Performance Evaluation of Computer Systems and Networks Lecture notes for CS 681

Varsha Apte
Department of Computer Science and Engineering
IIT Bombay. Spring - 2015.

[Lecture 1: January 5th, 2015]

1 Motivation

We experience the impact of contention for resources almost continuously in our daily lives. The traffic on the road, the lines at a supermarket, the crowd on a Sunday evening at a McDonalds - these are all examples of resources under contention.

Computing Systems and networks are no different. There are a variety of different resources in these systems that could easily come under contention. Can you name a few? In fact, as end-users of such systems we frequently experience the effects of this contention ("slow" website, "slow" network, etc).

Table 1 shows some resources and the "customers" that use these resources.

Resource	Users (Schedulable Entity)
CPU	Processes, Threads
Printer	Print Jobs
Web Server Threads	HTTP GET/POST requests
Data Network Link	Packets
Cellular Network Channel	Cellular Call

Table 1: Examples of resources and their users. "Users" are the scheduling unit, the entity that is chosen for resource allocation

Now consider one of these resources, let's say, a Web server, that is to be specific, the thread pool of a Web server process. The thread pool of a Web server is a *limited resource*. An arriving HTTP request is given a thread if there is an idle thread, or joins a queue of requests at the end, if all threads are busy.

What performance-related questions can we ask about the system? What are the variables that determine the performance of the system? When you think of such systems, always classify the quantities into *parameters* and *metrics*.

Thus we could ask the question as to "What is the average waiting time of a request", when the arrival rate is 100 requests/sec? Here, waiting time is an example of a *metric*.

The answer to this question will depend on various quantities, e.g., it will depend on the average processing time per request, request arrival rate or on the number threads that the Web server is configured with. These are examples of *parameters*.

	Parameter	Unit
1	CPU Speed	GHz, flops/sec , MIPs
2	Request Arrival Rate	requests/sec
3	Average resource requirements per request	Depending on the resource, e.g. for CPU MI/request
4	Disk read write rate	blocks/sec
5	Buffer Size	

Table 2: Parameters

Metric is something by which we judge a system. This is what we observe, measure, evaluate. *Parameter* is what we change to see how it affects a metric. This difference in metrics and parameters is important to note in system evaluation.

Figure 1: Parameters and Metrics

Think of various such performance metrics and parameters of a Web server system. Write them down . Then see Table 2 and 3 for examples of a few parameters and metrics, and see how many you were able to write.

1.1 Relationship between Metrics and Parameters

Before one proceeds to formally charactize relationships between performance metrics and performance parameters, it is useful to try to "guess" what those relationships are. When you are working in this field (or any other field for that matter), it is important to use common sense and intuition to predict behaviors, and only then use other rigourous techniques to characterize them formally.

Let us consider the metric of CPU utilization, in the Web server example, and the arrival rate parameter. What must the relationship between this metric and parameter? It is quite intuitive that CPU Utilization will increase with increase in request arrival rate. But, how does it increase? Figure 2 shows various relationships that could be possible. .

Similarly, think about the relationship of the waiting time of a request with CPU speed. Again, it is intuitive that waiting time should decrease with increasing CPU speed. Figure 3 shows a various possible ways in which this could happen.

1.2 Precise Analysis of the Relationship

How do we go from an intuitive/qualitative analysis of a relationship between a metric and a parameter, to a precise function?

We have two options: measurement and modeling. Measurement implies directly measuring the metrics of interest in a working system. Measurement involves lot of resources (hardware, software, people, time). To create a measured graph of metric vs parameter, one must perform experiments

	Metric	Unit
1	Max. number of users it can support	
2	Average Response Time (R)	msec,sec
3	Average Throughput(Λ) (Number of completed requests per unit time.)	requests/sec
4	Queue Length (N)	
5	Resource utilization measure (ρ) (Fraction of time a resource is busy e.g. CPU,Network Bandwidth,Memory,Disk Bandwidth,etc.)	% of time
6	Number of active requests in server (Applicable for Multithreaded system.)	
7	Probability(fraction) of arriving requests being dropped	
8	Probability(fraction) of arriving requests being timed out	
9	Waiting Time	msec,sec

Table 3: Metrics

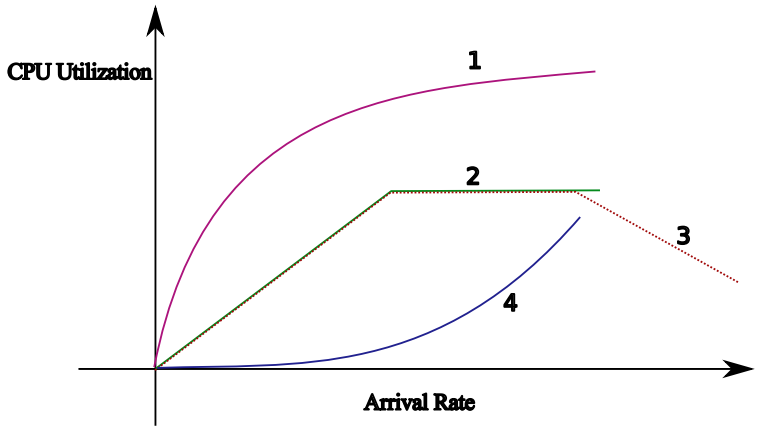


Figure 2: CPU utilization vs Arrival Rate: which of the above curves might be the right one?

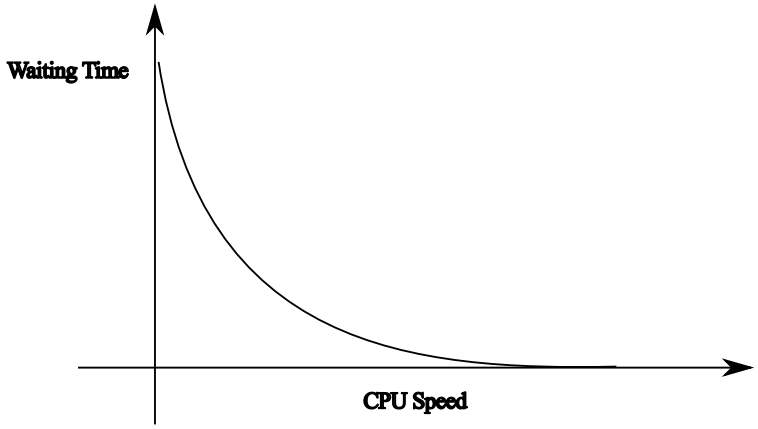


Figure 3: Average Waiting Time vs CPU Speed

at each parameter setting, take careful measurements, discard outliers, ensure that the testbed is accurate, etc.

Another option is to use a model. Models are of two types: simulation models, and analytical models. In a simulation model, we write a *program* that behaves like the system we want to study. We represent the state of the system using data structures, create and process "events" that change the state of the system. As the simulation runs, statistics are collected, and analyzed at the end.

In an analytical model we use logical and mathematical reasoning to reason about the behaviour of a system. One challenge here, is that quantitative parameters of systems (e.g. request processing time, inter packet-arrival time, etc) are not fixed numbers - rather, they are *random variables*. Thus, a most natural analysis approach is that of using probabilistic analysis. For example we may try to estimate the probability distribution of waiting time of a request at a server, if we know the distribution of request processing time and the distribution of request inter-arrival time.

Using probabilistic approaches we can write equations which express performance metrics in terms of system parameters. Mathematical relationships often offer much more insights than empirically observed relationships. The linearity/non-linearity, limiting values, sensitivity to various parameters etc is much more readily analyzable when an equation relates a metric to several parameters. This is the beauty that we will explore in this course.

However, we first start with queuing systems and their analysis in a much more intuitive manner.

[Lecture 2: January 8th, 2015]

2 Queueing Systems

Queueing systems arise when there are resources under contention and multiple user entities compete for those resources. A generic queueing (open) system is depicted in Figure 4. An open queueing system is where we do not model the sources from where the requests are coming.

An open queueing system is described using the following terms and characteristics, and statistical properties:

- Customers/Requests: these are the entities that need the resource under contention.

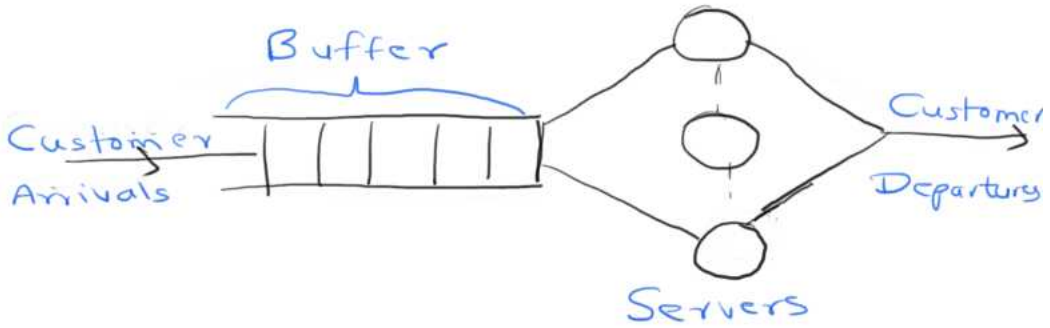


Figure 4: A Queueing System

- Server: this is the resource that has to be "held" by the customer during its "service" . E.g. a printer, a cpu, a network link etc are servers. A print job, a process, a packet, are "customers" of these servers, respectively.
- Customer Interarrival Distribution: Distribution of time between two customer arrivals into the queueing system.
- Service time, and its distribution: Service time is the time a customer actually "holds" the resource for service. This does NOT including time spent waiting for this service. This time is described by its distribution, mean, variance etc.
- Buffer: This is the area where customers wait for service - i.e. where the queue is formed. This is described by the size (in terms of number of customers).
- Scheduling Policy: This is the policy which determines how the next customer is selected for service, when a server becomes idle. Examples are FCFS, LCFS, Shortest Job First, priority, etc

The above characteristics of the system are abbreviated using a notation called *Kendall notation*: A/B/C/D/E/F

- A: Symbol for interarrival time distribution. M: Memoryless (exponential), D: Deterministic, G: General.
- B: Symbol for service time distribution ($M, D, G \ldots$).
- C: Number of servers.
- D: Buffer size (this symbol is dropped if buffer is assumed to be infinite)
- E: Scheduling Policy. E.g. FCFS, LCFS, PS (Processor Scheduling), PRIORITY, etc. This symbol is also dropped if scheduling discipline is FCFS, or otherwise understood from the contxt.

- F: Population. If the sources of the requests are modeled, then this denotes the number of users in a "request-response" cycle with the server. For open systems, population is implicitly assumed to be infinite, so this part is also usually not written.

Examples of queuing systems expressed using this notation are:

- $M/M/1$: Single server exponentially distributed service time and interarrival time, infinite buffer, FCFS, open system.
- $M/G/1/20$: Single server, exponentially distributed interarrival time, finite buffer of size 20, FCFS, open system.
- $G/G/c/K$: c servers, generally distributed service and interarrival time, finite buffer of size K, FCFS, open system.

The essential quantitative parameters of an open queueing system are:

- Request Arrival rate: λ requests/sec (Mean Interarrival time is $\frac{1}{\lambda}$.
- Service Rate (maximum rate at which one server can provide service): μ requests/sec.
- Average Service Time: $\tau = \frac{1}{\mu}$
- Sometimes, the variance of service time.
- Number of servers: c
- Buffer size: K

The *metrics* that we usually analyze and are dependent on the above parameters are:

Λ**, Average Throughput** : Number of requests completed per unit time (also called *departure rate*).

ρ**, Average Server Utilization** : (or just "utilization"): Fraction of time a server is busy. For multiple servers, this is the average fraction over all the servers.

W**, Average Waiting Time:** Time spent in queue before service starts

R**, Average Residence/System/Response Time:** Waiting Time + Service Time

N**, Average Number of customers in the system (including in service)**

Q**, Average Queue Length** : Average number waiting in the queue

2.1 Observational/Operational Laws

Some relationships between parameters can be derived with simple intuitive analysis of a queueing system. Suppose we have a *single server infinite buffer queueing system* under observation.

Let
- T Observation/measurement period
- A Number of arrivals in time T
- C Number of completions/departures in time T
- B Total time that system was busy in time T

Assume that T is very large. Note that we do not make any assumptions about the *distributions* of service time and inter-arrival time.

Then the following is obvious:

$$\begin{aligned}
\text{Arrival Rate } \lambda &= \frac{A}{T} \\
\text{Throughput } \Lambda &= \frac{C}{T} \\
\text{Mean Service Time } \tau &= \frac{B}{C} \\
\text{Utilization } \rho &= \frac{B}{T} \\
&= \frac{B}{C}.\frac{C}{T} \\
&= \tau\Lambda
\end{aligned}$$

Here,

$$\rho = \Lambda\tau$$

is called **the Utilization Law** is an important law in queueing systems.

For a lossless system, where $\lambda < \mu$, for a long observation period, the number of departures must be almost equal to the number of arrivals, thus the throughput $\Lambda = \lambda$ and $\rho = \lambda\tau$.

Thus:

$$\begin{aligned}
\text{Throughput } \Lambda &= \lambda \text{ if } \lambda < \mu \\
&= \mu \text{ if } \lambda \geq \mu
\end{aligned}$$

and

$$\begin{aligned}
\text{Utilization } \rho &= \lambda\tau \text{ if } \lambda < \mu \\
&= 1 \text{ if } \lambda \geq \mu
\end{aligned}$$

A single server lossless queueing system is said to be **stable** if $\lambda < \mu$.

Now consider a finite buffer system where request is dropped when buffer is full. Let p_l be the request loss probability (i.e. buffer full probability). Then

$$\Lambda = \lambda\ (1 - p_l) \qquad \forall\lambda \geq 0$$

. and

$$\rho = \lambda\ (1 - p_l)\tau \qquad \forall\lambda \geq 0$$

.

Thus, a finite buffer system is **always stable**. This is because as λ grows, p_l also grows.

For multiple server (c servers), infinite buffer, the throughput and server utilization is as follows:

$$\begin{aligned}
\text{Throughput } \Lambda &= \lambda \text{ if } \lambda < c\mu \\
&= c\mu \text{ if } \lambda \geq c\mu
\end{aligned}$$

and

$$\begin{aligned}
\text{Utilization } \rho &= \frac{\lambda\tau}{c} \text{ if } \lambda < c\mu \\
&= 1 \text{ if } \lambda \geq c\mu
\end{aligned}$$

[Lecture 3: January 13th, 2015]

For multiple server (c servers), finite buffer, the throughput and server utilization remains is:

$$\Lambda = \lambda\ (1 - p_l) \qquad \forall \lambda \geq 0$$

. and

$$\rho = \frac{\lambda\ (1 - p_l)\ \tau}{c} \qquad \forall \lambda \geq 0$$

.

2.2 Asymptotic Analysis of Open Queues

The above queueing systems can also be analyzed for some limiting values without making use of complicated "maths". We primarily look at limiting values as the load (arrival rate in this case) goes either to zero, or to infinity.

In the analysis regarding waiting time and response time, we assume thet the service time distribution has the *memoryless property* . In continuous distributions only the *exponential distribution* has this property. This property implies this: suppose we observe the server when it is busy, and a time u has elapsed in the current service. Let Y be the random variable denoting the *remaining time* of service. Then Y has the same distribution (and hence the same mean) as the total service time distribution. Thus it is as if there is "no memory" of how much work has already been done on this customer. Hence such a distribution is called *memoryless*. We will prove this property mathematically when we do our mathematical background review.

	$G/G/1$		$G/G/c$		$G/G/1/K$		$G/G/c/K$	
Metric	$\lambda \to 0$	$\lambda \to \infty$	$\lambda \to 0$	$\lambda \to \infty$	$\lambda \to 0$	$\lambda \to \infty$	$\lambda \to 0$	$\lambda \to \infty$
Throughput	0	μ	0	c μ	0	μ	0	c μ
Utilization	0	1	0	1	0	1	0	1
Queue Length	0	∞	0	K	0	∞	0	K
Number in System	0	∞	0	$K+1$	0	∞	0	$K+c$
Waiting Time	0	∞	0	$K\tau$	0	∞	0	$\frac{K}{c}\tau$
Response Time	0	∞	0	$(K+1)\tau$	0	∞	0	$(\frac{K}{c}+1)\tau$

Question: where exactly are we using the memoryless property in the entries above?

2.3 Little's Law

Consider any system through which customers traverse for service (see Figure 5). Consider the box with the boundary drawn in that system. Customers enter that box, get some service there and leave that region. Let R be the amount of time the customers spend inside that box. Let N be the average number of customers inside that box, and let Λ be the rate of "flow" of customers through that box (in other words, the *throughput* through that box).

Suppose further that the servers in this system are *work conserving* - i.e., they will not be idle if there is a customer in the queue, and work is not "lost" after it enters the system. Further suppose that the scheduling policy does not require knowledge of the service time requirement of the customers.

Then *Little's Law* states that:

$$N = \Lambda R \tag{1}$$

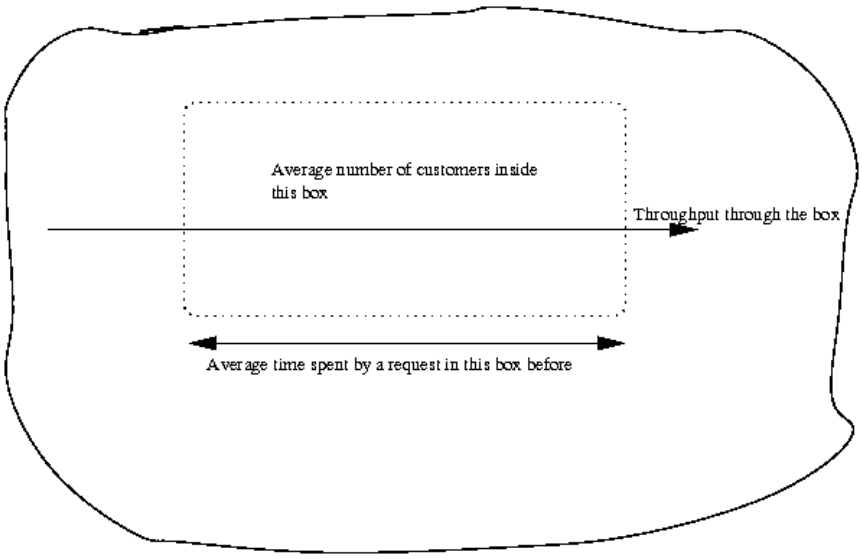


Figure 5: Pictorial view of Little's Law

Little's Law is remarkable because at first glance it seems to do something non-intuitive. For example, if you imagine a queuing system whose average number in the system is known to be N, and average service time is τ, the "intuitive" thing to do is to multiply τ by $N+1$ (arriving customer sees N in the system on an average, will on an average need $N\tau$ waiting time and then τ service time to finish). Why doesn't this reasoning work? It comes down again to the *memorylessness* property. In the reasoning where we arrive at $(N+1)\tau$, we assume that the mean of the remaining service time of the customer in service is also τ, which is not the case of service time is not memoryless. Hence this reasoning does not work.

This should imply that for a memoryless service time distribution, the intuitive reasoning should work. In fact, it does. We will prove later that for an $M/M/1$ queue,

$$N = \frac{\rho}{1-\rho}$$

Now by the direct reasoning we get:

$$R = (N+1)\tau = (\frac{\rho}{1-\rho} + 1)\tau = \frac{\tau}{1-\rho}$$

By applying Little's law we get:

$$\begin{aligned} R &= \frac{N}{\lambda} = \frac{\rho}{1-\rho} \times \frac{1}{\lambda} \\ &= \frac{\tau}{1-\rho} \end{aligned} \tag{2}$$

which is the same as we get with the "intuitive" reasoning.

Various queueing theorists have their own way of intuitively understanding Little' Law. Mor Harchol-Balter explains that since Λ is the throughput, which is also the "departure rate", the average inter-departure time is $\frac{1}{\Lambda}$. This time can be interpreted as the effective time by the system

to complete one customer. So if there are average N customers in the system, the time required to process all of them is $N \times \frac{1}{\Lambda}$. This intuition may work or not work for you!

Kleinrock explains the formula as follows: suppose we start with the average response time. This is the average time each customer spends in the system. For stable lossless system, $\lambda \times R$ is the number of customers that will arrive during this time and you could interpret it as the average number of customers in the system.

[Lecture 4: January 16th, 2015]

2.3.1 A visual proof of Little's Law

Consider a single server queueing system. Let us plot two curves on the same graph:

- $A(t)$: Cummulative number of arrivals upto time t
- $D(t)$: Cummulative number of departures upto time t

These curves will be as shown in Figure 6. Note that the two graphs are identical except how the enclosed area is divided ("horizontal rectangles" vs "vertical" rectangles").

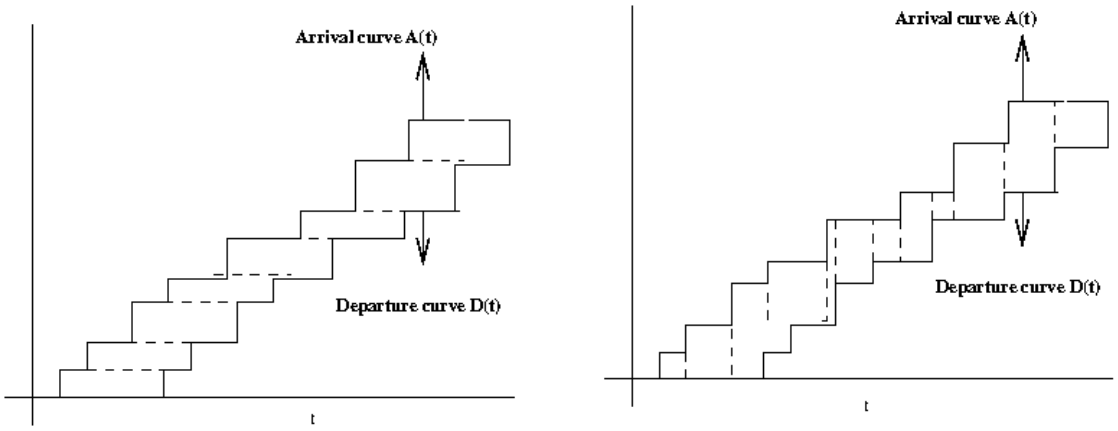


Figure 6: Visual Proof of Little's law

We prove Little's Law by calculating the area enclosed in within the two plots above in two was: first by adding the areas of the "horizontal rectangles", and then by adding the areas of the "vertical rectangles" and equating these two.

The width of the horizontal rectangles is the response time of the customer in the system. The height is 1. Let the number of arrivals (and number of departures) be A. Area by adding horizontal rectangles =

$$\begin{aligned}
\sum_{i=1}^{A}(R_i \times 1) &= \sum_{i=1}^{A} R_i \\
&= \left(\frac{\sum_{i=1}^{A} R_i}{A}\right) A \\
&= \overline{R}A
\end{aligned}$$

Here $\overline{R} = \frac{\sum_{i=1}^{A} R_i}{A}$ is the average response time.

The height of the vertical rectangles is the number of customers in the system (because the height is equal to the cumulative number of arrivals minus cumulative number of departures, so this difference should be what is left in the system). Denote these by N_i, for the ith rectangle. The width is the time difference between the two consecutive events (arrival or departure), denoted by $T_1, T_2, \ldots, T_k, \ldots$. Note that this means that there are N_i customers in the system for T_i amount of time.

Thus, the area is:

$$\begin{aligned} \sum_{i=1}^{k} N_i \times T_i &= \left[\sum_{i=1}^{k} N_i \left(\frac{T_i}{T}\right)\right] T \\ &= \overline{N}T \end{aligned}$$

note that here $\frac{T_i}{T}$ is the fraction of time that the number of customers in the system are N_i, therefore $\overline{N} = \sum_{i=1}^{k} N_i \left(\frac{T_i}{T}\right)$ is average number of customers in the queueing system.

Equating two areas

$$\begin{aligned} \overline{R}A &= \overline{N}T \\ \overline{N} &= \frac{A}{T}\overline{R} \\ \overline{N} &= \lambda \overline{R} \end{aligned}$$

2.4 Closed Queuing Systems

Closed queueing systems are models that arise when the *sources* of the requests are modeled explicitly. An example of a closed system is one in which there is a a fixed number of users or clients each of whom issues a request to the server, waits for a response, "thinks" for a while (e.g processes the response itself) and then issues the next request to the server.

A "canonical" one-server, and M user system is visualized as follows

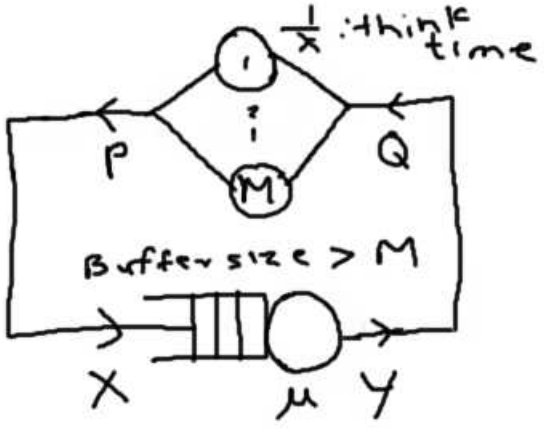


Figure 7: Canonical Closed System

Let:

- Average think time be denoted by $\frac{1}{\lambda}$.
- Service rate be denoted by μ and average service time by $\tau = \frac{1}{\mu}$.
- Number of users be denoted by M
- Throughput= Λ, Server Response time = R, server utilization = ρ.

2.4.1 Canonical Closed System: Operational and Asymptotic Analysis

Just like for open queues, we can analyze this closed system to some extent, without needing to use advanced mathematical models.

Some preliminary observations - since this is a closed system, it is important to realize that at equilibrium, the rate of requests flowing through the edges anywhere in this system is going to be the same. e.g. the rate of request flow at points P, Q, and X, Y are the same at equilibrium. Specifically, the aggregate rate at which users issue requests = arrival rate at server = throughput at server = aggregate response rate at users.

How does one understand this? We must realize that the server and the users "modulate" each other. E.g., what is the rate at which ONE USER issues requests? This is best understood by looking at one user's timeline.

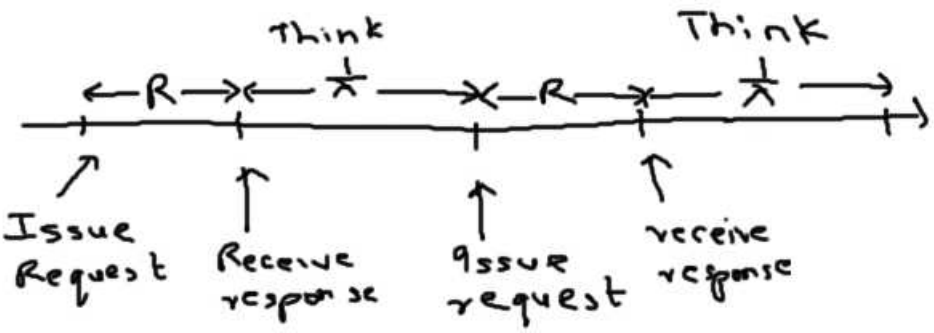


Figure 8: User Timeline

It is immediately apparent from this timeline, that one user issues a request ever $R + \frac{1}{\lambda}$ time. Thus, an individual user's request rate *slows down* as the server's response time increases. (This is very different from an open system.)

Now consider this:

$$\text{Average request issue rate by 1 client} = \frac{1}{E[R] + \frac{1}{\lambda}}$$

$$\text{Total request arrival rate} = \frac{M}{E[R] + \frac{1}{\lambda}} \text{Total throughput}$$

$$\Lambda = \frac{M}{R + \frac{1}{\lambda}}$$

$$R = \frac{M}{\Lambda} - \frac{1}{\lambda}$$

$$\text{At high load, } \lim_{M \to \infty} R = \frac{M}{\mu} - \frac{1}{\lambda}$$

$$\text{At high load, } \lim_{M \to \infty} R = M\tau - \frac{1}{\lambda}$$

$$\text{At low load, } \lim_{M \to 0} R = \tau$$

Thus, the response time has two linear asymptotes. We still do not know enough to calculate the response time for values of M between the asymptotes. But based on what we will learn later, we know that the response time curve looks like this:

Figure 9: Response Time vs Number of Users

The nature of this curve can be used to derive a very important metric for this system - that of how many users can this system support? Note that the high load asymptote is reached when server utilization is close to 1. When the server utilization is 1, we say that the system is *saturated*. Since the system is essentially a "finite" system, it is still mathematically *stable*, but it is *saturated* and adding more users just increases the response time, while does nothing to the throughput.

Thus the point at which or just before which the high load asymptote is reached should be considered as the maximum load this system can support. The shape of this curve can be exploited to derive a heuristic that gives us this number as shown in the following figure.

The M^* shown in the figure can be calculated as follows:

Low load asymptote: τ
High load asymptote: $M\tau - \frac{1}{\lambda}$

$$\begin{aligned} M\tau - \frac{1}{\lambda} &= \tau \\ M\tau &= \tau + \frac{1}{\lambda} \\ M^* &= 1 + \frac{1}{\lambda\tau} \\ &= 1 + \frac{\mu}{\lambda} \end{aligned}$$

M^* is called the "saturation number".

Note that the saturation number has a very intuitive explanation. First rewrite it as:

$$M^* = 1 + \frac{\text{thinktime}}{\text{servicetime}}$$

This makes the number immediately clear: look at the server timeline shown here. Suppose the server has just finished serving a request from User 1 at time t_1, then we know that User 1 is expected to send his next request only after "thinktime" amount of time. In the "thinktime" amount of time, the server can process '

$$\frac{\text{thinktime}}{\text{servicetime}}$$

requests, one from one user each. From the figure, clearly, the number of "simultaneous users" this server can handle is 1+ thinktime/service time - which is the saturation we dervided!

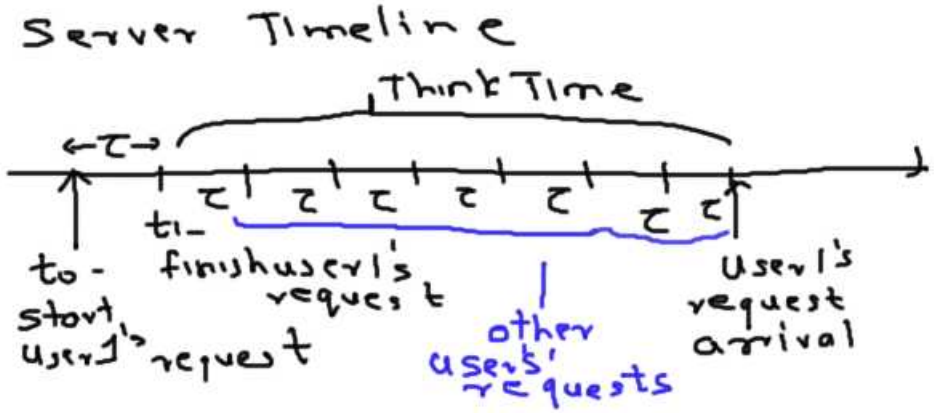


Figure 10: Server Timeline

2.4.2 Closed System: Little's Law

Little's Law can be applied to a closed system if we visualize the "boundary" to which we have to apply Little's Law as shown in this Figure:

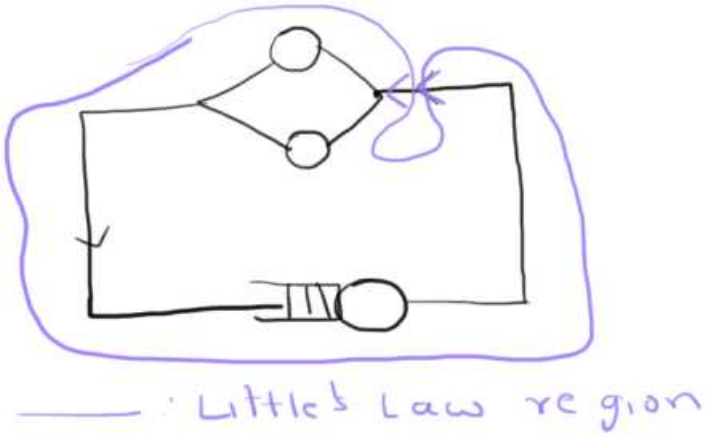


Figure 11: Little's Law for the Closed Queueing System

For us to apply Little's Law we have to abstract the closed system as a system in which there are a fixed number of requests M, that "circulate" through the system. The request is either at the server or "at the client" (when the user is thinking). Thus, the average number of customers in the system is *fixed* at M.

What is the time through the region? Clearly it is : server response time + think time. Thus, we can derive the same equilibrium equation as we got earlier, by using Little's Law:

$$M = \Lambda \left(\frac{1}{\lambda} + R \right)$$

2.5 Queueing Networks

So far we have studied systems in which there is just one server to which a request has to go. In reality though, a request may need to go through multiple queueing systems until it is completely finished. E.g. a request in a typical "multi-tier" Web application needs service from a Web Server, Database server, perhaps an Authentication server, etc before it is fully done.

Such systems are represented by *queueing networks*.

2.5.1 Closed Tandem Queuing Network

First, let us look at a simple queuing network as shown in the following figure. Here a request first needs service from Server 1, then from Server 2 and then is "done". Queues in such a configuration are called *tandem queues*.

Let us carry out asymptotic analysis on this queueing system as done earlier. Suppose service time at Server 1 is τ_1 and at Server 2 is τ_2.

At $M = 1$:

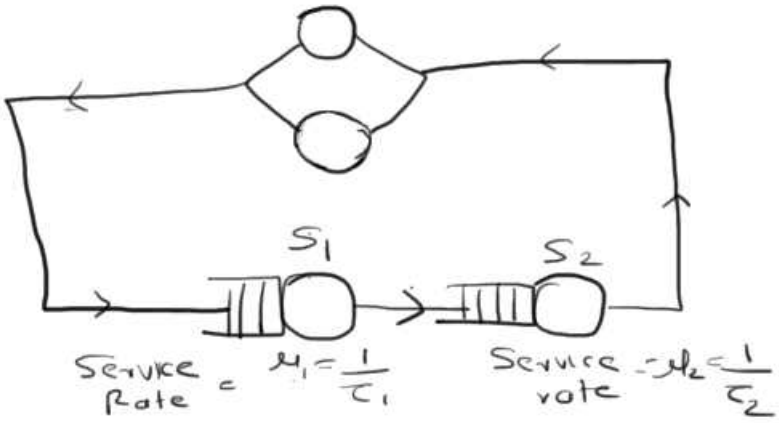


Figure 12: Tandem Queues

$$R_{sys} = \tau_1 + \tau_2 \tag{3}$$

$$\Lambda = \frac{1}{\frac{1}{\lambda} + \tau_1 + \tau_2} \tag{4}$$

Applying Little's Law at any user level M, we have:

$$M = \Lambda \times (R_1 + R_2 + \frac{1}{\lambda})$$

Let $R_{sys} = R_1 + R_2$ be the overall system response time. Then

$$R_{sys} = \frac{M}{\Lambda} - \frac{1}{\lambda}$$

where R_1, R_2 are the response times at server 1 and 2 respectively. In case of the tandem queuing system, for large M, $\Lambda \rightarrow \min \mu_1, \mu_2$. That is, the maximum throughput will be whatever the slower server can support. We term this server the *bottleneck server* of the system.

Let $\mu_B = \min \mu_1, \mu_2$. Then for large M

$$R_{sys} = \frac{M}{\mu_B} - \frac{1}{\lambda}$$

We can apply Kleinrock's saturation heuristic here also to find the maximum number of customers the system can support.

$$R_{low} = \tau_1 + \tau_2 = R_{high} = \frac{M^*}{\mu_B} - \frac{1}{\lambda}$$

$$M^* = \mu_B \left[\tau_1 + \tau_2 + \frac{1}{\lambda} \right]$$

2.6 Closed Queueing Networks - Mean Value Analysis

Now consider a general queuing network as shown in the figure below.

//insert closed QN

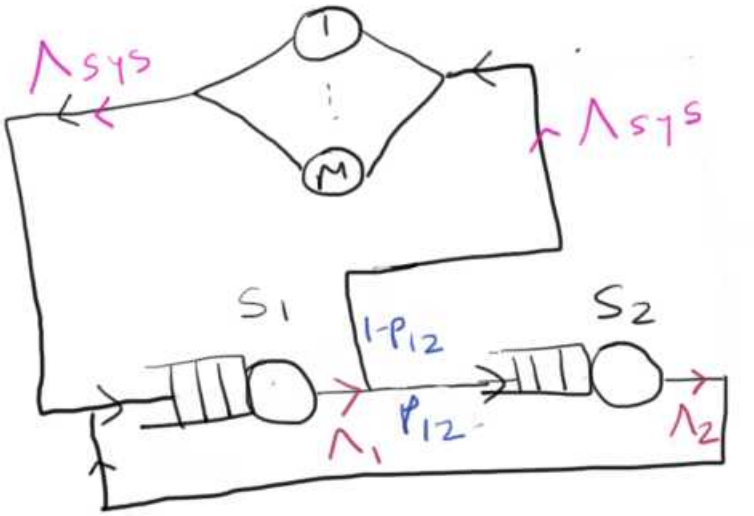


Figure 13: Closed Queueing Network

In general we assume that there are $m + 1$ queuing stations - station 0 is the client node and stations 1 to m are service stations. In this queuing network, we assume that when a request is done at a certain service station i, it goes to service station j with a probability p_{ij}. Note that in this generalized closed queuing network, *feedback* is possible. That is, we can have $p_{ii} > 0$, meaning that as soon as a request is done at station i, it may need service at the same station immediately again, with probability p_{ii}. Feedback may also happen from another station - i.e. a request goes from station i to j and then back to i. Feedback results in a request needing multiple visits to a station before it is *completed*.

The probabilities p_{ij} determine the average number of visits a request makes to any station before it is completed. Let this average be denoted by v_i. Thus v_i is the average number of visits made by a request to station i before it leaves (i.e. goes back to the client node). This is also called the *visit count*.

In the given figure, how do we calculate, say v_1? We see every request issued by the client visits server 1 at least once. Also, every request that visits server 2 also visits Server 1. Thus it is fair to say that

$$v_1 = 1 + v_2$$

.

Conversely, we can see that a fraction p_{12} of requests that visit server 1 also visit server 2. Thus

$$v_2 = p_{12} \times v_1$$

Solving the above equations shows:

$$v_1 = \frac{1}{1 - p_{12}}$$

and

$$v_2 = \frac{p_{12}}{1 - p_{12}}$$

In general, visit count to station j is given by the following system of linear equations:

$$v_i = \sum_{i=1}^{M} p_{ij} \times v_i, i = 1, 2, 3, \ldots$$

Once visit counts are calculated, a very important quantity of a queuing network is defined, called the *service demand*. This is the *total* time over multiple visits that a request uses service from a server.

Thus service demand at station i is given by

$$B_i = v_i \times \tau_i$$

For queuing networks, we can define two sets of metrics - one set at the "system level" and one set for each node i. Furthermore, it makes most sense to study these metrics as functions of the "load level" which is the number of users (or clients), M. Thus we have:

- $R_{sys}(M)$, "System Response Time": Time taken by a request from entering the server network to leaving the server network. This is the complete request response time as "experienced" by a user.

- $R_{cycle}(M)$, "Cycle Time": This is the time taken by a request to complete a cycle through the entire queueing network. Thus $R_{cycle} = \text{thinktime} + R_{sys}$.

- $\Lambda_{sys}(M)$, "System Throughput": The rate at which requests are *completed*, which will also be equal to the total rate at which M users issue requests to the system.

- $R_i(M)$, "Node Response Time": Response time at node i during *one visit* by a request. Note that because a request visits station i on an average v_i times, and each time spends R_i amount of time at server i.

$$R_{sys}(M) = \sum_{i=1}^{M} v_i R_i(M)$$

- $\Lambda_i(M)$, "Node throughput": Request throughput at a node i, is given by

$$\Lambda_i(M) = v_i \times \Lambda_{sys}(M)$$

- $N_i(M)$, "Average number of customers at node i: Note that

$$N_0(M) + N_i(M) + \ldots N_m(M) = M$$

- $\rho_i(M)$, "Utilization at node i". As usual,

$$\rho_i(M) = \Lambda_i(M) \times \tau_i$$

2.6.1 Asymptotic Analysis

As usual, let us start with analyzing the above metrics at low load ($M = 1$).

The response time at low load is

$$R_i(1) = \tau_i$$

and

$$R_{sys}(1) = \sum_i v_i \times \tau_i$$

The throughput is given by the same reasoning as for the canonical system (this can be derived both by Little's Law and by observing one user's timeline).

$$\Lambda_{sys}(1) = \frac{1}{\frac{1}{\lambda} + R_{sys}(1)}$$

The utilization of each service station is given by:

$$\rho_i(M) = (\Lambda_{sys}(M) \times v_i) \times \tau_i$$

This formula holds for $M = 1$ as well as $M > 1$.

Expanding the above formula for $M = 1$, we can see that it can also be interpreted as the fraction of time spent in the server i during the cycle of one request through the system (this logic works only when there is just one request in the server).

$$\rho_i(1) = \frac{v_i \tau_i}{\text{thinktime} + \sum_{i=1}^{M} v_i \tau_i}$$

Also, for any M the Little's Law equality for closed systems holds:

$$M = \Lambda_{sys} \times R_{cycle}$$

$$M = \Lambda_{sys} \times \left[\frac{1}{\lambda} + \sum_{i=1}^{M} v_i R_i \right]$$

Now let us consider the high load asymptotes, i.e., for large M.

At what value does system throughput flatten out as M goes to infinity?

The maximum rate at which the system completes the request will be determined by the most "slowest" server in the system. Note that, in a queueing network, whether a server's relative capacity is determined not only by its raw service rate ($\mu_i = \frac{1}{\tau_i}$) but the rate at which it can *complete* requests, which accounts for the average number of visits to a server.

Consider a simple example of some sort of an office (a government office, or a bank) which has multiple counters for serving customers. Suppose there the clerk at counter 1 needs an average of 5 minutes to service a customer, but a customer needs an average of 3 visits on an average to this counter before his work in this office is complete. Suppose the clerk at counter 2 needs an average of 10 minutes per customer, but each customer needs an average of only 1 visit to this counter. It is clear that the clerk at counter 1 has a "raw" service rate of 12 tasks per hour, but can fully process only 4 customers per hour. On the other hand the clerk at counter 1 has a raw service rate of 6 tasks per hour, but can fully process 6 customers per hour. It should be obvious then that the customer processing capacity of this office is 4 customers per hour. Thus although the clerk at counter 2 was "slower" - he is not the one that is limiting the throughput of this office.

This discussion should lead you to the obvious conclusion. The maximum throughput possible for a queuing network is given by:

$$\lim_{M\to\infty} \Lambda_{sys}(M) = \frac{1}{\max_i v_i\tau_i}$$

Here $\max_i v_i\tau_i$ is termed as the *bottleneck service demand* and the server with the maximum service demand is termed the *bottleneck server*.

Let $B_{max} = \max_i v_i\tau_i$ be the bottleneck service demand.

Then for large M,

$$M \approx \frac{\left[\frac{1}{\lambda} + R_{sys}\right]}{B_{max}}$$

$$R_{sys}(M) \approx M\dot{B}_{max} - \frac{1}{\lambda}$$

Thus the high load asymptote of R_{sys} is linear with slope B_{max}.

You can apply Kleinrock's saturation number derivation heuristic here also. We leave it to the reader to prove that the saturation number by this heuristic is:

$$M^* = \frac{\frac{1}{\lambda} + \sum_{i=1}^{M} v_i\tau_i}{B_{max}}$$

Suppose we define $\mu_{min} = \frac{1}{B_{max}}$. That is, this is the maximum system throughput capacity of the queuing network. One can also define an even rougher heuristic where we approximate the whole network as a single server with service rate of μ_{min}. In that case, the saturation number is given by:

$$1 + \frac{\mu_{min}}{\lambda}$$

We leave it to the reader to think about how these two heuristics compare.

2.6.2 Mean Value Analysis

So far in most of our analysis we have only focussed on asymptotes, and on some other simple metrics that can be derived without involving advanced mathematical models (e.g. stochastic processes).

It turns out that in case of Closed Queueing Networks, if we make use of a very useful theorem called the Sevcik-Mitrani Arrival Theorem, we can derive all metrics even for the "medium load" values of M (i.e neither too low, nor too high), with a very elegant and intuitive algorithm called Mean Value Analysis.

First, we must understand the Arrival Theorem. For this, we make use of all the metrics defined above and one additional one:

- $N_i^A(M)$: Average number of customers at node i *as seen by an arriving request.*

Thus, $N_i^A(M)$ is a *conditional* average, based on this number as noted only at request arrival times at i (and which does not include the arriving request). $N_i(M)$ is an *unconditional* average of this number. In general, we do not expect $N_i^A(M)$ to be equal to $N_i(M)$ - i.e. we do not expect conditional and unconditional averages to be the same.

[Imagine you note the number of students in a classroom at several uniformly and evenly sampled times of the day and take an average. Now suppose you take another average conditioned on the time of the observation being between 6pm and 7pm. These two averages will certainly not be the same].

The arrival theorem applies for certain types of queueing networks called *Jackson networks*. Closed Jackson networks are the kind of queuing networks we have described so far, with one more constraint: the service times and think time are *exponentially distributed*.

With this constraint, the arrival theorem (informally) states that, for a closed queuing network with number of users M, the average number of customers at node i that an arriving request *sees*, is the unconditional average number of customers at node i, when the queueing network has *one less customer* in the system.

In other words, the arrival theorem states that:

$$N_i^A(M) = N_i(M-1)$$

.

This equivalence suggests a very powerful way of solving the model - that of recursion (or iteration). This method is called *mean value analysis*.

Let us start with what we know.

- At $M = 0$, we know that $N_i(0) =$ for all i. Thus, we have an intial value for the N_i's. Now we will assume that we know $N_i(M-1)$ and try to go step-by-step until we have derived $N_i(M)$. Then our recursion will be properly defined.

- The arrival theorem tells us that at load level M, an arriving request at node i will see an average of $N_i(M-1)$ requests in front of it. Further more, service time is *memoryless*. Thus, the expected response time at all the service stations will be:

$$R_i(M) = [N_i(M-1) + 1] \times \tau_i$$

- Now we can write R_{sys} as

$$R_{sys}(M) = \sum_{i=1}^{M} v_i R_i(M)$$

- Once we know $R_{sys}(M)$, we know $R_{cycle}(M) =$ thinktime $+ R_{sys}(M)$. So we can calculate $\Lambda_{sys}(M)$ by applying Little's Law to the whole system:

$$\Lambda_{sys}(M) = \frac{M}{R_{cycle}(M)}$$

- Now that we know the system throughput, we can calculate node througput as

$$\Lambda_i(M) = v_i \times \Lambda_{sys}(M)$$

- Recall that we already calculated the per-visit node response time $R_i(M)$ in the first step itself. So we can apply Little's Law at the node level:

$$N_i(M) = \Lambda_i(M) \times R_i(M)$$

  This completes the recursion.

In this way Mean Value Analysis (MVA) can be used to calculate performance metrics at all load levels M for a Jackson closed queueing network.

2.7 Summary

So far we learnt concepts and laws in queuing systems such as:

- The Kendall Notation
- The utilization law
- Asymptotic metrics for the $M/M/c/K$ class of queues
- Calculating averages of all metrics for Jackson closed queuing networks using Mean Value Analysis. We had to understand Sevcik-Mitrani's Arrival Theorem for this purpose.
- The importance of memorylessness in making some conclusions.

We still do not know, for example, what is the average response time of an $M/M/1$ queue. We know that $N = \lambda R$ and if we know N we can determine R. But how do we determine N (the average number of customers in the system) ? It turns out that we need to determine the *distribution* of the number of customers in the queuing system to then find the average. And for this, the mathematical toolbox we need is of *stochastic processes*.

First, we need to refresh some probability basics needed for our purposes. The field of probability and statistics is vast, we will only revise those parts that we need.

3 Probability and Random Variables

Queuing systems analysis depends on a thorough understanding of *random variables* and their *distribution functions*. For that we need to understand basic probability. So, before we go to random variables, let us first quickly do some warm-up exercises in probability.

3.1 Simple Probability Models

Example 1: Let bit error rate/probability (b.e.r.) on a link be e, then what is the probability that a packet of length L bits is correctly transmitted? Assume:

(a) All bits need to be correct.
Solution:
P[packet is correct] = P[all bits correct] $=(1-e)^L$
This assumes that bit errors are **independent**.

(b) Error correcting code can correct 2 bit errors.
Solution:
P[packet is correct = P[$\leq$ 2 bits in error]
γ = P[0 bit error] + P[1 bit error] + P[2 bit error]
$\gamma = (1-e)^2 + Le(1-e)^{L-1} + {}^{L}C_2 e^2 (1-e)^{L-2}$

(c) Suppose packet can go on one of two links with b.e.r. e_1, e_2 respectively. What is the probability of succesful transmission?
Solution:
P[packet goes on link1] = p_1
P[packet goes on link2] = p_2

From case(b), we have:
γ_1 ... success on link1
γ_2 ... success on link2

A : packet is successful
B_1 : Link1 is used
B_2 : Link2 is used

$P[A|B_1] = \gamma_1$ (this is **conditional probability**)
$P[A|B_2] = \gamma_2$

$P[A] = P[A|B_1] + P[A|B_2] = \gamma_1 P_1 + \gamma_2 P_2$

This is theorem of **total probability**.

Definition of conditional probability:

$$P[A|B] = \frac{P[A \cap B]}{P[B]}$$

Definition of total probability:
Let $S = B_1, B_2, \ldots, B_n$ be the event space where,
$B_1, B_2, \ldots, B_n$ are mutually exclusive and exhaustive events.
Then, by law of total probability, we have,

$$P[A] = \sum_{i=1}^{n} P(A|B_i) \cdot P(B_i)$$

Bayes' Theorem The event space can be viewed as a collection of **Mutually exclusive and Collectively exhaustive** events. Depending on our intuition or assumptions we can have a prior estimate of the events which can be termed as Prior probability. Given that an event occured in the random experiment the probabilities for the events under consideration will change. This probability can be termed as the Posterior probability. Bayes theorem helps in determining the posterior probability given some prior information of the events.

Let the event space consists of B_j mutually exculsive and collectively exhaustive events. Knowing the prior probabilities and a event A occured the posterior probabilities can be calculated by:

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum_i P(A|B_i)P(B_i)}$$

Example:

Suppose, in the two-link case, the packet arrived successfully. What is the probability that it came on link1?

Solution:

$$P(B_j|A) = \frac{P(A \cap B_j)}{P(A)} = \frac{P(A \cap B_j)}{\sum_i P(A|B_i)P(B_i)} = \frac{P(A|B_j)P(B_j)}{\sum_i P(A|B_i)P(B_i)}$$

P[Link1 | packet is succesful] :

$$P(B_1|A) = \left(\frac{P(A|B_1)P(B_1)}{\gamma_1 P_1 + \gamma_2 P_2} \right)$$

Law of diminishing returns:

P[server is up] = r = 0.9

2 Server system :

P[system is up] = P[at least one server is up]
r = 1 - P[both servers are down]
r = 1 - $(1-r)^2$

For an n-server parallel system, system is up when at least one server is up.
$R_{\text{server}} = 1 - (1-r)^n$

3.2 Efficiency of ALOHA

3.2.1 Pure ALOHA

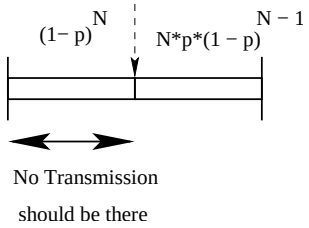


Figure 14: Pure ALOHA

Lets do a "quick and dirty" probabilistic analysis of ALOHA. For the purposes of our analysis let us assume all packets are of equal size, and transmission time is unit time. Suppose the probability of a station transmitting in this unit time is p and the number of stations is N.

Note that this analysis is a bit "hand-wavy" and inaccurate. But it is popular and acceptable as a first rough model. .

What is the expected throughput of such a system? e.g. How many successful packets transmitted per unit time? Since transmission time is the unit, this is really just the probability of successful transmission in the channel. Thus,
Prob[successful transmission]
= Prob[no transmission in previous unit slot] × Prob[no other transmission in this slot]

S = $(1-p)^N \cdot Np(1-p)^{N-1}$

$$S = Np(1-p)^{2N-1}$$

For N stations, what should be the load per station, that maximizes the throughput? We can try to guess this intutively. If there are N stations, one expects that if each one's probability of transmission at any time is $\frac{1}{N}$, that feels like a "full and fair share" that should mazimize the throughput.

Formally, the probability of transmission that maximumizes throughput is given by:

$$\begin{aligned} \frac{dS}{dp} &= N(1-p)^{2N-1} - Np(2n-1)(1-p)^{2N-2} = 0 \\ p &= \frac{1}{2N} \\ S_{\max} &= \frac{1}{2}\left(1 - \frac{1}{2N}\right)^{2N-1} \end{aligned}$$

Thus, note that our analysis tells us that our expectation was *not* met. Stations need to produce much lesser load than we hoped, to maximize network throughput of pure ALOHA.

Now, as N increases, throughput decreases. The limiting throughput is:

$$\lim_{N\to\infty} S = \frac{1}{2e} \sim 0.18 = 18\%$$

See the corresponding spreadsheet to understand this behaviour better and play with the parameters.

3.2.2 Slotted ALOHA

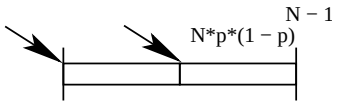


Figure 15: Slotted ALOHA

It is clear from the previous analysis that the throughput is reduced because transmissions from the previous "slot" can also collide with the current transmission. Slotted ALOHA improves this by actually dividing time into slots, and requiring that stations transmit only at the beginning of a slot. In this case,

P[successful transmission] = P[max. 1 transmission in the slot]

$$S = Np(1-p)^{N-1}$$

The per-station load that maximimizes success probability is given by

$$\frac{dS}{dp} = N(1-p)^{N-1} + Np(N-1)(1-p)^{N-2} = 0,$$

Thus, highest success rate is at $p = \frac{1}{N}$. Thus slotted ALOHA meets our expectation. At this value of p,

$$\begin{aligned} S &= \left(1 - \frac{1}{N}\right)^{N-1} \\ \lim_{N\to\infty} S &= \frac{1}{e} \sim 0.37 = 37\% \end{aligned}$$

We can see that this is twice of the throughput achievable by pure ALOHA.

This was just an illustration of using simple probabilistic methods to analyze protocol performance. We will now move on to study random variables and probability distributions.

3.3 Random Variables

Random variables are results of random experiments - that is experiments that can have random, unpredictable outcomes. Formally, a Random Variable is a *function* that maps a "sample space" to set of real numbers. A sample space is the set of possible outcomes of an experiment.

E.g. suppose the experiment is of sending a packet across a link. The outcome is a successful or unsuccessful transmission. We define a random variable X as a function which maps success to the number 1 and failure with the number 0.In practice, though when we talk about random variables, we do not use a "function" notation. In the above example, conventionally we can say that "X is a random variable that takes value 1 when the packet transmits successfully, and 0 if the transmission fails.".

There are two types of random variables (r.v.): discrete, and continuous. Discrete random variables take discrete values, while continues random variables take continues values.

3.3.1 Discrete Random Variables illustrated with a Go-back-N ARQ System

Let us take an example of a link between two nodes and packets being transmitted over these links using a sliding window protocol, namely, the Go-Back-N ARQ protocol (Figure 16).

Let the window size be n, and retransmission timout be $= n-1$ packet transmission times. That is we assume that packet transmission times are unit time. (This example is from the Bertsekas and Gallagher book).

What random variables can be find in this example? What random *experiment* is happening in this example?

The experiment with a random outcome in this example is each packet transmission. The outcome of a packet transmission is either a success or a failure. Such an experiment, with *two outcomes* is called a *Bernoulli trial*.

The random variable associated with a Bernoulli trial is a *Bernoulli random variable*, which we define as follows $X = \begin{cases} 1, & \text{if packet transmission successful} \\ 0, & \text{otherwise} \end{cases}$

Also let $P[X = 1]$ be p, then $P[X = 0]$ is $1 - p$.

The following random variables can also be found in this example:

- Y = Number of packets in one window that were transmitted successfully. Note here that $Y \in \{0, 1, \ldots, n\}$.

  $$P[Y = i] = \binom{n}{i} p^i (1-p)^{n-i}, \quad i = 0, 1, 2, ...n$$

  This is a *Binomial random variable*.

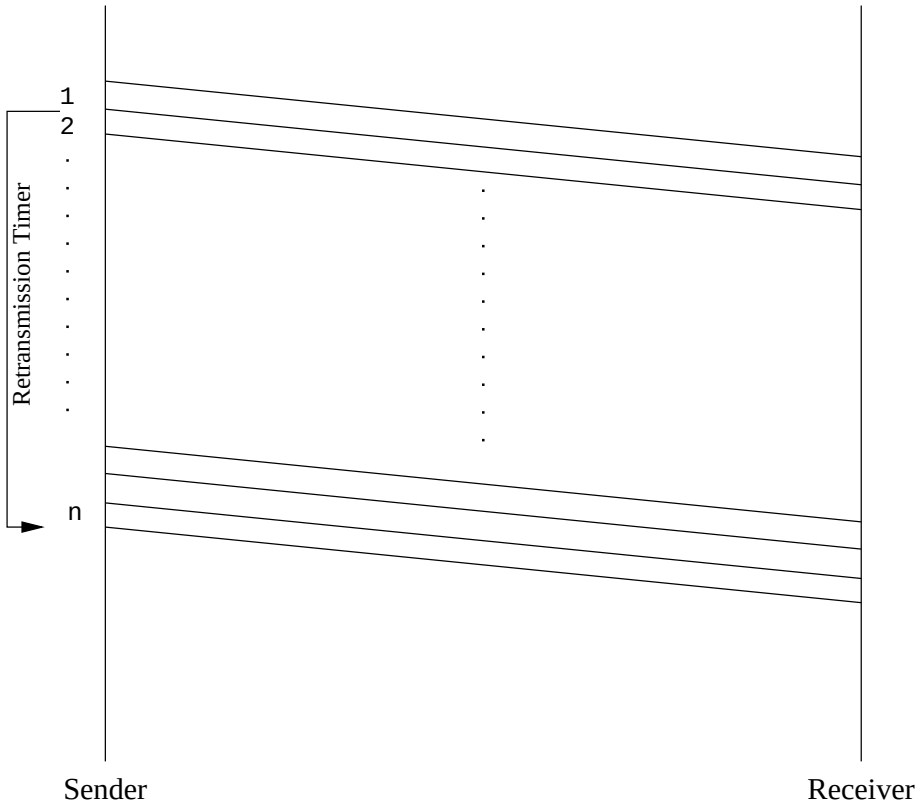


Figure 16: Go-back-n ARQ system

- N = Number of transmissions required to send a packet correctly. $N \in \{1, 2, 3, ...\}$. Here, exactly k transmissions will be required if $k-1$ transmission fail followed by one successful transmission. Thus,
$$P[N = k] = (1-p)^{k-1}p$$
.

  This is the Geometric random variable.

- N_R = Number of re-transmissions requried to send a packet correctly. In other words, number of failures before the packet was transmitted correctly. The packet may succeed in the first transmission itself, so $N \in \{0, 1, 2, 3..\}$ and
$$P[N_R = k] = (1-p)^k p$$

- D = total time required for a successful transmission of a packet (including time retransmissions). Let's call it packet transfer delay. Note that here we start counting the time required for a packet to be sent, only when all the packets before it in the window have been successfully sent (removed from the packet buffer).

  If the first transmission is a success, the packet needs only 1 time unit.

  If first transmission is a loss, this will be discovered after the retransmission timer expires - i.e. $n-1$ time units after transmission. Then we need one more time unit to transmit. Thus, if two transmissions are required, time taken is $1 + n - 1 + 1 = n + 1$.

  If k transmissions are a failure, and time required will be $kn + 1$. Thus,
$$P(D = 1 + kn) = (1-p)^k p \quad \text{for} \quad k = 0, 1, 2, \ldots$$

  This is not a well-known distribution and has no name.

- If Z = the size of the sliding window as observed at any random time, it is fair to assume that it will be of any size between 1 and n with equal probability. This is the *Discrete Uniform Distribution*. Thus
$$P[X = k] = \frac{1}{n}, \quad k = 1, 2, \ldots, n$$
.

There is one more important distribution that we can associate with this example. This distribution needs a slightly involved derivation.

Suppose we want to ask the question: how many packets are likely to arrive at the link for transmission, in a given timeframe? This question characterizes the arrival process and the "load" on the link.

A common set of assumptions that characterize the arrival process are as follows:

- Packets arrive into the source in an infinitesimal interval $\triangle t$ with probability $\lambda \triangle t$, where $\lambda > 0$. (Note that λ can be greater than 1. We assume that Δt is small enough that the product is a probability.)

- P[two or more arrivals in $\triangle t$] $\rightarrow 0$.

- Arrivals in different intervals are independent of each other.

Under these assumptions, what is the probability that k jobs arrive in time interval $(0,t)$? Let $N_A(t)$ be the random variable denoting the number of jobs arriving in $(0,t)$. Then we want to know $P[N_A = k]$. We proceed as follows: Divide time $(0,t)$ into n slots such that $\frac{t}{n} = \triangle t$.

$$\begin{aligned}
P[k \text{ jobs in } (0,t)] &= \text{exactly k out of n slots have job arrivals} \\
&= \binom{n}{k}(\lambda\triangle t)^k(1-\lambda\triangle t)^{n-k} \\
&= \binom{n}{k}(\lambda\frac{t}{n})^k(1-\lambda\frac{t}{n})^{n-k} \\
&= \left(\frac{n!}{(n-k)!n^k}\right)\frac{(\lambda t)^k}{k!}\frac{\left[\left(1-\frac{\lambda t}{n}\right)^{-n/\lambda t}\right]^{-\lambda t}}{\left(1-\frac{\lambda t}{n}\right)^k}
\end{aligned}$$

Since the assumption is that$\Delta t$is small, $we have$

$$\begin{aligned}
P[k \text{ jobs in } (0,t)] &= \lim_{n\to\infty}\frac{\overbrace{n\cdot n-1\cdots(n-k+1)}^{k \text{ terms}}}{n\cdot n\cdots n}[\cdots] \\
&= \frac{(\lambda t)^k}{k}\lim_{n\to\infty}\frac{\left[\left(1-\frac{\lambda t}{n}\right)^{-n/\lambda t}\right]^{-\lambda t}}{\left(1-\frac{\lambda t}{n}\right)^k} \\
&= \frac{(\lambda t)^k}{k}\frac{\left[\lim_{n\to\infty}\left(1-\frac{\lambda t}{n}\right)^{-n/\lambda t}\right]^{-\lambda t}}{\left(1-\lim_{n\to\infty}\frac{\lambda t}{n}\right)^k} \\
\lim_{n\to\infty} &= \frac{(\lambda t)^k}{k}e^{-\lambda t} \\
&\sim Poisson(\lambda t)
\end{aligned}$$

Let $\lambda t = \alpha$.

$$N_A \sim Poisson(\alpha)$$
$$\Rightarrow P[N_A = k] = \frac{\alpha^k e^{-\alpha}}{k!}$$

3.4 Important Distributions of Discrete Random Variables

We now define some terms about random variables and their distributions semi-formally, and summarize the discrete distributions that we saw above.

The **Probability Mass Function** $P_X(x)$ of a random variable X gives the probability of occurance of each value of the random variable.

$$P_X(X = x) \longrightarrow [0,1]$$

where

$$x \in \text{Sample Space}(S)$$

Some properties of a PMF are:

- $0 \le P_X(x) \le 1$

- $\sum_x P_X(x) = 1$

The **Cumulative Distribution Function (CDF)** $F_X(t)$ of a random variable X represents the cumulative probability of X till point $(x = t)$.

$$F_X(t) = P(x \leq t) = \sum_{x \leq t} P_X(x)$$

where $t \in \mathbf{R}$ and t can be continuous.

Example: $X = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

$$P_X(x) = \frac{1}{10} \qquad F_X(8.6) = \frac{8}{10}$$

Some properties of a CDF, also referred-to as simply the *Distribution Function*, are the following:

- $0 \leq F_X(t) \leq 1$
- Monotonically nondecreasing.
- $\lim_{t \to \infty} F_X(t) = 1$
- $\lim_{t \to -\infty} F_X(t) = 0$

Now we summarize the pmfs and CDFs of some well-known distributions. For each distribution, we mention what is described as the *parameter* of the distribution.

3.4.1 Bernoulli PMF

Parameter: p

$$X \in \{0, 1\}$$

$$P_X(1) = p \qquad P_X(0) = 1 - p$$

$$F_X(t) = \begin{cases} 0, & \text{for } t < 0 \\ 1 - p, & \text{for } 0 \leq t < 1 \\ 1, & \text{for } t \geq 1 \end{cases}$$

3.4.2 Binomial PMF

Parameters: n, p

$$X \longrightarrow \text{Number of successes } i \text{ in } n \text{ Bernoulli Trials}$$

$$X = 0, 1, 2, \ldots, n$$

$$P(success) = p$$

$$P_X(i) = \binom{n}{i} p^i (1 - p)^{n-i}$$

The Poisson PMF can be used as a convenient approximation to the binomial PMF when n is large and p is small (that is $n \to \infty$ and $p \to 0$, in such a way that $np \to \alpha$):

$$\binom{n}{i} p^i (1-p)^{n-i} \approx \frac{e^{-\alpha}\alpha^i}{i!}$$

where $\alpha = np$

Example: Bit errors in a packet of length L (packet size in bits is large, and error probability is very small).

3.4.3 Geometric Distribution

Parameter: p

$$X \longrightarrow \text{Number of trials upto and includng a success}$$

$$X = 1, 2, 3, \ldots, \infty$$

$$P(success) = p$$

$$P_X(i) = (1-p)^{i-1}p$$

CDF is the probability that we will need less than or equal to i trials. This is one minus the probability of needing more than i trials. This happens w.p. $1-p)^i$, thus:

$$F_X(i) = 1 - (1-p)^i$$

Geometric distribution has the memoryless property. This is apparent if you think like this: Suppose you have been tossing a coin to get a head, but have gotten 20 tails in a row. What is the probability that you will need 20 *more* tosses to get a head? Think for a second and you will know that this probability is the same as if you had just started tossing the coin - i.e. the experiment has no *memory* of number of trials already done.

Formal proof of **Markov Property (memorylessness)**:

X is the total number of trials for success. The additional number of trials required for success, after i unsuccessful trials is denoted by random variable Z. Note that if we know that i trials were unsuccessful, that means we know that X has to be strictly greater than i. This we *know*, so it is a *given condition*. Thus finding the probability of $Z = k$ means we need to find the probability that $X = k + i$, *given* that $X > i$

$$\begin{aligned} P[Z = k] &= P[X = k + i \mid X > i] \\ &= \frac{P[X = k + i]}{P[X > i]} \\ &= \frac{(1-p)^{k+i-1}p}{(1-p)^i} \\ &= (1-p)^{k-1}p \end{aligned}$$

$\Rightarrow Z \to Geom(p)$

Example: Bit error rate on a channel is b. Bit errors are independent. In a packet, the first 10 bits have come error-free. What is the probability that the first bit error is seen at exactly the kth bit after these 10 bits?

Solution:

By memoryless property, $Z \sim Geom(b)$

$P[Z = k] = (1-b)^{k-1}b$

3.4.4 Modified Geometric Distribution

Parameter: p

$$X \longrightarrow \text{Number of failures before a success}$$
$$X = 0, 1, 2, \ldots, \infty$$

$$P_X(i) = (1-p)^i p$$

Example: while not B do S;
X : # of times S is executed

$$\begin{aligned} P(B = true) &= p \\ P(X = k) &= (1-p)^k p \end{aligned}$$

$X \sim ModGeom(p)$

$P[X > k]$ is probability that there are more than k failures, that is there are at least $k+1$ failures $= (1-p)^{k+1}$.

$$F_X(k) = P[X <= k] = 1 - P[X > k] = 1 - (1-p)^{k+1}$$

3.4.5 Poission

Parameter: α

$$P_X(i) = \frac{e^{-\alpha}\alpha^i}{i!}$$

$$F_X(i) = \sum_{k=0}^{k=i} \frac{e^{-\alpha}\alpha^k}{k!}$$

3.4.6 Uniform

Parameter: N

$$X = \{x_1, x_2, x_3, \ldots, x_N\}$$

$$P_X(x_i) = \frac{1}{N}$$

$$F_X(t) = \frac{t}{N}$$

3.4.7 Constant / Deterministic

Parameter: c

$$X = c$$

$$P_X(x) = \begin{cases} 1, & \text{for } x = c \\ 0, & \text{for } x \neq c \end{cases}$$

$$F_X(t) = \begin{cases} 0, & \text{for } t < c \\ 1, & \text{for } t \geq c \end{cases}$$

3.4.8 Indicator Random Variable

$X = 1$ if event A and $X = 0$ if event $\overline{A}$.

3.5 Expectation

The expectation (mean), denoted by $E[X]$, of a random variable X is defined as:

$$E[X] = \sum_i x_i p(x_i) \qquad X \rightarrow \text{discrete random variable} \tag{5}$$

3.5.1 Uniform

$X = \{1, 2, \ldots, n\}$ $\qquad p_X(x) = \frac{1}{n}$

$$E[X] = \sum_{i=1}^{n} \frac{i}{n} \quad = \frac{n+1}{2}$$

3.5.2 Bernoulli pmf

Parameter: p

$$E[X] = p \tag{6}$$

3.5.3 Binomial

Parameters: n, p

$$E[X] = np \tag{7}$$

$$E[\sum_i X_i] = \sum_i E[X_i] \qquad (X_i s \text{ need not be independent}) \tag{8}$$

3.5.4 Geometric

Parameter: p

$$p_X(i) = (1-p)^{i-1} p$$

$$E[X] = \frac{1}{p} \qquad \text{(prove as homework)} \tag{9}$$

3.5.5 Modified Geometric

Parameter: p

$$p_X(i) = (1-p)^i p$$

$$E[X] = \frac{1-p}{p} \qquad \text{(prove as homework)} \tag{10}$$

3.5.6 Poisson

Parameter: α

$$p_X(k) = \frac{\alpha^k e^{-\alpha}}{k!}$$

$$\begin{aligned} E[X] &= \sum_{k=0}^{\infty} \frac{\alpha^k e^{-\alpha}}{k!} \\ &= \alpha e^{-\alpha} \sum_{k=1}^{\infty} \frac{\alpha^{k-1}}{(k-1)!} \\ &= \alpha e^{-\alpha} e^{\alpha} \\ E[X] &= \alpha \end{aligned} \tag{11}$$

3.6 Superposition and splitting of Poisson Arrivals

Suppose (open) arrivals to a queue come from two sources - each of which are Poisson. Then the merged arrival stream is also Poisson. This is because the sum of two Poisson random variables is also Poisson. Recall that "arrivals are Poisson" is short for saying that the number of arrivals in time $(0,t)$ has Poisson distribution with parameter λt where λ is the "rate" parameter.

Let $X \sim Poisson(\alpha_1)$ and $Y \sim Poisson(\alpha_2)$ and let $Z = X + Y$.

What is the probability that $Z = k$? If we know that say, $Y = j$, then this is the probability that $Z = k - j$. Thus, using the Law of Total Probability we can write:

$$\begin{aligned} p_Z(k) &= \sum_{j=0}^{k} p_X(k-j) p_Y(j) \qquad (12) \\ &= \sum_{j=0}^{k} \frac{e^{-\alpha_1} \alpha_1^{k-j}}{(k-j)!} \frac{e^{-\alpha_2} \alpha_2^{j}}{j!} \\ &= \frac{e^{-(\alpha_1+\alpha_2)}}{k!} \sum_{j=0}^{k} \frac{k!}{(k-j)!j!} \alpha_1^{k-j} \alpha_2^{j} \\ &= \frac{e^{-(\alpha_1+\alpha_2)}}{k!} (\alpha_1 + \alpha_2)^k \qquad \text{Binomial Expansion} a \end{aligned}$$

Thus $Z \sim Poisson(\alpha_1 + \alpha_2)$.

Now suppose that an arrival stream which is Poisson is *split* probabilistically. That is, say, an arrival goes to server 1 with probabilty p and server 2 with probability $1 - p$.

Let N denote the number of arrivals in $(0,t)$ in the original stream. Then $N \sim Poisson(\lambda t)$. Let $\alpha = \lambda t$.

Let N_1 denote the number of arrivals going to server 1 in time $(0,t)$. Then what is the probability that $N_1 = k$?

We can condition this on N. If we know that $N = j$, then what is the probability that $N_1 = k$. It is 0 if $k > j$ and otherwise it is the probability that k out of j arrivals went to server 1. Thus

$$\begin{aligned} p_{N_1}(k) &= \sum_{j=k}^{\infty} p_{N_1|N}(k|j) p_N(j) \qquad (13) \\ &= \sum_{j=k}^{\infty} \binom{j}{k} p^k (1-p)^{j-k} \quad \frac{e^{-\alpha}\alpha^j}{j!} \\ &= \frac{e^{-\alpha}(\alpha p)^k}{k!} \sum_{j=k}^{\infty} \frac{\alpha^{j-k}(1-p)^{j-k}}{(j-k)!} \\ &= \frac{e^{-\alpha}(\alpha p)^k}{k!} \quad e^{\alpha(1-p)} \\ &= \frac{e^{-\alpha p}(\alpha p)^k}{k!} \end{aligned}$$

3.7 Continuous Random Variables

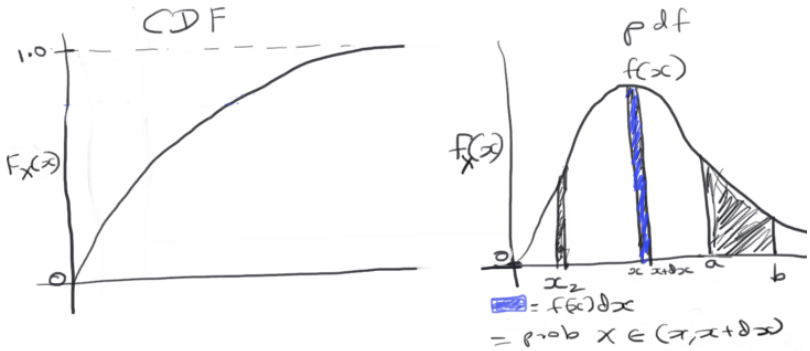


Figure 17: CDF and pdf of continuous random variables

Random variables such as processing time, interarrival time may not always described by discrete values - they are better described by continuous random variables (c.r.vs).

For a continuous random variable, say X, the probability of taking a certain value is 0. i.e.

$P[X = c] = 0$ for any specific value c. This is because the r.v. can take such a large (infinite) set of values, that the probability of taking any one value is essentially zero.

Thus, for c.r.v. we can only find probabilities of the r.v. being in a certain *range*. Thus, for continuous random variables, we define a *cummulative distribution function*, sometimes called just the "distribution function":

$$F_X(x) = P[X <= x] \qquad -\infty < x < \infty$$

The CDF has the same properties as the discrete CDF(Figure 17):

$$\begin{aligned} 0 \leq F_X(x) &\leq 1, -\infty < x < \infty \\ \lim_{x \to -\infty} &= 0 \\ \lim_{x \to -\infty} &= 0 \end{aligned}$$

and $F_X(x)$ is a *monotone non-decreasing function.*

Consider a small range $(x, x + dx)$. We know that

$$P[x < X < x + dx] = F(x + dx) - F(x)$$

.

Thus, for a small dx, this quantity captures the probability of values close to x. This intuition is represented by the *probability density function* (pdf) of a continuous random variable, which is given by

$$f_X(x) = \frac{dF_X(x)}{dx}$$

Thus $f_X(x)dx$ approximates the probability of values in the range $(x, x + dx)$. Note that $f_X(x)$ itself is *not a probability.*

A PDF is a function that must satisy the following properties:

$$\begin{aligned} f_X(x) &\geq 0, \quad -\infty < x < \infty \\ \int_{-\infty}^{\infty} f_X(x)dx &= 1 \end{aligned}$$

It follows from the above definitions that

$$\begin{aligned} F_X(x) &= \int_{-\infty}^{x} f_X(t)dt \\ P[a < X <= b] &= F_X(b) - F_X(a) = \int_a^b f_X(t)dt \end{aligned}$$

It follows that

$$P[X = c] = \int_c^c f_X(t)dt = 0$$

In other words, for continuous r.v.s, the *area under the curve* of a pdf function is the probability - "pieces" of this area will be probabilities, not the values. So, although $P[X = c] = 0$, $P[c < X < c + \delta c]$ is defined and can be non-zero. Also, while the value of the pdf is not a probability, it is good indicator of ranges of the random variable that are more likely then the other. E.g. in Figure 17, values in a small range around x are more likely than in a range around x_2 (this is because the area slice under $f_X(x)$ will be be larger than the area slice under $f_X(x_2)$).

3.7.1 Exponential Distribution

This is a distribution which has some unique properties, which make it a very useful distribution in modeling.

If a random variable X has a distribution given by

$$F(x) = 1 - e^{-\lambda x}, \quad X \geq 0, \lambda > 0$$

then X is said to have the *exponential* distribution with parameter λ. This distribution is also denoted by $EXP(\lambda)$.

The pdf of such a random variable is given by:

$$f(x) = \lambda e^{-\lambda x}$$

The exponential distribution has the *memoryless* property. An example of this is as follows: suppose you are at a phonebooth while somebody is on the phone. Suppose this person has been already talking for 10 minutes. What is the probability that he will hold the phone for 10 more minutes? If the call holding time distribution is exponential (which it often is), the probability of talking for 10 hold minutes is the same as if the person had just started talking - i.e. it is "memoryless". Let us prove this mathematically:

Let X be the random variable denoting the holding time. Suppose we know that $X > t$ ("person has already been talking for t time units"). Let Y be the remaining time. Then $X = Y + t$.

$$\begin{aligned}
P[Y \leq y] &= P[X \leq y + t | X > t] \\
&= \frac{P[X \leq y + tX > t]}{P[X > t]} \\
&= \frac{P[t < X \leq y + t]}{P[X > t]} \\
&= \frac{F(y+t) - F(t)}{[1 - F(t)]} \\
&= \frac{(1 - e^{-\lambda(y+t)}) - (1 - e^{-\lambda t})}{e^{-\lambda t}} \\
&= \frac{e^{-\lambda t}[1 - e^{-\lambda y}]}{e^{-\lambda t}} \\
&= 1 - e^{-\lambda y}
\end{aligned}$$

3.7.2 Relationship of Poisson distribution and Exponential Distributions

Poisson distribution can be derived in the context of arrivals of requests to a system. Suppose that number of arrivals Nm to a system in an interval $(0, t]$ follows Poisson distribution.

$$\Rightarrow P[N = k] = \frac{(\lambda t)^k e^{-(\lambda t)}}{k!}$$

What is the distribution of the interarrival time (X)? We can pose this question as what is $P[X > t]$. In words, if one arrival happenned at time 0, then the probability that there has been no arrival till at least time t is that there were 0 arrivals is the interval $(0, t]$. Thus $P[X > t] = P[N = 0] = e^{-(\lambda t)}$. Thus$P[X < t] = 1 - e^{-(\lambda t)}$. Thus X has $EXP(\lambda)$ distribution.

Poisson and Exponential distributions complement each other. When the arrivals are Poisson, the inter-arrival time is Exponential. If arrivals are Poisson with rate λ, interarrival time is exponentially distributed with mean $1/\lambda$.

3.7.3 Order Statistics

If we are given a set of random variables $X_1, X_2, \ldots, X_n$, we often need to find the distribution of the minimum, or the maximum of these variables. Thus let Y and Z be random variables defined as

$$Y = \min_i\{X_i\}$$

and

$$Z = \max_i\{X_i\}$$

. Then if we know $F_{X_i}(t)$'s, can we determine $F_Y(t)$ and $F_Z(t)$?
It is clear that Y, the minimum, will be greater than some y if all X_i's are greater than y. Thus:

$$P[Y \geq y] = P[X_1 \geq y]P[X_2 \geq y] \ldots P[X_n \geq y]$$

which means

$$1 - F_Y(y) = \Pi_{i=1}^n(1 - F_{X_i}(y))$$
$$F_Y(y) = 1 - \Pi_{i=1}^n(1 - F_{X_i}(y))$$

Similarly, Z, the maximum will be less than some y if all X_i's are less than y. Thus:

$$P[Z \leq y] = P[X_1 \leq y]P[X_2 \leq y] \ldots P[X_n \leq y]$$

which means

$$F_Z(y) = \Pi_{i=1}^n F_{X_i}(y)$$

3.7.4 Minimum of Exponentially Distributed Random Variables

Let X_i's be random variables, each with distribution $EXP(\lambda_i)$. Let $Y = \min_i\{X_i\}$. Then

$$\begin{aligned} F_Y(y) &= 1 - \Pi_{i=1}^n(1 - F_{X_i}(y)) \\ &= 1 - \Pi_{i=1}^n e^{-\lambda_i y} \\ &= 1 - e^{-(\sum_i \lambda_i)y} \end{aligned}$$

Thus the minimum is $EXP(\sum_i \lambda_i)$.

3.8 Distributions of Continuous Random Variables

Hypoexponential Distribution

Sum of exponential stages.

Consider $X = Y + Z$, where $Y \rightarrow \text{EXP}(\lambda_1)$ and $Z \rightarrow \text{EXP}(\lambda_2)$.

Then $X \rightarrow \text{HYPO}(\lambda_1, \lambda_2)$ and it has pdf:

$$f(t) = \frac{\lambda_1\lambda_2}{\lambda_2 - \lambda_1}(e^{-\lambda_1 t} - e^{-\lambda_2 t}), \qquad t > 0$$

Expectation:
Parameters: $\lambda_1, \lambda_2, \ldots, \lambda_n$

$$E[X] = \sum_{i=1}^n \frac{1}{\lambda_i}$$

Erlang Distribution

Special case of HYPO where all stages have identical distribution.

$$f(t) = \frac{\lambda^r t^{r-1} e^{-\lambda t}}{(r-1)!}, \qquad t > 0, \quad \lambda > 0, \quad r = 1, 2, \ldots$$

Represented as ERLANG(n, λ) which is quivalent to sum of n EXP(λ).
Expectation:
Parameters: n, λ

$$E[X] = \frac{n}{\lambda}$$

Hyperexponential Distribution

Mixture of exponential phases where one and only one of the alternative phases is chosen. The pdf of a 2-way hyperexponential random variable is:

$$f(t) = \alpha_1 \lambda_1 e^{-\lambda_1 t} + \alpha_2 \lambda_2 e^{-\lambda_2 t}$$

Expectation:
Parameters: α_i, λ_i where $i = 1, 2, \ldots, n$

$$E[X] = \sum_{i=1}^{n} \frac{\alpha_i}{\lambda_i}$$

//notes about squared coeff of variation to be added

4 A first principles analysis of the M/M/1 queue

Consider a system with one server and infinite queue capacity. The arrivals are Poisson with rate λ and the service time is exponentially distributed with rate μ.

Let $N(t)$ = number of jobs in the system at time t. We call $N(t)$ the *state* of the system at time t.

Suppose $N(t) = k$. Then what is $N(t+\Delta t)$? (Where Δt is an infinitesimally small time interval.)

In the small interval Δt, we can have an arrival, or a departure. We know by the definition of Poisson arrivals that an arrival happens in a small interval Δt with probability $\lambda \Delta t$. Also two arrivals cannot happen in this interval.

If $k \geq 1$, then what is the probability of a departure in an interval Δt? Since service time distribution is $EXP(\mu)$, this probability is:

$$\begin{aligned} 1 - e^{-\mu \Delta t} &= 1 - (1 - \mu \Delta t + \frac{(\mu \Delta t)^2}{2!} + \ldots) \\ &= (\mu \Delta t - \frac{(\mu \Delta t)^2}{2!} + \ldots) \\ &\approx \mu \Delta t \end{aligned} \tag{14}$$

We can similarly argue that the probability of two departures in a small interval is negligible.
Pr[Arrival in Δt] is $\lambda \Delta t$.
Pr[Departure in Δt] is $\mu \Delta t$.
Pr[Arrival & Departure in Δt] is $\lambda \mu (\Delta t)^2 \approx 0$.

Then $N(t+\Delta t) = k$ if
A: $N(t) = k$ and no jobs arrive/depart in $(t, t+\Delta t]$ w.p. $1 - \lambda\Delta t - \mu\Delta t$
B: $N(t) = k-1$ and 1 job arrives in $(t, t+\Delta t]$ w.p. $\lambda\Delta t$
C: $N(t) = k+1$ and 1 job departs in $(t, t+\Delta t]$ w.p. $\mu\Delta t$

Unconditional Probability:
Let $P_k(t) = P[N(t) = k]$. Then

$$P_k(t+\Delta t) = P_k(t) + [\lambda P_{k-1}(t) - (\lambda+\mu)P_k(t) + \mu P_{k+1}(t)]\Delta t$$

$$\text{The rate of change of probability} = \lim_{\Delta t \to 0} \frac{P_k(t+\Delta t) - P_k(t)}{\Delta t} = \frac{dP_k(t)}{dt}$$

$$\frac{dP_k(t)}{dt} = \lambda P_{k-1}(t) - (\lambda+\mu)P_k(t) + \mu P_{k+1}(t)$$

We are often interested in the state of the system as $t \to \infty$. Under certain conditions, $\lim_{t\to\infty} P_k(t)$ exists and converges to a unique value π_k. This probability is called the *steady state probability* of the system being in state k.

If steady-state exists, $\lim_{t\to\infty} \frac{dP_k(t)}{dt} \to 0$.

$$\begin{aligned}
(\lambda+\mu)\pi_k &= \lambda\pi_{k-1} + \mu\pi_{k+1} \quad , \qquad k = 1, 2, \cdots \\
\lambda\pi_0 &= \mu\pi_1 \\
(\lambda+\mu)\pi_1 &= \lambda\pi_0 + \mu\pi_2 \\
(\lambda+\mu)\pi_2 &= \lambda\pi_1 + \mu\pi_3 \\
&\vdots \\
\pi_k &= \frac{\lambda}{\mu}\pi_{k-1}
\end{aligned}$$

$$\text{Let} \quad \frac{\lambda}{\mu} = \rho.$$

Then the above equations can be written as

$$\begin{aligned}
\pi_1 &= \frac{\lambda}{\mu}\pi_0 = \rho\pi_0 \\
\pi_2 &= \frac{\lambda}{\mu}\pi_1 = \rho\pi_1 \\
\pi_3 &= \frac{\lambda}{\mu}\pi_2 = \rho\pi_2 \\
&\vdots \\
\pi_k &= \rho\pi_{k-1} \\
&= \rho \cdot \rho\pi_{k-2} \\
&\vdots \\
&= \rho^k\pi_0 \quad , \qquad k = 1, 2, \cdots, \infty
\end{aligned}$$

$$\text{Since} \quad \sum_{k=0}^{\infty} \pi_k = 1$$

$$\sum_{k=0}^{\infty} \rho^k \pi_0 = 1$$

$$\Rightarrow \pi_0 = \frac{1}{\sum_{k=0}^{\infty} \rho^k}$$

$$\pi_0 = 1 - \rho$$

$$\rho = 1 - \pi_0 \ldots \text{Probability that the server is busy}$$

$$\pi_k = \rho^k \cdot (1 - \rho)$$

Thus π_k is the "Modified Geometric" distribution with parameter $(1 - \rho)$

If N is the number of jobs in the system at steady state, then

$$E[N] = \frac{1 - (1 - \rho)}{1 - \rho} \qquad = \frac{\rho}{1 - \rho} = \text{Avg. Queue length}$$

$$By Little's Law:$$

$$E[R] = \frac{E[N]}{\lambda} \qquad = \frac{1}{\mu(1 - \rho)} = \text{Avg. Response time}$$

Thus, using a first principles approach, we can calculate various metrics of the M/M/1 queueing system.

However $\{N(t) | t \geq 0\}$ is actually an example of a special kind of *stochastic process*. We will now study Stochastic Processes, and see how those methodologies based on stochastic processes can make the above analysis very straightforward.

5 Stochastic Processes

A stochastic process is a family of random variables $\{X(t) | t \in T\}$ defined on a given probability space, indexed by the parameter t, where $t \in T$. Values of $X(t)$ define the state space and the parameter t is time in many cases. Based on the combinations of state and parameter types, we have the following classification of stochastic processes:

State / Parameter	Discrete	Continuous
Disrete	Discrete-parameter chain	Discrete-paremeter continuous state process
Continuous	Continuous-parameter chain	Continuous-parameter continuous state process

Examples from queueing systems:
DPDS: N_k: number of jobs in the system seen by the k^{th} arrival
DPCS: waiting time of the k^{th} customer
CPDS: $N(t)$ number of jobs in the system at time t
CPCS: remaining work in the system at time t

5.1 Markov Process

$\{X(t)|t \in T\}$ is called a Markov process if for any $t_0 < t_1 < t_2 < \cdots < t_n < t$
$P[X(t) \leq x|X(t_n) = x_n, \cdots X(t_1) = x_1, X(t_0) = x_0] = P[X(t) \leq x|X(t_n) = x_n]$

The interpretation of the above property is as follows: if we label time t_n as "current", then times $t_0, t_1, ...t_{n-1}$ are the "past" and time t is the "future". Also $X(t)$ typically represents a system "state" that is changing over time. Then the above property says that for a Markov process, the future state of the system depends only on the current state of the system, not on the path taken to come to the current state of the system.

We implicitly used this property when we did the first-principles analysis of $M/M/1$ queue. We predicted the value of $N(t + \Delta t)$ based only the value of $N(t)$ not on $N(t - \Delta t)$ and older states.

Further if $P[X(t) \leq x|X(t_n) = x_n] = P[X(t - t_n + u) \leq x|X(u) = x_n]$,
the Markov proces is said to be *time-homogenous*. This means that if a process is in state x_n now, then probability of being in some state x, some v time units later, depends only on x_n and the time difference v, not on the actual current time.

What exactly does this mean? This means that the system parameters which determine the behaviour of the stochastic process are not changing over time, and hence we do not have to keep track of how much time has "elapsed" since the "beginning". Recall again that our $N(t)$ of the $M/M/1$ queue had this property. We wrote the transition probabilities in terms *only* of the *time difference*, which was Δt. In the "RHS" of the expressions of the state transition probabilities, there is only Δt, and no t.

If, on the other hand, for example, the arrival rate was time dependent ($\lambda(t)$), the corresponding stochastic process would not be time-homogeneous.

A continuous-time, discrete state stochastic process can only have the time-homogeneous Markov property if the distribution of all the events that lead to state changes of the stochastic process are exponential. Only in such a case can a discrete state capture all the information required to predict the future.

Then the time-homogeneous Markov property of a process implies that the distribution of time spent in a state is exponential. Only in this distribution, can the time remaining in a certain state have the same distribution, no matter how much time the system has already spent in that state.

A stochastic process which has discrete states is called a *chain*. Thus a continuous time, discrete state Markov process is commonly termed as *Continuous Time Markov Chain (CTMC)*.

5.2 Analysis of time-homogeneous CTMCs

Consider a time-homogeneous CTMC $\{X(t)|t \geq 0\}$. Then we know that the time spent in a state has exponential distribution. It follows that the duration of all "events" that cause state changes must be exponentially distributed. The basic goal of analysis of a CTMC is to determine:

$$P_k(t) = P[X(t) = k], \quad \text{the probability that the system is in state } k \text{ at time } t.$$

which in vector form is denoted by $\mathbf{P(t)} = [P_0(t), P_1(t), \ldots]$

A time-homogeneous CTMC (henceforth, we call this just "CTMC") is characterized by the following:

- The definition of a state space of the system. For this analysis, let the state space be $\{0, 1, 2, \cdots\}$, without loss of generality.
- The *transition rate* q_{ij} at which the CTMC changes state from i to j. In other words, q_{ij} is the *parameter* of the **exponential distribution** of the time for going from state i to state j.

- An initial state probability vector $\mathbf{P(0)} = [P_0(0), P_1(0), \ldots]$, which gives the probability that the system was in state k at time $t = 0$.

The CTMC is often represented pictorially with a state-transition diagram.

An example is $\{N(t)|t \geq 0\}$ from Section 4, which denoted the number of customers in an M/M/1 queue at time t. This CTMC is characterized by:

- State space $S = \{0, 1, 2, \ldots\}$.
- Transition Rates:
  - $q_{i,i+1} = \lambda, \forall i = 0, 1, 2, \ldots$
  - $q_{i,i-1} = \mu, \forall i = 1, 2, 3, \ldots$
  - $q_{ij} = 0$ otherwise.
- $\mathbf{P(0)} = [1, 0, 0, \ldots]$. That is, we can assume that at the beginning the system is empty, with probability 1.

Now, we can follow a derivation very similar to the one in Section 4, to arrive at a general equation for $\mathbf{P(t)}$ in terms of the parameters of the CTMC, namely $\mathbf{P(0)}$ and q_{ij}'s.

Then, consider an infinitesimal small interval Δt. What is the probability that the system will be in state j at time $t + \Delta t$? That is

$$\text{What is } \; P_j(t + \Delta t) = P[X(t + \Delta t) = j]?$$

The CTMC can be in state j at time $t + \Delta t$ if:

- $X(t) = i, \; i \neq j$ and a transition from i to j happened in a small interval Δt. Recall that since the distribution of the time for this transition is $EXP(q_{ij})$, we can use the same reasoning as we used in Section 4, Equation 14 and state that this will happen with probability $q_{ij}\Delta t$.
- $X(t) = j$ and no state transition happened in the interval Δt. Now note that in state j many simultaneous transitions are "in progress" - the transition from state j to a state k is in progress with distribution $EXP(q_{jk})$. Thus the time to the first transition which will achieve a state change, is a minimum of all transition times, and thus, the distribution is

$$EXP(\sum_{k,k\in S,k\neq j} q_{jj})$$

Thus, the probability that there will be a transition out of state j in time Δt is (see Section 4)

$$(\sum_{k,k\in S,k\neq j} q_{jk})\Delta t$$

Therefore, the probability that there is **no transition is:**

$$1 - \sum_{k,k\in S,k\neq j} q_{jk}\Delta t$$

Thus, the unconditional probability $P_k(t + \Delta t) = P[X(t + \Delta t) = j]$ is given by the law of total probability:

$$\begin{aligned} P_j(t + \Delta t) &= \sum_{i,i\in S,i\neq j} q_{ij}\Delta t \; P_i(t) + (1 - \sum_{k,k\in S,k\neq j} q_{jk}\Delta t) \; P_j(t) \\ \frac{P_j(t + \Delta t) - P_j(t)}{\Delta t} &= \sum_{i,i\in S,i\neq j} q_{ij}P_i(t) - \sum_{k,k\in S,k\neq j} q_{jk}tP_j(t) \end{aligned}$$

Define

$$q_j = \sum_{k,k \in S, k \neq j} q_{jk}$$

.

Then, taking limit of Δt tending to zero on both sides, we have:

$$\frac{dP_j(t)}{dt} = \sum_{i,i \in S, i \neq j} q_{ij} P_i(t) - q_j P_j(t) \tag{15}$$

Now, if we define the matrix $\mathbf{Q}$ as follows:

$$\begin{aligned} Q_{ij} &= q_{ij},\ i \neq j \\ Q_{jj} &= -q_j \end{aligned} \tag{16}$$

Then the above equation can be written in matrix form as:

$$\frac{\mathbf{dP(t)}}{\mathbf{dt}} = \mathbf{P(t)Q}$$

The solution to the above equation (stating without proof) is

$$\mathbf{P(t)} = \mathbf{P(0)e^{Qt}}$$

where

$$e^{Qt} = I + \sum_{n=1}^{\infty} Q_n \frac{t^n}{n!}$$

Again, we are interested in the system state probabilities as $t \to \infty$. For this, we state the following definition and theorem:

A CTMC is said to be irreducible if every state is reachable from every other state.

Theorem: For an irreducible CTMC the following limits always exist and are independent of the initial state i.

$$\pi_j = \lim_{t \to \infty} P_j(t), \forall j \in S$$

If the underlying system is *stable*, then as $t \to \infty$, the CTMC is said to achieve *statistical equilibrium* and the limiting probabilities are called the *steady state probabilities* of the CTMC.

At steady state, $P_j(t)$ becomes constant, hence its derivative becomes zero. Taking limits as $t \to \infty$ on Equation 15, we get

$$\begin{aligned} 0 &= \sum_{i \neq j} \pi_i q_{ij} - \pi_j q_j \\ \Rightarrow \pi_j q_j &= \sum_{i \neq j} \pi_i q_{ij} \\ \Rightarrow \text{rate out of state } j &= \text{rate in to state } j \end{aligned}$$

These are called *steady state balance equations* of a CTMC. We can solve for the steady state probability vector $\pi = [\pi_0, \pi_1, \ldots]$ by using the above balance equations, and the normalizing condition:

$$\pi \mathbf{Q} = \mathbf{0} \quad \text{and} \sum_{j \in S} \pi_j = 1$$

6 Markovian queueing systems analysis using CTMCs

Now we study a series of Markovian queues using CTMCs.

6.1 M/M/1 queue

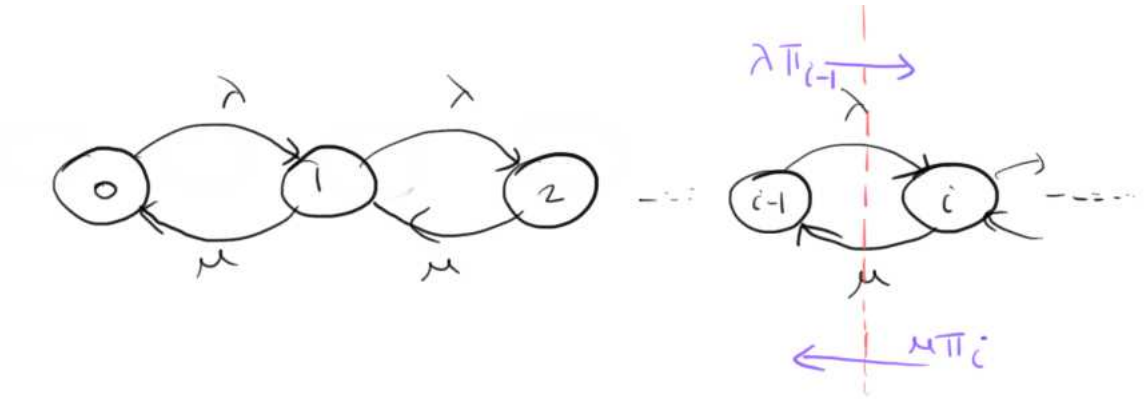


Figure 18: M/M/1 state transition diagram

Here, the state is the number of customers in the queue. Thus $S = \{0, 1, 2, \ldots\}$.

The transition rates were defined in the above section.The CTMC corresponding to the M/M/1 queue can be drawn as shown in Figure 18.

Note that the balance equations can also be written by drawing the CTMC as a graph, and then making "cuts" in the graph that divide the graph into two parts. Then equate the average "flow" in one direction of the cut to the average "flow" in the other direction of the cut.

E.g. for the M/M/1 queue, if the cut as shown in Figure 18, then the average steady state flow from left to right is $\lambda\pi_{i-1}$ and from right to left is $\mu\pi_i$. Thus we have:

$$\lambda\pi_{i-1} = \mu\pi_i$$

which is the same as we got using the "first principles analysis". The rest of the analysis follows similarly.

Note that the sum of the infinite series in these derivations converges only under certain conditions. The CTMC achieves a steady state only under these conditions. The underlying system is then considered *stable*. We term these conditions the *stability criteria* for the system. For M/M/1 queue, the stability criterion is

$$\lambda < \mu$$

6.2 M/M/c/∞ queue

Here we have multiple servers and a single queue. m is the number of servers, each with $EXP(\mu)$ service time. The total capacity of the system is $c\mu$ requests per unit time.

Interarrival $\sim$ EXP(λ)

The state of the system here is also the number of customers in the queueing system. Thus $S = \{0, 1, 2, \ldots\}$.

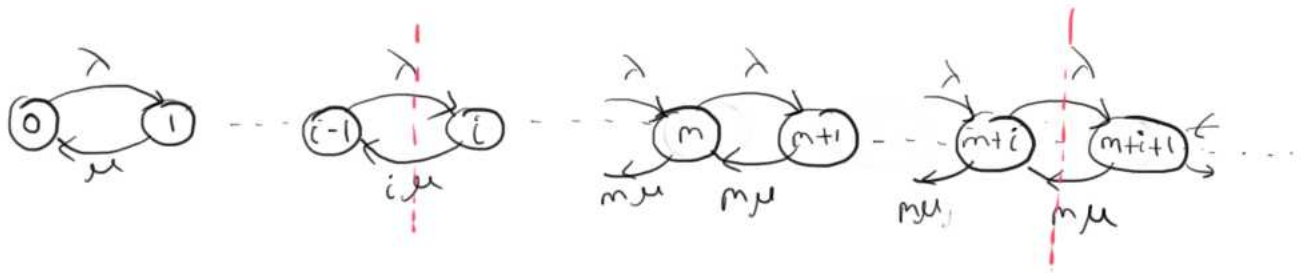


Figure 19: M/M/c state transition diagram (substitute m by c in the diagram above

Now consider a state $i < c$. What are the "events in progress" in this state?
There is one $EXP(\lambda)$ arrival that can happen.If the arrival happens first, the state will be $i + 1$. This happens at the "rate" λ.
There are i "services" or "departures" in progress. The distribution of time to the first departure is $EXP(i\mu)$, since it is the minumum of the i service times each of which are $EXP(\mu)$. A departure will change the state to $i - 1$.

Now consider a state $i \geq c$. Here, there are more than c customers in the system, but at max c servers can be busy serving customers. Thus, distribution of time to the next departure is $EXP(m\mu)$.

Figure 19 shows these state transitions. We can again write the balance equations by doing the "cuts" as shown.

$$\begin{aligned} \lambda\pi_{j-1} &= j\mu\pi_j \ \forall 1 \leq j \leq c \\ \lambda\pi_{c+j-1} &= c\mu\pi_{c+j} \ \forall j \geq 0 \end{aligned}$$

Let $\rho = \frac{\lambda}{c\mu}$.
For $1 \leq j \leq c$, this implies

$$\begin{aligned} \pi_j &= \tfrac{\lambda}{j\mu}\pi_{j-1} \\ \pi_1 &= \tfrac{\lambda}{\mu}\pi_0 = c\rho\pi_0 \\ \pi_2 &= \tfrac{\lambda}{2\mu}\pi_1 = \tfrac{c\rho}{2}\pi_1 = \tfrac{(c\rho)^2}{2!}\pi_0 \\ \pi_3 &= \tfrac{\lambda}{3\mu}\pi_2 = \tfrac{c\rho}{3}\pi_2 = \tfrac{(c\rho)^3}{3!}\pi_0 \\ &\vdots \\ \pi_j &= \tfrac{c\rho}{j}\pi_{j-1} \\ &= \tfrac{c\rho}{j} \cdot \tfrac{c\rho}{j-1}\pi_{j-2} \\ &\vdots \\ \pi_j &= \tfrac{(c\rho)^j}{j!}\pi_0 \end{aligned} \tag{17}$$

For $c + j \geq c$, this implies

$$\begin{aligned}
\pi_{c+j} &= \tfrac{\lambda}{c\mu}\pi_{c+j-1}, \quad j \geq 0 \\
\pi_{c+1} &= \tfrac{\lambda}{c\mu}\pi_c = \rho\pi_c \\
\pi_{c+2} &= \tfrac{\lambda}{c\mu}\pi_{c+1} = \rho^2\pi_c \\
\pi_{c+3} &= \tfrac{\lambda}{c\mu}\pi_{c+2} = \rho\pi_{c+2} = \rho^3\pi_c \\
&\vdots \\
\pi_{c+j} &= \rho\pi_{c+j-1} \\
&= \rho \cdot \rho\pi_{c+j-2} \\
&\vdots \\
\pi_{c+j} &= \rho^j\pi_c = \tfrac{c^c\rho^{c+j}}{c!}\pi_0
\end{aligned} \tag{18}$$

Now, we can determine π_0 using:

$$\sum_{j=0}^{\infty} \pi_j = 1$$

We have,

$$\begin{aligned}
\sum_{j=0}^{c-1} \pi_j + \sum_{j=0}^{\infty} \pi_{c+j} &= 1 \\
\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!}\pi_0 + \sum_{j=0}^{\infty} \rho^j\pi_c &= 1 \\
\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!}\pi_0 + \frac{(c\rho)^c}{c!}\pi_0 \sum_{j=0}^{\infty} \rho^j &= 1 \\
\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!}\pi_0 + \frac{(c\rho)^c}{c!}\frac{\pi_0}{1-\rho} &= 1 \quad \text{if } \rho < 1 \\
\pi_0 &= \left[\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!} + \frac{(c\rho)^c}{c!}\frac{1}{1-\rho}\right]^{-1} \quad \text{if } \rho < 1
\end{aligned} \tag{19}$$

We note here that we have discussed the notion of *stability* earlier but we can see its mathematical definition here. The queuing system is stable if the p_j's above are well-defined. In the above, the steady state probabilities exist and are well-defined if $\rho < 1$. Thus, the *Stability criterion* for this queue is $\lambda < c\mu$.

Let us now derive some performance metrics. The throughput and utilization for this system are given as usual, we do not need Markov chain analysis for that.(Throughput of the queueing system is λ when $\lambda < m\mu$ and $m\mu$ when $\lambda \geq m\mu$. Utilization is ρ).

But we now have the distribution of N: the number of customers in the system at steady state. Recall that what we have just calculated is

$$\lim_{t\to\infty} P[N(t) = j] = \pi_j$$

.

Then the Expectation of N is;'

$$E[N] = \sum_{j=0}^{\infty} j\pi_j$$

The expected response time at steady state is then

$$E[R] = \frac{E[N]}{\lambda}$$

Let M denote the number of busy servers at steady state. Then

$$E[M] = \sum_{j=0}^{c-1} j\pi_j + \sum_{j=c}^{\infty} c\pi_j$$

Let N_Q denote the number of customers in the buffer (waiting). Then

$$E[N_Q] = \sum_{j=c+1}^{\infty} (j-c)\pi_j$$

The expected waiting time, W, at steady state is then

$$E[W] = \frac{E[W]}{\lambda}$$

We can also find the probability of "queueing", that is the probability that an arriving request has to wait in the buffer (i.e. has non-zero waiting time).

For this we first need to understand a result called *PASTA*: "Poisson Arrivals See Time Averages". This result states that when arrivals are Poisson, the probability of an arriving customer seeing k customers in the system is just the unconditional state probability π_k. (Recall the example given in class of a D/D/1 queue. Complete this example to show how this is not true in general.)

The request has to queue if it sees c or more servers busy. Thus, using PASTA,

$$\text{P[queuing]} = \sum_{j=c}^{\infty} \pi_j$$

6.3 M/M/c/K queue

The state transition diagram for this queuing system is also almost the same as $M/M/c$- except that the "last" state is $c + K$ after which an arrival is dropped, hence the state does not change.

The analysis for the this queue is almost the same as for $M/M/c$, however we get

$$\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!}\pi_0 + \frac{(c\rho)^c}{c!}\pi_0 \sum_{j=0}^{K} \rho^j = 1 \tag{20}$$

$$\pi_0 = \left[\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!} + \frac{(c\rho)^c}{c!}\sum_{j=0}^{K} \rho^j\right]^{-1} \tag{21}$$

$$\pi_0 = \left[\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!} + \frac{(c\rho)^c}{c!}(K+1)\right]^{-1} \quad \text{if } \rho = 1 \tag{22}$$

$$\pi_0 = \left[\sum_{j=0}^{c-1} \frac{(c\rho)^j}{j!} + \frac{(c\rho)^c}{c!}\frac{1-\rho^{K+1}}{1-\rho}\right]^{-1} \quad \text{if} \rho \neq 1 \tag{23}$$

This expression is well-defined for all values of ρ, thus there is no stability criterion for this queue.

Let us now calculate the performance metrics of the system. Note that ρ here is *not* the utilization.

The probability of arriving request being *dropped* is given by the probability that an arriving request sees the buffer FULL:

$$p_{loss} = \pi_{c+K} \quad \text{by PASTA}$$

.

Thus throughput is given by:

$$\Lambda = \lambda(1 - \pi_{c+K})$$

.

Then, average server utilization is given by:

$$\frac{\Lambda}{c\mu}$$

Then the Expectation of N is;'

$$E[N] = \sum_{j=0}^{c+K} j\pi_j$$

The expected response time at steady state is then

$$E[R] = \frac{E[N]}{\Lambda}$$

The metrics $E[M], E[N_Q], E[W]$ can be found similarly as for $M/M/c$ queue (with sums going to $c + K$ instead of ∞).

6.4 Cyclic Queuing Model of a Multiprogramming system

So far we have learnt that closed queuing networks are used to model systems in which a fixed (small) set of users are in a closed request-response loop with a server system. However, there is one more scenario where closed queuing networks can be used as an abstraction which faithfully captures the behaviour of the system.

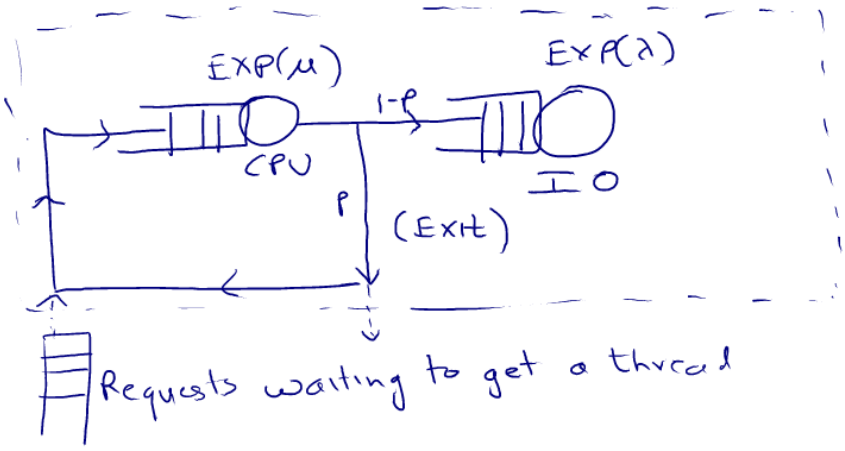
Consider a server, e.g. a multi-threaded Web server that is configured with a fixed number of threads, running on a computer. A thread in this server will typically circulate between using the CPU, the disk or the network several times before the request is complete.Thus, the thread is a "customer" in a queuing network comprising of the CPU, Disk and the Network. Now suppose that we want to analyze this system during its peak hour, when we know that all the threads of the Web server are always busy, and arriving requests have to queue for getting an idle thread. In the peak hour of usage, the following is expected to happen: a thread starts on a request on the CPU, visits CPU/disk/network multiple times, visits CPU one last time, and then the request is finished. As soon as this request is finished, since the load is high, there will be another request in the queue, and the thread will immediately pick up this request from the queue.

Now if we assume that these requests are *statistically similar*, then picking up the next request from the queue is mathematically equivalent to *the previous request re-joining the CPU queue*. Thus it is as if there is a fixed number of customers (threads) circulating "forever" between these three queuing stations: CPU, Disk, Network.

This is the abstraction that leads us to model this system as a closed queuing network. Note that in this network there is no "think station". The threads are "always busy" circulating between the hardware resources. The hardware resources, on the other hand may or may not be busy, depending on the number of threads in the system, and the relative service demands of the various resources.

The figure shows the queuing network model corresponding to a simple CPU-IO system. We assume that the thread starts on a request at the CPU. After leaving the CPU, it may be done, with probability p - which means it re-joins the CPU with probability p, or goes to the I/O queue with probability $q = 1 - p$. After doing some I/O the thread always returns to the CPU queue.

This queuing network can be solved by the MVA algorithm that we have learnt earlier. But it can also be modeled directly using CTMCs. In this section, we will analyze it using CTMCs.



Let n be the maximum number of threads allowed (also called "multiprogramming level.")

Let CPU service time be EXP(μ) and I/O service time be EXP(λ).

We model this as a CTMC as follows:

State: i = # of jobs at the CPU.

Then $i \in \{0, 1, 2, \ldots n\}$

The state-transition diagram is given by:

Following the usual methods, we can find the steady state probability of i jobs at the CPU as:

$$\pi_i = \left(\frac{\lambda}{\mu_q}\right)^i \pi_0, \qquad i = 0, 1, 2, \cdots n$$

Normalizing, we get

$$\sum_{i=0}^{n} \pi_i = \sum_{i=0}^{n} \left(\frac{\lambda}{\mu_q}\right)^i \pi_0 = 1$$

Let $\rho = \frac{\lambda}{\mu_q}$ ($\neq$ utilization!!)

$$\pi_0 = \frac{1-\rho}{1-\rho^{n+1}} \quad \text{if } \rho \neq 1$$

$$= \frac{1}{n+1} \quad \text{if } \rho = 1$$

$$\Rightarrow \pi_i = \rho^i \cdot \left(\frac{1-\rho}{1-\rho^{n+1}}\right), \quad \rho \neq 1$$

$$= \frac{1}{n+1}, \quad \rho = 1$$

Thus CPU utilization is given by

$$U_0 = 1 - \pi_0 = \frac{\rho - \rho^{n+1}}{1-\rho^{n+1}}, \quad \rho \neq 1$$

$$= \frac{n}{n+1}, \quad \rho = 1$$

I/O utilization is given by

$$U_{IO} = 1 - \Pi_n = 1 - \frac{\rho^n(1-\rho)}{1-\rho^{n+1}}, \quad \rho \neq 1$$

$$= \frac{n}{n+1}, \quad \rho = 1$$

Throughput of this system is given by the rate of request flow along the arc marked "exit":

$$\Lambda_{sys} = \mu p \times [\text{Prob CPU busy}] + 0 \times [\text{Prob CPU not busy}]$$

$$= U_{CPU} \cdot \mu \cdot p$$

The quantity "ρ" defined here has a particular interpretation. First, let us calculate the average number of visits to the CPU by a request before it exits.

# of visits to CPU is Geom(p), thus, the average is given by $\frac{1}{p}$.

Similarly, # of visits to I/O is modGeom(p), Thus, the average is $\frac{1-p}{p} = \frac{q}{p}$.

Thus,

$$\rho = \frac{\lambda}{\mu q} = \frac{\frac{1}{\mu}}{q/\lambda} = \frac{\frac{1}{\mu\rho}}{\frac{q}{\lambda\rho}}$$

If $E[B_{CPU}]$ denotes average service demand at CPU & $E[B_{IO}]$ denotes average service demand at IO, Then we can see that

$$E[B_{CPU}] = \frac{1}{\mu} \times \frac{1}{\rho} \quad \&$$

$$E[B_{IO}] = \frac{1}{\lambda} \times \frac{1-\rho}{\rho} = \frac{q}{\lambda\rho}$$

Thus $\rho = \dfrac{E[B_{CPU}]}{E[B_{IO}]}$

Now one may ask, since the purpose of multiprogramming is to increase the utilization, and therefore the throughput, are there conditions under which that purpose is acheived?

You can show that when $\rho \to 1$, that is service demands are almost equal, both resource utilizations go to 1, as $n \to \infty$. Similarly, you can show that if $\rho < 1$, CPU utilization flattens out at a value < 1, while IO goes to 1, as $n \to \infty$, and vice versa if $\rho > 1$. Do this exercise yourself. On either case, the throughput should flatten out to $\dfrac{1}{\text{Bottleneck Service Demand}}$

7 Open Queuing networks

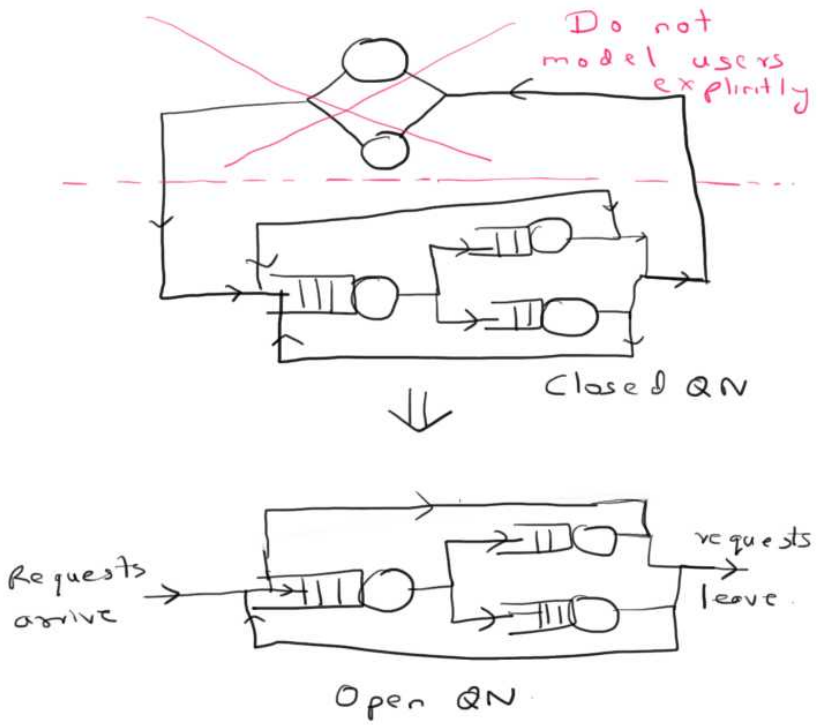


Figure 20: Open Queueing Network

So far we have studied closed queuing networks, where clients who issue requests are modeled explicitly. However, sometimes the number of simultaneous users of the system are very large (e.g. 50,000), and it is not feasible to model the clients explicitly. Furthermore, if their think time is also large, then it can be shown that the arrival process as seen at the server entry point starts looking like a Poisson process. Thus, it makes sense to "ignore the clients" and focus on the request arrivals. Figure 20 shows an example open queuing network.

Open queueing network are queueing networks where "external arrivals" are possible to every node. A request entering the queuing network may visit the queuing stations several times before exiting. Note that the total number of requests circulating in an open queueing network is not fixed.

For an open QN, as usual, we want to ask the usual questions:

- What is the maximum throughput that this network is capable of ?
- Which is the bottleneck server?
- For a certain arrival rate, what is the system throughput? What are the server utilizations?

- For a certain arrival rate, what is the system response time of a request?

A little bit of thought will make it clear to you that the first two questions are independent of whether the queuing network is being modeled as closed or open. The number of visits a request makes to a station can be determined in the same way as was done for closed queueing networks. This also implies that the *service demand* at each node is calculated in the same way. This in turn determines the *bottleneck system throughput* as the reciprocal of the maximum service demand.

What remains then is the last two questions. Let us first consider throughput, when the total arrival rate to the system is λ. It is clear that the asymptote of this as arrival rate increases is the bottleneck system throughput, which was defined above. Now, if we assume infinite buffers at all nodes, it follows that as long as the arrival rate is lesser than this value, the throughput will be equal to arrival rate. Thus

$$\begin{aligned} \Lambda_{sys} &= \lambda \text{ if } \lambda < \Lambda_{max} && (24) \\ &= \Lambda_{max} \text{ if } \lambda \geq \Lambda_{max} && (25) \\ & && (26) \end{aligned}$$

where if v_i is the average visit count at node i and $E[B_i]$ is the expected service demand at node i, then

$$\Lambda_{max} = \frac{1}{\max_i E[B_i]}$$

To find the average response time (or the number of customers at the queue), one could ask the question - "can these queues be evaluated in isolation"? Then we can simply apply the formulas for the single server queuing systems that we know!

We see now that this can in fact, be done. First, we consider a simple example.

7.1 Two-Server Tandem Queuing network

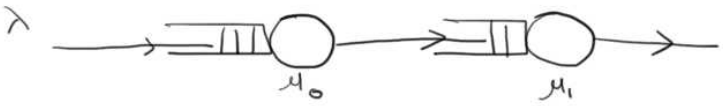


Figure 21: Open Tandem Queueing Network

Consider a two-stage tandem network as shown in the Figure 21. The sysetm consists of two nodes with respective service rates μ_0 and μ_1. The external arrival is Poisson with rate λ. The service-time distribution at both nodes is exponential.

Since all interevent tmes are exponentially distributed, the system can be modeled as a Continuous Time Markov chain, whose states are specified by pairs $(k_0, k_1), k_0 \geq 0, k_1 \geq 0$, where k_0 and k_0 are number of jobs at server 0 and server 1 respectively.

The state-transition diagram corresponding to a generic state of this CTMC is shown in Figure x.

Let $P(k_0, k_1)$ be the joint probability of k_0 jobs at the first server and k_1 jobs at the next server, in the steady state. Equating the rates of flow into and out of the state, we obtain the following balance equations:

$$(\mu_0 + \mu_1 + \lambda)P(k_0, k_1) = \mu_0 P(k_0 + 1, k_1 - 1) + \mu_1 P(k_0, k_1 + 1) + \lambda P(k_0 - 1, k_1), \qquad k_0 > 0, k_1 > 0.$$

For the boundary states, we have:

$$\begin{aligned}(\mu_0 + \lambda)P(k_0, 0) &= \mu_1 P(k_0, 1) + \lambda P(k_0 - 1, 0), \qquad k_0 > 0 \\ (\mu_0 + \lambda)P(0, k_1) &= \mu_0 P(1, k_1 - 1) + \mu_1 P(0, k_1 + 1), \qquad k_1 > 0 \\ \lambda P(0, 0) &= \mu_1 P(0, 1)\end{aligned}$$

The normalization is provided by:

$$\sum_{k_0 \geq 0} \sum_{k_1 \geq 0} P(k_0, k_1) = 1$$

It is easily shown by direct substitution that the following equation is the soluton to the above balance equations:

$$P(k_0, k_1) = (1 - \rho_0)\rho_0^{k_0}(1 - \rho_1)\rho_1^{k_1} \qquad = P(k_0)P_(k_1)$$

where, $\rho_0 = \lambda/\mu_0$ and $\rho_1 = \lambda/\mu_1$, and $P(k_0)$ and $P(k_1)$ are the steady state probabilities of having k_0 an k_1 customers at queue 1 and 2 respectively, if each queue was an independent $M/M/1$ queue. Thus, the tandem queues in the figure behave like 2 independent $M/M/1$ queues. The joint distribution is the product of individual distributions.

Expected number of jobs in the system is:

$$E[N] = \frac{\rho_0}{1 - \rho_0} + \frac{\rho_1}{1 - \rho_1}$$

Average Response time is:

$$E[R] = \frac{E[N]}{\lambda}$$

7.2 Product Form (Jackson) Open Queuing Networks

It turns out that even if queuing networks have splitting, joining and feedback, the steady state joint distribution of jobs at each queueing station is a product of the marginal distribution of jobs at each queue. In fact, Jackson's theorem can be stated as follows:

Consider an open queuing network with $(m + 1)$ stations, where the i^{th} station consists of c_i exponential servers, $i = 0, 1, \ldots, m$ each with mean service time of $1/\mu_i$ seconds. External Poisson sources contribute γ_i jobs/second to the average rate of arrival to the i^{th} station.

Then the total external arrival rate to the queueing network is given as:

$$\lambda = \sum_{i=0}^{m} \gamma_i$$

.

The routing matrix is given by :

$$X = [x_{ij}]$$

and is such that the matrix power series

$$\sum_{k=0}^{\infty} X^k$$

converges. (This property implies that after some cycles, a job must leave the open queuing network.)

Then, according to Jackson's result, each queue behaves like an independent $M/M/c_i$ queueing system with service rate of μ_i and effective arrival rate λ_i as given by:

$$\lambda_i = \gamma_i + \sum_{j=0}^{m} x_{ji}\lambda_j, i = 0, 1, \ldots, m$$

Also, for each queue i, we need $\lambda_i < c_i\mu_i$

This means that if $\pi_i(k_i)$, is the probability of having k_i jobs at station i, then

$$\pi(k_0, k_1, \ldots, k_m) = \pi_0(k_0)\pi_1(k_1)\ldots\pi_m(k_m)$$

E.g. if $c_i = 1$ for all the queues, then:

$$\pi(k_0, k_1, \ldots, k_m) = \Pi_{i=0}^{m}\rho_i^{k_i}(1 - \rho_i)$$

where $\rho_i = \frac{\lambda_i}{\mu_i}$

7.3 Example

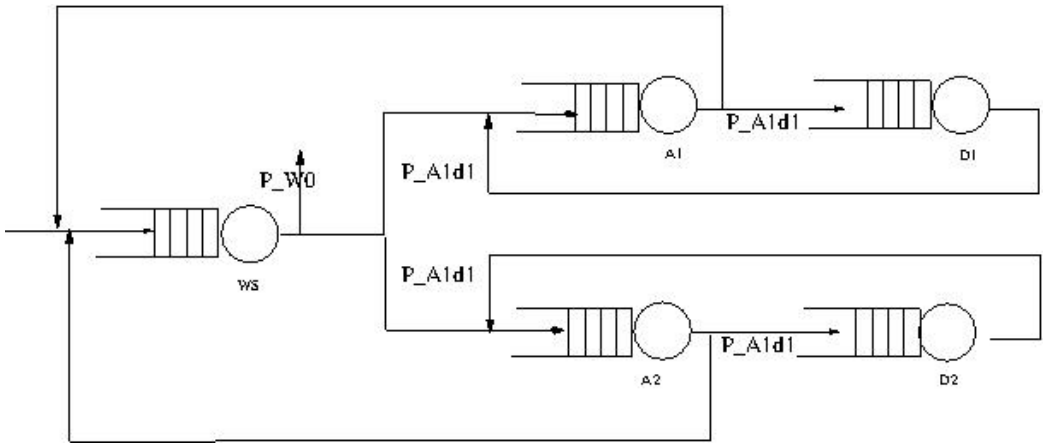


Figure 22: Open Queuing Network Example

Consider the system as shown in figure 22. Let the service requirement per visit of the individual servers be $\tau_{WS}, \tau_{A1}, \tau_{A2}, \tau_{D1}, \tau_{D2}$. Let arrivals be Poisson with rate λ and service times be exponential.

We find the average number of visits per server, i.e. $V_W, V_{A1}, V_{A2}, V_{D1}, V_{D2}$ as:

$$\begin{aligned} V_{A1} &= P_{WA1}V_W + P_{A1D1}V_{A1} \\ V_{A2} &= P_{WA2}V_W + P_{A2D2}V_{A2} \\ V_{D1} &= P_{A1D1}V_{A1} \\ V_{D2} &= P_{A2D2}V_{A2} \\ V_W &= \frac{1}{P_{W0}} \end{aligned} \tag{27}$$

By back-substitution, we get:

$$\begin{aligned} V_{A1} &= P_{WA1}V_W + P_{A1D1}V_{A1} = P_{WA1}V_W/(1-P_{A1D1}) = P_{WA1}/(P_{W0}(1-P_{A1D1})) \\ V_{A2} &= P_{WA2}V_W + P_{A2D2}V_{A2} = P_{WA2}V_W/(1-P_{A2D2}) = P_{WA2}/(P_{W0}(1-P_{A2D2})) \\ V_{D1} &= P_{A1D1}V_{A1} = \frac{P_{A1D1}P_{WA1}}{P_{Q0}(1-P_{A1D1})} \\ V_{D2} &= P_{A2D2}V_{A2} = \frac{P_{A2D2}P_{WA2}}{P_{Q0}(1-P_{A2D2})} \\ V_W &= \frac{1}{P_{W0}} \end{aligned} \tag{28}$$

The effective arrival rates at individual servers in terms of (request/sec)*(visits/request) would be:

$$\begin{aligned} \lambda_W &= V_W\lambda \\ \lambda_{A1} &= V_{A1}\lambda \\ \lambda_{A2} &= V_{A2}\lambda \\ \lambda_{D1} &= V_{D1}\lambda \\ \lambda_{D2} &= V_{D2}\lambda \end{aligned} \tag{29}$$

The average service demand at each of the servers would be:

$$\begin{aligned} E[B_W] &= V_W\tau_W \\ E[B_{A1}] &= V_{A1}\tau_{A1} \\ E[B_{A2}] &= V_{A2}\tau_{A2} \\ E[B_{D1}] &= V_{D1}\tau_{D1} \\ E[B_{D2}] &= V_{D2}\tau_{D2} \end{aligned} \tag{30}$$

The maximum among the above will be the *bottleneck service demand*, and the corresponding server will be the *bottleneck server*. The maximum throughput supportable will be the reciprocal of the bottleneck service demand.

Response time on individual servers would be:

$$\begin{aligned} E[R_W] &= \left(\frac{\tau_W}{1-\lambda_W\tau_W}\right) \\ E[R_{A1}] &= \left(\frac{\tau_{A1}}{1-\lambda_{A1}\tau_{A1}}\right) \\ E[R_{A2}] &= \left(\frac{\tau_{A2}}{1-\lambda_{A2}\tau_{A2}}\right) \\ E[R_{D1}] &= \left(\frac{\tau_{D1}}{1-\lambda_{D1}\tau_{D1}}\right) \\ E[R_{D2}] &= \left(\frac{\tau_{D2}}{1-\lambda_{D2}\tau_{D2}}\right) \end{aligned} \tag{31}$$

Total Response time for a request entering the system is the summation of total time taken at each of these servers. i.e.

$$\begin{aligned} E[R] &= \sum_i \text{No. of visits to server 'i' * Response time at server 'i' per visit} \\ &= \sum_i V_i * E[R_i] \\ &= \sum_i \frac{v_i\tau_i}{1-\lambda_i\tau_i} \\ &= \sum_i \frac{v_i\tau_i}{1-\lambda v_i\tau_i} \\ &= \sum_i \frac{B_i}{1-\lambda B_i} \end{aligned} \tag{32}$$

where $B_i = v_i\tau_i$ is the average service demand at server i.

We can see from the way the above expression is written that the average response time in this network can also be expressed as the sum of response times at an *equivalent tandem queuing network* where we have one server corresponding to each of the servers in the original queuing network, whose average service time is equal to the average service demand of the original queuing network (Figure 23).

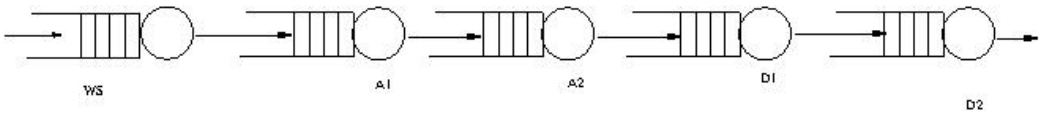


Figure 23: Equivalent Tandem Queuing Network to the Open Queuing Network in Figure 22. Arrival rate is λ and average service time = average total service demand at each station in the original network.

8 Discrete Time Markov Chains

A Markov process with discrete state space and discrete parameter space is known as a Discrete-Time Markov Chain (DTMC).

Let $\{X_n \mid n = 0, 1, 2, \ldots\}$ represent the state of the system at time step n. X_n takes discrete values. If the following property is satisfied:

$$P(X_n = i_n \mid X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1}) = P(X_n = i_n \mid X_{n-1} = i_{n-1})$$

then $\{X_n \mid n = 0, 1, 2, \ldots\}$ is a Discrete Time Markov Chain. As seen earlier in CTMCs the above property means that the future (X_n) depends only on the present state of the system ($X_{n-1} = i_{n-1}$,) and is independent of its past ($X_0 = i_0, X_1 = i_1, \ldots, X_{n-2} = i_{n-2}$.

Let $p_{jk}(m, n) = P(X_n = k \mid X_m = j)$, $0 \leq m \leq n$ denote the probabililty that the DTMC makes a transition from state j at step m to state k at step n. Then a DTMC is homegeneous if $p_{jk}(m, n) = P[X_n = k \mid X_m = j] = P[X_{n-m} = k \mid X_0 = j]$.

In other words, for homegeneous DTMCs, the number of time steps elapsed "from the beginning" also does not matter. The only thing that matters is the state the system is in at the current step. Thus, for homogeneous Markov chains:

$$p_{jk}(n) = P(X_{m+n} = k \mid X_m = j)$$

That is, the probability in the RHS is a function only of the time step difference n.

A DTMC is completely described by:

- A state space S. Without loss of generality we assume this to be $S = \{0, 1, 2, \ldots\}$
- Initial state probabilities i.e. $P\{X_0 = i\} \quad \forall \ i$.
- *One-step transition probabilities*: $p_{jk} = p_{jk}(1) = P(X_n = k \mid X_{n-1} = j), \qquad n \geq 1$

For a DTMC, the matrix $\mathbf{P} = [p_{ij}]$ must be a *stochastic matrix*, i.e. $\sum_j p_{ij} = 1, \ \forall \ i$

Given this information, we are generally interested in the probability that the DTMC is in state j at time step n, denoted by $p_j(n) = P(X_n = j)$.

We can find this using a special case of the Chapman-Kolmogorov equation:

$$p_j(n) = \sum_i p_i(n-1) p_{ij} \tag{33}$$

Let $\mathbf{p}(n) = [p_0(n), p_1(n), \ldots, p_j(n), \ldots]$. Then this can be written as

$$\begin{aligned} \mathbf{p}(n) &= \mathbf{p}(n-1)\mathbf{P}. \\ &= \mathbf{p}(n-2)\mathbf{P}.\mathbf{P} \\ &\vdots \\ &= \mathbf{p}(0)\mathbf{P}^n \end{aligned}$$

Limiting State Probabilities In the condition that the Markov chain is irreducible, aperiodic, recurrent and non-null, the following limit exists and is independent of initial probability:

$$v_j = \lim_{n \to \infty} p_j(n), \quad j = 0, 1, \ldots$$

Here

- *Irreducible* means that every state is reachable from every other state.

- *Aperiodic* means that there is no periodic state in the DTMC. A state is periodic if the number of steps in which one can return to the state after leaving it is a multiple of an integer greater than 1. For an irreducible DTMC, either all states are periodic, or all are aperiodic.
- *Recurrent non-null* roughly captures the property of "stability". We know from our study of CTMCs that limits of probabilities do not exist if the system is not stable. We do not define this phrase formally here, but note the requirement that if the DTMC is infinite-state, stability property will have to be met for limits to exist.

Now we have, from Equation 33:

$$\begin{aligned} \lim_{n\to\infty} p_j(n) &= \lim_{n\to\infty} \sum_i p_i(n-1)p_{ij} \\ \Rightarrow v_j &= \sum_i v_i p_{ij} \\ \mathbf{v} &= \mathbf{v}P \\ \text{where } \sum_i v_i &= 1 \qquad \text{and} \qquad v_j \geq 0 \end{aligned}$$

Thus when limits exist they can be found by solving the above set of linear equations.
Now we will look at a few examples of modeling using DTMCs.

8.1 Page Hit rate analysis

A program's address space typically consists of continuous pages represented by the indices $1, 2, \ldots, n$. For the purpose of studying a program's reference behaviour, it can be represented by the reference string $w = x_1, x_2, \ldots, x_t, \ldots$. Successive references are assumed to form a sequence of independent, identically distributed random variables with the pmf:

$$P(X_t = i) = \beta_i, \qquad 1 \leq i \leq n; \qquad \sum_{i=1}^{n} \beta_i = 1.$$

The number of page frames is in most cases smaller than the number of pages referenced. Let n denote the number of pages and m denote the number of page frames available. Clearly, $1 \leq m \leq n$. The internal state of the paging algorithm at time t denoted by $q(t)$, is an ordered list of the m pages currently in main memory. We assume that on a page fault, the rightmost page in the ordered list $q(t)$ will be replaced. On the page hit, there is no replacement but the list $q(t)$ is updated to $q(t+1)$, reflecting the new replacement priorities. It is clear that the sequence of states $q(0), \ldots, q(n), \ldots, q(t), \ldots$ forms a discrete-time homogeneous Markov chain with the state space consisting of $n!/(n-m)!$ permutations over $1, 2, \ldots, n$.

LRU paging algorithm

As an example, consider the LRU paging algorithm with $n = 3$ and $m = 2$. $q(t) = (i, j)$ implies that the page indexed i was more reently used than page indexed j, and, therefore, page j will be the candidate for replacement. The state space I is given by:

$$I = (1,2), (2,1), (1,3), (3,1), (2,3), (3,2)$$

Let the current state $q(t) = (i, j)$. Then the next state $q(t+1)$ takes one of the following values:

$$q(t+1) = \begin{cases} (i,j), & \text{if } x_{t+1} = i, \text{with associated probability } \beta_i, \\ (j,i), & \text{if } x_{t+1} = j, \text{with associated probability } \beta_j, \\ (k,i), & \text{if } x_{t+1} = k, k \neq i, k \neq j, \text{with associated probability } \beta_k. \end{cases}$$

Then the transition probaility matrix P is given by:

$$P = \begin{bmatrix} \beta_1 & \beta_2 & 0 & \beta_3 & 0 & 0 \\ \beta_1 & \beta_2 & 0 & 0 & 0 & \beta_3 \\ 0 & \beta_2 & \beta_1 & \beta_3 & 0 & 0 \\ 0 & 0 & \beta_1 & \beta_3 & \beta_2 & 0 \\ \beta_1 & 0 & 0 & 0 & \beta_2 & \beta_3 \\ 0 & 0 & \beta_1 & 0 & \beta_2 & \beta_3 \end{bmatrix}$$

It can be verified that the avove Markov chain is irreducible and aperiodic; hence a unique steady-state probability vector v exists. This vector is obtained by solving the system of equations:

$$\begin{aligned} v &= vP, \qquad \text{and} \\ \sum_{(i,j)} v_{(i,j)} &= 1 \end{aligned} \tag{34}$$

Solving this system of equations, we get:

$$v_{(i,j)} = \frac{\beta_i \beta_j}{1 - \beta_i}$$

A page fault occurs in state (i, j), provided that a page other than i or j is referenced. The associated conditional probabiity of this event is $1 - \beta_i - \beta_j$; we hence regard $r_{(i,j)} = 1 - \beta_i - \beta_j$ to state (i, j). The steady-state page fault probability is then given by:

$$E[Z] = F(LRU) = \sum_{(i,j) \in I} (1 - \beta_i - \beta_j) \frac{\beta_i \beta_j}{1 - \beta_i}$$

8.2 More precise model of Slotted ALOHA

There are m nodes sending packets on a channel using a slotted ALOHA-type protocol. The basic mechanism of an ALOHA protocol is to just try to send a packet at a slot boundary. If it collides, the station tries it again with some probability.

- n are 'backlogged', i.e nodes which tried to send a packet, unsuccessfully
- $m - n$ 'not backlogegd', i.e nodes which do not have packets to send

Assumptions:

- Slot time is unit time
- First transmission: Arrivals to a station in a slot are Poisson(λ/m)
- Retransmission: A backlogged node retransmits with probability q_r
- Stations have no buffer, so arrivals to backlogged nodes are dropped

Probability that an unbacklogged station has a packet to transmit in a slot
= P[arrivals in previous slot] $= 1 - e^{-\lambda/m} = q_a$

P[i backlogged nodes transmit in a slot] $= Q_r(i, n) = \binom{n}{i} q_r^i (1 - q_r)^{n-i}$
P[i unbacklogged node first transmit] $= Q_a(i, n) = \binom{m-n}{i} q_a^i (1 - q_a)^{m-n-i}$

We define a DTMC: number of backlogged users the end of a slot intervals. Now if this number is n at the end of an interval k, what can its value be at the end of the next interval $k + 1$?

- $n-1$: This can be if one backlogged user has a successful transmission, which will be if there is exactly 1 backlogged transmission and no unbacklogged transmission
- n: The number of backlogged remains the same if either
  - Exactly one unbacklogged transmits, and 0 backlogged transmits.
  - Zero unbacklogged transmit, and two or more backlogged transmit
  - zero backlogged and zero unbacklogged transmit
- $n+1$: The number will increase by one if exactly one unbacklogged transmits, and at least one or more backlogged transmit
- $n+i, \quad i \geq 2$: This will happen if exactly i unbacklogged transmit, and any number of backlogged transmit.

The corresponding one-step transition probabilities are:
$p_{n,n-1} = Q_a(0,n) \cdot Q_r(1,n)$
$p_{n,n} = Q_a(1,n)Q_r(0,n) + Q_a(0,n)[1 - Q_r(1,n)]$
$p_{n,n+1} = Q_a(1,n)[1 - Q_r(0,n)]$
$p_{n,n+i} = Q_a(i,n) \qquad 2 \leq i \leq m-n$

Now we can solve for steady-state probabilities by solving:

$$\mathbf{v} = \mathbf{vP}, \qquad \sum v_i = 1$$

where $\mathbf{P} = [P_{ij}]$
Now the expected number of backlogged nodes is: $\mathbf{E}[\text{backlogged nodes}] = \sum_{i=0}^{n} iv_i$

If you apply Little's law to a "box" which contains all the stations that are trying to transmit, then the delay through this box is given by:

$$\left[\frac{\sum_{i=0}^{n} iv_i}{\text{Throughput}}\right]$$

The throughput is the number of packet being transmitted successfully per unit time. Since the unit time is the slot time, this is number of packets being transmitted successfully per slot. Since only one packet can be transmitted per slot, this is the same as the probability of the slot.

We first find the conditional probability of successful transmission given that there are n backlogged nodes.

$$
\begin{aligned}
P_{succ}(n) &= Q_a(1,n)Q_r(0,n) + Q_a(0,n)Q_r(1,n) \\
&= (m-n)q_a(1-q_a)^{m-n-1}(1-q_r)^n + (1-q_a)^{m-n}nq_r(1-q_r)^{n-1} \\
&= (1-q_a)^{m-n-1}(1-q_r)^{n-1}[(m-n)q_a(1-q_r) + (1-q_a)nq_r] \\
\text{for small } \quad q_a \text{ and } \quad q_r &= \\
&= (1-q_a)^{m-n}(1-q_r)^n[(m-n)q_a - (m-n)q_aq_r + n(1-q_a)q_r] \\
&= (1-q_a)^{m-n}(1-q_r)^n[(m-n)q_a - mq_aq_r + nq_r] \\
&= e^{(m-n)q_a}e^{nq_r}[(m-n)q_a + nq_r] \\
&= [(1-x)^y = e^{-xy} \text{ for x small}] \\
&= G(n)e^{-G(n)}
\end{aligned}
$$

where

$$G(n) = nq_r + (m-n)q_a$$

The unconditional probability of successful transmission is:

$$P_{succ} = \sum_n v_n P_{succ}(n)$$

Throughput =Successful transmission per unit time $= 1.P_{succ} + 0.(1 - P_{succ}) = P_{succ}$

9 M/G/1 queuing system

This is a single-server queuing system whose arrival process is Poisson with the average arrival rate λ. The job service times are independent and identically distributed with some distribution function F_B and pdf f_B, which need not be exponential. Jobs are scheduled for service in their order of arrival; that is, scheduling is FCFS.

Let $N(t)$ denote the number of jobs in the system at time t. If $N(t) \geq 1$, then a job is in service, and since the general service time distribution need not be memoryless, besides $N(t)$, we also require knowledge of time spent by the job in service in order to predict the future behaviour of the system. It follows that the stochastic process $N(t)|t \geq 0$ is **NOT** a Markov chain.

The trick to capture this system as a Markov chain is to observe it in a way that does not "interrupt" a service in progress. We do this by observing the system at times of departure of jobs. Let X_n denote the number of jobs in the system just after the departure of the n job. This is a discrete-state, discrete-time stochastic process. But is it a Markov chain? It will be if we can say something about X_{n+1} if only X_n is known.

See Figure **??**. First suppose that $X_n = 0$; that is, the nth departure left the server idle. What state can the $(n+1)$th departure leave the server in, and under what events? Note that the $(n+1)$th departure happens after the $(n+1)$th arrival. So after the nth departure, some time will elapse until an arrival, and then after one service time, there will be the $(n+1)$th departure. This is when we will observe the system. Thus note that we will observe the system after an arrival and a departure - in other words, we are given that there was an arrival, and a depature after that arrival. Thus, the number of jobs in the system left behind after the departure will depend only on how many jobs arrived to the server during the $(n+1)$ job was being served.

X_{n+1} will be zero, if no arrivals come during the $(n+1)$th service. X_{n+1} will be k, if k arrivals come during the $(n+1)$th service.

Let Y_{n+1} = No. of arrivals during $(n+1)^{\text{th}} service$

Now, if $X_n = i > 0$ then that means the nth departure left the server busy with i jobs in the system. The next departure will be exactly after one service time and we will observe the system just after this departure. How many jobs will be left in the system? Well, one less since one job has left, and k more, if k arrive during the service of the $(n+1)$th job. Thus X_{n+1} will be $X_n - 1 + Y_{n+1}$.

To summarize:

$$X_{n+1} = Y_{n+1}, \text{ if } X_n = 0$$
$$X_{n+1} = X_n - 1 + Y_{n+1}, \text{ if } X_n > 0$$

Let

$$P[Y_{n+1}] = a_j$$

and

$$P_{ij} = P[X_{n+1} = j | X_n = i]$$

Then

$$P_{ij} = P[X_n - 1 + Y_{n+1} = j | X_n = i] \text{ if } i > 0$$
$$P_{ij} = P[Y_{n+1} = j - i + 1 | X_n = i] = a_{j-i+1} \text{ if } j \geq i - 1$$
$$P_{0j} = P[Y_{n+1} = j] = a_j, \text{ if } i = 0$$

Thus, the one-step transition probability matrix can be written as:

$$P = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & \cdots \\ a_0 & a_1 & a_2 & a_3 & \cdots \\ 0 & a_0 & a_1 & a_2 & \cdots \\ 0 & 0 & a_0 & a_1 & \cdots \\ 0 & 0 & 0 & a_0 & \cdots \\ \cdot & \cdot & \cdot & \cdot & \cdots \end{bmatrix} \tag{35}$$

Now let us consider the case where $G = D$, that is when service time distribution is deterministic. Let this service time be τ.

Then a_j = Probability that there are j arrivals during one service is easily found, since arrivals are Poisson with rate λ.

$$a_j = \frac{(\lambda\tau)^j e^{-\lambda\tau}}{j!}$$

Thus, for this case we can solve the DTMC equations for $\mathbf{v}$, and find the steady-state probabilities of there being k jobs in the system at departure epochs.

However, how do we get the *unconditional* probability distribution of jobs in the system? Turns out that for Poisson arrivals, the unconditional distribution is also the distribution at departure epochs (stating without proof). Thus, we have what we want.

9.1 M/G/1 waiting time using mean residual time approach

We could solve the DTMC equations precisely, and get the expected number of jobs in the system, apply Little's Law and then get response time. But instead, we will use an interesting and elegant derivation to get the waiting time *directly* - by using the reasoning of *mean residual time.*

Some definitions:

W_i : waiting tme of i^{th} customer

T_i : Residual service time seen by the i^{th} customer

S_i : Service time of i^{th} customer

N_i : Number of customers found in queue by the i^{th} customer

$$\begin{aligned} W_i &= T_i + S_{i-1} + S_{i-2} + \cdots + S_{i-N_i} \\ E[W_i] &= W = T + E[N].E[S] \\ &= R + \tau N_q \end{aligned}$$

Now, by Little's Law

$$\begin{aligned} N_q &= \lambda W \\ W &= T + \lambda \tau W \\ W(1-\rho) &= T \\ W &= \frac{T}{1-\rho} \end{aligned}$$

So now, to find W, we have to estimate mean residual service time.

Let $r(x)$ denote residual service at time x. It is clear that at the beginning of any service S_i, $r(x)$ will have the value of the full service (i.e. $r(x) = S_i$), and it will decrease linearly and at $(x + S_i)^-$ will have the value 0.

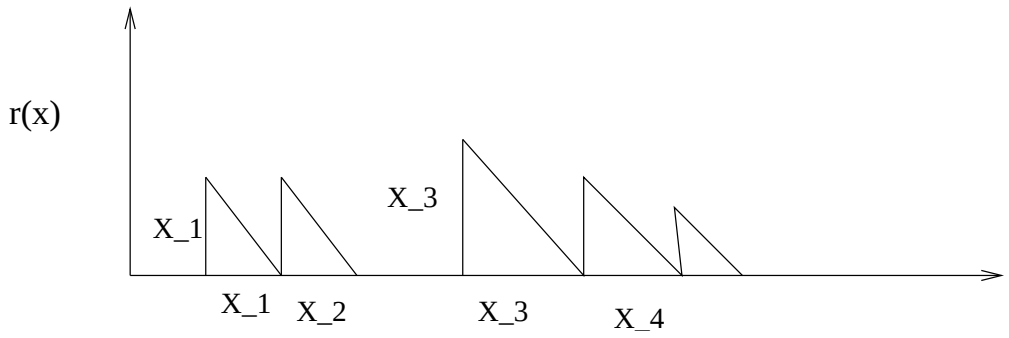


Figure 24: Figure shows "X" instead of "S"

Let t be such that $r(t) = 0$. If we assume that the time average of the residual time is also the instantaneous average, then the mean residual time in the interval $[0, t]$ is given by

$$\frac{\int_0^t r(x)dx}{t}$$

The numerator is the area under the curve. Let $M(t)$ be the number of service completions upto time t. Since the curve is a perfect sawtooth and has a series of right-angled isoceles triangles ($M(t)$ in number) whose sides are of length S_i, the total area is easy to write:

$$\begin{aligned} T &= \lim_{t\to\infty} \frac{\int_0^t r(x)dx}{t} \\ &= \lim_{t\to\infty} \frac{1}{t} \sum_{i=1}^{M(t)} \frac{1}{2} S_i^2 \\ &= \lim_{t\to\infty} \frac{M(t)}{t} \frac{1}{2} \frac{\sum_{i=1}^{M(t)} S_i^2}{M(t)} \end{aligned}$$

$M(t)/t$: throughput = arrival rate for stable system

$$\therefore T = \frac{\lambda \overline{S^2}}{2}$$

$$\therefore W = \frac{\lambda \overline{S^2}}{2(1-\rho)}$$

Here, $\overline{S^2}$ is the *second moment* of the service time distribution. Now note that the variance of the service time (σ_S^2) is given by

$$\sigma_S^2 = \overline{S^2} - \tau^2$$

where τ is mean service time as usual. Also, the squared coefficient of variation of service time is given by:

$$c_S^2 = \frac{\sigma_S^2}{\tau^2}$$

Recall that squared coefficient is a more "normalized" indicator of variability of a random variable, than variance. Then the waiting time formula can be re-written as:

$$\begin{aligned} W &= \frac{\lambda \overline{S^2}}{2(1-\rho)} \\ &= \frac{\lambda(\sigma_s^2 + \tau^2)}{2(1-\rho)} \\ &= \frac{\lambda(c_s^2\tau^2 + \tau^2)}{2(1-\rho)} \\ &= \frac{\lambda\tau^2(c_s^2+1)}{2(1-\rho)} \end{aligned} \tag{36}$$

The response time is then given by:

$$R = \tau + \frac{\lambda\tau^2(c_s^2+1)}{2(1-\rho)}$$

The above formula is called the **Pollaczek-Khinchin Formula for M/G/1 average response time**.

10 Random Sums

Random sums arise when a variable is a sum of a random number of random variables. E.g., the waiting time of an arriving request to a queue with exponential service time is a sum of the service times of the customers in front of it. The number of customers itself is a random number.Thus the waiting time is a random sum. Suppose packet transmission time is a random variable, and there are a random number of retransmissions. Thus total time to successfully transmit a packet is a random sum.

Formally, let $X_i, i = 1, 2, \ldots$ be random variables which are *independent and identically distributed*. Let N be a discrete random variable. Then if

$$T = X_1 + X_2 + \ldots X_N$$

then T is a *random sum*.

E.g. N could be a geometric random variable: $GEOM(p)$ and T could be $Uniform(a, b)$. Suppose the expectation and variance of X and N is known: $E[X], Var[X, E[N], Var[N]$ respectively. Then the following can be derived by conditioning on the value of N:

$$E[T] = E[X].E[N]$$
$$Var[T] = Var[X].E[N] + (E[X])^2 Var[N]$$

.