9 Queueing Networks

9.1 Introduction

Mathematicians have studied queues for approximately 100 years, and one of their first applications was to telephone exchanges (*Erlang's loss formula*). However they gained popularity with Computer Scientists about 35 years ago when it was realised that single queues, and networks of queues could be used as performance models of computer systems. More recently, developments in many of those systems have grown beyond the expressiveness of queueing networks. Nevertheless, for many people *performance analysis* is synonymous with *queueing theory*.

Queueing theory was first used in the late 1960s to model time-sharing computer systems. In particular single queues were used to study allocation policies for CPUs. Analysis of the single queues led to a new qualitative, as well as quantitative, understanding of some aspects of operating system and disk management system design. However, these single queue models were too restrictive to represent systems of interacting computing devices, as computers came to be viewed.

Later developments in queueing theory therefore studied the interaction between service centres or queues—*queueing networks*. Contemporary computers at the time could be viewed as a set of loosely coupled hardware components through which weakly interacting *jobs* or *transactions* were circulating. The success of queueing theory as a performance modelling paradigm relied on this view of computers. Computer and communication systems today do not fit into this model so readily; however, queueing networks remain a useful performance modelling paradigm, in some circumstances.

9.2 Single queues

The basic scenario for a single queue is that *customers*, who belong to some *population* arrive at the *service facility*. The service facility has one or more *servers* who are capable of performing the service required by customers. If a customer cannot gain access to a server it must join a queue, in a *buffer*, until a server is available. When service is complete the customer departs, and the server selects the next customer from the buffer according to the *service discipline*.

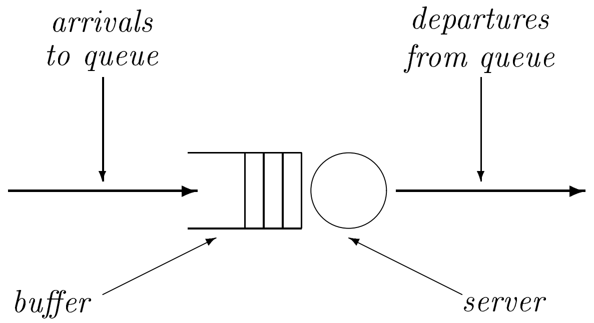

Figure 18: A Simple Queue

In order to describe a service facility accurately we need to know details about each of the terms emphasised above:

Arrival Pattern of Customers The ability of the service facility to provide service for an arriving stream of customers depends not only on the mean arrival rate λ, but also on the pattern in which they arrive, i.e. the *distribution function* of the inter-arrival times. In this course we will only be considering queues in which the times between arrivals are assumed to be exponentially distributed[1]. However, you should be aware that there are other possibilities.

Service Time Distribution The service time is the time which a server spends satisfying a customer. As with the inter-arrival stream, the important characteristics of this time will be both its average duration and the distribution function. Similarly, in this course we will only consider service times which obey an exponential distribution. If the average duration of a service interaction between a server and a customer is $1/\mu$ then μ is the *service rate*.

Number of Servers If there is more than one server in the service facility the processing of customers can go on concurrently: one customer at each server. It is assumed that the servers are indistinguishable from each other, so that it does not matter which server actually serves a customer—the results, and the service characteristics, will be the same.

The most commonly considered cases are of a *single server* or an *infinite server*. In the first case, we assume that the service facility only has the capability to serve one customer at a time; waiting customers will stay in the buffer until chosen for service; how the next customer is chosen will depend on the service discipline. In an infinite server facility we assume that there are always at least as many servers as there are customers, so that each customer can have a dedicated server as soon as it arrives in the facility. There is no queueing, (and no buffer) in such facilities. Between these two extremes there are *multiple server* facilities. These have a fixed number of servers (usually denoted c), each of which can service a customer at any time. If the number of customers in the facility is less than or equal to c there will no queueing—each customer will have direct access to a server. If there are more than c customers, the additional customers will have to wait in the buffer.

Buffer Capacity Customers who cannot receive service immediately that they require it must wait in the buffer and wait for a server to become available. Clearly the number of customers waiting may grow, filling the places of the buffer. If the buffer has finite capacity there is the possibility that it may become full. There are two possibilities in this case:

- either, the fact that the facility is full is passed back to the arrival process and arrivals are suspended until the facility has spare capacity (created by the completion of a customer who is currently being served);
- or, arrivals continue and arriving customers are turned away and lost until the facility has spare capacity again.

[1] Equivalently, the arrival stream obeys a Poisson distribution.

In some systems the buffer capacity is so large as to never affect the behaviour of the customers; in this case the buffer capacity is assumed to be infinite.

Population The characteristic of the population which we are interested in is usually the size. Clearly, if the size of the population is fixed, at some value N, no more than N customers will ever be requiring service at any time. When the population is finite the arrival rate of customers will be affected by the number who are already in the service facility. The arrival rate will be zero when all the population is already in the facility. Sometimes the size of the population is so large that this situation will never arise. In this case we assume that the population is infinite—this simplifies the mathematics.

Ideally, members of the population are indistinguishable from each other. When this is not the case we divide the population into *classes* whose members all exhibit the same behaviour. A class will represent a set of customers who share the same characteristics. Different classes differ in one or more characteristics, for example, arrival rate or service demand.

Service Discipline When more than one customer is waiting for service there has to be a rule for selecting which of the waiting customers will be the next one to gain access to a server. The commonly used service disciplines are

- FCFS *first-come-first-serve* (or FIFO *first-in-first-out*).
- LCFS *last-come-first-serve* (or LIFO *last-in-first-out*).
- RSS *random-selection-for-service.*
- PRI *priority.* The assignment of different priorities to elements of a population is one way in which classes are formed.

We will only be considering systems which obey the FCFS service discipline.

A shorthand notation for these six characteristics of a system is provided by *Kendall's notation* for classifying queues. In this notation a queue is represented as $A/S/c/m/N/SD$:

A denotes the arrival process; usually M, to denote Markov (exponential), G, general, or D, deterministic distributions.

S denotes the service rate and uses the distribution identifiers as above.

c denotes the number of servers available in the service facility.

m denotes the capacity of the **service facility** (buffer + server(s)), infinite by default.

N denotes the customer population, also infinite by default.

SD denotes the service discipline, FCFS by default.

Example: Consider a network gateway, at which measurements have shown that packets arrive at a mean rate of 125 packets per second, and are buffered until they can be transmitted. The gateway takes 2 milliseconds on average to transmit a packet. The buffer currently has 13 places, including the place occupied by the packet being transmitted and packets that arrive when the buffer is full are lost. We wish to find out if the buffer

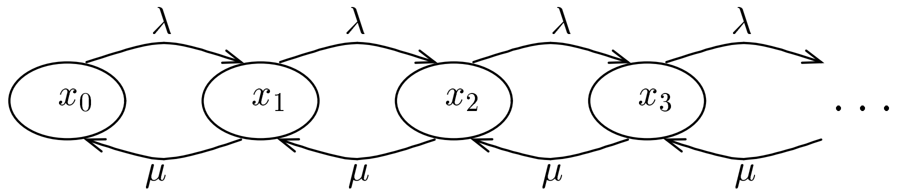

Figure 19: The state transition diagram for a simple $M/M/1$ queue

capacity is sufficient to ensure that less than one packet per million gets lost. Making exponential assumptions about the arrival rate and the service rate we would model the gateway as an $M/M/1/13/\infty/FCFS$ queue.

Example: Consider the mobile phone antenna discussed in Sections 1.5 and 3.2.1. Assuming that we are interested in the simpler characterisation of the system as a random variable, whose value denotes the number of frequencies in use, we can now see that this is a $M/M/6/6/\infty/FCFS$ queueing system. Since the last two characteristics are default values we would usually omit them and talk about an $M/M/6/6$ queue.

Example: A PEPA model of a $M/M/1/N$ queue is as follows:

$$
\begin{aligned}
Arrivals &\stackrel{\text{def}}{=} (arrive, \lambda).Arrivals \\[1em]
Queue_0 &\stackrel{\text{def}}{=} (arrive, \top).Queue_1 \\
Queue_i &\stackrel{\text{def}}{=} (arrive, \top).Queue_{i+1} + (depart, \mu).Queue_{i-1} \qquad 1 \leq i \leq N-1 \\
Queue_N &\stackrel{\text{def}}{=} (depart, \mu).Queue_{N-1} \\[1em]
System &\stackrel{\text{def}}{=} Arrivals \underset{\{arrive\}}{\bowtie} Queue_0
\end{aligned}
$$

9.3 Traffic Intensity

The two most important features of a single queue are the arrival rate of customers λ, and the service rate of the server(s), μ. These are combined into a single parameter which characterises a single or multiple server system[2], the *traffic intensity.*

$$\boxed{\text{traffic intensity, } \rho = \frac{\lambda}{c \times \mu}}$$

As we will see in the next lecture, many of the important performance measures associated with a queue can be expressed in terms of ρ. These are measures such as *utilisation* (the proportion of time the server is busy), *residence time* (the average time spent at the

[2] $c = 1$ in the single server case

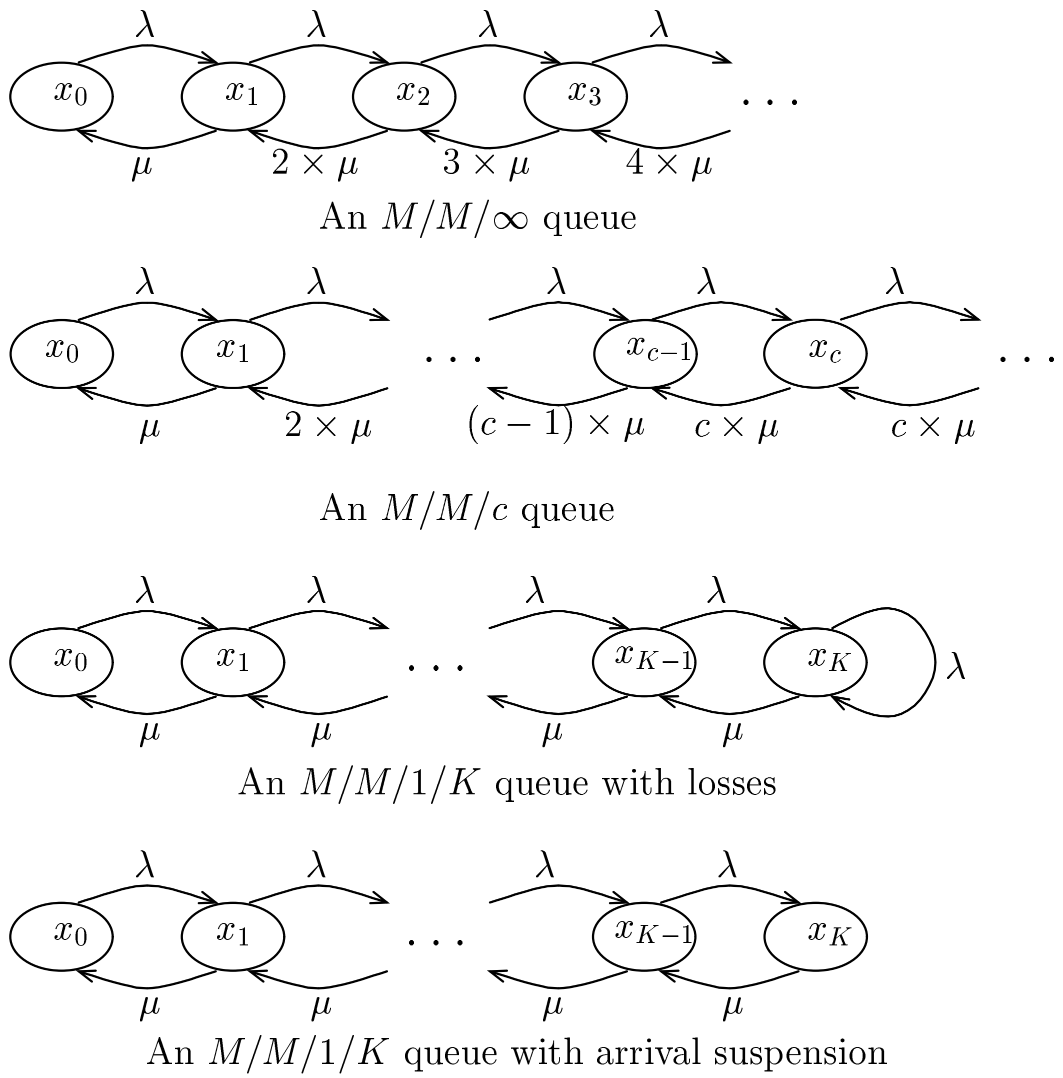

Figure 20: State transition diagrams for some other commonly occurring queues

service facility by a customer, both queueing and receiving service), *waiting time* (the average time spent queueing), *queue length* (the average number of customers at the service facility, both waiting and receiving service), and *throughput* (the rate at which customers pass through the service facility). For the moment we just point out that for the system to be *stable*, ρ must be less than 1: that is the arrival rate of customers must be less than the rate at which they are served. The alternative, if the arrival rate is greater than the service rate, would result in a queue which grew unboundedly.

9.4 Networks of Queues

If we view a computer as an interaction of devices, which jobs visit sequentially, then it is natural to model the system as a *queueing network*. Each device or resource of the system is represented by a separate service facility, now termed a *service centre*. Customers in the queueing network correspond to the users or transactions in the computer system.

It is assumed that after service at one service centre a customer may progress to other service centres in the system, following some previously defined pattern of behaviour, representing the task which the customer wishes to achieve. Note that this is very similar to the view of systems taken in operational analysis.

The characteristics of each service centre are analogous to the single queue case, i.e. the arrival rate, the service rate, the number of servers, the buffer capacity, the population and the service discipline. However, note that these characteristics are not completely independent for the service centres within a queueing network. For example, the population will usually be the same for all service centres, and the arrival rate at one service centre will often depend on the arrival rate at another. If a service centre with finite capacity buffer becomes full, service may be suspended at the service centres which send customers to this queue.

Defining a queueing network model of a particular system is made relatively straightforward by the close correspondence between the attributes of queueing network models and the attributes of the systems.

A queueing network can be represented as a directed graph in which the nodes represent the service centres (see Figure 21). Various conventions have been developed for this representation but you should note that this is an informal representation. In the case of GSPN the graph representation had a formal interpretation from which the dynamic behaviour of the system could be deduced. In contrast, the diagram of a queueing network is schematic, representing the topology of the system, and the route that customers take between resources. In the diagram the routing of customers is represented by the *routing probabilities*, which label branches in the paths between service centres. For example, in the network shown in Figure 21, a customer leaving the first service centre goes to the second service centre with probability p and leaves the system with probability $1 - p$.

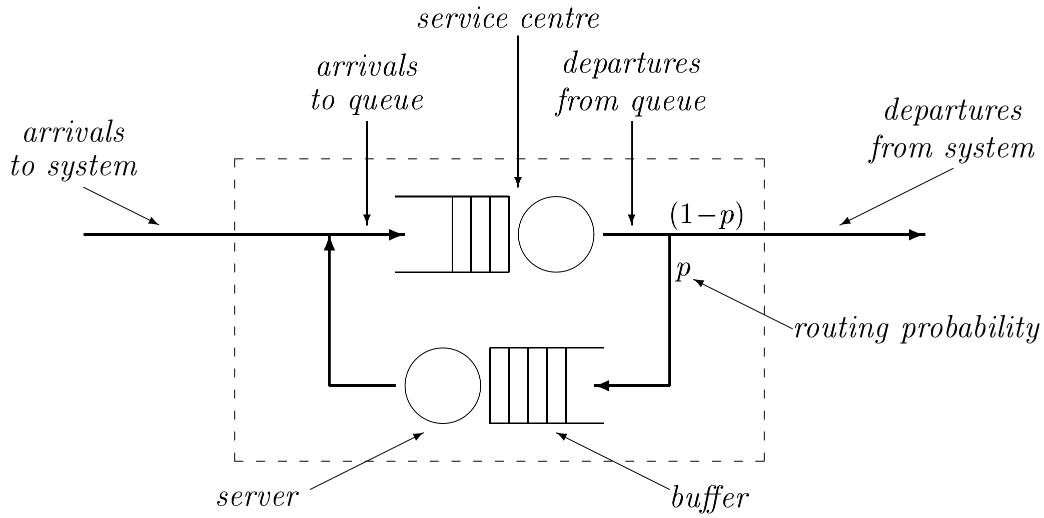

Figure 21: A Simple Open Queueing Network

The state of the system is typically represented as the number of customers currently occupying each of the service centres. There may be a number of different *classes* of customers each exhibiting different characteristics within the network. In this case the state is the number of customers of each class at each service centre. As well as differences in arrival rate and/or service demand, classes may be distinguished by the routes which customers take through the network.

A network may be *closed*, *open* or *mixed* depending on whether a fixed population of customers remain within the system; customers may arrive from, or depart to, some external environment; or there are classes of customers within the system exhibiting open and closed patterns of behaviour respectively.

9.5 Expressiveness

As mentioned in the Introduction, contemporary computer and communication systems often have features which are not easily represented by queueing networks. Some of the most important of these features are listed below. In some cases ways of adapting queueing models to represent, and provide solutions for, systems with such features have been found. However, these are new approaches and largely remain topics for research.

Simultaneous resource possession In a computer system a job may continue to compute while its I/O requests are progressing in parallel. Thus, the job may be simultaneously using two or more resources of the system.

Fork and Join Fork and Join primitives are used in computer systems to create and synchronise subprocesses. This causes the number of jobs in the system to change and also invalidates the assumption of independence among jobs.

Bulk and train arrivals In communication networks the arrival of packets may occur in batches (bulk arrivals) or in quick succession (train arrivals). Such situations do not fit assumptions of independence between customers, nor exponential interarrival times.

Load dependent arrivals Computer networks and distributed systems have intelligent load-balancing policies that cause new jobs or packets to go to one of a number of devices depending upon the load observed in the recent past.

Defections from the queue Communication systems sometimes set a maximum limit on the time that a packet or request is able to stay in a queue. Thus, the packets that have been in the queue for a long time are dropped from the queue on the assumption that they may have already been retransmitted by higher protocol layers.