What is a queue?
In general, a queue is a line of people or things waiting to be handled, or a group of jobs waiting to be executed, usually in sequential order starting at the beginning or top of the line or sequence. In computer technology, a queue is a sequence of work objects that are waiting to be processed. The possible factors, arrangements, and processes related to queues is known as queueing theory
In computer science, queuing refers to lining up jobs for a computer or device. For example, if you want to print a number of documents, the operating system (or a special print spooler) queues the documents by placing them in a special area called a print buffer or print queue. The printer then pulls the documents off the queue one at a time. Another term for this is print spooling .

The order in which a system executes jobs on a queue depends on the priority system being used. Most commonly, jobs are executed in the same order that they were placed on the queue, but in some schemes certain jobs are given higher priority

Queuing Theory
Queueing Theory is the study of queues (sometimes called waiting lines). Most people are familiar with the concept of queues; they exist all around us in daily lives. Much of queueing theory is very complex. Almost all of it relies heavily on mathematics, especially statistics. Thus, most important results will be stated, but not derived in this course

A queue can be studied in terms of:

- The source of each queued item,
- how frequently items arrive on the queue,
- how long they can or should wait,
- whether some items should jump ahead in the queue,
- how multiple queues might be formed and managed, the rules by which items are enqueued and dequeued

Queuing Theory in Computer Science

In computer science, queueing theory is the study of queues as a technique for managing processes and objects in a computer. Queueing Theory can be used to describe real world queues, or more abstract queues, such as are often found in many branches of computer science, for example in operating system design.
The queues that a computer manages are sometimes viewed as being in stacks. In most systems, an item is always added to the top of a stack. In programming, a queue is a data structure in which elements are removed in the same order they were entered. This is often referred to as FIFO (first in, first out). In contrast, a stack is a data structure in which elements are removed in the reverse order from which they were entered. This is referred to as LIFO (last in, first out).

Basic Terminology of Queueing Theory

Terminology for study of queueing systems tends to be fairly standard. Some variation sometimes occurs, and where there are popular alternative forms of terminology, this will be made clear.

The three main concepts in queueing theory are customers, queues, and servers (service mechanisms). The meaning of these terms is reasonably self-evident. In general, in a queueing system, customers for the queueing system are generated by an input source. The customers are generated according to a statistical distribution (at least, that is the simplifying assumption made for modelling purposes) and the distribution describes their interarrival times, in other words, the times between arrivals of customers. The customers join a queue. At various times, customers are selected for service by the server (service mechanism). The basis on which the customers are selected is called the queue discipline. The head of the queue is the customer who arrived in the queue first. Another piece of terminology which is sometimes used is the tail of the queue. The meaning of this varies depends upon the context and the source. It normally means either all of the queue except the head or the last item in the queue, in other words the customer who arrived last and is at the back of the queue. Both uses are in common usage, and the terminology front and back of the queue will be used to describe the customers who arrived least recently and most recently (respectively) to avoid ambiguity.

We will now look at each of these pieces of terminology in more detail.

Input Source

The input source is a population of individuals, and as such is called the calling population. The calling population has a size, which is the number of potential customers to the system. The size can either be finite or infinite. As will become apparent, if the calling population is infinite, various simplifying assumptions can be made which make the process of modelling queues much easier. Most queueing models assume that the population is infinite.

Queue

Queues can be either infinite or finite. If a queue is finite, it can only hold a limited number of customers. Most queueing models assume an infinite queue, even though this is almost certainly not strictly true in the majority of applications of queueing theory. This assumption is made because it makes the modelling process simpler. Also, if the maximum queue size is significantly larger than the likely number of customers at any one time, then to all intents and purposes it is infinite in size. The amount of time which is a customer waits in the queue for is called the queueing time. The number of customers who arrive from the calling population and join the queue in a given period of time is modelled by a statistical distribution.

Queue Discipline

The queue discipline is the method by which customers are selected from the queue for processing by the service mechanisms (also called servers). The queue discipline is normally first-come-first-served (FCFS), where the customers are processed in the order in which they arrived in the queue, such that the head of the queue is always processed next. Most queueing models assume FCFS as the queue discipline, and only this discipline will be considered in any detail in this project.

Service Mechanism

The service mechanism is the way that customers receive service once they are selected from the front of a queue. A service mechanism is also called a server (in fact, this is the more common terminology). The amount of time which a customer takes to be serviced by the server is called the service time. A statistical distribution is used to model the service time of a server. Some queueing models assume a single server, some multiple servers. For most general analysis, most queueing models assume that the system has either a single server or allow the number of servers to become a variable. This convention is explored further in the section on Kendall notation.

Applications Of Queueing Theory

Queueing Theory has a wide range of applications, and this section is designed to give an illustration of some of these. It has been divided into 3 main sections,

- Traffic Flow,
- Scheduling and
- Facility Design and Employee Allocation.

The given examples are certainly not the only applications where queuing theory can be put to good use, some other examples of areas that queueing theory is used are also given.

Traffic Flow

This is concerned with the flow of objects around a network, avoiding congestion and trying to maintain a steady flow, in all directions.

Queueing on roads

Queues at a motorway junction, and queueing in the rush hour

Scheduling

- Computer scheduling

Facility Design and Employee Management

- Queues in a bank
- A Mail Sorting Office

Some Other Examples

- Design of a garage forecourt
- Airports - runway layout, luggage collection, shops, passport control etc.
- Hair dressers
- Supermarkets

- Restaurants
- Manufacturing processes
- Bus scheduling
- Hospital appointment bookings
- Printer queues
- Minimising page faults in computing

Elements of Queuing Systems

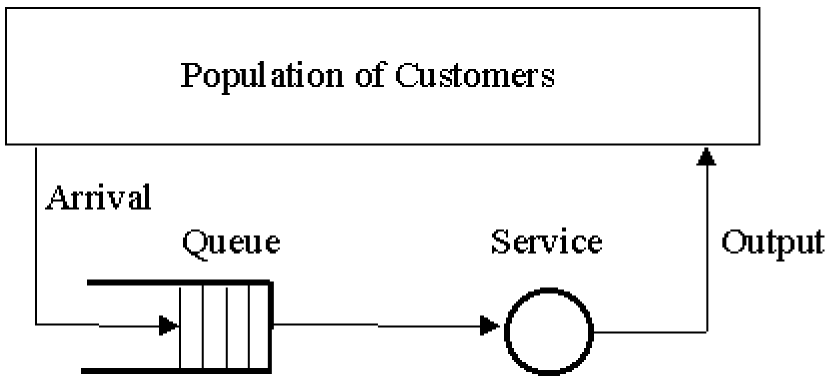


Figure 1

Population of Customers can be considered either limited (closed systems) or unlimited (open systems). Unlimited population represents a theoretical model of systems with a large number of possible customers (a bank on a busy street, a motorway petrol station). Example of a limited population may be a number of processes to be run (served) by a computer or a certain number of machines to be repaired by a service man. It is necessary to take the term "customer" very generally. Customers may be people, machines of various nature, computer processes, telephone calls, etc.

Arrival defines the way customers enter the system. Mostly the arrivals are random with random intervals between two adjacent arrivals. Typically the arrival is described by a random distribution of intervals also called*Arrival Pattern*.

Queue represents a certain number of customers waiting for service (of course the queue may be empty). Typically the customer being served is considered not to be in the queue. Sometimes the customers form a queue literally (people waiting in a line for a bank teller). Sometimes the queue is an abstraction (planes waiting for a runway to land). There are two important properties of a queue: *Maximum Size* and *Queuing Discipline*.

Maximum Queue Size (also called *System capacity*) is the maximum number of customers that may wait in the queue (plus the one(s) being served). Queue is always limited, but some theoretical models assume an unlimited queue length. If the queue length is limited, some customers are forced to renounce without being served.

Queuing Discipline represents the way the queue is organised (rules of inserting and removing customers to/from the queue). There are these ways:

1) FIFO (First In First Out) also called FCFS (First Come First Serve) - orderly queue.

2) LIFO (Last In First Out) also called LCFS (Last Come First Serve) - stack.

3) SIRO (Serve In Random Order).

4) Priority Queue, that may be viewed as a number of queues for various priorities.

5) Many other more complex queuing methods that typically change the customer’s position in the queue according to the time spent already in the queue, expected service duration, and/or priority. These methods are typical for computer multi-access systems.

Most quantitative parameters (like **average queue length**, **average time spent in the system**) do not depend on the queuing discipline. That's why most models either do not take the queuing discipline into account at all or assume the normal FIFO queue. In fact the only parameter that depends on the queuing discipline is the variance (or standard deviation) of the waiting time. There is this important rule (that may be used for example to verify results of a simulation experiment):

The two extreme values of the waiting time variance are for the FIFO queue (minimum) and the LIFO queue (maximum).

Theoretical models (without priorities) assume only one queue. This is not considered as a limiting factor because practical systems with more queues (bank with several tellers with separate queues) may be viewed as a system with one queue, because the customers always select the shortest queue. Of course, it is assumed that the customers leave after being served. Systems with more queues (and more servers) where the customers may be served more times are called *Queuing Networks.*

Service represents some activity that takes time and that the customers are waiting for. Again take it very generally. It may be a real service carried on persons or machines, but it may be a CPU time slice, connection created for a telephone call, being shot down for an enemy plane, etc. Typically a service takes random time. Theoretical models are based on random distribution of service duration also called *Service Pattern*. Another important parameter is the number of servers. Systems with one server only are called *Single Channel Systems*, systems with more servers are called *Multi Channel Systems*.

Output represents the way customers leave the system. Output is mostly ignored by theoretical models, but sometimes the customers leaving the server enter the queue again ("round robin" time-sharing systems).

Queuing Theory is a collection of mathematical models of various queuing systems that take as inputs parameters of the above elements and that provide quantitative parameters describing the system performance.

Because of random nature of the processes involved the queuing theory is rather demanding and all models are based on very strong assumptions (not always satisfied in practice). Many systems (especially queuing networks) are not soluble at all, so the only technique that may be applied is simulation.

Nevertheless queuing systems are practically very important because of the typical trade-off between the various costs of providing service and the costs associated with waiting for the service (or leaving the system without being served). High quality fast service is expensive, but costs caused by customers waiting in the queue are minimum. On the other hand long queues may cost a lot because customers (machines e.g.) do not work while waiting in the queue or customers leave because of long queues. So a typical problem is to find an optimum system configuration (e.g. the optimum number of servers). The solution may be found by applying queuing theory or by simulation.

Kendall's Notation for Classification of Queue Types

There is a standard notation for classifying queueing systems into different types. This was proposed by D. G. Kendall (1953) and exists in several modifications. The most comprehensive classification uses 6 symbols and are described by the notation:

A/B/s/q/c/p

where:

A	The arrival pattern (distribution of intervals between arrivals).
B	is the service pattern (distribution of service time or duration).
s	Number of servers
q	The queuing discipline (FIFO, LIFO, ...). Omitted for FIFO or if not specified.
c	System capacity or maximum total number of customers which can be accommodated in system. The value is omitted for unlimited queues
p	The population size (number of possible customers). This is omitted for open systems or infinite polulation

The arrival patterns (A) and service patterns (B) can take any of following distribution types:

M	Poisson (Markovian) process with exponential distribution of intervals or service duration respectively
D	Degenerate or Deterministic (known) arrivals and constant service duration
E_k	Erlang Distribution of intervals or service duration. (k = shape parameter)
G	General Distribution (arbitrary distribution) GI is a general (any) distribution with independent random values

<u>Notes</u>: If G is used for **A**, it it sometimes written GI. **c** is usually infinite or a variable, as is **p**. If **c** or **p** are assumed to be infinite for modelling purposes, they can be omitted from the notation (which they frequently are). If **p** is included, **c** must be, to ensure that one is not confused between the two, but an infinity symbol is allowed for **c**.

Examples

- D / M / *n* - This would describe a queue with a degenerate distribution for the interarrival times of customers, an exponential distribution for service times of customers, and *n* servers.
- E_k / E_l / 1 - This would describe a queue with an Erlang distribution for the interarrival times of customers (with a shape parameter of k), an exponential distribution for service times of customers (with a shape parameter of l), and a single server.
- M / M / *m* / K / N - This would describe a queueing system with an exponential distribution for the interarrival times of customers and the service times of customers, *m* servers, a maximum of K customers in the queueing system at once, and N potential customers in the calling population.
- **D/M/1** = Deterministic (known) input, one exponential server, one unlimited FIFO or unspecified queue, unlimited customer population.
- **M/G/3/20** = Poisson input, three servers with any distribution, maximum number of customers 20, unlimited customer population.
- **D/M/1/LIFO/10/50** = Deterministic arrivals, one exponential server, queue is a stack of the maximum size 10, total number of customers 50.

Poisson Distribution

In statistics and Probability, the Poisson Distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time and/or space if these events occur with a known average rate and independently of the time since the last event. The Poisson distribution can also be used for the number of events in other specified intervals such as distance, area or volume.

For instance, suppose someone typically gets 4 pieces of mail per day on average. There will be, however, a certain spread: sometimes a little more, sometimes a little fewer, once in a while nothing at all. Given only the average rate, for a certain period of observation (pieces of mail per day, phone calls per hour, etc.), and assuming that the process, or mix of processes, that produces the event flow is essentially random, the Poisson distribution specifies how likely it is that the count will be 3, or 5, or 10, or any other number, during one period of observation. That is, it predicts the degree of spread around a known average rate of occurrence. The Poisson process describes the distribution of the number of customers arriving in a fixed interval of time length, t. Because of its mathematical properties, the assumption of Poisson input in the analysis of queuing systems often leads to the easiest mathematics. Specifically, we have, as a consequence of the Markov property, the following important facts:

With Poisson input at rate λ, the distribution of the length of time from an arbitrary instant until the next am'va1 epoch is the same as the distribution of the length of time between successive arrival epochs, that is, exponential with mean λ^{-1}. Similarly, looking backward from an arbitrary instant, the distribution of the length of time separating an arbitrary instant and the preceding arrival epoch is also exponentially distributed with mean λ^{-1} (assuming, of course, that the process is not measured from a finite origin).

Arrivals occurring according to a Poisson process are often said to occur at random. This is because the probability of arrival of a customer in a small interval of length t is proportional to the length t, and is independent of the amount of elapsed time from the arrival epoch of the last customer. That is, when customers are arriving according to a Poisson process, a customer is as likely to arrive at one instant as any other, regardless of the instants at which the other customers arrive. However, in that case, the length I of the interval containing the points is known. One might conjecture, there- fore, that when it is known that a given number n of arrivals generated by a Poisson process have occurred in an interval (0,1), these arrival epochs are (conditionally) uniformly distributed throughout the interval. This is indeed true.

Poisson Distribution versus Exponential Distribution

Poisson Distribution and Exponential Distribution are two distinct distributions, but both relate to the same process. The exponential distribution is the probability distribution that describes the time between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate. It is the continuous analogue of the geometric distribution, and it has the key property of being memoryless. In addition to being used for the analysis of Poisson processes, it is found in various other contexts.

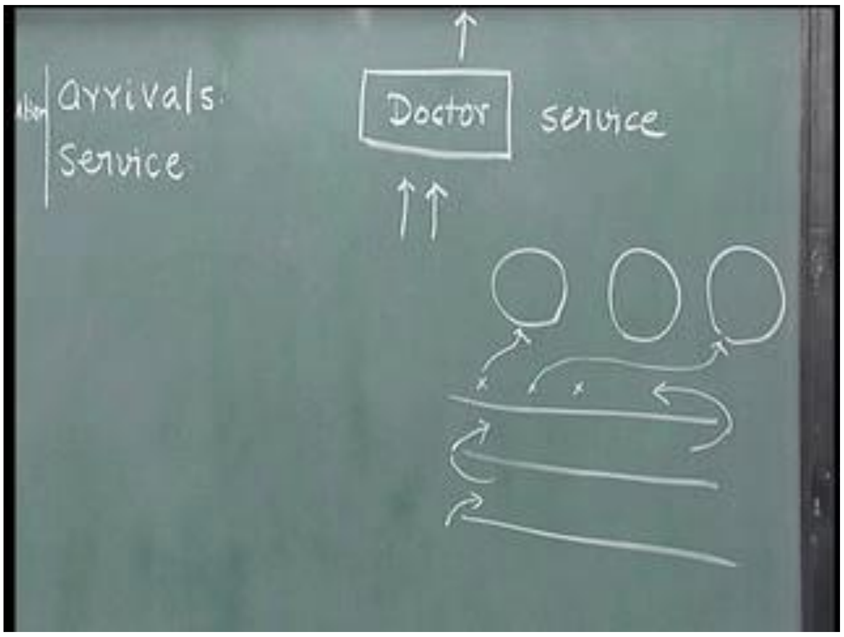
Given a **Poisson Process**, the following have an **Exponential Distribution**:
* the time until the first event
* the time from now until the next occurrence of an event
* the time interval between two successive events

The following has a POISSON DISTRIBUTION:
* the number of events in a given time period

Queuing Models

In queueing theory a model is constructed so that queue lengths and waiting times can be predicted. Queueing models have been proved to be very useful in many practical applications in areas such as, e.g., production systems, inventory systems and communication systems. These applications concern in particular design problems, where we need to answer questions like: Is the capacity sufficient?, What should be the layout? or How do we have to divide work among several capacities? In many applications the variability in the arrival and service processes are essential to the performance of the system. Queueing models help us to understand and quantify the effect of variability

Queuing models are also called waiting line models. Here, we study what happens when an individual or a set of people come and join queues. If you take a typical example of a doctor; people arrive, spend some time, get served and people leave. What characterizes this queuing system or what makes it little different from the earlier models is the fact that these arrivals and service are assumed to follow some distributions. They are not deterministic but they are assumed to follow certain distributions. So, people arrive according to the given distribution or according to a certain distribution, the service is also provided according to certain given distributions.



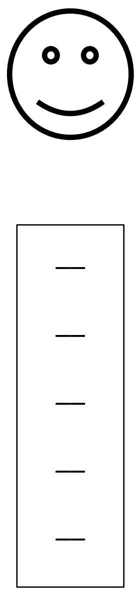
single server and multiple server queuing model

Queuing models can be of several types. First category is called a **single server queuing model** where there is only one server. We also have **multiple server queuing** models where there are multiple servers.

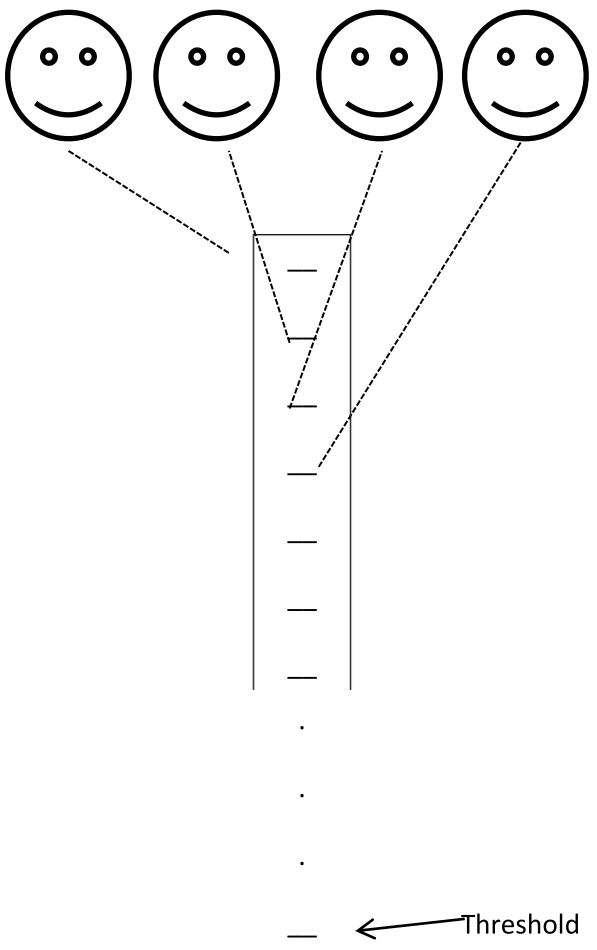
A good example of a multiple server queuing model is a railway reservation system where we could have several counters where people who come join this line and whichever server is free the first person will go, then the next and then the next person goes and so on. So, we have a multiple server model where there is more than one server. Here, there is a common line and as soon as a server is free, this person will join, get the service and leave.

Queue length

Within the single server and multiple servers there are two categories. One is called a finite queue length and infinite queue length. The infinite queue length model assumes, that every person who comes joins the line, for example, if already three people are waiting for the doctor, the fourth person will join the line and so on. There is no restriction on the number of people who are actually waiting or there is no restriction on the length of the queue. The queue length can theoretically be infinite so it can go on and on. In finite queue length models we try to restrict the queue length to a certain limit after which we say that if this threshold limit is reached, people who come into the system do not join the system



Single server finite queue length model



Multiple server infinite queue length model

A good example of a finite queue length model is a garage. Let us say there is a garage with a single server or a single mechanic. There is a mechanic here and let us say this garage has space to park, say ten cars. For example, someone is coming into the garage to give his or her car for service if some slots are available which means the number in the system is less than 10.

Remember, these are the spaces for the garage, this is not garage this is the server. These are the places in the garage, if the number of cars in the garage including the one that is being serviced is less than 10, then the person who is coming in will have a space to park the car, the person will leave the car and go. When some person is coming, the garage is full which means all the ten slots have been taken up including the car that is being serviced, then this person does not find a place to park his or her car. The person will leave the system without joining the line without getting served. Such models are finite queue length models, whereas ordinarily we have infinite queue length models.

Finite and infinite population models

There is one further classification which is called finite population models and infinite population models. If we take the example of the doctor or the reservation system or the car mechanic they all come under the category of what are called infinite population models. The doctor’s case is an example of a single server, infinite queue length and infinite population, while the mechanic is an example of a single server, finite queue length and infinite population models. How do we categorize something as an infinite population and finite population? In this case, the population simply represents anybody who can come for treatment or service to this doctor and therefore it can be infinite.

A good example of a finite population model is shown below:

If we have a factory and if this factory has about thirty machines and if there is a dedicated maintenance team, this maintenance team attends to calls every time these machines breakdown. Then we have a system where the breakdown represents the arrival of a job for the maintenance team. The service is the time or the service provided by the maintenance team to attend to these breakdowns. There is a finite population because only any one of these thirty machines when they breakdown will be attended to. This is an example of a finite population situation.

Ordinarily we discuss more of infinite population situation compared to finite population situations

Distribution of Arrival and Service

The queuing system is also characterized by the **distribution of arrival** and **distribution of service**. The arrivals and services can follow any given distribution or we can observe physically, what is happening in the queuing system. From the data, we can fit a corresponding distribution. Most of the times it also observed that arrivals follow a Poisson distribution, with arrival rate λ per hour (λ/h)

λ usually denotes the arrival rate in a queuing system.

It is also observed from practice, that service times are exponentially distributed, at the rate of µ per hour. We also observe that Poisson distribution and exponential distribution are related. Poisson distribution has an important property called the memory less property based on which we will derive some expressions for the performance of the queuing systems. Essentially, in the context of this study, we will assume that arrivals follow a Poisson distribution with λ per hour (λ/h) and service times are exponential, denoted by µ per hour (µ/h).

What is the relationship between λ and µ

We also need to realise that

λ/µ is less than 1, particularly, if we have an infinite queue length models. For example, we assume

λ = 5 per hour and

$\mu = 6$ per hour, then

λ/μ is less than 1.

What happens, here $\lambda = 5$ per hour means on an average five people enter the system every hour, which means, on an average every twelve minutes a person enters the system and on an average every ten minutes a person gets served and leaves the system.

What happens when λ/μ is greater than 1?

For example, if $\mu \neq 6$ per hour, say,

$\mu = 4$ per hour. Then, every hour five people on an average enter the system and four people on an average leave the system, so, the queue length will automatically increase by 1 every hour and this means that somebody who joins the queue will never get served. For a queuing system to be efficient, particularly, when we have infinite queue length, λ/μ has to be less than 1.

If we have finite queue lengths we may have

$\lambda/\mu > 1$

because in finite queue length models depending on what we have, people may not join the line or join the system. Therefore, we are not that worried about λ/μ being greater than 1. Since some people do not join the system and go away, everybody who joins the system there will get an opportunity to be served. However, we have to be very careful to have λ/μ less than 1, Particularly, when we have infinite queue length models and λ/μ can be greater than 1.

If $\lambda = 5$, $\mu = 6$, $\lambda/\mu < 1$. This means, every hour on an average five people enter the system while six people will leave the system. We may be tempted to ask a question why should we have a queue at all or a line at all, when five people enter the system and six people can leave? The answer comes from the fact that, this is an average or expected value of a distribution. 5 per hour does not mean that exactly every twelve minutes a person gets in.

The inter arrival times are exponential; the arrival rate is Poisson with lambda per hour, it means, on an average five people enter the system following a Poisson distribution. It may happen between two consecutive arrivals, it could be five minutes, it could be twenty minutes, but at steady state, if we measure then we will have five people per hour entering the system. Therefore, there will be a line even though λ is less than μ. We are interested in analyzing the performance of this queuing system or this line. Before we start deriving some expressions for the queuing models. There are three other terms which are often associated with queuing, which we need to discuss. These three are called balking, reneging and jockeying.

Balking

Assume that, you are going to book a ticket in a railway reservation system. consisting of three servers. Assuming you are in a hurry and observe that there are already some sixty people waiting in the line. As you enter, you form an opinion of how much time it is going to take for you to finish this service and leave. If you thinks it is going to take a lot of time, then you leave the system without joining the line. If an arrival does not join the system and leave, then the arrival is set the balk.

Balking can be of two types: **Forced balking** and an **unforced balking**. For example, when you enter as a sixty-first person in a line and you think it is going to take a lot of time, therefore, you do not join the line and nobody is forcing you to leave. That is an example of an unforced balking.

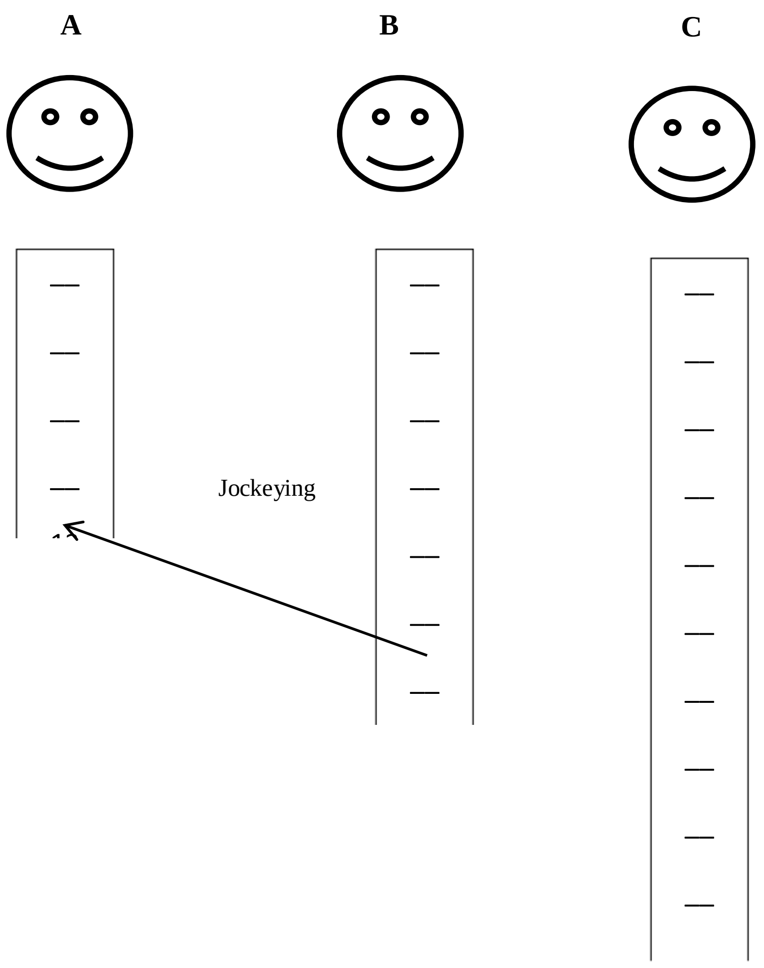
On the other hand, if you go take the mechanic example with space only for ten cars, including the car being serviced and you drive your car for service and you realise that all these ten cars are full then you do not have a choice, you leave the system then such a balking that is called **forced balking**. Whenever we have finite queue length models, then there is a chance that balking occurs.

Reneging

Again come back to the previous example let us say you enter the system to book a ticket and there are sixty people already waiting. Then you realise you wish to join and say you join as the sixty-first person in the line. When you join as the sixty-first person you expect the line to move in a certain speed. After a while, let us say you observe that there it is actually moving slower than what you thought and your time is running up, then half way through from somewhere in the middle of the line you decide to quit, you just come out the line and you go away. Such a phenomenon is called reneging. The person joins the system, but after some time decides not to continue and simply moves out of the system, called reneging.

Jockeying

Jockeying is an interesting phenomenon. For example, if in the above system, we have multiple servers but there is a common line. People are waiting here now depending on which server is free, the person will automatically go to the corresponding server. Assuming we had a system where there are three servers and each server has a dedicated line let us say you come to the system and you find about nineteen people on one line, about twenty people on a second line and fifty nine people on the third line including the people who are being served we normally would join one of these lines. Many times there will be a tendency to join the shortest line. Let us say, you join the shortest line as a twentieth person. All the three lines are moving towards the server and more people have actually come in and joined. After five minutes you realise that, the third line is moving slightly faster. You have now become the eighteenth on the first line, but if you realise that, if you actually shift this line, you can become the seventeenth person in the third line, because this line is moving faster.



There will be a tendency to shift from one line to another line for a while and come back depending on what we think which is the rate at which people in these lines are moving. Such a phenomenon is called jockeying. Ordinarily, jockeying happens within the first few minutes of joining the line. Once we realise that there are four or five people behind us. If you skip this, if you are the eighteenth person here and then if you jockey, you will become the twenty-third person here then you will not like to jockey. Jockeying is something that is common to multiple server queuing system.

Case Studies

Smartqueue Integrated multimedia queue management system

DigiQ queue management system by Ctronix

Q-net Queue Management systems by Q-net International Ltd

Q-Flow Queue Management system by Nexa Group

QLess queue management system by QLess group

http://en.europeum.cz/queuing-systems

http://www.airport-technology.com/contractors/lighting/marimils/marimils1.html

minkhollow.ca/books/?page_id=34

http://www.mathworks.com/help/simevents/examples/m-m-1-queuing-system.html