616</td>
</tr>
<tr>
<td>5.00</td>
<td>5.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.5875</td>
<td>0.3241</td>
<td>0.1673</td>
<td>0.0805</td>
</tr>
</tbody>
</table>

Table 2.1. Tabular representation of Erlang - C

2.3.3 Graphical Interpretation

Erlang – C can be also represented graphically. It is shown on Fig. 2.2.
2.3.4 MATLAB Code for Erlang C

```matlab
function B = erlangb (n, A)
% This function computes the Erlang B probability that a system with n
% servers, no waiting line, Poisson arrival rate lambda, service rate
% (per server) mu, and intensity A = lambda / mu will have all servers busy.
% The probability is B=(A^m / m!) / (sum (A^k / k!), k=0...m)
% The recurrence is B (0, A) = 1.
% B (n, A)=(A*B (n - 1, A) / n) / (1+A*B (n - 1, A) / n)
if ( (floor (n) ~= n) | (n < 1) )
    warning ('n is not a positive integer');
    A=NaN;
    return;
end;
if (A < 0.0)
    warning ('A is negative!');
    A=NaN;
    return;
end;
B=1;
for k=1:n,
    B=( (A*B) / k) / (1 + A*B / k);
end;
```

```matlab
function C=erlangc (n, A)
% This function computes the Erlang C probability that a system with n
% servers, infinite waiting line, Poisson arrival rate lambda, service rate
% (per server) mu, and intensity A=lambda / mu will have all servers busy.
% It uses the formula C (n, A)=n*B (n, A) / (n-A*(1 - B (n, A))
if ( (floor (n) ~= n) | (n < 1) )
    warning ('n is not a positive integer');
    A=NaN;
    return;
end;
if (A < 0.0)
    warning ('A is negative!');
    A=NaN;
    return;
end;
B=erlangb (n, A);
C=n*B / (n - A*(1 - B));
```

2.3.5 Recursive Erlang-C Formula and Its Inversion
Another expression for Erlang – C formula can be obtained in terms of its recursion. The appropriate derivation begins with its inverse representation \((2.27)\). This form is also known as Sander’s formula, which will be also derived in section 2.3.7.

\[
E_{2,n}(A) = \frac{n \cdot E_{1,n}(A)}{n - A \cdot (1 - E_{1,n}(A))}
\]  
\[(2.24)\]

\[
E_{1,n}(A) = \frac{(A - n) \cdot E_{2,n}(A)}{A \cdot E_{2,n}(A) - n}
\]  
\[(2.25)\]

\[
E_{1,n-1}(A) = \frac{(A - (n - 1)) \cdot E_{2,n-1}(A)}{A \cdot E_{2,n-1}(A) - (n - 1)}
\]  
\[(2.26)\]

\[
\frac{1}{E_{2,n}(A)} = \frac{1}{E_{1,n}(A)} - \frac{1}{E_{1,n-1}(A)}
\]  
\[(2.27)\]

\[
\frac{1}{E_{2,n}(A)} = \frac{A \cdot E_{2,n}(A) - n}{(A - n) \cdot E_{2,n}(A)} - \frac{A \cdot E_{2,n-1}(A) - (n - 1)}{(A - (n - 1)) \cdot E_{2,n-1}(A)}
\]  
\[(2.28)\]

\[
\frac{1}{E_{2,n}(A)} = \frac{E_{2,n}(A) \cdot [A \cdot ((A - (n - 1)) \cdot E_{2,n-1}(A) - (A - n)) \cdot (A - n)] - n(A - (n - 1)) \cdot E_{2,n-1}(A)}{[(A - n) \cdot E_{2,n}(A)] \cdot [(A - (n - 1)) \cdot E_{2,n-1}(A)]}
\]  
\[(2.29)\]

\[
E_{2,n}(A) = \frac{(A - n) \cdot ((A - (n - 1)) \cdot E_{2,n-1}(A) + n(A - (n - 1)) \cdot E_{2,n-1}(A))}{A \cdot ((A - (n - 1)) \cdot E_{2,n-1}(A) - (A - n)) \cdot [A \cdot E_{2,n-1}(A) - (n - 1)]}
\]  
\[(2.31)\]

\[
E_{2,n}(A) = \frac{A \cdot ((A - (n - 1)) \cdot E_{2,n-1}(A))}{A \cdot ((A - (n - 1)) \cdot E_{2,n-1}(A) - (A - n)) \cdot [A \cdot E_{2,n-1}(A) - (n - 1)]}
\]  
\[(2.32)\]

\[
E_{2,n}(A) = \frac{A \cdot ((A - (n - 1)) \cdot E_{2,n-1}(A))}{A \cdot ((A - (n - 1)) \cdot E_{2,n-1}(A) - (A - n)) \cdot [A \cdot E_{2,n-1}(A) - (n - 1)]}
\]  
\[(2.33)\]

For \(n \gg 1\), we obtain the following simplified expression:
2.3.6 MATLAB Code for Inverse Erlang - C

function InvB=erlangbinnv (n, A)
% This function computes the Inverse Erlang B probability that a system with n
% servers, no waiting line, Poisson arrival rate lambda, service rate
% (per server) mu, and intensity A=lambda / mu will have all servers busy.
% The inverse probability is InvB = 1 + (1 / A)*InvB (n - 1, A)
% The recurrence is InvB (0, A) = 1
% InvB (n, A) = 1 + (1 / A)*InvB (n - 1, A)
if ( (floor (n) ~= n) | (n < 1) )
    warning ('n is not a positive integer');
    A=NaN;
    return;
end;
if (A < 0.0)
    warning ('A is negative!');
    A=NaN;
    return;
end;
InvB=1;
for k=1:n,
    InvB=1 + (1/A)*InvB (n-1, A);
end;

function InvC=erlangcinv (n, A)
% This function computes the inverse Erlang C probability that a system with n
% servers, infinite waiting line, Poisson arrival rate lambda, service rate
% (per server) mu, and intensity rho=lambda / mu will have all servers busy.
% It uses the formula InvC (n, A) = InvB (n, A) – InvB (n - 1, A)
if ( (floor (n) ~= n) | (n < 1) )
    warning ('n is not a positive integer');
    A=NaN;
    return;
end;
if (A < 0.0)
    warning ('A is negative!');
    A=NaN;
    return;
end;
InvB=erlangbinnv (n, A);
InvC (n, A) = InvB (n, A) – InvB (n - 1, A);

2.3.7 Sander’s Formula

As it can be noticed this formula gives the inverse expression of Erlang – C in terms of Erlang – B.
and it is by far the most accurate one.
The derivation procedure starts as follows:

\[ E_{2,n}(A) = \frac{nE_{1,n}(A)}{1 - A \left[1 - E_{1,n}(A)\right]} \]  \hspace{1cm} (2.36)

\[ \frac{1}{E_{2,n}(A)} = \frac{n - A \left[1 - E_{1,n}(A)\right]}{n \cdot E_{1,n}(A)} \hspace{1cm} \frac{1}{E_{1,n}(A)} = \frac{A \left[1 - E_{1,n}(A)\right]}{n \cdot E_{1,n}(A)} \]  \hspace{1cm} (2.37)

Using the recursion formula for evaluation of Erlang - B the second term in (2.37) can be expressed by \(E_{1,n-1}(A)\).

\[ E_{1,n}(A) = \frac{A \cdot E_{1,n-1}(A)}{n + A \cdot E_{1,n-1}(A)} \]  \hspace{1cm} (2.38)

Now we may apply (2.38) to obtain the Sander’s formula for evaluating the Erlang - C formula.

\[ \frac{1}{E_{2,n}(A)} = \frac{1}{E_{1,n}(A)} - \frac{A \left[1 - \frac{A \cdot E_{1,n-1}(A)}{n + A \cdot E_{1,n-1}(A)}\right]}{n \cdot \frac{A \cdot E_{1,n-1}(A)}{n + A \cdot E_{1,n-1}(A)}} \]  \hspace{1cm} (2.39)

\[ \frac{1}{E_{2,n}(A)} = \frac{1}{E_{1,n}(A)} - \frac{1}{E_{1,n-1}(A)} \]  \hspace{1cm} (2.40)

3. Queue Length Distribution (QLD) – M/M/n
The queue length distribution (QLD) is determined by the probabilities of observing the queuing system in the steady states $p(n+1)$, $p(n+2)$ etc i.e. the probability of having one, two etc. customers in the queue. Thus, the respective probability density function (PDF) is discrete.

\[ p_s(L) = p(n+1)\delta(L) + p(n+2)\delta(L+1) \ldots \quad (3.1) \]

The steady state probabilities have already been derived in section 2. The first few values of importance are:

\[ p(n+1) = \frac{A}{n} p(n) \quad (3.2) \]

\[ p(n+2) = \frac{A}{n} p(n+1) = \left( \frac{A}{n} \right)^2 p(n) \quad (3.3) \]

An arbitrary steady state probability becomes:

\[ p(n+i) = \frac{A}{n} p(n+i-1) = \left( \frac{A}{n} \right)^i p(n) \quad (3.4) \]

Therefore, we find the following expression for PDF of QLD:

\[ p_s(L) = \frac{A}{n} p(n)\delta(L) + \left( \frac{A}{n} \right)^2 p(n)\delta(L+1) + \ldots + \left( \frac{A}{n} \right)^i p(n) + \ldots \quad (3.5) \]

\[ p_s(L) = p(n) \frac{A}{n} \left[ \delta(L) + \frac{A}{n} \delta(L+1) + \left( \frac{A}{n} \right)^2 \delta(L+2) + \ldots \right] \quad (3.6) \]

In terms of Erlang-C formula, (3.6) can be expressed by:

\[ p_s(L) = p(n) \frac{n}{n-A} \left[ \frac{A}{n} \delta(L) + \left( \frac{A}{n} \right)^2 \delta(L+1) + \left( \frac{A}{n} \right)^3 \delta(L+2) + \ldots \right] \quad (3.7) \]

\[ p_s(L) = E_{2,n} \left( \frac{n-A}{n} \right) \left[ \frac{A}{n} \delta(L) + \left( \frac{A}{n} \right)^2 \delta(L+1) + \left( \frac{A}{n} \right)^3 + \ldots \right] \quad (3.8) \]

We can now proceed towards determining an expression for QLD. The probability of having customers in queue at a random point of time is denoted by:
The last relation can be expressed in terms of Erlang - C. That is,

\[
P(L > 0) = \frac{A}{n-A} \cdot p(n) = \frac{A}{n} E_{2,n} (A) \tag{3.10}
\]

It should be noted that \(p(L > 0)\) does not coincide with \(p(W>0) = E_{2,n}(A)\). They differ in that we do not include the steady state \(p(n)\) when performing the summation for determining \(p(L > 0)\) since the buffer is still empty \((L = 0)\) when the system is in the steady state probability \(p(n)\). The relation between the both of them is determined by the scaling factor \(A/n\).

Accordingly, the probability of having more than \(l\) customers in the queue is:

\[
P(L > l) = \sum_{i=l+1}^{\infty} p(i) = \frac{1}{1 - \frac{A}{n}} \cdot p(n + l + 1) = \frac{n}{n-A} \left( \frac{A}{n} \right)^{l+1} p(n) \tag{3.11}
\]

\[
P(L > l) = \left( \frac{A}{n} \right)^{l+1} E_{2,n} (A) \tag{3.12}
\]

Since we need the probability \(P(L < l)\) in order to obtain QLD, we find

\[
F_s (L) = 1 - P(L > l) = 1 - \left( \frac{A}{n} \right)^{l+1} E_{2,n} (A) \tag{3.13}
\]

### 3.1 MATLAB Code for PDF and QLD – M/M/n

```matlab
% The following script computes the queue length distribution (QLD) and the respective
% probability density function (PDF) that a system with n servers, infinite waiting line,
% Poisson arrival rate lambda, service rate (per server) mu, and intensity A = lambda / mu
% will have. It should be satisfied that A > 0, n > 0, A < n, n – positive integer.
% It uses the expressions (3.8) for PDF and (3.13) for QTD
n=20;
A=18;
C=erlangc (n, A);
for L=1:20
    ps (L) = (C*(n - A) / n)* ( (A / n)^L);
    Fs (L) = 1 - C* ( (A / n)^L+1 )
end
subplot (2,1,1); stem (ps);
xlabel ('Queue Length')
```


3.2 Mean Queue Length – M/M/n

We distinguish between the queue length at an arbitrary point of time and the queue length when there are customers waiting in the queue.

3.2.1 Mean Queue Length at an arbitrary point of time

The queue length $L$ at an arbitrary point of time is called the virtual queue length. This is the queue length experienced by an arbitrary customer, as the PASTA-property is valid due to the Poisson arrival process (time average = call average). We get the mean queue length $L_n = E\{L\}$ at an arbitrary point of time:
\[ L_n = 0 \cdot \sum_{i=0}^{n} p(i) + \sum_{i=n+1}^{\infty} (i - n) p(i) \]  
(3.14)

\[ L_n = \sum_{i=n+1}^{\infty} (i - n) p(n) \left( \frac{A}{n} \right)^{i-n} \]  
(3.15)

\[ L_n = p(n) \sum_{i=n+1}^{\infty} i \left( \frac{A}{n} \right)^{i} \]  
(3.16)

We can observe in (3.16) that the integrand can be expressed through the derivative of \((A/n)^i\).

\[ i \left( \frac{A}{n} \right)^{i} = \frac{A}{n} \frac{\partial}{\partial (A/n)} \left( \frac{A}{n} \right)^{i} \]  
(3.17)

Inserting (3.17) into (3.16) we have

\[ L_n = p(n) \frac{A}{n} \sum_{i=n+1}^{\infty} \frac{\partial}{\partial (A/n)} \left( \frac{A}{n} \right)^{i} \]  
(3.18)

As \(A/n < 1\), the geometric series is uniformly convergent, and the differentiation operator might be put outside the summation:

\[ L_n = p(n) \frac{A}{n} \frac{\partial}{\partial (A/n)} \left( \frac{A}{n} \right)^{i} = p(n) \frac{A/n}{1 - A/n} \]  
(3.19)

\[ L_n = p(n) \frac{n - A/n}{n - A} \]  
(3.20)

Eventually,

\[ L_n = E_{2,n}(A) \frac{A}{n - A} \]  
(3.21)

### 3.2.2 Mean Queue Length, if the queue is greater than zero

The time average is also in this case equal to the call average. The conditional mean queue length becomes:

\[ L_{mq} = \frac{\sum_{i=n+1}^{\infty} (i - n) p(i)}{\sum_{n+1}^{\infty} p(i)} \]  
(3.22)

The last equation amounts to
Comparing (3.24) with (3.21), we can state that

\[ L_{mq} = \frac{L_n}{P(L > 0)} \]  

(3.25)
4. Waiting Time Distribution (WTD) – M/M/n

The waiting time distribution $W_s(t)$ for a random customer usually has a positive probability mass (atom) at $t = 0$, because some of the customers get service immediately without waiting. We thus have $W_s(0) > 0$. The waiting time distribution $W_r(t)$ for customers having positive waiting times becomes:

$$W_r(t) = \frac{W_s(t) - W_s(0)}{1 - W_s(0)}$$  \hspace{1cm} (4.1)

Denoting the probability of a positive waiting time $[1 - W_s(0)]$ by $D$ (probability of delay) then we get:

$$D \cdot [1 - W_r(t)] = 1 - W_s(t)$$  \hspace{1cm} (4.2)

4.1 Waiting Time Distribution - M/M/n, FCFS

Queuing systems where the service discipline only depends upon the arrival times, all have the same mean waiting times. In this case the strategy has only influence upon the distribution of waiting times for the individual customer.

Let us consider an arbitrary customer. Upon arrival to the system, the customer is either served immediately or has to wait in the queue. We now assume that the customer considered has to wait in the queue, i.e. the system may be in state $(n+k)$, $(k = 0, 1, 2, \ldots)$, where $k$ is the number of occupied waiting positions just before the arrival of an arbitrary customer. The customer has to wait until $(k + 1)$ customers have completed their service before an idle server becomes available. When all $n$ servers are operating, the system services customers with a constant rate $n \mu$, i.e. the departure process is a Poisson process with this intensity.

The probability $p(W \leq t) = F(t)$ of experiencing a positive waiting time less than or equal to $t$ is equal to the probability that in a Poisson arrival process with intensity $(n \mu)$ at least $(k+1)$ customers arrive during the interval $t$:

$$F(t / k) = \sum_{i=k+1}^{\infty} \frac{(n \mu)^i}{i!} \cdot e^{-n \mu t}$$  \hspace{1cm} (4.3)

This cumulative probability was based on the assumption that our customer has to wait in the queue. The conditional probability that the customer when arriving observes all $n$ servers busy and $k$ waiting customers $(k = 0, 1, 2\ldots)$ is:
The unconditional waiting time distribution then becomes:

\[ F(t) = \sum_{k=0}^{\infty} \left( 1 - \frac{A}{n} \right) \left( \frac{A}{n} \right)^k \sum_{i=k+1}^{n} \left( \frac{n\mu t}{i!} e^{-n\mu t} \right) \]  

By few mathematical transformations and calculations we can obtain the following expression that represents the waiting time distribution for customers that are waiting in the buffer to be served.

\[ F(t) = 1 - e^{-(n-A)t} = 1 - e^{-(n\mu - A)t} > A, \quad t > 0 \]  

It is clear that \( A \) represents the offered traffic whereas \( 1/\mu \) is the mean service time. If we recall the equation (4.2) it is clear that \( D \) equals Erlang - C formula. On the other hand, \( W_s = F(t) \) represents the unconditional probability of experiencing a positive waiting time less than or equal to \( t \) and is the probability that in a Poisson arrival process with intensity \( (n\mu) \) at least \( (k+1) \) customers arrive during the interval \( t \).

\[ E_{2,n}(A) \cdot [1 - F(t)] = 1 - F_s(t) \]  

The expression for \( F(t) \) is again obtained to be as (4.8). At this point it is intuitively reasonable that we shall further continue with finding the expression for the waiting time distribution for all customers (those who are served immediately and those who have to wait in the queue). This can be easily obtained if (4.8) is inserted in (4.10) as follows:

\[ F_s(t) = 1 - E_{2,n}(A)e^{-(n-A)t} \]  

In order to obtain this PDF we should find the derivative of \( F_s(t) \) in the following way.

\[ p_s(t) = \frac{dF_s(t)}{dt} = \frac{d\left[1 - E_{2,n}(A)e^{-(n-A)t}\right]}{dt} = E_{2,n}(A) \cdot (n - A)\mu \cdot e^{-(n-A)t} \]
4.2 MATLAB Code for PDF and WTD – M/M/n

% The following script computes the waiting time distribution (WTD) and the respective
% probability density function (PDF) that a system with n servers, infinite waiting line,
% Poisson arrival rate lambda, service rate (per server) mu, and intensity A = lambda / mu
% will have. It should be satisfied that A > 0, n > 0, A < n, n – positive integer.
% It uses the expressions (4.12) for PDF and (4.8) for WTD
% Mean service time, measured in time units, offered traffic, measured in Erlang
s = 10; mu = 1/s;
n = 20;
A = 18;
t = 0:0.25:30;
C = erlangc(n, A);
ps = mu*(n - A)*C*exp(-(n - A)*mu*t);
Fs = 1 - C*exp(-(n - A)*mu*t);
plot(t, ps, 'g');
hold on
plot(t, Fs, 'r');
xlabel('Waiting time (Time Units)')
ylabel('PDF (green), WTD (red)')
title('Calculation of Probability Density Function and Waiting Time Distribution')

Figure 4.1. Waiting Time Distribution (WTD) and its Probability Density Function (PDF) of M/M/n
4.3 Waiting Time Distributions for Other Queuing Disciplines

Apart from FIFO, there are several other queuing disciplines used in teletraffic engineering. Those, which are of particular interest for us, are LCFS (Last In First Served) and SIRO (Service In Random Order). LCFS corresponds to the stack principle and is also used in storages, on shelves of shops etc. In case of SIRO all customers who are waiting in the queue have the same probability of being chosen for service. FIFO and LCFS disciplines only take arrival times into consideration, while SIRO does not consider any criteria at all and so does not require any memory.

For the three above-mentioned disciplines the total waiting time for all customers is the same. The queuing discipline only decides how waiting time is allocated to the individual customers. In a program-controlled queuing system there may be more complicated queuing disciplines. In queuing theory we in general assume that the total offered traffic is independent of the queuing discipline.

Their probability density functions for the waiting time distribution of the above disciplines are shown in Fig. 4.2.

![Figure 4.2. Probability Density Function (PDF) for the Waiting Time Distribution for the Queuing Disciplines LCFS, LCFS, and SIRO](image)

For all three cases the mean waiting time for delayed calls is 5 time-units. The form factor is 2 for FCFS, 3.33 for LCFS, and 10 for SIRO. The number of servers is 10 and the offered traffic is 8 Erlang. The mean service time is $s = 10$ time-units.
4.4 Mean Waiting Time – M/M/n

Here there are also two items of interest: the mean waiting time $W$ for all customers, and the mean waiting time $w$ for customers experiencing a positive waiting time. The first one is an indicator for the service level of the whole system, whereas the second one is of importance for the customers, which are delayed. Time averages will be equal to call averages because of the PASTA-property.

4.4.1 Mean Waiting Time - All customers

The Little’s theorem states that the average queue length is equal to the arrival intensity multiplied by the mean waiting time.

$$L_n = \lambda W_n \quad (4.13)$$

In this relation $L_n = L_n(A)$ and $W_n = W_n(A)$. Moreover, by making use of (3.21) we obtain

$$W_n = \frac{L_n}{\lambda} = \frac{1}{\lambda} E_{2,n}(A) \frac{A}{n - A} \quad (4.14)$$

As $A/\lambda$ equals $s$, where $s$ is the mean service time, we finally get

$$W_n = E_{2,n}(A) \frac{s}{n - A} \quad (4.15)$$

4.4.2 Mean Waiting Time - Delayed customers

If we are interested in the mean waiting time averaged over all customers over customers that experience positive waiting times only, we can define:

$$W_n = w_n E_{2,n}(A) \quad (4.16)$$

In the last equation, $w_n$ denotes the mean waiting time averaged over customers over customers that experience positive waiting times only (delayed customers). By combining (4.15) and (4.16), we have:

$$w_n = \frac{s}{n - A} \quad (4.17)$$

We can notice that as $A$ approaches 0, one would eventually get $w_n = s/n$. Put another way, if a customer experiences waiting time, then this customer will be the only one in the queue. The customer must wait until a server becomes idle. This happens after an exponentially distributed time interval with mean value $s/n$. Therefore, $w_n$ never becomes less than $s/n$. 

26
5. Improvement Functions for M/M/n

There are several ways to express the marginal improvement of the traffic when one server is added in the system. In general the improvement function is denoted by $F_n (A)$ whereas in our case, considering Erlang - B formula, it is $F_{2,n} (A)$ and its concrete definition would be the decrease in the proportion of the total traffic that experience delay.

$$F_{2,n} (A) = A \cdot [E_{2,n} (A) - E_{2,n+1} (A)] \tag{5.1}$$

In the following lines we derive the relation for $F_{2,n}(A)$ so that it is expressed in terms of Erlang - B formula.

$$F_{2,n} (A) = A \cdot [\frac{E_{1,n} (A)}{1 - \frac{A \cdot [1 - E_{1,n} (A)]}{n}} - \frac{E_{1,n+1} (A)}{1 - \frac{A \cdot [1 - E_{1,n+1} (A)]}{n+1}}] \tag{5.2}$$

$$F_{2,n} (A) = A \cdot \left[ \frac{E_{1,n} (A)}{1 - \frac{A \cdot [1 - E_{1,n} (A)]}{n}} - \frac{\frac{A}{n+1} \cdot E_{1,n} (A)}{1 - \frac{A \cdot [1 - E_{1,n} (A)]}{n+1}} \right] \tag{5.3}$$

The expression of $E_{2,n}(A)$ in terms of $E_{1,n}(A)$ is given by (2.22 ) and $E_{2,n+1}(A)$ is derived below.

$$E_{2,n+1} (A) = \frac{A \cdot E_{1,n} (A)}{(n+1) + A \cdot E_{1,n} (A)} \tag{5.4}$$

$$E_{2,n+1} (A) = A \cdot \left[ \frac{A \cdot E_{1,n} (A)}{(n+1) + A \cdot E_{1,n} (A)} \right] \tag{5.5}$$

$$E_{2,n+1} (A) = \frac{A \cdot E_{1,n} (A)}{(n+1)(n+1) + A \cdot E_{1,n} (A)} - A(n+1) = \frac{A \cdot E_{1,n} (A)}{(n+1) - A(1 - E_{1,n} (A))} \tag{5.6}$$
\[ E_{2,n+1}(A) = \frac{A}{n+1} E_{1,n}(A) \frac{1}{1 - \frac{A}{n+1} [1 - E_{1,n}(A)]} \]  

(5.7)

In (5.2) we use the recursion formula for evaluating the Erlang - B formula.

\[ E_{1,n+1}(A) = \frac{A E_{1,n}(A)}{(n+1) + A E_{1,n}(A)} \]  

(5.8)

Another expression for the marginal improvement of the traffic carried when one server is added could be defined as the decrease in mean queue length using Little’s law \(L_n = \lambda \cdot W_n\). It is also given in terms of Erlang - B formula.

\[ F_{L,n}(A) = L_n(A) - L_{n+1}(A) = \lambda [W_n(A) - W_{n+1}(A)] \]  

(5.9)

\[ F_{L,n}(A) = \lambda [E_{2,n}(A) \cdot w_n - E_{2,n+1}(A) \cdot w_{n+1}] \]  

(5.10)

Combining (5.7) and (5.10), we have

\[ F_{L,n}(A) = \lambda \cdot \left\{ \frac{E_{1,n}(A)}{A[1 - E_{1,n}(A)]} \cdot w_n - \frac{A}{n+1} \cdot E_{1,n}(A) \frac{1}{1 - \frac{A}{n+1} [1 - E_{1,n}(A)]} \cdot w_{n+1} \right\} \]  

(5.11)

where \(W_n(A)\) is the mean waiting time for all costumers when the offered traffic is \(A\) and the number of servers is \(n\). \(W_{n+1}\) is in case when the number of servers is \(n+1\).

Both are given in (5.12) and (5.13), respectively.

\[ W_n(A) = E_{2,n}(A) \cdot \frac{s}{n - A} = E_{2,n}(A) \cdot w_n \]  

(5.12)

\[ W_{n+1}(A) = E_{2,n+1}(A) \cdot \frac{s}{(n+1) - A} = E_{2,n+1}(A) \cdot w_{n+1} \]  

(5.13)

Here \(w_n\) is the mean waiting time for customers who experience positive waiting times and \(s\) is the mean service time.
5.1 MATLAB Code for Improvement Function of M/M/n

% The following script computes the Improvement Function that a system with n servers,
% infinite waiting line, Poisson arrival rate lambda, service rate (per server) mu, and
% intensity % A = lambda / mu with respect to the offered traffic.

n=25;
for A1=1:2500-1
    A=A1 / 100
    for i=1:2
        C (i, A1)=A*(erlangc (n-1+i, A) );
        L (i, A1)=C (i, A1) / (n-1+i-A);
    end
    F_2n (A1) = C (1, A1) – C (2, A1);
    F_Ln (A1) = L (1, A1) - L (2, A1);
end
subplot (2,1,1); plot (F_2n);
xlabel ('Offered Traffic (100*Erlang)')
ylabel ('F_2n')
title ('Calculation of the improvement function for M/M/n system')
subplot (2,1,2); plot (F_Ln);
xlabel ('Offered Traffic (100*Erlang)')
ylabel ('F_Ln')

Figure 5.1 Calculation of the Improvement Function for M/M/n system
6. Finite-buffer Queuing System

In real systems we always have a finite queue. In computer systems the size of the storage is finite and the IP routers and ATM switches have finite buffers. In the following section we present a short analysis on some of these systems. For single-server systems only state probabilities derivations of the state probabilities are considered, whereas \( M/M/n/k \) systems is analyzed more detail.

6.1 General Considerations

Obtaining the state probabilities of a queuing system with finite buffer is not an easy task as in the case of queuing system with infinite buffer positions. Still, calculations of the steady state probabilities could be made. However, at the beginning, it is much convenient to start out by determining the state probabilities of single server system. We can then proceed towards deriving the steady state probabilities in the multiple-server system.

6.1.1 Single-server system

The state probabilities of the single-server finite-buffer system are obtained in the similar manner as the state probabilities of the infinite buffer system. In a system with one server and \((k-1)\) queuing positions we have \((k+1)\) states \((0, 1 \ldots k)\). The state diagram is shown below.

\[
\begin{align*}
0 & \quad \lambda & \mu \\
1 & \lambda & \mu \\
& \vdots & \vdots \\
& \lambda & \mu \\
k & \lambda & \\
\end{align*}
\]

Figure 6.1 State transition diagrams for single-server finite-buffer system

6.1.1.1 M/M/1/k

The balance equations are the same as in M/M/1 except now, two boundary conditions for \(i = 0\) and \(i = k\) exist. They are

\[
\begin{align*}
\lambda \cdot p(i) &= \mu \cdot p(i+1) \\
(\lambda + \mu) p(i) &= \lambda \cdot p(i-1) + \mu \cdot p(i+1)
\end{align*}
\]

We can establish the following relation:
In the relation above $\rho$ is given by

$$\rho = \frac{\lambda}{\mu} \quad (6.4)$$

Further on, using the normalization requirement:

$$\sum_{i=0}^{k} p(i) = 1 \quad (6.5)$$

$$p(0)(1 + \rho + \rho^2 + ... + \rho^k) = 1 \quad (6.6)$$

Thus we have got the finite geometric series. Hence,

$$p(0) = \frac{1 - \rho}{1 - \rho^{k+1}} \quad (6.7)$$

Finally, the steady state probabilities become

$$p(i) = \frac{1 - \rho}{1 - \rho^{k+1}} \rho^i \quad (6.8)$$

Now it is an easy task to obtain the probability for the queue to be full i.e. the probability of blocking. It is the case where $i = k$. Consequently,

$$P(\text{blocking}) = p(k) = \frac{1 - \rho}{1 - \rho^{k+1}} \rho^k \quad (6.9)$$

6.1.1.1. MATLAB Code for Blocking Probability in M/M/1/k

```matlab
n=1;
k=15;
for ro = 1:1000
    rho = ro / 100;
    pb (ro)=( (1 - rho)*(rho^k) ) / (1 - rho^(k + 1) );
end
plot (pb);
xlabel ('Offered traffic (100*Erlang)')
ylabel ('Blocking probability')
title ('Calculation of blocking probability with respect to the offered traffic')
```
There exists a simple relation between the state probabilities $p(i)$ ($i = 0, 1, 2, \ldots$) of the infinite system $M/G/1$ and the state probabilities $p_k(i)$, ($i = 0, 1, 2, \ldots k$) of $M/G/1/k$, where the total number of positions for customers is $k$, including the customer being served.

$$p_k(i) = \frac{p(i)}{(1 - A \cdot Q_k)} , \quad i = 0, 1, 2, \ldots, k-1 \quad (6.10)$$

$$p_k(i) = \frac{(1 - A) \cdot Q_k}{(1 - A \cdot Q_k)} \quad (6.11)$$

where $A < 1$ is the offered traffic, and

$$Q_k = \sum_{j=k}^{\infty} p(j) \quad (6.12)$$

There exists algorithms for calculating $p(i)$ for arbitrary holding time distributions. We can notice that the above is only valid for $A < 1$, but for a finite buffer we also obtain statistical equilibrium for $A > 1$. 

**Figure 5.2.** Blocking Probability of a M/M/1/15 system
6.1.1.3 M/D/1/k

The balance equations for states \{0, 1,...,k-2\} can be set up in the same way as Fry’s equations of state i.e.

\[ p_{i+1}(i) = \{ p_i(0) + p_i(1) \} p(i, h) + \sum_{j=2}^{i+1} p_i(j) \cdot p(i - j + 1, h) \]  

(6.13)

In (6.13), \( p(j, h) \) is given by

\[ p(j, h) = \frac{(\lambda h)^j}{j!} e^{-\lambda h}, j = 0, 1, 2,... \]  

(6.14)

It denotes the probability of \( j \) new call appearing during the time interval \( h \).

However, it is not possible to write down simple time-independent equations for state \( k-1 \) and \( k \). However, the first \((k - 2)\) equations together with the normalization requirement (6.5) and the fact that the offered traffic equals the carried traffic plus the rejected traffic (PASTA property):

\[ A = 1 - p(0) + Ap(k) \]  

(6.15)

The result is \((k+1)\) independent equations, which could be solved easily.

6.2 Multiple-server system

The state probabilities of the multiple-server finite-buffer system (particularly \( M/M/n/k \)) can be obtained in the similar manner as the state probabilities of the infinite buffer system. In a system with \( n \) server and \( k \) queuing positions we have \((k + 1)\) states \((0, 1,\ldots, k)\).

Typically, in \( k \) there are included \( n \) queuing positions – those determined by the number of servers. (Alternatively, in some other notation where this is not the case, the number of states are determined by \( n + k + 1 \)) The state diagram is shown below.

![Figure 6.2 State transition diagram for Multiple-server Finite-buffer System](image)

6.2.1 Steady State Probabilities

We can derive the state probabilities of M/M/n/k queuing system in the similar manner as in M/M/n. The relation (2.7) still holds in this case. However, the normalization condition is finite.
\[ p(i) = 1 \quad (6.16) \]

\[ p(0) \left(1 + \frac{A}{1} + \frac{A^2}{2!} + \ldots + \frac{A^n}{n!} \right) \left(1 + \frac{A}{n} + \frac{A^2}{n^2} + \ldots + \frac{A^{k-n}}{n^{k-n}} \right) = 1 \quad (6.17) \]

\[ p(0) = \frac{1}{\sum_{i=0}^{n-1} \frac{A^i}{i!} + \frac{A^n}{n!} \frac{1 - \left( \frac{A}{n} \right)^{k-n+1}}{1 - \frac{A}{n}}} \quad (6.18) \]

Now the state probabilities are as follows:

\[ p(i) = \begin{cases} 
      p(0) \cdot \frac{A^i}{i!} , & 0 \leq i \leq n \\
      p(n) \cdot \left( \frac{A}{n} \right)^{k-n} , & n \leq i \leq k 
   \end{cases} \quad (6.19) \]

### 6.2.2 Blocking Probability – M/M/n/k

The blocking occurs when all servers, together with all waiting positions in the buffer are busy. Therefore, the probability of blocking is equal to \( p(k) \). Using (2.7), (6.18), and (6.19), the probability of blocking becomes:

\[ p(blocking) = \frac{A^n A^{k-n}}{n! n^{k-n}} \sum_{i=0}^{n-1} \frac{A^i}{i!} + \frac{A^n}{n!} \frac{1 - \left( \frac{A}{n} \right)^{k-n+1}}{1 - \frac{A}{n}} \quad (6.20) \]

### 6.2.2.1 MATLAB Code for the Blocking Probability in M/M/n/k

% The following script computes the blocking probability that a system with n servers, k-n available waiting positions in the buffer, Poisson arrival rate lambda, service rate % mu, and intensity rho = lambda/mu will have with respect to the offered traffic. The % offered traffic is measured in Erlang. It uses the expression (6.20).
% $n = 10$;
% $k = 15$;
% for ro = 1:3000
% rho = ro/100;
% nom (ro) = (rho^n / factorial (n)) * ( (rho^n(k-n)) / (n^n(k-n)) ) ;
for i=1:n-1
    t(i)=(rho^i) / (factorial(i));
end
s(ro)=sum(t);
m(ro)=rho^n / factorial(n);
l(ro)=m(ro)*(1-((rho/n)^k)+1)/((1-(rho/n)));
denom(ro)=s(ro)+l(ro);
pb(ro)=nom(ro)/denom(ro);
end
plot(pb);
xlabel('Offered traffic (100*Erlang)')
ylabel('Blocking probability')
title('Calculation of blocking probability with respect to the offered traffic')

Figure 6.3. Blocking Probability of M/M/10/15 system
6.2.3 Queue Length Distribution – M/M/n/k

QLD is easy to obtain from (3.6). It is:

$$p_s(L) = p(n+1)\delta(L) + p(n+2)\delta(L+1) + \ldots + p(K)\delta(K)$$  \hspace{1cm} (6.21)

Similarly,

$$p_s(L) = \frac{A}{n} \cdot p(n) \cdot \delta(L) + \left( \frac{A}{n} \right)^2 \cdot p(n) \cdot \delta(L+1) + \ldots + \left( \frac{A}{n} \right)^{K-n+1} \cdot p(n) \cdot \delta(L+K-N)$$  \hspace{1cm} (6.22)

$$p_s(L) = \frac{A}{n} p(n) \left( \delta(L) + \left( \frac{A}{n} \right) \delta(L+1) + \ldots + \left( \frac{A}{n} \right)^{K-n} \delta(L+K-N) \right)$$  \hspace{1cm} (6.23)

In terms of Erlang – C, (3.8) can be rewritten as:

$$p_s(L) = \frac{A}{n} E_{2,n}(A) \left( \delta(L) + \left( \frac{A}{n} \right) \delta(L+1) + \ldots + \left( \frac{A}{n} \right)^{K-n} \delta(L+K-N) \right)$$  \hspace{1cm} (6.24)

As for the QLD function, following (2.10), we can write:

$$P(L > l) = \sum_{i=n+1}^{k} p(i) = \frac{1 - \left( \frac{A}{n} \right)^{k-n-i}}{1 - \frac{A}{n}} \left( \frac{A}{n} \right)^{l+1} p(n)$$  \hspace{1cm} (6.25)

$$P(L > l) = \sum_{i=n+1}^{k} p(i) = E_{2,n}(A) \left( 1 - \left( \frac{A}{n} \right)^{k-n-i} \right) \left( \frac{A}{n} \right)^{l+1}$$  \hspace{1cm} (6.26)

Accordingly,

$$F_s(L) = 1 - P(L > l) = 1 - E_{2,n}(A) \left( 1 - \left( \frac{A}{n} \right)^{k-n-i} \right) \left( \frac{A}{n} \right)^{l+1}$$  \hspace{1cm} (6.27)

6.2.3.1. MATLAB Code for PDF and QLD – M/M/n/k

% The following script computes the queue length distribution (QLD) and the respective
% probability density function (PDF) that a M/M/n/k system, Poisson arrival rate lambda,
% service rate (per server) mu, and intensity A = lambda / mu will have. It should be
% satisfied
% that A > 0, n > 0, A < n, n - positive integer. It uses the expressions (6.24) for PDF
% and (6.27) for QTD. Mean service time, measured in time units, offered traffic,
% measured in Erlang
s = 10;
u = 1/s;
n = 20;
A = 19;
k = 35;
C = erlangc(n, A);
for L = 1:k-n
    ps(L) = (C*(n - A) / n)((A / n)^L);
    Fs(L) = 1 - C*(1 - (A / n)^(k - n - L));
end
stem(ps,'g','o');
hold on
stem(Fs,'r','x');
xlabel('Queue Length')
ylabel('PDF (green), QLD (red)')
title('Calculation of Probability Density Function and Queue Length Distribution')

Figure 6.4. Queue Length Distribution and its Probability Density Function of M/M/n/k
6.2.3.2 Mean Queue Length – M/M/n/k

Again, the derivation of MQL resembles much to the one in M/M/n case. Consequently, we start out from:

\[
L_n = 0 \cdot \sum_{i=0}^{n} p(i) + \sum_{i=n+1}^{K-n} (i-n) p(i) \tag{6.28}
\]

\[
L_n = \sum_{i=n+1}^{K-n} (i-n) p(n) \left( \frac{A}{n} \right)^{i-n} \tag{6.29}
\]

Moreover,

\[
L_n = p(n) \sum_{i=1}^{K-2n} \frac{\partial}{\partial \left( A/n \right)} \left( \frac{A}{n} \right)^i \tag{6.30}
\]

Further on,

\[
L_n = p(n) \frac{A}{n} \sum_{i=1}^{K-2n} \frac{\partial}{\partial \left( A/n \right)} \left( \frac{A}{n} \right)^i \tag{6.31}
\]

We can put the differential operator outside integral. This yields:

\[
L_n = p(n) \frac{A}{n} \frac{\partial}{\partial \left( A/n \right)} \sum_{i=1}^{K-2n} \left( \frac{A}{n} \right)^i \tag{6.32}
\]

\[
L_n = p(n) \frac{A}{n} \frac{\partial}{\partial \left( A/n \right)} \left( 1 - \left( \frac{A}{n} \right)^{K-2n-1} \right) \tag{6.33}
\]

Finally, solving the derivative, we arrive at

\[
L_n = p(n) \frac{A}{n} \left( 1 - \left( \frac{A}{n} \right)^{K-2n-1} \right) / \left( 1 - \frac{A}{n} \right) \tag{6.34}
\]

In terms of Erlang – C, (6.41) can expressed by
6.2.4. Waiting Time Distribution – M/M/n/k

The derivation of WTD function follows the line of thought as in the case of M/M/n. The relation (4.3) is also valid here. Since the issue here is WTD function of a finite-buffer system with \( K+1 \) steady states, the conditional probability that a customer upon arriving to the system observes all \( n \) servers busy and \( m \) waiting customers \((m = 0,1,2\ldots K-n)\) now becomes:

\[
p_w(m) = \frac{\lambda p(n + m)}{\lambda \sum_{s=0}^{K-n} p(n + s)} = \frac{p(n) \left( \frac{A}{n} \right)^m}{\sum_{s=0}^{K-n} \left( \frac{A}{n} \right)^s}
\]

\[
p_w(m) = \frac{p(n) \left( \frac{A}{n} \right)^m}{1 - \left( \frac{A}{n} \right)^{K-n+1}}
\]

\[
p_w(m) = \left( \frac{A}{n} \right)^m \frac{1 - \left( \frac{A}{n} \right)^{K-n+1}}{1 - \left( \frac{A}{n} \right)^{K-n+1}}, \quad m = 0,1,2\ldots K-n
\]

Now that we have found the conditional probability, we go further onto determining the unconditional waiting time distribution. Therefore,

\[
L_n = p(n) \frac{n}{n-A} A \left\{ \frac{1 - \left( \frac{A}{n} \right)^{K-2n+1}}{1 - \left( \frac{A}{n} \right)^{K-2n+1}} - (K-2n+1) \left( \frac{A}{n} \right)^{K-2n+1} \right\}
\]

\[
L_n = E_{2,n}(A) \left\{ \frac{1 - \left( \frac{A}{n} \right)^{K-2n+1}}{n-A} - (K-2n+1) \left( \frac{A}{n} \right)^{K-2n+1} \right\}
\]
\[ F(t) = \sum_{m=0}^{K-n} p_n(m) F(t/m) \]  

(6.40)

Furthermore, substituting (4.3) and (6.23) into (6.24), we have

\[ F(t) = \sum_{m=0}^{K-n} \left( \frac{A}{n} \right)^m \frac{1 - \left( \frac{A}{n} \right)}{1 - \left( \frac{A}{n} \right) n} \sum_{k=1}^{K-n+1} \frac{(n \mu t)^k}{k!} e^{-n \mu t} \]  

(6.41)

It is very difficult to obtain the solution for this function. However, by utilizing Mathematica 4.2, we arrive at the following expression:

\[ F(t) = e^{-\alpha r m} \frac{\Gamma(1-n) \Gamma(n \mu t + 1) - \Gamma(n \mu t)}{\Gamma(1-n) \Gamma(n \mu t)} \]  

(6.42)

In (6.42), we have introduced Gamma function. We consider Gamma function in Appendix A.

The PDF function has even more complicated form and therefore, it will not be considered here.

### 6.2.4.1. Mean Waiting Time – M/M/n/k

Similarly, as in the case of M/M/n, we can apply the Little’s formula to determine MWT. Hence,

\[ W_n = \frac{L_n}{\lambda} = \frac{1}{\lambda} \frac{1 - \left( \frac{A}{n} \right)^{K-2n+1}}{\left( 1 - \frac{A}{n} \right)^2} - \frac{(K-2n+1) \left( \frac{A}{n} \right)^{K-2n}}{1 - \frac{A}{n}} \]  

(6.43)

(6.27), in terms of Erlang – C, this can be expressed by:

\[ W_n = \frac{L_n}{\lambda} = \frac{1}{\lambda} E_{2,n}(A) \left( \frac{1 - \left( \frac{A}{n} \right)^{K-2n+1}}{A - \frac{A}{n}} - (K-2n+1) \left( \frac{A}{n} \right)^{K-2n+1} \right) \]  

(6.44)
7. Cost Function

Moe first derived his principle for queuing systems. He studied the subscribers waiting times for an operator at the manual exchanges in Copenhagen Telephone Company. Let us consider \( k \) independent queuing systems. A customer being served at all \( k \) systems has the total average waiting time 

\[ P_i W_i, \]

where 

\[ W_i \] is the mean waiting time of \( i \)'th system which has 

\[ n_i \] servers and is offered the traffic \( A_i \). The cost of a channel is \( c_i \), eventually plus a constant cost, which is included in the constant \( C_0 \) below. Thus the total costs for channels becomes:

\[ C = C_0 + \sum_{i=1}^{k} n_i \cdot c_i \]  

(7.1)

If the waiting time is also considered as a cost, then the total costs that is to be minimised becomes 

\[ f = f (n_1, n_2, \ldots, n_k). \] This is to be minimised as a function of number of channels \( n_i \) in the individual systems. If the total average waiting time is \( W \), then the allocation of channels to the individual systems is determined by:

\[ \min\{f(n_1, n_2, \ldots, n_k)\} = \min\left\{C_0 + \sum_{i} n_i \cdot c_i + \theta \cdot \left( \sum_{i} W_i - W \right) \right\} \]  

(7.2)

where \( \theta \) (theta) is the Lagrange’s multiplier.

As \( n_i \) is integral, a necessary condition for minimum, which in this case also can be shown to be sufficient condition, becomes:

\[ 0 < f(n_1, n_2, \ldots, n_i+1, \ldots, n_k) - f(n_1, n_2, \ldots, n_i, \ldots, n_k), \]  

(7.3)

\[ 0 \leq f(n_1, n_2, \ldots, n_i, \ldots, n_k) - f(n_1, n_2, \ldots, n_i+1, \ldots, n_k), \]  

(7.4)

which corresponds to:

\[ W_{n_i+1}(A_i) - W_{n_i}(A_i) > \frac{c_i}{\theta} \]  

(7.5)

\[ W_{n_i}(A_i) - W_{n_i+1}(A_i) \leq \frac{c_i}{\theta} \]  

(7.6)

where \( W_n (A_i) \) is given by (4.15). Expressed by the improvement function for the waiting time \( F_{W,n}(A_i) \) in (5.4) the optimal solution becomes:

\[ F_{W,n_i-1} > (A_i) \frac{c_i}{\theta} \geq F_{W,n_i}(A_i), \quad i=1,2\ldots k \]  

(7.7)

The function \( F_{W,n}(A) \) is tabulated in Moe’s Principle (Jensen, 1950 [5]). Similar optimizations can be carried out for other improvement functions.
8. Priority of customers

In real life customers are often divided into $N$ priority classes, where a customer belonging to class $p$ has higher priority than a customer belonging to class $p+1$. We distinguish between two types of priority:

- **Non-pre-emptive (HOL)** - A new arriving customer with higher priority than a customer being served waits until a server becomes idle (and all customers with higher priority have been served). This discipline is also called HOL (Head-Of-the-Line).

- **Pre-emptive** - A customer being served having lower priority than a new arriving customer is interrupted. We distinguish between:

  - **Pre-emptive resume (PR)** - The service is continued starting from the point where it was interrupted.

  - **Pre-emptive without re-sampling** - The service restarts from the beginning with the same service time

  - **Pre-emptive with re-sampling** - The service starts again with a new service time.

The two latter disciplines are applied in for example manufacturing systems and reliability. There are many different queuing models. However, there are only few general results in the queuing theory. Little’s theorem presented is the most general result, which is valid for an arbitrary queuing system. The theorem is easy to apply and very useful in many cases.

In general only queuing systems with Poisson arrival processes are simple to deal with. Concerning queuing systems in series and queuing networks (e.g. computer networks) it is important to know cases, where the departure process from a queuing system is a Poisson process.

These queuing systems are called symmetric queuing systems, because they are symmetric in time, as the arrival process and the departure process are of the same type. If we make a film of the time development, we cannot decide whether the film is run forward or backward.

Systems that deviate most from the classical models are the systems with a single server. However, these systems are also the simplest to deal with. In waiting time systems we also distinguish between call averages and time averages. The virtual waiting time is the waiting time, a customer experiences if the customer arrives at a random point of time (time average).

The actual waiting time is the waiting time, the real customers experiences (call average). If the arrival process is a Poisson process, then the two averages are identical.
8.1 Analysis of M/M/n/K Queues with Priorities

We shall further outline an approach that may be used to analyse a M/M/n/K queue \((n=1, 2, \ldots \text{ and } K = n, n+1, \ldots \infty)\) which has job classes of multiple priorities. The priority discipline followed may be either non-pre-emptive or pre-emptive in nature, as we have explained before. However, it should be noted that there might be loss of work in the pre-emptive non-resume priority case. Such loss of work will not happen in the case of the other two priorities. Since we assume that the service times are exponentially distributed, they will satisfy the memory-less property and that, therefore, the results will be the same both for the pre-emptive resume and pre-emptive non-resume cases.

For a \(P\) priority system, we consider jobs of class 1 to be the lowest priority and jobs of class \(P\) to be the highest priority. The job arrivals for class \(i\) come from a Poisson process at rate \(\lambda_i\). When a job of this class gets served by any of the servers in the queue, the service time is an exponentially distributed random variable with mean \(\mu^{-1}\). The traffic of priority class \(i\) offered to the queue is denoted by \(\rho = \lambda/\mu\). We consider here queues with both pre-emptive and non-pre-emptive priority disciplines. The approach suggested may be applied to both single-server and multi-server queues. Moreover, queues with both finite and infinite buffers may be analysed using the suggested approach.

In this section, we do not actually do a complete derivation of the state probabilities of a M/M/n/K queue. Instead, we outline the way in which such a queue may actually be analysed by solving its balance equations. As is usual with any M/M/-/- type queue, the basic approach to its solution is to define the system-states appropriately so that a state transition diagram of the queue may be drawn. Using the exponential (memory-less) distribution of the inter-arrival and service times, the probability flows in the state transition diagram are identified. The balance equations are then obtained by choosing proper closed boundaries. Thereafter we equate the flow across each boundary. These balance equations are then solved along with the normalisation condition (i.e. sum of the probabilities of all states equal to unity) to obtain the state probabilities. Once these have been obtained, the performance parameters of interest may be suitably calculated. In the following, we consider the cases of pre-emptive and non-pre-emptive priorities separately. For each case, we first give the analytical approach for the M/M/1 case and then show how it may be used to handle the M/M/1/K queue and the M/M/n/K queue.

8.1.1 M/M/-/- Queue with Pre-emptive Priority

We first consider the simple M/M/1 queue with infinite buffers. This queue may be analysed by drawing a state transition diagram with the states given by the P-tuple \((n_1, \ldots, n_P)\) where \(n_i\) is the number in the system of priority class \(i\). Note that the server will always be engaged by a job of the highest priority class present in the system, i.e. by a job of class \(j\) with service rate \(\mu_j\) if \(n_j \geq 1\) and \(n_{j+1} = \ldots = n_P = 0\).

The state transition diagram for a 2-priority M/M/1 queue with infinite buffers operating with a pre-emptive priority discipline is shown in Fig. 7.1. Since all the flows are known, writing and solving the corresponding balance equations with the normalisation condition may find the equilibrium probabilities of each of the states. Some of the balance equations for this case are given below as an example.
Since the rules for transition between the states may be stated logically, we can actually use a computer program to generate the actual balance equations. These equations may then be given to an appropriate mathematical package to get the state probabilities. For the infinite buffer case, we may have some computational difficulties because of the infinite number of states. If this happens to be the case, then a convenient approximation would be to limit the state space to some large values of the total number in the queue and solve this approximate model instead.

This would also actually be the way one can solve a $M/M/-/-$ pre-emptive priority system with finite buffers. As an example, consider the 2-priority $M/M/1/3$ queue with pre-emptive priority. In this case, the state transition diagram will be as shown in Fig. 7.2. Note that unlike Fig. 7.1, Fig. 7.2 shows all the states of the system. Moreover, the circled states are the ones where the system is full and any jobs arriving in these states will be lost.

\[
\begin{align*}
p_0(\lambda_1 + \lambda_2) &= p_{0,1}\mu_2 + p_{1,0}\mu_1 \\
p_{0,1}(\lambda_1 + \lambda_2 + \mu_1) &= p_{0,2}\mu_2 + p_{0,1}\lambda_2 \\
p_{1,0}(\lambda_1 + \lambda_2 + \mu_1) &= p_{2,0}\mu_1 + p_{1,1}\mu_2 + p_0\lambda_1 \\
p_{1,1}(\lambda_1 + \lambda_2 + \mu_1) &= p_{1,0}\lambda_2 + p_{1,2}\mu_2 + p_{0,1}\lambda_1
\end{align*}
\] (8.1)

Figure 8.1. State transition diagram for 2-priority M/M/1 queue with pre-emptive priority

Figure 8.2. State transition diagram for a 2-priority M/M/1/3 Queue with Pre-emptive Priority
The following balance equations may be written for this $M/M/1/3$ system.

\[
\begin{align*}
    p_0(\lambda_1 + \lambda_2) &= p_{0,1}\mu_2 + p_{1,0}\mu_1 \\
    p_{0,1}(\lambda_1 + \lambda_2 + \mu_2) &= p_{0,2}\mu_2 + p_0\lambda_2 \\
    p_{1,0}(\lambda_1 + \lambda_2 + \mu_1) &= p_{2,0}\mu_1 + p_{1,1}\mu_2 + p_0\lambda_1 \\
    p_{1,1}(\lambda_1 + \lambda_2 + \mu_2) &= p_{1,0}\lambda_2 + p_{1,2}\mu_2 + p_{0,1}\lambda_1 \\
    p_{1,2}\mu_2 &= p_{1,1}\lambda_2 + p_{0,2}\lambda_1 \\
    p_{2,1}\mu_2 &= p_{1,1}\lambda_1 + p_{2,0}\lambda_2 \\
    p_{0,2}(\lambda_1 + \lambda_2 + \mu_2) &= p_{0,1}\lambda_2 + p_{0,3}\mu_2 \\
    p_{2,0}(\lambda_1 + \lambda_2 + \mu_1) &= p_{1,0}\lambda_1 + p_{2,1}\mu_2 + p_{3,0}\mu_1 \\
    p_{0,3}\mu_2 &= p_{0,2}\lambda_2 \\
    p_{3,0}\mu_1 &= p_{2,0}\lambda_1
\end{align*}
\]  

(8.2)

with the normalisation equation as follows

\[
\begin{align*}
    p_0 + p_{0,1} + p_{1,0} + p_{1,1} + p_{2,0} + p_{1,2} + p_{2,1} + p_{3,0} + p_{3,3} = 1
\end{align*}
\]  

(8.3)

To solve for the state probabilities in this case, we need any nine of the ten equations of (8.2) and the normalisation condition. Note that in this case, the job loss probability (or the blocking probability) will be $p_{1,2} + p_{2,1} + p_{3,0} + p_{3,3}$.

If the $M/M/-/-$ pre-emptive multi-priority queue has more than one server, then that can be taken into account while drawing the state transition diagram. Specifically, consider the situation where there are $n_P$ jobs of priority class $P$ and $c$ servers in the system. If $c < n_P$ then all the servers will be taken by $c$ of the $n_P$ (class $P$) jobs and no free servers will be available for lower priority jobs (if any). In this case, the overall service rate will be $c$.

On the other hand, if $c > n_P$, then the $n_P$ high priority jobs will each get a server and their overall service rate will be $n_P\mu_P$. The remaining $(c-n_P)$ servers will be available for the jobs of priority class $P-1$ and lower and will be used by these jobs according to the number in these classes. This approach will allow us to define the flows between each of the states in the state transition diagram. Note that the representation of states in the state transition diagram may still be done as the $P$-tuple $(n_1, \ldots, n_P)$ where $n_i$ is the number in the system of priority class $i$.

Since the priority discipline is pre-emptive, we do not need to separately specify the priority classes of the jobs engaging the servers. An example of this approach for obtaining the state transition diagram of a $M/M/2/3$ queue is shown in Fig. 7.3. Using this, one can follow the same solution approach as the one outlined for the $M/M/1/3$ queue earlier to obtain the various state probabilities.
8.1.1.1 Mean Waiting Time for Pre-emptive Resume Queuing Discipline – M/M/n

We now assume that an ongoing service is interrupted by the arrival of a customer with a higher priority. Later the service continues from where it was interrupted. This situation is typical for computer systems, which will be covered by a concrete example later in this section. For a customer with the priority $p$ there is no customer with lower priority. The mean waiting time $W_p$ for a customer in class $p$ consists of two contributions.

- Waiting time due to customers with higher or same priorities that are already in the queuing system. This is the waiting time experienced by a customer in a system without priority where only the first $p$ classes exists:

$$\frac{V_p}{1 - A_p}, \text{ where } V_p = \sum_{i=1}^{p} \frac{\lambda_i}{2} \cdot m_{2,i}$$  \hspace{1cm} (8.4)

is the expected remaining service time due to customers with a higher or the same priority and $A_p$ is given by

$$A_p = \sum_{i=0}^{p} A_i$$  \hspace{1cm} (8.5)

- The waiting time due to the customers with higher priority who arrive during the waiting time or service time and overtake the customer considered:

$$(W_p + s_p) \cdot \sum_{i=1}^{p-1} s_i \cdot \lambda_i = (W_p + s_p) \cdot A_{p-1}$$  \hspace{1cm} (8.6)

We thus get:

$$W_p = \frac{V_p}{1 - A_p} + (W_p + s_p) \cdot A_{p-1}$$  \hspace{1cm} (8.7)
This can be rewritten as follows:

\[ W_p (1 - A'_{p-1}) = \frac{V_p}{1 - A_p} + s_p \cdot A'_{p-1} \]  

resulting in:

\[ W_p = \frac{V_p}{(1 - A'_{p-1}) \cdot (1 - A_p)} + \frac{A'_{p-1} \cdot s_p}{1 - A'_{p-1}} \]  

In the same way we may write the formula for the average waiting time for the SJF–queuing discipline with pre-emptive resume. The total response time becomes:

\[ T_p = W_p + s_p \]  

8.1.1.2 Mean Queue Length for Pre-emptive Resume Queuing Discipline – M/M/n

The main waiting time in this case can be easily obtained by using the Little’s theorem since it is valid for all queuing systems. It gives the following expression.

\[ L_p = \lambda \cdot W_p = \lambda \left( \frac{V_p}{(1 - A'_{p-1}) \cdot (1 - A_p)} + \frac{A'_{p-1} \cdot s_p}{1 - A'_{p-1}} \right) \]  

8.1.2 M/M/-/- Queue with Non-Pre-emptive Priority

The basic approach to analysing an M/M/-/- queue with a non-pre-emptive priority discipline would be fairly similar to the approach suggested for the case where the priority discipline is pre-emptive in nature. The state definition would need to be modified in this case in order to describe the job class currently being served for each of the servers.

This is required, as unlike the M/M/-/- queue with pre-emptive priority, the ongoing service (if any) is not interrupted in the non-pre-emptive case because of the arrival of a job of a higher priority class than the one currently being served at a server (when no free servers are available).

For a M/M/n/K queue with P priority classes of jobs, a possible state descriptor would be a \((P + n)\)-tuple of the following form.

\textit{State Representation} - \((n_1, \ldots, n_P, s_1, \ldots, s_n)\) where \(n_j\) is the number of jobs of class \(j\) in the system \(j=0,1,\ldots,P\) and \(s_k\) represents priority class of the service currently on-going at server \(k\) \((k=1,\ldots,n)\). Note that \(n_1+\ldots+n_P \leq K\) for a finite capacity system.
An alternative representation using a \(2_p\)-tuple may give a more compact representation if the number of priority classes is less than the number of servers. In this alternative approach, the state description will be of the following form.

State Representation - \((n_1, \ldots, n_p, s_1, \ldots, s_P)\) where \(n_j\) is the number of jobs of class \(j\) in the system \(j=0,1,\ldots,P\) and \(s_k\) represents the number of servers currently busy serving jobs of priority class \(k\), \((k=1,\ldots,P)\).

As an example, we consider using the first method of representation to represent the states of a \(M/M/1/3\) 2-priority queue following a non-pre-emptive priority discipline. The state diagram for this queue is shown in Fig. 8.4.

For this, we can write balance equations in the same way as it was done earlier. These are given in (8.4).

\[
\begin{align*}
p_0(\lambda_1 + \lambda_2) &= p_{0,1,2}\mu_2 + p_{1,0,1}\mu_1 \\
p_{0,1,2}(\lambda_1 + \lambda_2 + \mu_2) &= p_0\lambda_2 + p_{0,2,3}\mu_2 \\
p_{1,0,1}(\lambda_1 + \lambda_2 + \mu_1) &= p_0\lambda_1 + p_{2,0,1}\mu_1 + p_{1,1,2}\mu_2 \\
p_{1,1,2}(\lambda_1 + \lambda_2 + \mu_1) &= p_{1,0,1}\lambda_2 \\
p_{1,1,2}(\lambda_1 + \lambda_2 + \mu_2) &= p_{0,1,2}\lambda_1 + p_{1,2,2}\mu_2 + p_{2,1,1}\mu_1 \\
p_{0,2,2}(\lambda_1 + \lambda_2 + \mu_2) &= p_{0,1,2}\lambda_2 + p_{1,2,1}\mu_1 + p_{0,2,3}\mu_2 \\
p_{2,0,1}(\lambda_1 + \lambda_2 + \mu_1) &= p_{1,0,1}\lambda_1 + p_{2,1,3}\mu_2 + p_{3,0,1}\mu_1 \\
p_{1,2,2}\mu_2 &= p_{1,1,2}\lambda_2 + p_{0,2,2}\lambda_1 \\
p_{2,1,2}\mu_2 &= p_{1,1,2}\lambda_1 \\
p_{2,1,1}\mu_1 &= p_{1,1,1}\lambda_1 + p_{2,0,1}\lambda_2 \\
p_{1,2,1}\mu_1 &= p_{1,1,1}\lambda_2 \\
p_{0,3,2}\mu_2 &= p_{0,2,2}\lambda_2 \\
p_{3,0,1}\mu_1 &= p_{2,0,1}\lambda_1
\end{align*}
\]
We would need any twelve of the thirteen equations of (8.4) along with the normalisation condition

\[
P_0 + P_{1,0,1} + P_{0,1,2} + P_{0,2,1} + P_{0,3,2} + P_{2,0,1} + P_{3,0,1} \\
+ P_{1,1,1} + P_{1,1,2} + P_{1,2,1} + P_{2,1,1} + P_{2,1,2} + P_{2,2,1} = 1
\]

(8.5)

to obtain the actual state probabilities. Once these have been calculated, one can use them to obtain the usual queue performance parameters. Note that, in this case, the loss probability will be given by \((P_{0,3,2} + P_{3,0,1} + P_{1,2,2} + P_{2,1,2} + P_{1,2,1} + P_{2,1,1})\) as the sum of the probability of all those states where the system is full, i.e. the total number in the system is three for this \(M/M/1/3\) queue.

As for the case of the \(M/M/-/-\) queue with pre-emptive priority classes, the approach given above for the non-pre-emptive \(M/M/1/3\) queue may also be generalised. Using the same approach, one can also analyze queues where

- The number of priority classes are more than two
- The buffer capacity is infinite
- The queue has more than one server.

Note that the complexity of the approach suggested here is basically because of the large number of states in the state transition diagram and the resultant complexity in the balance equations. Since the rules for transitions between states are straightforward, one can actually use a computer program to identify the transitions between states and directly write the corresponding balance equations. Using an appropriate numerical package to get the actual state probabilities, even if it is large, may directly solve this set of equations. These may then be used to obtain the queue performance parameters required.

### 8.1.2.1 Mean Waiting Time for Non-pre-emptive Queuing Discipline (M/M/n)

We may also generalise Erlang classical waiting time system \(M/M/n\) with non-pre-emptive queuing disciplines, when all classes of customers have the same exponentially distributed service time distribution with mean value \(s = \mu^{-1}\). Denoting the arrival intensity for class \(i\) by \(\lambda_i\), we have the mean waiting time \(W_p\) for class \(p\):

\[
W_p = V + \sum_{i=1}^{p} \frac{S}{n} \cdot L_i + W_p \sum_{i=1}^{p-1} \frac{S}{n} \cdot \lambda_i
\]

(8.6)

\[
W_p = E_{2,n}(A) \cdot \frac{S}{n} + \sum_{i=1}^{p} \frac{S \cdot \lambda_i}{n} \cdot W_i + W_p \sum_{i=1}^{p-1} \frac{S}{n} \cdot \lambda_i
\]

(8.7)

\(A\) is the total offered traffic for all classes whereas \(V\) is the residual service time for the customer under service when the customer we consider arrives. It is given by the following formula.

\[
V = \sum_{i=1}^{N} \frac{\lambda_i}{2} \cdot m_{2i}
\]

(8.8)
In (8.8) \( m_{2i} \) is the second moment of the service time distribution of the \( i \)'th class. The probability \( E_{2,n}(A) \) for waiting time is given by Erlang-C formula, and customers are terminated with the mean inter-departure time \( s/n \) when all customers are busy. In general we find:

\[
W_p = \frac{n \cdot s \cdot E_{2,n}(A)}{[n - A_{p-1}][n - A_p]} \quad (8.9)
\]

### 8.1.2.2 Mean Queue Length for Non-pre-emptive Queuing Discipline (M/M/n)

Similar with the mean waiting time for pre-emptive resume queuing discipline we can also apply here the Little’s theorem in order to give the expression for the mean waiting for \( M/M/n \) with non-preemptive priority.

\[
L_p = \lambda \cdot W_p = \lambda \left( \frac{n \cdot s \cdot E_{2,n}(A)}{[n - A_{p-1}][n - A_p]} \right) \quad (8.10)
\]

### 8.2 Concrete example for a Non-pre-emptive Priority System

The explosion of data-intensive applications and high quality of service requirements has caused a change in traditional storage systems. The server-to-disk approach is being replaced by storage area networks (SAN). Emerging as a data communication platform, SAN interconnect servers and storage devices to create a pool of storage that users can access directly. A storage-networking environment consists of three components: servers, storage devices (disks) and interconnect devices. In this section, we propose an analytical model for disks in SAN. Assuming a finite disk buffer and non-pre-emptive priority of read requests over write requests, we model a typical disk in SAN as a 2-priority \( M/M/1/N \) queue and evaluate its performance measures such as the blocking probability and response times for both read and write requests. In traditional storage systems, each server has its own private connection to storage, as illustrated in Fig. 8.5. Servers and clients share a communication network, which is used both for messaging and I/O related traffic. However, such a storage system is now suffering from problems such as unavailability of data in case of a failure on server, bandwidth saturation during backups and many other limitations when implementing large configurations.

![Figure 8.5 A traditional storage system](image_url)
Traditional storage systems are replaced rapidly by storage area networks (SAN). Essentially, SAN uncouple storage devices from the LAN and put them on their own network (Fig. 8.6). A high-speed network similar to LAN, SANs can interconnect technologies similar to the ones used in LANs, namely hubs and switches. SANs provide easy access for multiple servers to multiple storage devices (such as disks, disk arrays, and tape drives) in the same way that LANs allow the client easy access to many servers. SANs enable storage to be externalized from the server and hence allow storage to be shared among multiple servers without impacting the performance of the primary communication network. SANs improve traditional storage systems in many aspects such as computer clustering, topological flexibility, fault tolerance, high availability, and remote management.

![Figure 8.6 A typical SAN environment](image)

Disks are a central part of a SAN, which communicate with the server by means of two I/O operations: read and write operations. Modelling disks in a SAN is a difficult task because of the burst I/O arrival patterns and complicated response times for I/O requests. In order simulation modelling of disks in SANs to be simplified it is assumed equal priority of read and write requests, while in reality read requests should have higher priority than write requests in order to improve read response time. Moreover it could be also considered that the disk buffer has infinite capacity, which is unreasonable in practice. Below in this section, we present an analytical queuing model for disks in a SAN by considering a non-pre-emptive priority policy with read requests having higher priority than write requests, and finite disk buffering.

### 8.2.1 Disks in SAN

Physically a SAN consists of three main components: server systems, storage devices (mainly magnetic disks) and interconnect devices including the interconnect network (switch fabric). Both servers and disks are attached to the interconnect network by means of a network interface card.

#### 8.2.1.1 Read and Write Requests

Servers initiate the communication with disks. Disks take actions according to the requests previously sent by servers. For I/O read requests, the server first sends a request message to the selected disk indicating the number of sectors required, and the initial sector address in the disk. Once the disk has processed the request, it accesses the information, and finally sends the data back to the server through the storage network. The I/O read request lasts since the server initiates it until
the total amount of data is received at the server. For the I/O write requests, servers start them by sending the data and the initial sector address to the selected disk. Once the data is stored, the disk sends back an acknowledge message to the server. The I/O write request lasts since the server initiates it until the acknowledgement is received.

Both the acknowledgement for write requests and the read request are short messages of a small amount of bytes (typically 16 or 32 bytes). The size of data messages usually ranges from a few bytes, say, 1, 2, 4 or 8 KB, until several KB (32 or 64 KB).

8.2.1.2 Disk Response Time

The disk response time of an I/O request includes three main components: seek time, rotational latency, and transfer time. Seek time is modelled by the function:

$$a\sqrt{x-1} + b(x-1) + c$$

(8.11)

where $x$ is the distance in cylinders. Coefficients $a$, $b$ and $c$ depend the disk geometry. Rotational latency lasts since the disk head arrives at the selected cylinder until the first sector of the request passes under the head.

Finally, transfer time is the time needed to transfer the data from/to the disk to/from the disk buffer.

8.3 Analysis Model

In this section, we present a finite queuing model for a typical disk in SAN and evaluate its performance measures such as the blocking probability and mean response time. The state transition diagram for $N = 3$ is same as that in Fig. 8.3.

8.3.1 Basic Assumptions

We model a typical disk in a SAN by a single server queue with a FIFO discipline. We assume the disk buffer is finite with size for $N$ requests. Disk response depends on the operation type requested by the server. If it is a read request, the disk sends the data back to the server. On the other hand, if it is a write request, only the acknowledgement message is generated.

To improve read response time, we assume read requests have non-pre-emptive priority over write requests. To achieve a tractable model, we make the following simplifying assumptions. Both read and write requests arrive at the disk according to a Poisson process with rate $\lambda_1$ and $\lambda_2$, respectively.

We assume the disk response time for both read and write requests to be exponentially distributed with mean $1/\mu_1$ and $1/\mu_2$, respectively.
8.3.2 Finite Queuing Model

8.3.2.1 State Probabilities

The disk can be modelled as a 2-priority \( M/M/1/N \) queue (non-pre-emptive priority). The state representation is a 3-tuple in the form \((n_1, n_2, s)\). Here, \(n_1\) and \(n_2\) represent the number of write and read requests, respectively, and \(s\) \((s = 0, 1, 2)\) indicates the server is now empty, serving a read request and serving a write request, respectively. There is only one state corresponding to \(s = 0\), which is the initial state \((0,0,0)\) and will be simplified as \(0\) hereafter. Note that \(n_1 + n_2 \leq N\). Let us take \(N = 3\) for example. The approach can be easily extended in the usual fashion to other bigger values of \(N\). In case \(N = 3\), there are a total of 13 states, whose transition diagram is shown in Fig.8.3. The balance equations for this non-pre-emptive queue and the normalization condition are the same to that from (8.4) and (8.5), respectively.

8.3.2.2 Performance Measures

We are interested in the following performance measures:

- \(\bar{N}_1\) : the mean number of write requests;
- \(\bar{N}_2\) : the mean number of write requests;
- \(\bar{N}\) : the mean number of requests;
- \(T_p\) : the mean throughput of the server;
- \(\bar{T}\) : the mean response time of requests;
- \(P_B\) : the blocking probability.

A little thought will show that

\[
\bar{N}_1 = \sum_{n_1} n_1 \cdot p(n_1, n_2, s) \\
= p_{1,0,1} + p_{1,1,1} + p_{1,1,2} + p_{1,2,1} + p_{1,3,2} \\
+ 2(p_{2,2,1} + p_{1,1,1} + p_{2,1,1}) + 3p_{3,0,1}.
\]

\[
\bar{N}_2 = \sum_{n_2} n_2 \cdot p(n_1, n_2, s) \\
= p_{0,1,1} + p_{1,1,1} + p_{1,1,2} + p_{2,1,1} + p_{3,1,2} \\
+ 2(p_{4,2,1} + p_{3,1,1} + p_{2,1,2}) + 3p_{3,0,1}.
\] (8.12)

\[
\bar{N} = \sum_{n_1, n_2} (n_1 + n_2) \cdot p(n_1, n_2, s) = \bar{N}_1 + \bar{N}_2.
\]

\[
T_p = \mu_1 \sum_{n_1} p(n_1, n_2, s) + \mu_2 \sum_{n_2} p(n_1, n_2, s)
\]
By Little’s law, the mean response time of requests is

\[ \overline{T} = \frac{N}{T_p} \]  \hspace{1cm} (8.13)

A new request (either write or read) will be blocked when \( n_1 + n_2 = 3 \). Thus, the blocking probability \( P_B \) is

\[ P_B = \sum_{n_1+n_2=3} p(n_1,n_2,2) \]

\[ = P_{0,1,2} + P_{1,2,2} + P_{2,1,1} + P_{2,2,2} + P_{3,0,1} \]  \hspace{1cm} (8.14)

### 8.3.2.3 Numerical Results

In this section, we present numerical results to illustrate our analytical model. In Fig. 8.7 and 8.8, we fix \( \lambda_2 = 1, \mu_1 = 15, \mu_2 = 20 \) and let \( \lambda_1 \) increase from 1 to 10. We see that both blocking probability and mean response time are increasing as \( \lambda_1 \) increase. In Figures 8.9 and 8.10, we fix \( \lambda_1 = 1, \mu_1 = 15, \mu_2 = 20 \) and let \( \lambda_2 \) increase from 1 to 10. Again we see that both blocking probability and mean response time are increasing as \( \lambda_2 \) increases.

![Figure 8.7 Blocking Probability (Fixed \( \lambda_2 \))](image)

![Figure 8.8. Mean Delay (Fixed \( \lambda_2 \))](image)

![Figure 8.9 Blocking Probability (Fixed \( \lambda_1 \))](image)

![Figure 8.10. Mean Delay (Fixed \( \lambda_1 \))](image)
9. Systems with Limited Waiting Time

We often need strategy for limiting the number of customers in the system. One solution is to impose an upper limit to the allowed waiting time. This may be due to impatient customers, which give up waiting, or due to that the system enforces them to leave the queue.

In the following we consider Erlang delay system $M/M/n$ having Poisson arrival process ($\lambda$), exponential service times ($\mu$), $n$ servers and FCFS-queuing discipline.

9.1 Exponentially distributed Time-out Interval

Typical example of a queuing system with limited waiting time is Palm's waiting time model, where customers in queue leave the system with rate $\mu_r$. We then get a system with arrival rates $\lambda_i$, respectively departure rates $\mu_i$, in state $i$:

$$\lambda_i = \lambda \quad 0 \leq i < \infty \quad (9.1)$$

$$\mu_i = \begin{cases} i \cdot \mu & 0 \leq i \leq n, \\ n \cdot \mu + (i - n) \cdot \mu & n \leq i \end{cases} \quad (9.2)$$

This system is a classical queuing system, where waiting customers becomes impatient and leave the system after an exponentially distributed waiting time. The classical systems are special cases of this:

$\mu_r = 0$ : Erlang's waiting time system ,
$\mu_r = \mu$ : Molina's Lost-Calls-Held system ,
$\mu_r = \infty$ : Erlang's Lost-Calls-Cleared system .

9.2 Constant Time-out Interval

A time-out interval is often applied in technical systems after a constant time interval. In old switching systems this was implemented by heating a resistor, in new digital switching systems it is implemented by software.

For example, the dial tone is terminated, if no dialling takes place within approximately 10 seconds. Also, it is only possible to give ringing tone to a B-subscriber during a limited time interval (2-3 minutes) if there is no answer.
9.2.1 State Probabilities – Systems with Constant Time-out Interval

It has been shown that the state probabilities satisfy the following relations:

\[ p(i) = p(0) \cdot \frac{A}{i!} p(i), \quad 0 \leq i \leq n \] (9.3)

\[ p(n + i) = p(n) \cdot \frac{\lambda^i}{\prod_{j=1}^{n} \left( n \cdot \mu + \frac{1}{c_{n+j}} \right)}, \quad 1 \leq i \leq \infty \] (9.4)

In terms of Erlang – C:

\[ p(n + i) = E_{2,n}(A) \frac{n - A}{n} \left( \frac{\lambda}{n \mu + \frac{1}{c_{n+j}}} \right), 1 \leq i \leq \infty \] (9.5)

From (8.4) we can see that the state probabilities become increasingly complicated as \( i \) approaches infinity. Even for small values of \( i \), the expressions are much difficult. To elaborate, we present the first value for \( i = 1 \).

\[ p(n + 1) = E_{2,n}(A) \frac{n - A}{n} \frac{\lambda}{n \mu + \frac{1}{c_{n+j}}} \left( \exp\left(\frac{\mu \tau}{4}\right) \right) \left( \frac{\pi}{\sqrt{\pi} \cdot \text{Erf}\left(\frac{\sqrt{\mu \tau}}{2}\right)} \right) \] (9.6)

9.2.2 Queue Length Distribution – Systems with Constant Time-out Interval

The probability density function of the QLD is determined by the same expression as in the \( M/M/n \) case.

\[ p_s(L) = p(n + 1) \delta (L) + p(n + 2) \delta (L + 1) + ... \] (9.7)

However, since the state probabilities are so complicated it will not be presented in further detail. The same goes for QLD. Only the general idea is given below.

\[ F_s(L) = 1 - P(L > l) = 1 - \sum_{i=\infty}^{l} p(i) \] (9.8)
9.2.2.1 Mean Queue Length - Systems with Constant Time-out Interval

Utilizing the well-known relation for determining QLD i.e.

\[ L_n = 0 \sum_{i=0}^{n} p(i) + \sum_{i=n+1}^{\infty} (i-n)p(i) \]  \hspace{1cm} (9.9)

\[ L_n = \sum_{i=n+1}^{\infty} (i-n)p(i) \]  \hspace{1cm} (9.10)

This is also very complicated to derive, so we will not go any further.

9.2.3. Waiting Time Distribution - Systems with Constant Time-out Interval

We consider Erlang delay system \((n, \lambda, \mu)\) and choose the mean service time \(1/\mu\) as time unit. We introduce the additional assumption that waiting calls give up waiting after a constant time interval \(\tau\), and that the queuing discipline is FCFS. It can be shown that the waiting time distribution for this becomes:

\[ W(t) = \frac{1 - E_{2,n}(A) \cdot e^{-(A - \tau)\mu}}{1 - \frac{n}{A} \cdot E_{2,n}(A) \cdot e^{-(A - \tau)\mu}} \]  \hspace{1cm} (9.11)

The formula (9.11) is valid only for \(A \neq n\). When \(A > n\) the exponent becomes positive, and \(E_{2}(A) = E_{2,n}(A)\) should not be considered as a probability, but simply as a values obtained from (2.22).

\[ E_{2,n}(A) = \frac{nE_{1,n}(A)}{n - A \cdot [1 - E_{1,n}(A)]} \]  \hspace{1cm} (9.12)

where \(E_{1,n}(A)\) is Erlang-B formula, which is valid for all \(A > 0\). By inserting this in (9.11) we get

\[ W(t) = \frac{n - A [1 - E_{1}(A)] - n \cdot E_{1}(A) \cdot e^{-(A - \tau)\mu}}{n - A [1 - E_{1}(A)] - A \cdot E_{1}(A) \cdot e^{-(A - \tau)\mu}} \]  \hspace{1cm} (8.13)

For \(t = \tau = 0\), \(W(0)\) becomes equal to the probability for experiencing a waiting time zero, i.e. immediate service, which in this case as expected becomes \(1 - E_{1,n}(A)\).

For \(\tau \rightarrow \infty\), we must demand that \(A < n\), and we find the waiting time distribution for the Erlang-C system (4.11). In the general case \(0 < T < \infty\) we get from (9.11):
\[ p(\text{immediate}) = W(0) = \frac{1 - E_2(A)}{1 - \frac{A}{n} E_2(A) \cdot e^{-(n-A)t}} \quad (9.14) \]

\[ p(\text{immediate}) = p[T = \tau] = 1 - p[T < \tau] \]

\[ p(\text{immediate}) = 1 - W(\tau) = 1 - \frac{1 - E_2(A) \cdot e^{-(n-A)t}}{1 - \frac{A}{n} E_2(A) \cdot e^{-(n-A)t}} , \quad A \neq n \quad (9.15) \]

For \( A = n \) we cannot calculate \( E_2 = E_{2,n}(A) \), because \( A - n \) appears in the denominator. If we apply the expression (9.13), both the numerator and the denominator becomes equal to zero for \( A = n \). Therefore, we apply l'Hospital rule and differentiate numerator and denominator separately with respect to \( A \). The derivative of Erlang-B formula for \( A = n \) becomes:

\[ E_1' = \frac{\partial E_1}{\partial A} = \left( \frac{n}{A} - 1 \right) E_1 + E_2^2 = E_i^2 \quad \text{for} \quad A = n \quad (9.16) \]

We thus get:

\[ W(t) = \frac{f(A)}{g(A)} \quad (9.17) \]

\[ \frac{f(A)}{g(A)} = \frac{- (1 - E_1) + A \cdot E_1' - n \cdot E_1 \cdot t \cdot e^{-(n-A)t} - E_1' \cdot n \cdot e^{-(n-A)t}}{- (1 - E_1) + A \cdot E_1' - E_1 \cdot [1 + A \cdot \tau] \cdot e^{-(n-A)t} - E_1' \cdot A \cdot e^{-(n-A)t}} \quad (9.18) \]

As \( A \) approaches \( n \), both \( f(A) \) and \( g'(A) \) has a limit value different from zero, and according to l’Hospital's rule \( W(t) \) will have the limiting value:

\[ \lim_{A \to n} W(t) = \lim_{A \to n} \frac{f'(A)}{g'(A)} \quad (9.19) \]

\[ \lim_{A \to n} W(t) = \frac{E_1 (A \cdot t - 1) + 1}{A \cdot E_1 \cdot \tau + 1} , \quad A = n, \ 0 \leq t \leq \tau \quad (9.20) \]

Following the line of thought as in the previous sections, we can derive the probability density function (PDF) as the derivative from waiting time distribution (WTD). Hence,

\[ p_S(t) = \frac{(n - A)E_{2,n}(A)e^{-(n-A)t}}{1 - \frac{A}{n} E_{2,n}(A) e^{-(n-A)t}} \quad (9.21) \]
9.2.3.1 MATLAB Code for PDF and WTD of Systems with Constant Time-out Interval

% The following script computes the waiting time distribution (WTD) and the respective
% probability density function (PDF) that a system with n servers, constant time-out
% interval, % finite waiting line, Poisson arrival rate lambda, service rate (per server) mu, and
% intensity % A = lambda / mu will have. It uses the expressions (9.21) for PDF of WTD
% and (9.11) for % WTD. Mean service time s, measured in time units, time-out interval in
time units
s=1; mu = 1/s;
n = 20; tau = 3;
A = 19; L = 15;
C = erlangc (n, A);
t = 0:0.25:L;
nom_ps = (n - A)*C*exp (- (n - A)*t);
nom_Fs = 1-C*exp (- (n - A)*t);
denom = 1 - (A / n)*C*exp (- (n - A)*tau);
ps = nom_ps / denom; Fs = nom_Fs / denom;
ps (4*tau: 4*L) = 0;
Fs (4*tau: 4*L) = 1;
plot (t, ps, 'g');
hold on
plot (t, Fs, 'r');
xlabel ('Waiting time (Time Units)')
ylabel ('PDF (green), WTD (red)')
title ('Calculation of Probability Density Function and Waiting Time Distribution')
System with Constant Time-out

9.2.3.2. Mean Waiting Time - Systems with Constant Time-out Interval

MWT will not be calculated in detail since it is extremely difficult to do that. We should only confirm the Little theorem.

\[
W_n = \frac{L_n}{\lambda} = \frac{1}{\lambda} \sum_{i=n+1}^{\infty} (i - n) p(i) \tag{9.22}
\]
10. Concluding Remarks

Upon reading this report one can notice that the emphasis is placed on derivation of Erlang - C and some different representations of the formula as well. Furthermore, we concentrate on different queue configurations, such as: $M/M/n$ and $M/M/n/k$. We investigated the waiting time and queue length distribution for both of them as well as their expression for the mean values. It also involves having a good understanding of queuing and probability theory.

However, there are several occasions when it is often hard or impossible to analyze some of the queuing systems in an exact analytical way. For instance, in the case of $M/M/n$ with constant time-out interval, the final formulae for queue length distributions as well as the mean waiting time and mean queue length were not derived. Also, this applies to systems with finite number of waiting positions. This is something we have encountered during our work on this mini-task.
Appendix A – Gamma Function

The (complete) gamma function $\Gamma(n)$ is defined to be an extension of the factorial to complex and real number arguments. It is related to the factorial by $\Gamma(n) = (n-1)!$. It is analytic everywhere except at $z = 0, -1, -2, ...$ and the residue at $z = -k$ is

$$\text{Res} \Gamma(z) = \frac{(-1)^k}{k!} \quad (1)$$

There are no points $z$ at which $\Gamma(z) = 0$. The graphical representation of the gamma function is given on Fig. 1.

![Figure 1. Gamma Function](image)

The gamma function can be defined as a definite integral for $\Re[z] > 0$ (Euler's integral form).

$$\Gamma(z) = \int t^{z-1}e^{-t}dt = 2\int e^{-t}t^{2z-1}dt \quad (2)$$

or

$$\Gamma(z) = \left[ \ln \left( \frac{1}{t} \right) \right]^{z-1} \quad (3)$$
Appendix B - Error Function

\text{Erf}(z)$ is the "error function" encountered in integrating the normal distribution (which is a normalized form of the Gaussian function). It is defined by

\[
\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt
\]  \hspace{1cm} (4)

The graphical presentation of the error function is given on Figure 2.

Figure 2. Error Function
Glossary of Acronyms

ATM Asynchronous Transfer Mode
FCFS First Come First Served
FIFO First In First Out
HOL Head Of Line
IP Internet Protocol
LAN Local Area Network
LCFS Last In First Served
LIFO Last In First Out
MQL Mean Queue Length
MWT Mean Waiting Time
PASTA Poisson Arrivals See Time Average
PCT-I Pure Chance Traffic Type One
PDF Probability Density Function
QLD Queue Length Distribution
SAN Storage Area Networks
SIRO Service In Random Order
WTD Waiting Time Distribution
References:


