Notes on Queueing Theory

New version, 2019

Student version

Last saved: 04/08/2022 09:52

Index

1 General Information

Prof. Ing. Giovanni Stea

Dipartimento di Ingegneria dell'Informazione, University of Pisa

Largo L. Lazzarino 1, 56122 Pisa - Italy

Ph. : (+39) 050-2217.653 (direct) .599 (switch)

Fax : (+39) 050-2217.600

E-mail: giovanni.stea@unipi.it

Useful references:

Most of the material found in these notes can be found on the following book:

Leonard Kleinrock, "QUEUEING SYSTEMS" John Wiley & Sons 1975

Another helpful book is:

Mor Harchol-Balter, "Performance Modeling and Design of Computer Systems: Queueing Theory in Action", Cambridge University Press, 18 Feb. 2013

Pre-requisites: Strong background in probability theory, algebra and mathematical analysis (derivatives, integrals, infinite sums). Some notions of linear algebra and differential equations.

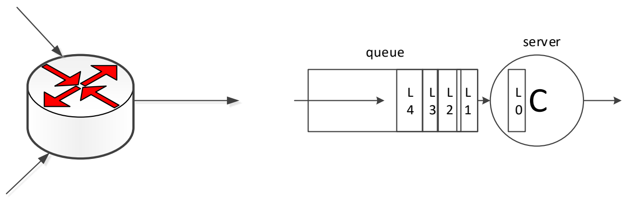
Module length: 18 hours theory, 4-5 hours exercises.

2 Introduction to Queueing Theory

Queueing theory is an **analytical technique** to model systems and get performance measures out of them. It is based on the observation that most of the work in computer systems/networks (and in many other, non-computer-related contexts) is performed by entities (e.g., a CPU, a disk, a peripheral) that **handle one job at a time**. Thus, if there are **many jobs** requiring service, they will **queue up** in a waiting queue, and will eventually get served when those ahead of them have been served.

Why should one employ a queue at all? Because **queues increase system utilization**. In fact, if there is no queue, then the system will just **reject jobs** when it is busy, and then will have to **wait for the next job arrival** when it is idle. Instead, if we allow jobs to queue up, then the next (queued) job will seize the server as soon as it finishes processing the current job. This comes at the price of **adding delay**. Queued jobs spend time doing nothing, waiting for their turn.

A typical example is **the output interface of a router**, with a line whose speed is C bits/s.



If packets arrive fast enough (e.g., from the other input interfaces) they queue up, and they are transmitted sequentially in FCFS order (or, possibly, according to other scheduling disciplines). According to the figure, there are **five** packets in the system: one is being transmitted (packet 0), i.e., is **in the server**; four are **queued** (packets 1 to 4). Let us assume that packet 0 has **just** started transmission at time t. Then we know that it will leave at time $t + L_0/C$, L_0/C being its **service time**. On the other hand, packet 4 will:

- Start being served at $t_4^S = t + \frac{\sum_{i=0}^{3} L_i}{C}$, i.e. it will stop **waiting in the queue** at that time instant;
- Leave at time $t_4^D = t_4^S + \frac{L_4}{C} = t + \frac{\sum_{i=0}^{4} L_i}{C}$.

Assuming that packet 4 has arrived at time $t_4^A < t_4^S < t_4^D$, then $t_4^S - t_4^A$ will be its **waiting time** (or **queueing time**), and $t_4^D - t_4^A$ will be its **response time.** We will always assume that our servers are **work conserving**, i.e., they always serve queued jobs if the queue is non-empty.

In a system like this we would like to answer the following questions:

- What is the **distribution of the number of packets** in the queue (or in the system)?
  You would need to know this, for instance, to **size the buffer,** to bound the probability of dropping a packet due to a buffer overflow;
- What is the distribution of the **response times**?
- What happens if you **change the link speed** to $2C$?

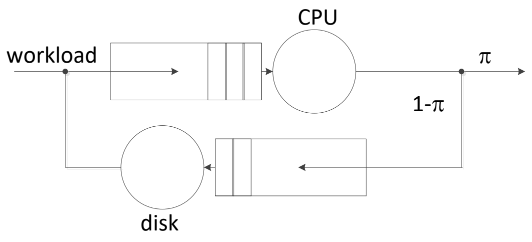
- How much do you need to increase the speed of the line if you want the 90th percentile of the response time to be less than x? (**capacity planning**)

In a system like this, most of the above questions can be answered once we define what the **workload** is. In this case, a workload is given by:

a) The **interarrival time** of the packets, which will be some random variable;
b) The **service demand**, i.e., the time for which they keep the server busy. In this case, it is given by the packet lengths (divided by a constant link speed).

Another example: a **till at the supermarket**. In this case the service demand may be inferred from the number of items in the shopping cart (assuming that the cashier keeps a constant speed). The model is the same, i.e. a queue plus a server, with interarrivals (new customers joining the queue) and departures (customers checking out).

Yet another example: consider a **web server**, that accepts HTTP requests, performs some computations, may or may not need to access a **database**, and then returns the response. In this case it would probably be **inappropriate** to model this using a simple queue and server. In fact, it is still true that requests are served in sequence, but they may interest several components, and visit the same component more than once.



In fact, some requests will only require **computations** at the CPU (and will not interest the disk), others will access the CPU first, then the disk and the CPU again, possibly several times. A more appropriate model for this example would then be the following:
This is a **queueing network**, where each **service center** can be modeled as a queue+server. There is **probabilistic routing**, i.e., jobs (or requests, or transactions, etc.) that leave the CPU may **leave the system** altogether (with probability π) or may be routed to the disk (with probability $1-\pi$). Obviously, the service demand for the same transaction at the two service centers **may be different** (it may differ by orders of magnitude, given the relative speeds of the devices), and if the same request traverses **twice** the same service center it will place a different service demand at that center.
For such a system, we can answer questions like the ones that we formulated previously both **separately**, i.e., per service center, and **globally,** for the system as a whole. In this case, we can also perform a **bottleneck analysis**, i.e., find out which of the two components **limits the performance of the system**. This is interesting, because the **bottleneck** is the component that we need to upgrade **first** if we want our system to scale up (i.e., be able to handle a higher workload). If the CPU is the

bottleneck, then upgrading the disk will **not** yield a faster response or allow this system to handle more transactions per unit of time. Upgrading to a faster CPU will instead provide tangible benefits. The bottleneck is the service center with the **highest utilization**. Utilization is defined as the **percentage of time for which a server is busy**, i.e. is dequeuing and serving jobs. The device with the highest utilization is the one to upgrade first. If you upgrade another – non-bottleneck - device, you will simply reduce *its* utilization (i.e., you will have it idle for a higher percentage of time), but the overall performance of the system will not improve.

The power of QT is that you can (almost always) obtain at least **average performance metrics** (e.g., mean response times, mean number of jobs in the queue), and **very often in a closed form**, just by solving **few simple equations**. In the simplest cases, you can often obtain more detailed performance metrics (e.g., a CDF of the response time), and not just average values. Having closed-form solutions is important if you want to **predict** what happens when the parameters change, e.g., to identify possible bottlenecks.

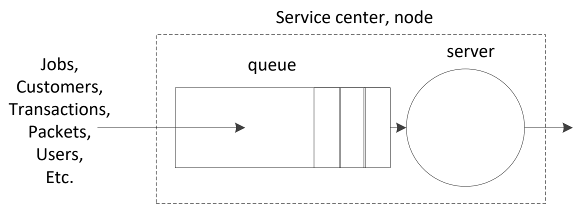
At the very least, you can use QT to **model** systems as queuing networks, and then **simulate their behavior** and get numerical results instead. This is less insightful, but you can always do it. Thus, QT is also a **modeling paradigm**. Its power derives from the fact that it is **quite abstract**: you need to describe your fragment of reality **at a very high level**, without going into too many details. If you can model a physical system in terms of service centers, queues, interarrivals and service demands, then you can **quickly get some insight** into the performance of that system.
Compare this modeling style, and the (weak) insight it requires, to the level of detail that you may want to attain in an in-depth simulation modeling (e.g., simulating the fetch and execution phases of a CPU with real instructions, etc.). The two are clearly different, and they will require different time and money.

On the other hand, QT has its limitations. It is quite apt an instrument if you want to do a **quick and dirty** evaluation, but QT modeling may incur the risk of **oversimplification.** Neglecting crucial aspects just because they complicate your QT model too much happens all too often.
What can you **expect** from QT?
Those in the know assert that, if your model is correct, then normally you obtain **fairly accurate throughput predictions** (say, within 10% of the actual throughput). On the other hand, **response times** tend to be **less accurate**, and the error that you get will be **load dependent** (the higher the load, i.e., the nearer your system is to **saturation**, the larger your errors are going to be).

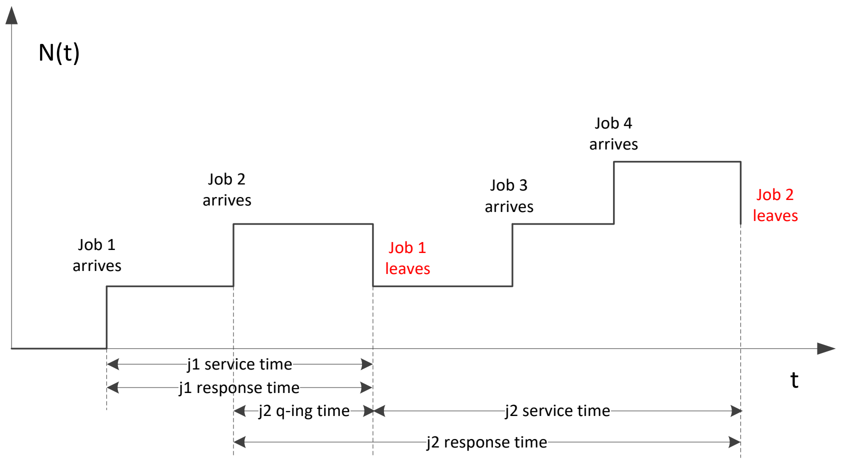
3 Analysis of queueing nodes in isolation

3.1 *Characterizing the state of a queueing node*

Let us start with an example: a single queue + server (something that we will study a lot in the future), often called a **service center** or **node**. This may model anything (e.g., a network interface queue just as well as a post office's).



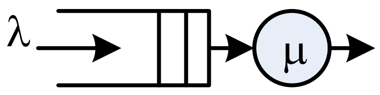
We need to characterize the **state** of this system at a given time t. The way we characterize its state depends **on what we want to observe**. For instance, we may be interested in the **number of jobs in the system** at time t (also called the **backlog** at that time)



$N(t)$ is a **discrete** quantity (it is an integer), which is a function of a **continuous parameter (time)**. The above is a **trajectory** (or realization, or sample path), which depends on the (possibly random) **interarrival times** of the customers at the queue, as well as on the (possibly random) **service times** (or service demands). Given different interarrival and service times, the trajectory is going to be different. We call such random trajectories **random (or stochastic) processes**.

Let us start from one where:

- The **interarrival times** between jobs are IID **exponentials**, with a rate λ (or a mean interarrival time $1/\lambda$).
- The **service demands** are IID exponentials with a mean $1/\mu$.
- Interarrivals and service demands are independent.
- The queue is infinite and FCFS.



In such a system (which is often called a **birth-death system**), jobs will get in, they will queue up, and whenever the system is non-empty the server will pick the head-of-line job in the queue, work on it for an exponential time, etc. etc.

We are interested in computing the **distribution** of the number of jobs in the system, from which (as we will see) we will be able to derive the one of the response time, etc.

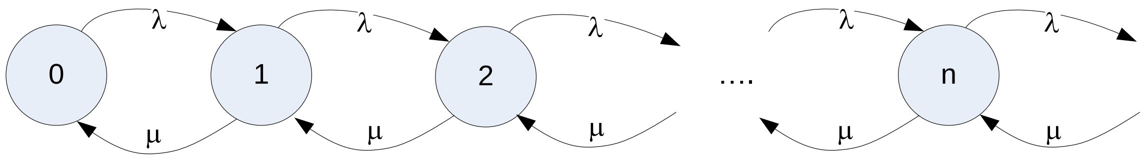
The trajectories of this system can go up and down. We first observe that, if both interarrivals and service times are exponentially distributed, **then $\boldsymbol{N(t)}$ describes the state of this process completely**. There is no need, in fact, to know the last time of arrival/departure in the past, since this does not yield any more insight: exponentials are memoryless. This means that the future evolutions of this system can be predicted (in a stochastic sense) only by knowing $\boldsymbol{N(t)}$.

As a counterexample, consider a system where:

- Arrivals are **exponential;**
- Service times are **constant.**

For the above system, $N(t)$ alone **would not be a complete state characterization**. The time at which the next departure event will occur is univocally determined by the time of the **last departure**, hence the future does depend on the past.

This means that we can setup a **state diagram**, describing the evolution of such a system in time, as follows:

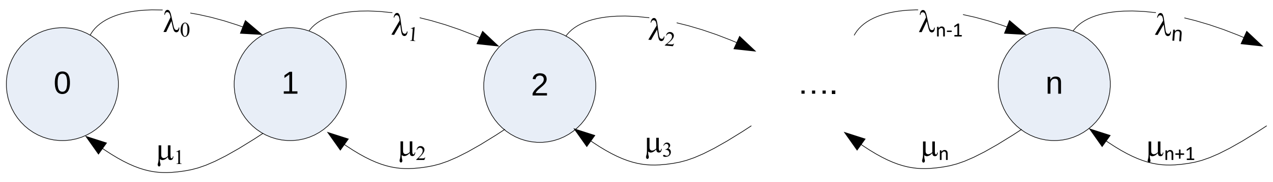


The circles are the **system states at time *t*,** and the arcs are the **transitions** from one state to another. The above is sometimes called a **transition-rate diagram**, since λ and μ are in fact transition rates between adjacent states, and – more often – **continuous-time Markov chain (CTMC)**.

We will always assume that λ and μ are **time-independent**, i.e. they do not change over time. They might, instead, be **state-dependent**, i.e. they may depend on the **state of the system**. There are many practical cases in which they do:

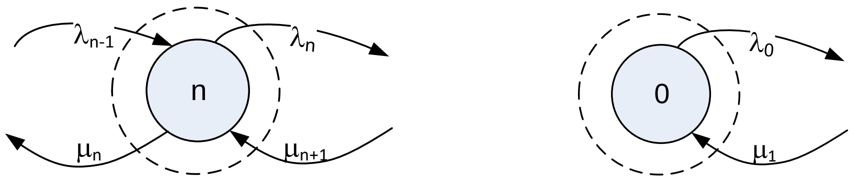
- In some CPUs, **the clock frequency is varied** depending on the number of tasks to be executed: more tasks mean higher frequency, hence higher service rates depending on the state of the system.
- Most people will be **less likely to join a queue** (e.g., to enter a museum) if the queue is long. In this case, the arrival rates would clearly depend on the system state.

Therefore, it may be worthwhile to use λ_n instead of λ, to denote the arrival rate **when the system holds *n* jobs**, and μ_n in place of μ. The resulting CTMC would be the following. Note that the earlier CTMC was a special case of this one, when λ_n=λ and μ_n=μ for all the values of n.



The probability of **two simultaneous events** (i.e., one arrival and one departure, or two arrivals, or two departures) is **negligible**. For this reason, there are only arcs **reaching out to the nearest (left or right) neighbors** in the graph. Systems that admit only nearest-neighbor transitions are quite easy to analyze.

Focus now on one state $n > 0$, and fix a time t. Call $p_n(t)$ the probability that there are n jobs at time t, i.e. $p_n(t) = P\{N(t) = n\}$. If you circle that state, you can quickly write a **probability flow-balance equation** involving that state, just by looking at the CTMC:



$$\begin{cases} \frac{d}{dt}p_n(t) = -(\lambda_n + \mu_n)\cdot p_n(t) + \mu_{n+1}\cdot p_{n+1}(t) + \lambda_{n-1}\cdot p_{n-1}(t) & n > 0 \\ \frac{d}{dt}p_0(t) = -\lambda_0 \cdot p_0(t) + \mu_1 \cdot p_1(t) & n = 0 \end{cases}$$

The intuitive explanation for the above set of equations is the following: the term on the left is the **variation in the flow of probability.** That variation stems from the balance of:

- An **outgoing flow**, with a minus sign
- An **incoming flow**, with a plus sign.

Both of which are at the right-hand side of the equations. This way, λ_n can be interpreted as the **transition rate** from state n to state $n + 1$, and μ_n as the transition rate from state n to $n - 1$. $\lambda_n \cdot p_n(t)$ is the **flow of probability** which is poured from state n to state $n + 1$, etc.

$$\underbrace{\frac{d}{dt}p_n(t)}_{\text{Variation of flow}} = \underbrace{-(\lambda_n + \mu_n)\cdot p_n(t)}_{\text{Outgoing flow}} + \underbrace{\mu_{n+1}\cdot p_{n+1}(t) + \lambda_{n-1}\cdot p_{n-1}(t)}_{\text{Incoming flow}}$$

The above equations are called **Chapman-Kolmogorov's equations**[1]. The one for $n = 0$ is slightly different, since the incoming and outgoing arcs from state 0 are different.

The evolution of such a system over time (i.e., for each t) is thus completely specified once we solve the CK system of (an infinite number of) differential equations. Systems of differential equations can be solved once **initial conditions** are given in the form of a PMF for the state at time 0, i.e. $p_n(0)\ \forall n$, such that $\sum_{n=0}^{+\infty} p_n(0) = 1$. A typical initial condition is $p_0(0) = 1$, $p_n(0) = 0$ for $n > 0$, i.e. the system is initially empty. This is tough, in general, and – as we will show – not necessary for our purposes. However, we now solve the CK system in two very simple cases (we will *not* need to solve it in general, but it pays to do it a couple of times to figure out how things are).

3.1.1 Birth-only process

This can be seen as an instance of the above, obtained by setting $\mu_n = 0\ \forall n$. In the simplest case $\lambda_n = \lambda,\ \forall n$, the CK equations become:

$$\begin{cases} \frac{d}{dt} p_n(t) = -\lambda \cdot p_n(t) + \lambda \cdot p_{n-1}(t) & n > 0 \\ \frac{d}{dt} p_0(t) = -\lambda \cdot p_0(t) & n = 0 \end{cases}$$

Assuming as initial conditions the usual ones, i.e., $p_0(0) = 1$, $p_n(0) = 0$ for $n > 0$, we can easily solve the equations. In fact:

- $n = 0$: $\frac{d}{dt} p_0(t) = -\lambda \cdot p_0(t)$ admits as a solution $p_0(t) = k \cdot e^{-\lambda t}$. The constant k can be set using the initial condition $p_0(0) = 1$, hence $k = 1$. Thus, $p_0(t) = e^{-\lambda t}$.
- $n = 1$: $\frac{d}{dt} p_1(t) = -\lambda \cdot p_1(t) + \lambda \cdot p_0(t) = -\lambda \cdot p_1(t) + \lambda \cdot e^{-\lambda t}$. The solution to this one is $p_1(t) = \lambda t \cdot e^{-\lambda t}$.
- $n > 1$: by generalizing the same computations, one easily gets $p_n(t) = \frac{(\lambda t)^n}{n!} \cdot e^{-\lambda t}$.

Therefore, the general expression is $p_n(t) = \frac{(\lambda t)^n}{n!} \cdot e^{-\lambda t},\ \forall n$.

The above one is a **Poisson distribution**, with a mean λt. This is why we normally call "Poisson processes" those whose interarrival times are IID exponentials[2]. Moreover, we get that, **as time increases**, $\lim_{t\to\infty} p_n(t) = 0,\ \forall n$. Again, this should not surprise us, since in a birth-only process the

[1] A more formal derivation of CK equations can be found in the Appendix

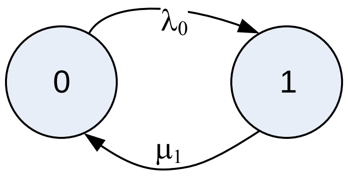
[2] A more formal definition of a Poisson process can be found in the Appendix

trajectory grows indefinitely with time, so the probability that n jobs have arrived in an infinite time must go to zero for each finite n.

3.1.2 Two-state birth-death process

This is a model for a single-slot buffer. If a job arrives when the system is in state 1, then that job is **discarded**. Thus, it is $\lambda_0 = \lambda$, $\lambda_1 = 0$, and $\mu_1 = \mu$ (the fact that $\mu_0 = 0$ is pretty obvious).

The CK equations for this system are the following:



$$\begin{cases} \frac{d}{dt} p_1(t) = -\mu \cdot p_1(t) + \lambda \cdot p_0(t) \\ \frac{d}{dt} p_0(t) = -\lambda \cdot p_0(t) + \mu \cdot p_1(t) \end{cases}$$

Summing both equations, we get $\frac{d}{dt}p_1(t) + \frac{d}{dt}p_0(t) = 0$, which means that $p_0(t) + p_1(t) = const$, which is obviously true, with $const = 1\ \forall t$.

We can solve this system using standard techniques (therein including using the LST), and get the following result:

$$\begin{cases} p_0(t) = \frac{\mu}{\lambda+\mu} + \left[p_0(0) - \frac{\mu}{\lambda+\mu}\right] e^{-(\lambda+\mu)t} \\ p_1(t) = \frac{\lambda}{\lambda+\mu} + \left[p_1(0) - \frac{\lambda}{\lambda+\mu}\right] e^{-(\lambda+\mu)t} \end{cases}$$

Now, these expressions describe the probability of being in either state as time progresses. They do **depend on the initial state**, i.e., on $p_i(0)$. It is always $p_0(t) + p_1(t) = 1$

If we let $t \to +\infty$, we observe the following:

$$\begin{cases} p_0 \triangleq \lim_{t\to+\infty} p_0(t) = \frac{\mu}{\lambda+\mu} \\ p_1 \triangleq \lim_{t\to+\infty} p_1(t) = \frac{\lambda}{\lambda+\mu} \end{cases}$$

And, again, $p_0 + p_1 = 1$. We call p_0, p_1 the **steady-state probabilities**. At the **steady state**, in fact, they do not depend on the time anymore. On the other hand, $p_i(t)$ is called the **transient probability**. Note that, while the transient probability does depend on the initial conditions (see the above formulas), **the steady-state probability does not**. It is **independent of the initial conditions**. Depending on the initial conditions, the steady-state probability will be approached from below (e.g., if $p_0(0) < \mu/(\mu+\lambda)$), or from above (if the opposite inequality holds). If, instead, $p_0(0) = \mu/(\mu+\lambda)$, then the system will be in the steady state $\forall t$. However, the fact that a steady state is reached **and** the value of the SS probabilities will not change.

Note that, if we are **only interested in SS probabilities**, there is a much quicker way to obtain them – notably, one that does not involve differential equations. In fact, by the very definition of steady state, we have that:

$$\forall n, \qquad \frac{d}{dt}p_n(t) = 0$$

Hence, under this hypothesis, we can compute the SS probabilities by solving the following system:

$$\begin{cases} 0 = -\mu \cdot p_1 + \lambda \cdot p_0 \\ 0 = -\lambda \cdot p_0 + \mu \cdot p_1 \end{cases}$$

Which is **only algebraic**. In this case, the two equations are clearly **not independent**, so we can discard one and use the normalization condition in its stead: $p_0 + p_1 = 1$. The system is thus:

$$\begin{cases} 0 = -\mu \cdot p_1 + \lambda \cdot p_0 \\ p_0 + p_1 = 1 \end{cases}$$

And its solution is the one that we have just found – yet computed considerably faster.

3.2 Steady-state analysis of birth-death systems

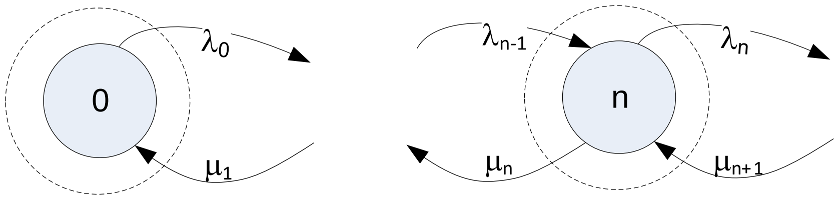
The above example reveals something that is indeed general. If we want to compute the steady-state probabilities in a birth-death system (whatever its number of states), there are two ways:

a) The **complex one**, which consists in formulating the CK (differential) equations, solving the system – thus obtaining a solution in the form $p_n(t)$, and getting $p_n = \lim_{t\to+\infty} p_n(t)$.
b) The **simple one**, which consists in equating $\frac{d}{dt}p_n(t) = 0\ \forall n$ in the CK equations, solving an algebraic system, and getting p_n, $\forall n$.

The complex one has the (slight) advantage of providing us with the transient probabilities as well, but these are normally uninteresting for our purposes, hence we will use the simple method from now on.

Note that the system of the first example – the birth-only process – **does not admit a steady state**. In fact, it is $p_n = \lim_{t\to\infty} p_n(t) = 0,\ \forall n$. It is an unpleasant fact that systems **may or may not admit a steady state**, and – when they do – they might reach it only **under specific conditions** (e.g., a constraint on the arrival rates, or something similar). The simple method can **only** be used **if the system does reach a steady state** (this is what allows us to set the derivatives to zero in the first place), and this hypothesis **must always be tested** *a posteriori*.

Both methods can be applied to systems with an arbitrary number of states, **as long as they do admit a steady state**. The physical interpretation of "reaching a steady state" is the following:



The flow of probability through the dashed surface, which is the derivative at the left-hand side of the CK equation, is **null**. Thus, the **outgoing and incoming flows must balance each other**, i.e.

$$\begin{cases}(\lambda_n+\mu_n)\cdot p_n=\lambda_{n-1}\cdot p_{n-1}+\mu_{n+1}\cdot p_{n+1} & n>0\\ \lambda_0\cdot p_0=\mu_1\cdot p_1\end{cases}$$

This said, computing the SS probabilities in a birth-death system is straightforward:

a) You draw the **CTMC**, according to the modeling of your system;
b) You formulate the above **steady-state equilibrium equations**;
c) You add the **normalization condition**, i.e. $\sum_{n=0}^{+\infty} p_n=1$

This way you get a **non-homogeneous algebraic system** (non-homogeneity been assured by the constant "1" in the normalization condition), which – as such – admits only **one solution**.

That solution can be computed quite easily, starting from $n=0$ and working your way up for increasing values of n.

- $n=0$: from the equation we get $p_1=\frac{\lambda_0}{\mu_1}\cdot p_0$
- $n=1$: we instantiate $(\lambda_n+\mu_n)\cdot p_n=\lambda_{n-1}\cdot p_{n-1}+\mu_{n+1}\cdot p_{n+1}$ and substitute $p_1=\frac{\lambda_0}{\mu_1}\cdot p_0$, thus obtaining:

$$(\lambda_1+\mu_1)\cdot p_1=\lambda_0\cdot p_0+\mu_2\cdot p_2$$

$$(\lambda_1+\mu_1)\cdot\frac{\lambda_0}{\mu_1}\cdot p_0=\lambda_0\cdot p_0+\mu_2\cdot p_2$$

$$p_2=\frac{1}{\mu_2}\left[(\lambda_1+\mu_1)\cdot\frac{\lambda_0}{\mu_1}-\lambda_0\right]\cdot p_0=\frac{\lambda_0\cdot\lambda_1}{\mu_1\cdot\mu_2}p_0$$

- $n>1$: after few algebraic manipulations, it is clear that we always obtain $p_n=\frac{\lambda_0\cdot\lambda_1\cdot\ldots\cdot\lambda_{n-1}}{\mu_1\cdot\mu_2\cdot\ldots\cdot\mu_n}p_0=\prod_{i=0}^{n-1}\frac{\lambda_i}{\mu_{i+1}}p_0$. Note that this expression holds also when $n=1$.

Thus, in the end, we get the following:

$$\begin{cases} p_n = \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} p_0, & n \geq 1 \\ \sum_{n=0}^{+\infty} p_n = 1 \end{cases}$$

And the normalization condition can be rewritten as follows:

$$p_0 \left[1 + \sum_{n=1}^{+\infty} \left(\prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} \right) \right] = 1$$

A **necessary and sufficient condition** for the system to admit a steady state (i.e., to be **stable**) is that **the above sum be finite**.

If that sum if finite, call *S* the term between square brackets, and we get:

$$\begin{cases} p_0 = \frac{1}{S} \\ p_n = \frac{1}{S} \cdot \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}}, & n \geq 1 \end{cases}$$

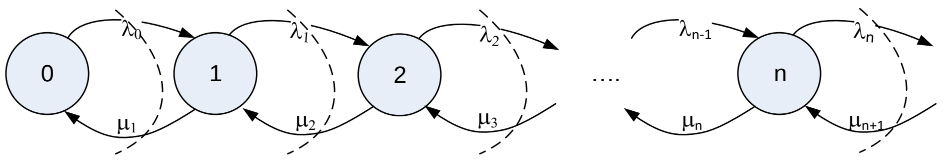
Otherwise, we get $p_n = 0 \quad \forall n$.

Note that the problem of **stability** only exists with systems with **an infinite number of states**. In fact, in system with **finite states** (such as the single-slot buffer) the above sum would only include a **finite number of terms**, hence would always be finite. Therefore, only systems with **infinite states may not reach a steady state**. Systems with finite states always do.

The above method for computing the SS probabilities is **entirely general** and can be applied to any birth-death system. The above way of computing probabilities, i.e. "circling" **each single state** and balancing its outgoing and incoming flow, leads to the so-called **global equilibrium equations**, and it is not the only one. In fact, at the equilibrium, the outgoing and incoming flow **through every surface,** circling any number of states, must balance each other out (otherwise some derivative would be non-null). Therefore, one may choose **arbitrary perimeters** across which to enforce the flow balance, and this sometimes leads to simpler computations.

For instance, **local equilibrium equations** are those written balancing the flows through perimeters including **all the states from 0 to *n* included**, and they are the following:

$$\begin{aligned}&\lambda_0 \cdot p_0 = \mu_1 \cdot p_1\\&\lambda_1 \cdot p_1 = \mu_2 \cdot p_2\\&\dots\\&\lambda_n \cdot p_n = \mu_{n+1} \cdot p_{n+1}\end{aligned}$$



From which we get $p_n = \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} p_0$, even more quickly than before.

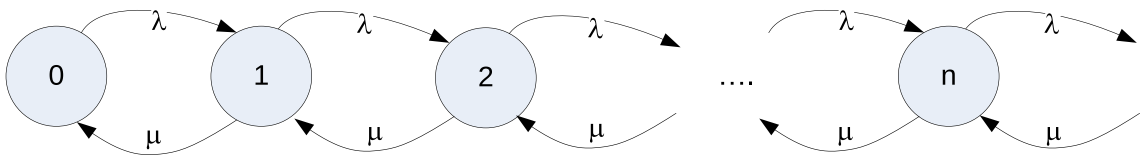
As will be apparent later on, **global equations** are always easy to write. **Local equations** are easy to write (possibly easier than global ones) when the CTMC is **simple**, but they quickly get **overly complicated** if the CTMC is messy: if there are too many arcs around, you run the risk of **forgetting something**.

3.3 M/M/1 systems

Let us now discuss in some detail the simplest birth-death system, which is called an M/M/1 system. The latter is known as **Kendall's notation**, and consists of (at least) three indications:

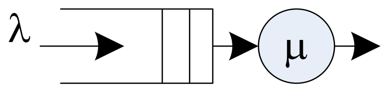
- The distribution of interarrival times: M for memoryless (D for deterministic, E for Erlang, G for Generic, etc.)
- The distribution of service times: the same letters can appear
- The number of servers, one in this case.

There can be other indications following these three, such as the **system capacity** (the max. number of jobs allowed in the system), or the **population** from which arrivals are drawn. These are both assumed to be infinite in our case, and – when they are – they need not be stated explicitly. We will discuss systems with finite queues and finite populations later on.



Assume $\lambda_n = \lambda, \quad \mu_n = \mu$, i.e. arrival and departure rates are **constant** (or state-independent, or load-independent), and the queue is infinite. In this case, the relationship derived for generic birth-death processes becomes:

$$p_n = \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} p_0 = \left(\frac{\lambda}{\mu}\right)^n p_0 \quad n \geq 0$$



We call $\rho = \lambda/\mu$ the **utilization** of this system (the reason why it is called like that will be given in a minute). Then the normalization condition is the following:

$$p_0\left[1+\sum_{n=1}^{+\infty}\left(\prod_{i=0}^{n-1}\frac{\lambda_i}{\mu_{i+1}}\right)\right] = 1 \Rightarrow p_0\left[\sum_{n=0}^{+\infty}\rho^n\right] = 1$$

Which means that the **stability condition is** $\rho < 1$. Under that condition, the above infinite sum converges to $\frac{1}{1-\rho}$, hence $p_0 = (1-\rho)$ and $p_n = (1-\rho)\cdot\rho^n$.

The fact that $p_0 = (1-\rho)$ justifies the name of utilization given to ρ: in fact, it is $\rho = 1 - p_0 = 0\cdot p_0 + 1\cdot(1-p_0)$, hence ρ is the **mean number of jobs in the server** – or the **fraction of time** for which it is busy, hence its utilization. It also helps us to get a physical explanation of the stability condition: if ρ is a utilization, it **cannot grow beyond one**: as it approaches one, the system becomes unstable, and queues grow to infinity.

In fact, $\rho < 1$ means $\lambda < \mu$, i.e. the mean interarrival time $1/\lambda$ is larger than the mean service time $1/\mu$. We call a system where $\rho < 1$ **positive recurrent.**

When the opposite occurs, $\lambda > \mu$, then the system accumulates **more and more jobs** in the queue as time progresses, hence $\lim_{t\to\infty} p_n(t) = 0$ for all finite values of n. A system where $\rho > 1$ is called **transient**.

The case $\rho = 1$, i.e. $\lambda = \mu$, is somewhat tricky to understand. In this case the system is not stable, and the reason why it is not is that it *may* happen that a **very large** service time occurs at least once (recall that exponentials have a tail extending to infinity), during which the queue gets so large that it is never able to empty again. We call a system where $\rho = 1$ **null recurrent**.

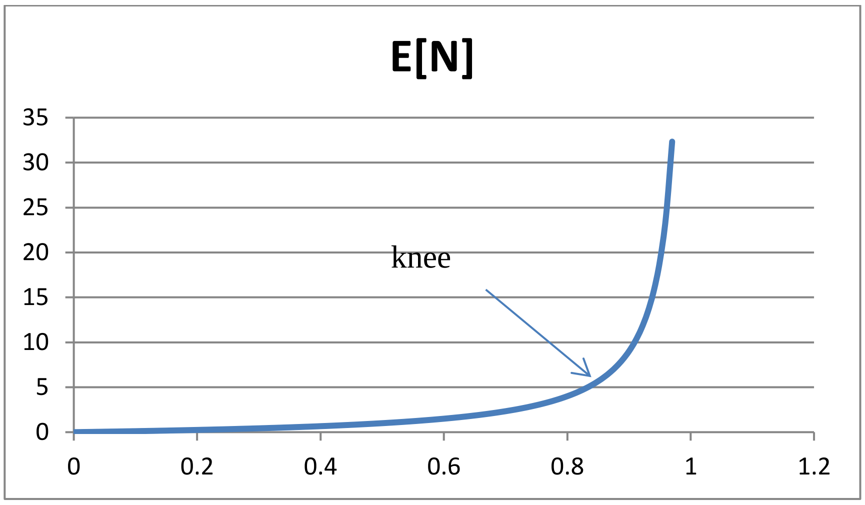
For a positive recurrent system, the distribution of the number of jobs in the system at the steady state is **geometric**: $p_n = (1-\rho)\cdot\rho^n$, with a success probability $p = 1-\rho$. Therefore, it is straightforward to compute its mean and its variance, i.e. the **main indexes of the number of jobs in the system**. It is:

$E[N] = \sum_{n=0}^{+\infty} n\cdot p_n = \frac{\rho}{1-\rho}$, $Var(N) = \frac{\rho}{(1-\rho)^2}$

It is particularly interesting to observe the behavior of $E[N]$ as a function of ρ. This function is called the **Kleinrock function** (also called the "hockey-stick"), and its shape is the one in the figure: it is practically **flat** until $\rho = 0.5$ (when it reaches 1), and then exhibits a **knee** and has a vertical asymptote for $\rho \to 1$. This is the typical behavior of systems under a varying workload:

- In **low-load** conditions, the mean number of jobs in the system is below one (i.e., the system is either empty or serving the one and only job present, most of the time).

- As ρ grows beyond 0.5, queueing starts to occur frequently. As $\rho \to 1$, the system **saturates**, hence a marginal increase in ρ translates to a huge increase in the number of jobs. When you provision a system, the region to the right of the knee point is the one which you will want to avoid.



Exercise
You have to dimension a network interface for your system. Its workload consists in packets whose length is exponentially distributed with a mean $1/\gamma$. Packet interarrival times at the interface with are exponential with a mean $1/\lambda$. Compute the line speed C so that:

1) the mean backlog is B packets
2) the 95th percentile of the backlog is Π packets

We observe that this is an M/M/1 system, with an arrival rate λ. As for the service rate, we get that $E[t_s] = \frac{E[length]}{C} = \frac{1}{\mu}$, hence we get $\mu = \gamma \cdot C$.

The first question can be readily answered by observing that the mean backlog is $B = E[N] = \frac{\rho}{1-\rho} = \frac{\lambda/(\gamma \cdot C)}{1-\lambda/(\gamma \cdot C)}$. We solve the latter for C and we get $C = \frac{\lambda \cdot (1+B)}{\gamma \cdot B}$.
Note that this only holds if the system *does* admit a steady state, hence if $\rho = \lambda/(\gamma \cdot C) < 1$.
The second question can be answered by solving the following equation: $P\{N \leq \Pi\} = 0.95$. However, we quickly get $P\{N \leq x\} = \sum_{n=0}^{x} p_n = \sum_{n=0}^{x} (1-\rho) \cdot \rho^n = (1-\rho) \cdot \frac{1-\rho^{x+1}}{1-\rho} = 1 - \rho^{x+1}$.
Therefore, the equation that we need to solve is $1 - \rho^{\Pi+1} = 0.95$, i.e.$C^{\Pi+1} = 20 \cdot (\lambda/\gamma)^{\Pi+1}$, i.e. $C = \sqrt[(\Pi+1)]{20} \cdot \frac{\lambda}{\gamma}$

■

3.3.1 Mean performance indexes

The important mean performance indexes in queueing systems are the following:

- The mean **number of jobs in the system** $E[N]$. We have already computed it.
- The mean **number of jobs in the <u>queue</u>** (i.e., not counting the one being served), called $E[N_q]$
- The mean **response time** $E[R]$, i.e. the mean time between the arrival and the departure of the same job.
- The mean **waiting time** $E[W]$ (or queueing time), i.e. the mean time between the arrival and the start of the service of the same job. This is what **people** care about, normally.

The optimum would be to be able to compute the **distributions** of all the above quantities. From these, in fact, we can compute everything: mean value, variance, percentiles, etc. However, this is only possible if the system is simple enough. If the system is too complex, we will have to settle for **the mean values** which are always easy to compute.

We have already discussed how to compute $E[N]$.

<u>Number of jobs in the queue</u>

We move to analyzing RV N_q.The latter takes on the following values:

- 0, with probability $p_0 + p_1$ (in fact, our queueing systems are work-conserving).
- 1, with probability p_2
- $k \geq 1$, with probability p_{k+1}.

From the above, computing the mean value is quite straightforward:

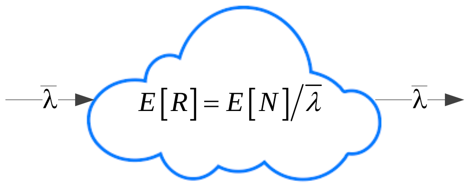
$$\begin{aligned} E[N_q] &= \sum_{k=1}^{+\infty} k \cdot p_{k+1} \\ &= \sum_{k=2}^{+\infty}(k-1) \cdot p_k = \sum_{k=1}^{+\infty}(k-1) \cdot p_k = \sum_{k=1}^{+\infty} k \cdot p_k - \sum_{k=1}^{+\infty} \cdot p_k \\ &= E[N] - (1 - p_0) \\ &= E[N] - \rho \end{aligned}$$

This result (which is common to all the systems with **one** server) could have been obtained much more easily by observing that **mean values are additive**, and that ρ is the server's utilization, i.e., the **mean number of jobs in the server**. Therefore, it **must be** $E[N_q] + \rho = E[N]$.

<u>Response time</u>

The mean response time can be computed using a general result, which is very useful in many cases. This is called **Little's Law (or Little's Theorem)**, and it states the following:

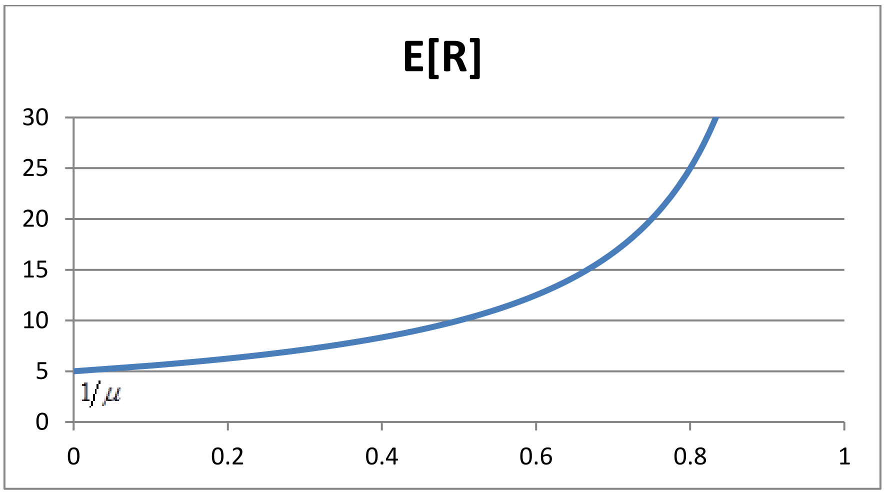
Consider a system in a steady state, such that ***no jobs are created/destroyed within the system*** *and let $\bar{\lambda}$ be its* <u>*mean*</u> *arrival rate: then the mean response time is $E[R] = E[N]/\bar{\lambda}$.*



Little's law can be applied to every system at the steady state, under very loose hypotheses: the system may not be FCFS, it may have non-exponential arrivals/departures, whatever. The only requirement is that no jobs are created/destroyed within the system, so that the average arrival rate $\bar{\lambda}$ is also the average **departure rate**.

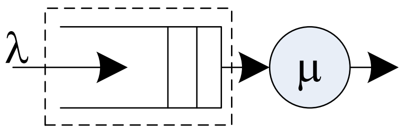
An intuitive rationale behind Little's law is the following: if the system is in a steady state, when a job arrives, the number of jobs that it sees **ahead of itself** is statistically equal to the number of jobs it will **leave behind** on its departure. The latter is equal to $E[N]$, and has arrived at a rate $\bar{\lambda}$ during the response time of that job $E[R]$. Hence, it makes sense that $E[R] = E[N]/\bar{\lambda}$. Note that Little's law only applies to **mean values**, not to distributions.

We can use Little's law to compute $E[R]$. In an M/M/1 system it is $\bar{\lambda} = \sum_{n=0}^{+\infty} \lambda_n \cdot p_n = \lambda \cdot \sum_{n=0}^{+\infty} p_n = \lambda$, so it is fairly easy to see that $E[R] = \frac{1}{\lambda} \cdot \frac{\rho}{1-\rho} = \frac{1/\mu}{1-\rho} = \frac{1}{\mu-\lambda}$.



When the load is **small** $\rho \ll 1$,the response time tends to $1/\mu$, which is in fact the **mean service time** $E[t_s]$. This makes perfect sense, since the system will always be empty, and any arriving job will only spend time in the server. As ρ increases, queueing starts to occur frequently, until the system saturates and the response time grows to infinity.

<u>Waiting time</u>



The mean waiting time can be computed by applying **Little's law to the queue** at the equilibrium. Note that Little's law can be applied anywhere, under very broad conditions.
Since the system "queue" is in equilibrium, then its arrival **and** departure rates are equal to λ, hence $E[W] = \frac{E[N_q]}{\lambda} = \frac{E[N]-\rho}{\lambda} = E[R] - \frac{1}{\mu}$.
The last expression is obvious, since mean values are additive and the mean service time is

$$E[t_s] = \frac{1}{\mu}$$

<u>**Throughput**</u>

In queueing systems, it is often required to compute the **throughput**, i.e., the number of jobs served per unit of time. The throughput is often denoted with γ. We start with an intuitive reasoning:

- If $\gamma > \lambda$, then it means that there are jobs that get out without having been injected in the system. In other words, the system should **create jobs internally** for this to be possible. This is not the case, of course.
- If, on the other hand, $\gamma < \lambda$, there would be jobs that stay in the queue indefinitely (since they do get in, but they never get out). This is impossible, since the system is FCFS and stable.

Therefore, the only possibility is that $\gamma = \lambda$. This is a given in systems without losses. The only case when $\gamma < \lambda$ is possible is when systems have **finite memory**: in this case, due to the interplay of the random arrival and service times, there might be cases when some jobs are rejected.
In any case, the **formal definition of throughput** is the following:
$\gamma \triangleq \sum_{n=1}^{+\infty} \mu_n \cdot p_n$, which in this case is

$$\gamma = \mu \sum_{n=1}^{+\infty} p_n = \mu \cdot (1 - p_0) = \mu \cdot \rho = \lambda$$

3.3.2 An alternative way to compute mean performance indexes

Sometimes computing the steady-state probabilities using the direct method is challenging, because the computations involved are non-trivial. In many cases, we can still compute the mean performance indexes **without** computing the SS probabilities. The method is **quite general** (i.e., it can be applied to any birth-death system) and will be exemplified on the M/M/1 for simplicity.
Consider the global steady-state equations:

$$\begin{cases} \lambda \cdot p_0 = \mu \cdot p_1 \\ (\lambda + \mu) \cdot p_n = \lambda \cdot p_{n-1} + \mu \cdot p_{n+1} \qquad n \geq 1 \end{cases}$$

The technique is as follows: you **multiply each equation by z^n**, $z \in \mathbb{C}, |z| < 1$, and then **sum everything up**. We obtain:

$$\begin{cases} \lambda \cdot z^0 \cdot p_0 = \mu \cdot z^0 \cdot p_1 \\ (\lambda+\mu) \cdot z^n \cdot p_n = \lambda \cdot z^n \cdot p_{n-1} + \mu \cdot z^n \cdot p_{n+1} \qquad n \ge 1 \end{cases}$$

$$\lambda \cdot z^0 \cdot p_0 + \sum_{n=1}^{+\infty} (\lambda+\mu) \cdot z^n \cdot p_n = \mu \cdot z^0 \cdot p_1 + \sum_{n=1}^{+\infty} \lambda \cdot z^n \cdot p_{n-1} + \sum_{n=1}^{+\infty} \mu \cdot z^n \cdot p_{n+1}$$

Then we recall the definition of **PGF of a discrete non-negative RV:** $\mathbf{P}(z) \triangleq E[z^N] = \sum_{k=0}^{+\infty} z^k \cdot p_k$. In this case, the state of the system N is a discrete and non-negative RV. We manipulate the above expression to obtain $\mathbf{P}(z)$:

$$\lambda \cdot z^0 \cdot p_0 + \sum_{n=1}^{+\infty} (\lambda+\mu) \cdot z^n \cdot p_n = \mu \cdot z^0 \cdot p_1 + \sum_{n=1}^{+\infty} \lambda \cdot z^n \cdot p_{n-1} + \sum_{n=1}^{+\infty} \mu \cdot z^n \cdot p_{n+1}$$

$$-\mu \cdot p_0 + (\lambda+\mu) \cdot \sum_{n=0}^{+\infty} z^n \cdot p_n = \lambda \cdot z \cdot \sum_{n=1}^{+\infty} z^{n-1} \cdot p_{n-1} + \mu \cdot \sum_{n=0}^{+\infty} z^n \cdot p_{n+1}$$

$$-\mu \cdot p_0 + (\lambda+\mu) \cdot \sum_{n=0}^{+\infty} z^n \cdot p_n = \lambda \cdot z \cdot \sum_{n=0}^{+\infty} z^n \cdot p_n + \frac{\mu}{z} \cdot \sum_{n=0}^{+\infty} z^{n+1} \cdot p_{n+1}$$

$$(\lambda+\mu) \cdot \mathbf{P}(z) - \mu \cdot p_0 = \lambda \cdot z \cdot \mathbf{P}(z) + \frac{\mu}{z} \cdot [\mathbf{P}(z) - p_0]$$

$$(\lambda+\mu) \cdot z \cdot \mathbf{P}(z) - \mu \cdot z \cdot p_0 = \lambda \cdot z^2 \cdot \mathbf{P}(z) + \mu \cdot [\mathbf{P}(z) - p_0]$$

We rearrange the terms and obtain:

$$\mathbf{P}(z) = \frac{\mu \cdot p_0 \cdot (z-1)}{(\lambda+\mu) \cdot z - \lambda \cdot z^2 - \mu} = \frac{\mu \cdot p_0 \cdot (z-1)}{\mu \cdot (z-1) - \lambda \cdot z \cdot (z-1)} = \frac{\mu \cdot p_0}{\mu - \lambda \cdot z} = \frac{p_0}{1 - \rho \cdot z}$$

The latter depends on p_0, which is unknown and can be set by imposing the normalization condition. From $\mathbf{P}(z) \triangleq E[z^N] = \sum_{k=0}^{+\infty} z^k \cdot p_k$ we obtain that $\mathbf{P}(1) \triangleq E[1^N] = \sum_{k=0}^{+\infty} 1^k \cdot p_k = 1$, which yields $p_0 = 1 - \rho$, hence:

$$\mathbf{P}(z) = \frac{1-\rho}{1-\rho \cdot z}$$

Note that the above expression can be anti-transformed (this is because this case is particularly simple). In fact, from $\mathbf{P}(z) \triangleq E[z^N] = \sum_{k=0}^{+\infty} z^k \cdot p_k = \frac{1-\rho}{1-\rho \cdot z} = (1-\rho) \cdot \sum_{k=0}^{+\infty} (\rho \cdot z)^k$ we immediately obtain that $p_k = (1-\rho) \cdot \rho^k$, which we already knew.

However, once you have $\mathbf{P}(z)$, you can compute average performance indexes **without** anti-transforming it, by only using the well-known properties of the PGF:

- Mean number of jobs: $E[N] = \frac{d}{dz}\mathbf{P}(z)|_{z=1}$. In this case, we get: $E[N] = \frac{d}{dz}\mathbf{P}(z)|_{z=1} = \left.\frac{\rho\cdot(1-\rho)}{(1-\rho\cdot z)^2}\right|_{z=1} = \frac{\rho}{1-\rho}$. From the latter using simple algebra, one can compute the missing mean performance indexes $E[N_q]$, $E[R]$, $E[W]$.
- Mean squared number of jobs: $E[N^2] = \left[\frac{d^2}{dz^2}\mathbf{P}(z) + \frac{d}{dz}\mathbf{P}(z)\right]_{z=1}$.

If needed, one can also compute **some SS probabilities** by deriving the PGF. In fact, it is:

- $p_0 = \lim_{z\to 0}\mathbf{P}(z)$
- $p_k = \frac{\mathbf{P}^{(k)}(0)}{k!} = \frac{1}{k!}\cdot\left.\frac{d^k}{dz^k}\mathbf{P}(z)\right|_{z=0}$

Therefore, at least the *first* few SS probabilities (often the most relevant) can be easily computed.

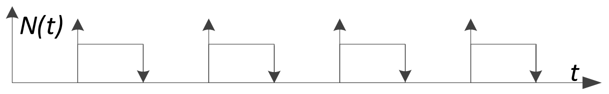
3.3.3 Arrival-time and random-observer probabilities

The SS probabilities that we have computed so far are those that a **random observer** would observe. In other words, if anyone looks at the system at a **random time** (in the steady state), p_n is the probability that she will observe n jobs in the system.

There is another important probability, which is the one **seen by an arriving job.** We call it **arrival-time** or **tagged-job SS probability**, to distinguish it from the random observer's one, and denote it with r_n. In general, the two SS probabilities are different. We show this via a simple yet illuminating example.

Consider a queueing system with **constant** interarrival times, equal to 2s, and **constant service times** equal to 1s (a D/D/1 system). Such a system is always in a steady state, being deterministic.

A trajectory of this system is the following:



A **random observer** will observe:

- **One job** in the system, half of the time
- **Zero jobs** in the system, half of the time.

Hence it is $p_0 = p_1 = 1/2$. However, an arriving job **always finds the system empty**, hence it is $r_0 = 1$, and $r_j = 0,\quad j > 0$. Therefore, it is in general $r_n \neq p_n$.

■

In a (generic) birth-death system with exponential interarrival times, arrival-time probabilities can be found using **Bayes' theorem**, as follows. Define:

$$r_n(t) = \lim_{\Delta t\to 0} P\{N(t) = n | A(t, t+\Delta t)\}$$

Where $A(t, t+\Delta t)$ means that there is an arrival in $[t, t+\Delta t)$[3]. We develop the computations as:

$$r_n(t) = \lim_{\Delta t \to 0} P\{N(t) = n | A(t, t+\Delta t)\}$$

$$= \lim_{\Delta t \to 0} \frac{P\{N(t) = n, A(t, t+\Delta t)\}}{P\{A(t, t+\Delta t)\}}$$

$$= \lim_{\Delta t \to 0} \frac{P\{A(t, t+\Delta t) | N(t) = n\} \cdot P\{N(t) = n\}}{\sum_{k=0}^{+\infty} P\{A(t, t+\Delta t) | N(t) = k\} \cdot P\{N(t) = k\}}$$

$$= \lim_{\Delta t \to 0} \frac{P\{A(t, t+\Delta t) | N(t) = n\} \cdot P\{N(t) = n\}}{\sum_{k=0}^{+\infty} P\{A(t, t+\Delta t) | N(t) = k\} \cdot P\{N(t) = k\}}$$

$$= \lim_{\Delta t \to 0} \frac{\left[\lambda_n + \frac{o(\Delta t)}{\Delta t}\right] \cdot p_n(t)}{\sum_{k=0}^{+\infty} \left[\lambda_k + \frac{o(\Delta t)}{\Delta t}\right] \cdot p_k(t)}$$

$$= \frac{\lambda_n \cdot p_n(t)}{\sum_{k=0}^{+\infty} \lambda_k \cdot p_k(t)}$$

The probability that the next arrival occurs by $t + \Delta t$ when the system is in state n is:

$$P\{A(t, t+\Delta t) | N(t) = n\} = 1 - e^{-\lambda_n \cdot \Delta t}$$

However, since we are letting $\Delta t \to 0$, we substitute the expansion of the exponential:

$$e^{-\lambda_n \cdot \Delta t} = \sum_{j=0}^{+\infty} \frac{(-\lambda_n \cdot \Delta t)^j}{j!} = 1 - \lambda_n \cdot \Delta t + o(\Delta t)$$

Hence:

$$1 - e^{-\lambda_n \cdot \Delta t} = 1 - \left(1 - \lambda_n \cdot \Delta t + o(\Delta t)\right)$$
$$= \lambda_n \cdot \Delta t + o(\Delta t)$$

The above equalities hold $\forall t$, hence it holds also at the steady state. At the steady state, we get:

$$r_n = \lim_{t \to +\infty} r_n(t) = \frac{\lambda_n \cdot p_n}{\sum_{k=0}^{+\infty} \lambda_k \cdot p_k} = \frac{\lambda_n}{\bar{\lambda}} \cdot p_n$$

where $\bar{\lambda}$ is the mean arrival rate.

Note that, when $\lambda_n = \lambda \quad \forall n$, and only under that condition, it is $r_n = p_n$. Systems where $r_n = p_n$ are said to possess **the PASTA property** (Poisson Arrivals See Time Average). Nevertheless, there is still a **conceptual** difference between the two distributions, even when they have the same values, hence we will take some care to use the correct symbol whenever possible.

Tagged-job probabilities are useful to compute the **distribution** of the response and waiting times. In fact, the response time of a job within a system does not start **at a random time instant**: it starts **when that job arrives**, hence its distribution must be related to probabilities r_n.

3.3.4 Distribution of response and waiting times

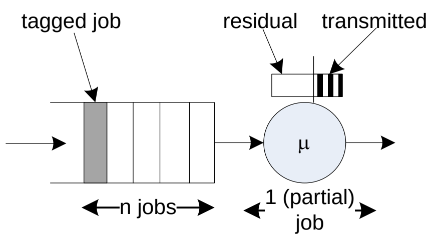
We have already computed the **mean** response and waiting time, $E[R], E[W]$, through Little's Law. We now show how to compute the **distribution** of the response and waiting times, i.e. $F_R(x), F_W(x)$. We start with the former.

[3] Recall that, with continuous RVs (such as arrival times), we cannot posit that anything occurs "at" time t with non-null probability. However, we can have it occur within an interval $[t, t+\Delta t)$, and then let $\Delta t \to 0$.

Suppose a job arrives in the system at time t. Let n be the number of jobs *already* in the system at time t^-. We know that – at the steady state – the probability to have n jobs in the system at the time of an arrival is r_n (which may or may not be equal to p_n in general – it is in this case).

Now, the time that the tagged job will have to spend in the system (i.e., its response time) is composed of:

- The **residual service time** of the one job being served at time t.
- The **sum of the service times** of all the other n jobs, including the tagged one.

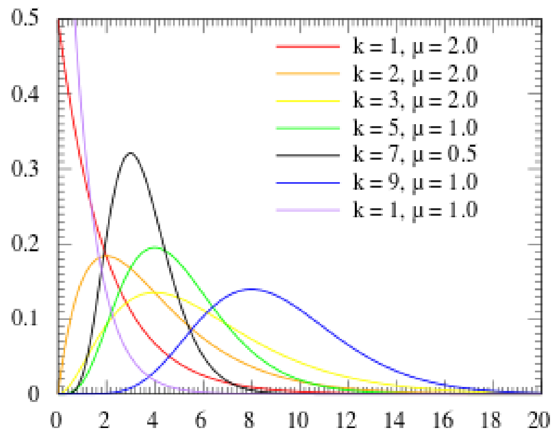


However, since service times are exponential (hence memoryless), the residual service time has the same distribution as the service time: $P\{t_s > x + y | t_s > x\} = P\{t_s > y\}$. Therefore, the response time is the **sum of the service times of $n + 1$ jobs**, **<u>if</u>** there are n jobs in the system at the time of arrival. This distribution of the sum of IID exponential is very common, and it is called **Erlang distribution**. The CDF of an n-stage Erlang is:

$$F_n(t) = 1 - \sum_{k=0}^{n-1} e^{-\mu t} \frac{(\mu t)^k}{k!}$$

The PDF is $f_n(t) = e^{-\mu t} \cdot \mu \cdot \frac{(\mu t)^{n-1}}{(n-1)!}$

Let us take a look at what an Erlang PDF looks like: When $n = 1$ it is an exponential. This is clear both intuitively and from the formulas. When $n > 1$ it starts **peaking** and then goes down. When n gets large, due to the CLT, it looks like a Normal.



We can compute $E[S_n]$ and $Var(S_n)$ leveraging **additivity** of mean values and **independence** (recall that the Erlang is the sum of n independent exponentials). Therefore, we get $E[S_n] = \frac{n}{\mu}$, $Var(S_n) = \frac{n}{\mu^2}$. From these we obtain $CoV(S_n) = \frac{\sqrt{Var(S_n)}}{E[S_n]} = \frac{1}{\sqrt{n}}$.

In other words, the CoV of an n-stage Erlang is **smaller than one** (and, specifically, it is smaller than an exponential's), and it gets smaller with n (which is again a consequence of the CLT).

We now know that the sum of $n+1$ IID exponentials with a rate μ is an $(n+1)$-stage Erlang distribution, i.e., $f_{n+1}(x)=\mu\cdot e^{-\mu\cdot x}\cdot\frac{(\mu\cdot x)^n}{n!}$. This is the PDF of RV R, **given that** there are n jobs in the system. Therefore, we can compute $f_R(x)$ using Total Probability as follows:

$$\begin{aligned}
f_R(x) &= \sum_{n=0}^{+\infty} f_{n+1}(x)\cdot r_n \\
&= \sum_{n=0}^{+\infty} \mu\cdot e^{-\mu\cdot x}\cdot\frac{(\mu\cdot x)^n}{n!}\cdot(1-\rho)\cdot\rho^n \\
&= \mu\cdot(1-\rho)\cdot e^{-\mu\cdot x}\cdot\sum_{n=0}^{+\infty}\frac{(\mu\cdot x\cdot\rho)^n}{n!} \\
&= \mu\cdot(1-\rho)\cdot e^{-\mu\cdot x}\cdot e^{\mu\rho x} \\
&= \mu\cdot(1-\rho)\cdot e^{-\mu(1-\rho)\cdot x} \\
&= (\mu-\lambda)\cdot e^{-(\mu-\lambda)\cdot x} \\
&= \frac{1}{E[R]}\cdot e^{-x/E[R]}
\end{aligned}$$

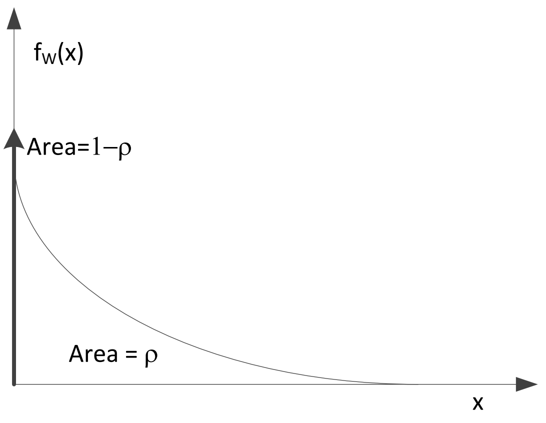
This is **an exponential distribution**, hence $F_R(x)=1-e^{-x/E[R]}=1-e^{-(\mu-\lambda)\cdot x}$.

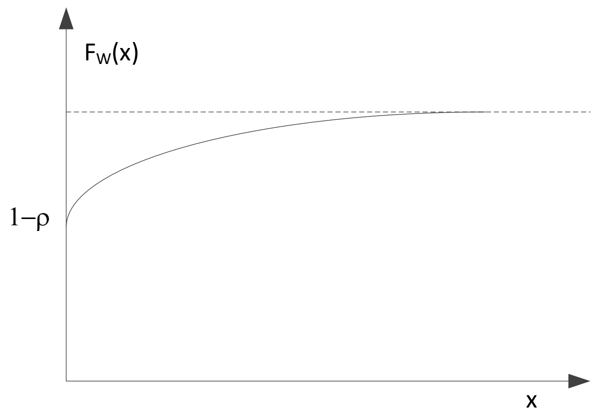
As far as the distribution of the **waiting time W is concerned**, things are only slightly different. We can repeat the same reasoning as for the **response time**; however, we need a little care: in fact, there is a possibility that the system may be **empty** at the time of arrival, hence the waiting time will be **zero** in that case. There is a non-null probability that the waiting time be **null,** equal to the probability of finding the system empty, i.e., $r_0=1-\rho$. Therefore, the PDF of the waiting time will be:

- **Zero**, with a non-null probability $r_0=1-\rho$.
- Equal to an n-stage Erlang distribution (mind the difference: it was $n+1$ for the response time) with a probability r_n, $n\geq 1$.

Note that the fact that $F_W(0)=P\{W=0\}=r_0>0$ implies that there is a discontinuity at $F_W(0)$. The PDF will have a Dirac's delta in zero. This said, we can use Total Probability again and compute $f_W(x)$:

$$
\begin{aligned}
f_W(x) &= (1-\rho)\cdot\delta(x) + \sum_{n=1}^{+\infty} f_n(x)\cdot r_n \\
&= (1-\rho)\cdot\delta(x) + \sum_{n=1}^{+\infty} \mu\cdot e^{-\mu\cdot x}\cdot\frac{(\mu\cdot x)^{n-1}}{(n-1)!}\cdot(1-\rho)\cdot\rho^n \\
&= (1-\rho)\cdot\delta(x) + \mu\cdot e^{-\mu\cdot x}\cdot(1-\rho)\cdot\rho\cdot\sum_{n=1}^{+\infty}\frac{(\mu\cdot x)^{n-1}}{(n-1)!}\cdot\rho^{n-1} \\
&= (1-\rho)\cdot\delta(x) + \rho\cdot\mu\cdot(1-\rho)\cdot e^{-\mu(1-\rho)\cdot x} \\
&= (1-\rho)\cdot\delta(x) + \rho\cdot\frac{1}{E[R]}\cdot e^{-x/E[R]}
\end{aligned}
$$





From the above we can easily obtain the distribution $F_W(x)$, which is such that:

$$F_W(x) = \begin{cases} 1-\rho & x = 0 \\ (1-\rho)+\rho\cdot\left(1-e^{-x/E[R]}\right) & x > 0\text{'} \end{cases}$$

Which boils down to: $F_W(x) = 1-\rho\cdot e^{-x/E[R]} \quad x \geq 0$

Now that we have **distributions,** we can solve problems with **percentile constraints**: select the server speed so that the 95th percentile of the response (waiting) time is below x, etc. You just need to solve $F_R(x) = 0.95$ and obtain *x* as a solution.

3.3.5 Exercise

Your boss says that the rate of contacts to your company's website is going to double next month. She wants you to add more capacity to your website, so that:

a) The mean response time will remain the same;
b) The 99th percentile of the response time will remain the same.

Assuming your web server can be modeled via an M/M/1 system, this boils down to increasing its service rate to match the requirements. Call λ, μ the *current* arrival rate and service rate of your website, and let $\lambda' = 2\lambda, \mu'$ be the *new* arrival and service rates. The equations are:

For case a), i.e. how to keep the mean response time constant:

$$E[R'] = E[R]$$
$$\frac{1}{\mu' - 2\lambda} = \frac{1}{\mu - \lambda}$$
$$\mu' = \mu + \lambda$$

For case b) we have to consider that $F_R(x) = 1 - e^{-x/E[R]} = 1 - e^{-(\mu-\lambda)\cdot x}$, and that the 99th percentile of the *current* response time is the solution $x_{.01}$ to $F_R(x_{.01}) = 1 - e^{-(\mu-\lambda)\cdot x_{.01}} = 0.99$, i.e.

$$\frac{1}{100} = e^{-(\mu-\lambda)\cdot x_{.01}}$$
$$x_{.01} = \frac{\log 100}{\mu - \lambda}$$

In the *new* configuration, the 99th percentile of the response time will be $x_{.01}' = \frac{\log 100}{\mu' - 2\lambda}$, which again means that $\mu' = \mu + \lambda$.

Those who instinctively thought $\mu' = 2\mu$ can check for themselves that doubling *both* the arrival and the service rate leads to *halving* the (mean or 99th percentile) response time.

As an aside, case b) shows that setting $\mu' = \mu + \lambda$ allows you to obtain exactly the same distribution of the response times: in fact, you can match any percentile through the same trick – you will just get a different argument for the logarithm in the above expressions.

3.4 M/M/C systems

So far, we have discussed systems with **only one server**. A frequent case is that of a queue which is drained by **more than one server**:

- Airport check-in, where the single queue is served by several desks;
- One queue of transactions waiting to be processed by several redundant disks;
- A server farm, with requests being routed to the first idle server;
- Networks with C parallel links bundled together to increase the capacity.

If the C servers are **equivalent** (meaning that they have the same rate μ) and everything else stays the same (i.e., exponential arrivals, exponential service times, infinite memory), then the system is called an M/M/C one. In an M/M/C system, an arriving job is sent to **an idle server** at random, if one such server exists, otherwise it queues up.

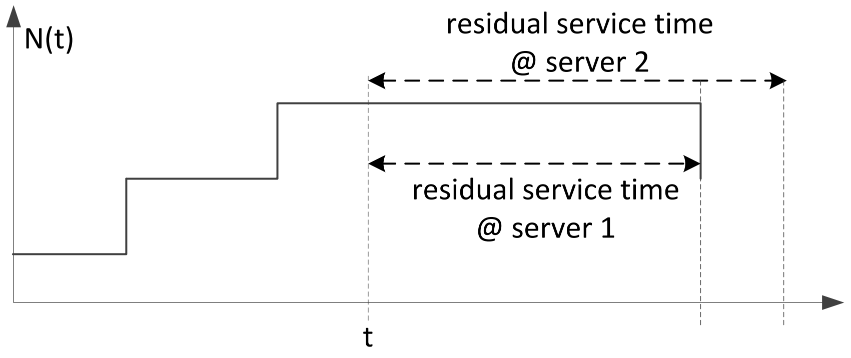
We want to derive the **SS probabilities** for an M/M/C system. We start with $C = 2$, and then we generalize to an arbitrary value of C.

It is quite easy to derive the **transition rates** in this system[4]. In fact:

[4] A formal derivation of CK equations for an M/M/2 system can be found in the Appendix.

- Arcs going to the **right** are the same (with a rate $\lambda_n = \lambda$);
- Arcs going to the **left** will have to consider that (at least when $n \geq 2$) **both servers are busy**.

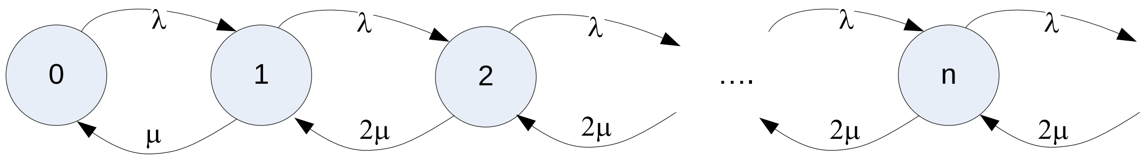
Assume you are at time t, in a trajectory whose value is $N(t) = n$, and $n \geq 2$. Both servers are busy, and a job may depart in the future from either of them (but not from both: as for the previous cases, we neglect the occurrence of **simultaneous events**, since it has **a negligible probability**).



When will the next downward step in the trajectory occur? When the **smallest of the two residual service times** (i.e., those of server 1 and of server 2) expires. However, we know that the service times at both servers are **IID exponentials with the same rate μ**, therefore:

- **"residual"** service times are themselves IID exponentials with a rate μ, since exponentials are memoryless. The adjective "residual" is thus immaterial.
- The **minimum** of two IID exponentials is an exponential with **double their rate**.

Therefore, arcs going to the left, out of states $n \geq 2$, will have a transition rate equal to 2μ.
What about the arc related to the service rate **in state 1**? In that case the transition rate to the left is still μ, since **only one server is busy**. The CTMC will then look as follows:



We stress again that the fact that the rate of departure is 2μ **does not mean that two jobs may leave simultaneously**: transitions are still of the **nearest-neighbor** type, and non-nearest-neighbor transitions have negligible probability.

Hence, we get a CTMC with $\lambda_n = \lambda$, and $\mu_n = \begin{cases} \mu & n = 1 \\ 2\mu & n > 1 \end{cases}$.

In this case, **service rates are load-dependent.**
Given the above diagram, we can easily write down both global and local equilibrium equations at the steady state, through visual inspection:

<u>Global equilibrium equations</u>

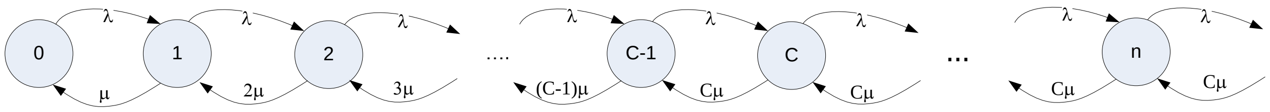
$$\lambda \cdot p_0 = \mu \cdot p_1$$
$$(\lambda + \mu) \cdot p_1 = \lambda \cdot p_0 + 2\mu \cdot p_2$$
$$(\lambda + 2\mu) \cdot p_n = \lambda \cdot p_{n-1} + 2\mu \cdot p_{n+1}, \quad n \geq 2$$

<u>Local equilibrium equations</u>

$$\lambda \cdot p_0 = \mu \cdot p_1$$
$$\lambda \cdot p_n = 2\mu \cdot p_{n+1}, \quad n \geq 2$$

Before solving the above system, we generalize the above reasoning to **an arbitrary number of servers $\boldsymbol{C}$.** We can write down the CTMC keeping in mind that:

- Arrival rates are constant;
- The service rates will be $\mu_n = \begin{cases} n \cdot \mu & n \leq C \\ C \cdot \mu & n \geq C \end{cases} = \min(C, n) \cdot \mu$



In order to compute the SS probabilities and the stability condition, we can specialize the general formulas that hold for any birth-death system with nearest-neighbor transitions, i.e.:

$$\begin{cases} p_n = \displaystyle\prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} p_0, \quad n \geq 1 \\ \displaystyle\sum_{n=0}^{+\infty} p_n = 1 \end{cases}$$

Define $u = \lambda/\mu$. Note that, with this system, **$\boldsymbol{\lambda/\mu}$ is not the utilization**. Symbol ρ denotes the utilization, hence we need a different symbol to avoid confusion. To write the above formulas, we distinguish the two cases: $n \leq C,\ n \geq C$.

- When $n \leq C$, we have

$$p_n = \frac{\lambda^n}{\mu \cdot 2\mu \cdot 3\mu \cdot \ldots \cdot n \cdot \mu} \cdot p_0 = \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{1}{n!} \cdot p_0 = \frac{u^n}{n!} \cdot p_0$$

- When $n \geq C$, we have

$$\begin{aligned} p_n &= \frac{\lambda^n}{(\mu \cdot 2\mu \cdot 3\mu \cdot \ldots \cdot C \cdot \mu) \cdot (C \cdot \mu \cdot \ldots \cdot C \cdot \mu)} \cdot p_0 \\ &= \left(\frac{\lambda}{\mu}\right)^C \cdot \frac{1}{C!} \cdot \left(\frac{\lambda}{C \cdot \mu}\right)^{n-C} \cdot p_0 \\ &= \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{1}{C!} \cdot \frac{1}{C^{n-C}} \cdot p_0 \\ &= \frac{u^n}{C^{n-C} \cdot C!} \cdot p_0 \end{aligned}$$

This said, the normalization condition becomes (recall that both expressions are correct when $n = C$, but we should not count equality twice):

$$p_0 \cdot \left[\sum_{n=0}^{C-1} \frac{u^n}{n!} + \sum_{n=C}^{+\infty} \frac{u^n}{C^{n-C} \cdot C!} \right] = 1$$

The **first summation includes a finite number of terms**, hence it is always finite. Thus, the stability condition is the one under which **the second summation is finite**, i.e.:

$$\sum_{n=C}^{+\infty} \frac{u^n}{C^{n-C} \cdot C!} = \frac{u^C}{C!} \cdot \sum_{n=C}^{+\infty} \frac{u^{n-C}}{C^{n-C}} = \frac{u^C}{C!} \cdot \sum_{j=0}^{+\infty} \left(\frac{u}{C}\right)^j$$

The condition is clearly $u < C$, i.e. $\lambda < C \cdot \mu$. This could be expected, since λ is the arrival rate, and $C \cdot \mu$ is the service rate at high loads. If the services cannot keep up with the arrivals when the load is high, then the state is bound to diverge. In fact, in this case it is $\rho = \frac{\lambda}{C \cdot \mu}$ (we will see why later on), hence the stability condition is still $\rho < 1$. This gives us an interesting insight on stability: what actually matters for stability is the balance of the CTMC rates "**to the right**", i.e. the fact that $\lambda_n < \mu_n$ $\forall n \geq n_0$. What happens "close to state 0" does not affect stability, although it still influences performance indexes. Under the above condition, the infinite sum converges to $1/(1-\rho)$, hence:

$$p_0 = \frac{1}{\sum_{k=0}^{C-1} \frac{u^k}{k!} + \frac{u^C}{C!} \cdot \frac{1}{1-\rho}}$$

and

$$p_n = \begin{cases} p_0 \cdot \frac{u^n}{n!} & n \leq C \\ p_0 \cdot \frac{u^n}{C^{n-C} \cdot C!} & n \geq C \end{cases}$$

Now that we have the SS probabilities, we can compute all the **performance indexes**. We start with $E[N_q]$, which is easier.

$$\begin{aligned} E[N_q] &= \sum_{n=C+1}^{+\infty} (n-C) \cdot p_n = \sum_{n=C+1}^{+\infty} (n-C) \cdot \frac{u^n}{C!\,C^{n-C}} \cdot p_0 = \\ &= \frac{u^C}{C!} \cdot p_0 \cdot \sum_{n=C+1}^{+\infty} (n-C) \cdot \frac{u^{n-C}}{C^{n-C}} \\ &= \frac{u^C}{C!} \cdot p_0 \cdot \sum_{n=1}^{+\infty} n \cdot \rho^n \\ &= \frac{u^C}{C!} \cdot p_0 \cdot \frac{\rho}{(1-\rho)^2} \end{aligned}$$

From the above, using Little's Law, we get:

$$E[W] = \frac{E[N_q]}{\lambda} = \frac{u^C}{C!} \cdot p_0 \cdot \frac{1/(C \cdot \mu)}{(1-\rho)^2}$$

In order to compute $E[R]$, we only need to sum up a mean service time to $E[W]$, i.e.

$$E[R] = E[W] + E[t_s] = \frac{u^C}{C!} \cdot p_0 \cdot \frac{1/(C \cdot \mu)}{(1-\rho)^2} + \frac{1}{\mu}$$

And from the latter, applying Little's Law backwards, we can get $E[N]$:

$$E[N] = \frac{u^C}{C!} \cdot p_0 \cdot \frac{\rho}{(1-\rho)^2} + \frac{\lambda}{\mu} = E[N_q] + C \cdot \rho$$

Again, it is $E[N] = E[N_q] + E[N_s]$, i.e. the mean number in the queue plus the mean number being served, which is $C \cdot \rho$.

The M/M/C system has the PASTA property, so it is $r_n = p_n$. As far as the throughput is concerned, we get:

$$\gamma = \sum_{n=1}^{+\infty} \mu_n \cdot p_n = \sum_{n=1}^{C} n \cdot \mu \cdot p_n + \sum_{n=C+1}^{+\infty} C \cdot \mu \cdot p_n$$

The computations are not straightforward. However, for pretty obvious physical considerations, the only possibility is that $\gamma = \lambda$, since the system is in a steady state, and what gets in must get out.

Finally, it is interesting to compute the **mean number of busy servers**. Call c the RV "number of busy servers". The expression is the following:

$$E[c] = \sum_{n=1}^{+\infty} \min(n, C) \cdot p_n = \sum_{n=1}^{C} n \cdot p_n + \sum_{n=C+1}^{+\infty} C \cdot p_n$$

Again, the computations are not straightforward, and – again – we can find a quicker workaround, which relies on Little's Law.

Apply Little's Law to the **sub-system consisting of the *C* servers**, and get the following:

- The average response time of the system is $E[t_s] = \frac{1}{\mu}$;
- The input-output rate at the steady state is $\gamma = \lambda$.

Thus, $E[c] = \lambda \cdot E[t_s] = \frac{\lambda}{\mu} = u$.

This justifies the fact that $\rho = \frac{\lambda}{C \cdot \mu} = \frac{E[c]}{C}$. ρ represents the utilization, hence it is the average **fraction of busy servers**. As this fraction approaches one, the system becomes **unstable**. Note that the above definition also holds for an M/M/1 system. SS probabilities can be rewritten using ρ instead of u via a few algebraic computations:

$$p_n = \begin{cases} p_0 \cdot \dfrac{(C \cdot \rho)^n}{n!} & n \le C \\ p_0 \cdot \dfrac{C^C \cdot \rho^n}{C!} & n \ge C \end{cases}$$

3.4.1 Exercise: comparison of the response time for queueing systems

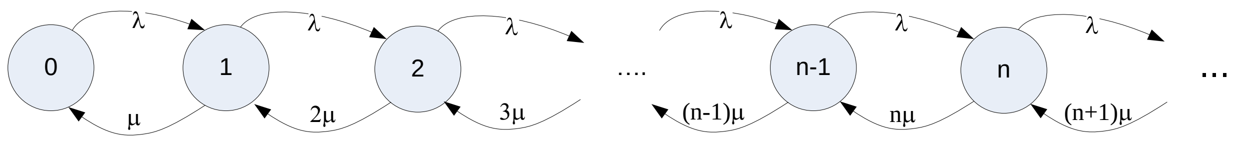
Suppose that you have to build up a computer system that processes transactions. Transactions arrive at a rate λ. You can buy a total computing power equal to $C \cdot \mu$. You are requested to choose among three alternative designs:

1) A system where the incoming traffic is split into C Poisson processes[5], each of which is sent to an M/M/1 system whose service rate is μ (**load balancing**).
2) An M/M/C system, i.e., where a single queue exists and each of the identical C servers have a service rate equal to μ.
3) An M/M/1 system with a more powerful server, whose service rate is $C \cdot \mu$.

Which of the three will yield the smallest $E[R]$? What is the rationale behind the answer?

3.4.2 Delay centers: M/M/∞ systems

If we take the limit $C \to +\infty$, we observe a peculiar behavior: since the number of severs is infinite, **every job** that arrives will find an available server, hence there will be **no queueing**. The CTMC is the following:



The local equilibrium equations are: $p_{n+1} = \frac{\lambda}{(n+1)\mu} \cdot p_n$.

The SS probabilities can be written quite easily by specializing the general formula:

$$\begin{cases} p_n = \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} p_0, & n \geq 1 \\ \sum_{n=0}^{+\infty} p_n = 1 \end{cases}$$, with $\lambda_n = \lambda$, $\mu_n = n \cdot \mu$.

We readily obtain $p_n = \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{1}{n!} \cdot p_0$, $n \geq 0$. The stability condition is $p_0 \cdot \sum_{n=0}^{+\infty} \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{1}{n!} = 1$. However, since $\sum_{n=0}^{+\infty} \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{1}{n!} = e^{\lambda/\mu}$, then this system **is always stable**, regardless of the values of λ, μ. This means that $p_n = e^{-\lambda/\mu} \cdot \frac{(\lambda/\mu)^n}{n!}$, $n \geq 0$. The SS probabilities have a **Poisson distribution**.

M/M/∞ systems are called **delay centers**. They model cases when arriving jobs are **delayed** by a random exponential time before being forwarded (e.g., a user think time between two successive page requests).

[5] If a Poisson process with a rate λ is **split probabilistically**, i.e. jobs are routed to server j with probability π_j, the arrivals at each server will themselves be independent Poisson processes, with rates $\lambda_j = \lambda \cdot \pi_j$.

For an M/M/∞ system we can easily compute the only relevant performance metric, i.e. $E[N] = E[C]$. By recalling the properties of Poisson distributions, we easily get $E[N] = E[C] = \frac{\lambda}{\mu}$.

3.4.3 Models, CTMCs and performance indexes

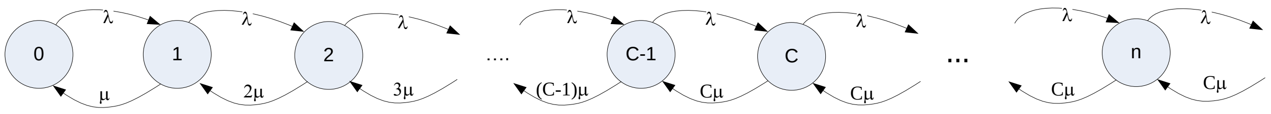
It is now time to observe that **very different models** may admit the same SS probabilities. It is important to understand that **these similarities may not extend to all the performance indexes,** hence some care must be taken when doing the computations. Here is an example.

Example

Consider the following two systems:

a) An M/M/C system;
b) An M/M/1 system with **load-dependent service-rate**. It is $\mu_n = \min(n, C) \cdot \mu$, i.e. the service rate increases with the load, but caps to a maximum of $C \cdot \mu$.

It is quite clear that both systems have the **same CTMC**, hence they will have the **same SS probabilities**. Since they do, it will be $E[N^{(a)}] = E[N^{(b)}]$. By Little, since $\gamma^{(a)} = \gamma^{(b)}$, it will also be $E[R^{(a)}] = E[R^{(b)}]$. Does this imply that **all the performance indexes are equal**?



No, it does not. In fact, we have:

$$E[N_q^{(a)}] = \sum_{n=C+1}^{+\infty} (n - C) \cdot p_n$$

whereas it is:

$$E[N_q^{(b)}] = \sum_{n=2}^{+\infty} (n - 1) \cdot p_n$$

Hence, in general $E[N_q^{(a)}] \neq E[N_q^{(b)}]$. The mean waiting time will thus be different as well.

■

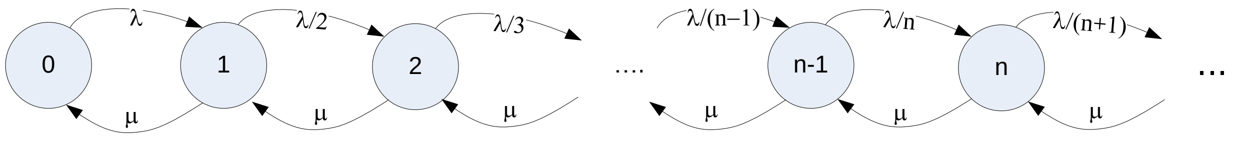
This is to remind to ourselves that one is never too careful around these systems, and that **not everything that you need to know can be found in the CTMC**.

3.5 Discouraged arrivals

Consider the (likely) case of a system where **the more jobs are in the queue, the fewer will join**. This happens, for instance, in museums or supermarket tills, where the fact that the queue is long **discourages** other users from joining it.

A common model is one where $\lambda_n = \lambda/(n+1)$, and the service rate is constant, $\mu_n = \mu$.

The CTMC is the following:



And the local equilibrium equations are $p_{n+1} = \frac{\lambda}{(n+1)\mu} \cdot p_n$, **which are the same as a delay center's**.

Therefore, we quickly get the following:

a) The system is always stable, whatever the values of λ, μ;

b) It is $p_n = e^{-\lambda/\mu} \cdot \frac{(\lambda/\mu)^n}{n!}$, $n \geq 0$. The SS probabilities have a **Poisson distribution**;

c) $E[N] = \frac{\lambda}{\mu}$.

However, the similarities between a discouraged-arrivals system and a delay center end here. In fact:

- Queueing occurs in this system, but not in a delay center. In fact, we have $E[N_q] = E[N] - (1 - p_0) = \frac{\lambda}{\mu} - \left(1 - e^{-\lambda/\mu}\right)$.
- The average arrival rate is different in the two systems, hence everything that is computed through Little (notably, $E[R]$) will be different.
- This system is non-PASTA, since the arrival rates are not constant, whereas the other is. Therefore, here we have $r_n \neq p_n$.

This means that we have to compute the average arrival rate, which is defined as $\bar{\lambda} = \sum_{n=0}^{+\infty} \lambda_n \cdot p_n$. However, we know that it must also be $\bar{\lambda} = \gamma \triangleq \sum_{n=1}^{+\infty} \mu_n \cdot p_n$. Since $\mu_n = \mu$, we obtain that $\bar{\lambda} = \mu \cdot \left(1 - p_0\right) = \mu \cdot \left(1 - e^{-\lambda/\mu}\right)$ with considerably fewer computations. Therefore, we get:

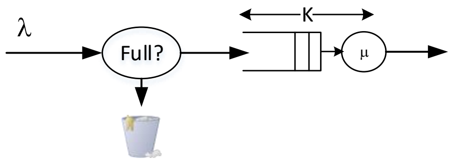
$$E[R] = \frac{E[N]}{\bar{\lambda}} = \frac{\lambda}{\mu^2 \cdot (1 - e^{-\lambda/\mu})}$$

$$E[W] = \frac{E[N_q]}{\bar{\lambda}} = \left[\frac{\lambda}{\mu} - \left(1 - e^{-\lambda/\mu}\right)\right] \cdot \frac{1}{\mu(1 - e^{-\lambda/\mu})} = \frac{\lambda}{\mu^2(1 - e^{-\lambda/\mu})} - \frac{1}{\mu} = E[R] - E[t_s]$$

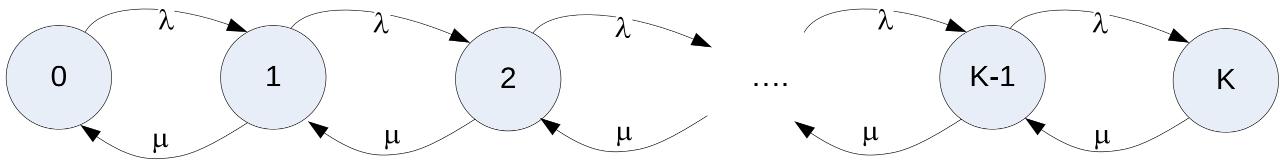
$$r_n = \frac{\lambda_n}{\bar{\lambda}} \cdot p_n = \frac{\lambda}{(n+1) \cdot \mu \cdot (1 - e^{-\lambda/\mu})} \cdot e^{-\lambda/\mu} \cdot \frac{(\lambda/\mu)^n}{n!} = \frac{p_{n+1}}{1 - e^{-\lambda/\mu}} = \frac{p_{n+1}}{1 - p_0}$$

3.6 Systems with finite memory: M/M/1/K

Infinite queues are often a useful abstraction. Real systems, however, have **finite queues**, hence they have **losses due to overflow**. This means that, in general, the case that $\gamma < \lambda$ may be given, since some of the jobs **will not enter the system**.



Call K the system **memory**, i.e., the maximum number of jobs that are allowed in the system at any time. When the system is in state K, the queue is **full**. Any arrival occurring when the system is in that state will be dropped. If the arrival and service rates are constant, the CTMC will be this:



From the latter, we can easily infer the local equilibrium equations $\lambda \cdot p_n = \mu \cdot p_{n+1}$, $0 \le n < K$, hence we will have $p_n = \left(\frac{\lambda}{\mu}\right)^n \cdot p_0$, $0 \le n \le K$.

The normalization condition now involves **a finite sum**, hence the system is **always positive recurrent** (all systems with finite states are), whether $\lambda < \mu$ or not. Call $u = \lambda/\mu$. We obtain:

$\begin{cases} p_n = u^n \cdot p_0, & n \ge 0 \\ p_0 \cdot \sum_{n=0}^{K} u^n = 1 \end{cases}$. However, $\sum_{n=0}^{K} u^n = \begin{cases} \frac{1-u^{K+1}}{1-u} & u \ne 1 \\ K+1 & u = 1 \end{cases}$, thus:

$$p_0 = \frac{1}{\sum_{n=0}^{K} u^n} = \begin{cases} \frac{1-u}{1-u^{K+1}} & u \ne 1 \\ \frac{1}{K+1} & u = 1 \end{cases}, \quad p_n = \begin{cases} \frac{1-u}{1-u^{K+1}} \cdot u^n & u \ne 1 \\ \frac{1}{K+1} & u = 1 \end{cases}.$$

Please do not forget to consider this case when solving classworks

The case $u = 1$, i.e. $\lambda = \mu$, is quite peculiar: in this case, in fact, the rates of left- and right-bound transitions are the same, hence all the states are equally likely.

If $u < 1$ we would expect “low” states to be observed with a larger probability than “high” ones: this is the case, in fact, since p_n is a decreasing sequence. If $u > 1$, instead, p_n is an increasing sequence, which is again expectable for the same reason.

We can compute **performance indexes** (assuming $u \ne 1$ from now on, otherwise they are trivial):

$$
\begin{aligned}
E[N] &= \sum_{n=0}^{K} n \cdot p_n = \frac{1-u}{1-u^{K+1}} \cdot \sum_{n=0}^{K} n \cdot u^n \\
&= \frac{1-u}{1-u^{K+1}} \cdot u \cdot \frac{d}{du} \sum_{n=0}^{K} u^n \\
&= \frac{1-u}{1-u^{K+1}} \cdot u \cdot \frac{d}{du} \left(\frac{1-u^{K+1}}{1-u} \right) \\
&= \frac{1-u}{1-u^{K+1}} \cdot u \cdot \frac{-(K+1)u^K \cdot (1-u) + 1 - u^{K+1}}{(1-u)^2} \\
&= \frac{u}{1-u} \cdot \frac{-(K+1)u^K \cdot (1-u) + 1 - u^{K+1}}{1-u^{K+1}} \\
&= \frac{u}{1-u} \cdot \left(1 - \frac{(K+1)u^K \cdot (1-u)}{1-u^{K+1}} \right) \\
&= \frac{u}{1-u} - \frac{(K+1)u^{K+1}}{1-u^{K+1}}
\end{aligned}
$$

Assume $u < 1$: the above formula is similar to the M/M/1 system's, but it has a negative offset to compensate for the missing states. In this case $\lim_{K \to +\infty} E[N] = \frac{u}{1-u}$, which is in fact what should happen in a stable M/M/1 system. On the other hand, if $u > 1$, as K increases, we get the following:

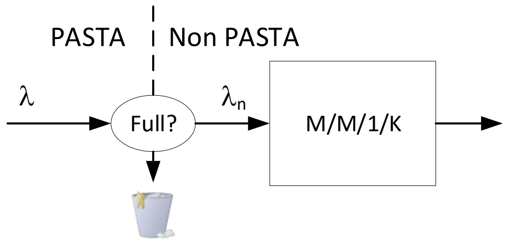
$$
E[N] = \frac{u}{1-u} - \frac{(K+1)u^{K+1}}{1-u^{K+1}} = \frac{(K+1)}{1 - \frac{1}{u^{K+1}}} - \frac{u}{u-1} \simeq K - \frac{1}{u-1}
$$

Hence, the number of jobs will be **close to K in any case**, since the transitions to the right occur more often than those to the left.

The mean number of jobs in the queue is:

$$
\begin{aligned}
E[N_q] &= E[N] - (1 - p_0) = \frac{u}{1-u} - \frac{(K+1)u^{K+1}}{1-u^{K+1}} - \left(1 - \frac{1-u}{1-u^{K+1}} \right) \\
&= \frac{u}{1-u} - \frac{K \cdot u^{K+1} + u}{1-u^{K+1}}
\end{aligned}
$$

An important performance metric of a finite-queue system is the **blocking probability** or **loss probability**, i.e. the probability that an arriving job is dropped because the system is full. This is the probability **that an arriving job** finds the system in state K. However, **to the left of the divide** this system is a **PASTA system**, since the arrival rates are independent of the state of the system (they are constant and equal to λ). Therefore, we have $\boldsymbol{p_L = p_K = \frac{1-u}{1-u^{K+1}} \cdot u^K}$. Knowing the relationship between p_L and K, u, allows one to **dimension the queue size K** based on the expected loss probability given the arrival and service rates. This should be done by solving the above equation numerically, possibly with the help of a spreadsheet.



Some care must instead be taken when computing response/waiting times through **Little's Law**, since response times are computed only for those jobs that pass the above vertical divide. For these λ_n is **not constant**. In fact, it is:

$$\lambda_n = \begin{cases} \lambda & 0 \leq n < K \\ 0 & n = K \end{cases}$$

Hence, **right of the divide,** this is a **non-PASTA system**, with $\bar{\lambda} = \sum_{n=0}^{K} \lambda_n \cdot p_n = \lambda \cdot (1 - p_K) < \lambda$.
This said, we can compute $E[R], E[W]$ using $\bar{\lambda}$ as a mean arrival rate: $E[R] = \frac{E[N]}{\bar{\lambda}}$, $E[W] = \frac{E[N_q]}{\bar{\lambda}}$.
Moreover, it is:

$$r_n = \frac{\lambda_n}{\bar{\lambda}} \cdot p_n = \begin{cases} \dfrac{p_n}{1 - p_L} & n < K \\ 0 & n = K \end{cases}$$

This has an intuitive explanation, since r_K must be equal to zero (no job can enter the system when the queue is full). Therefore, all the **other** probabilities $r_0, \ldots r_{K-1}$ **must sum to one *and* be proportional** to the related random-observer ones (the arrival rates are constant up to state $K - 1$), hence they can only be $p_n/(1 - p_L)$.
Last, but not least, we should compute the **throughput**. Due to physical reasons, the only possibility is $\gamma = \bar{\lambda} = \lambda(1 - p_L)$.

3.6.1 Adding queueing space does increase the utilization

We have discussed at the beginning that queues are there to **increase the utilization of a system**. We can now corroborate the above observation with numbers. Let us compute the utilization of an M/M/1/K as a function of K. We have:

$$\rho(K) = 1 - p_0(K) = 1 - \frac{1 - u}{1 - u^{K+1}} = u \cdot \frac{1 - u^K}{1 - u^{K+1}}$$

When $u < 1$ (i.e., $\lambda < \mu$), the above expression increases with K and $\rho(\infty) = \lim_{K \to +\infty} \rho(K) = \frac{\lambda}{\mu}$.
Hence, adding queueing **increases the system utilization**, up to a maximum achieved when the queue is infinite. For instance, an M/M/1/1 system having $\frac{\lambda}{\mu} = 0.8$ can only achieve a utilization of $\rho(1) \approx 0.44$ **just because** jobs are not allowed to queue up when the system is busy. Moreover, increasing the queue also increases the **throughput,** since it is:

$$\gamma(K) = \mu \cdot (1 - p_0(K)) = \mu \cdot \rho(K)$$

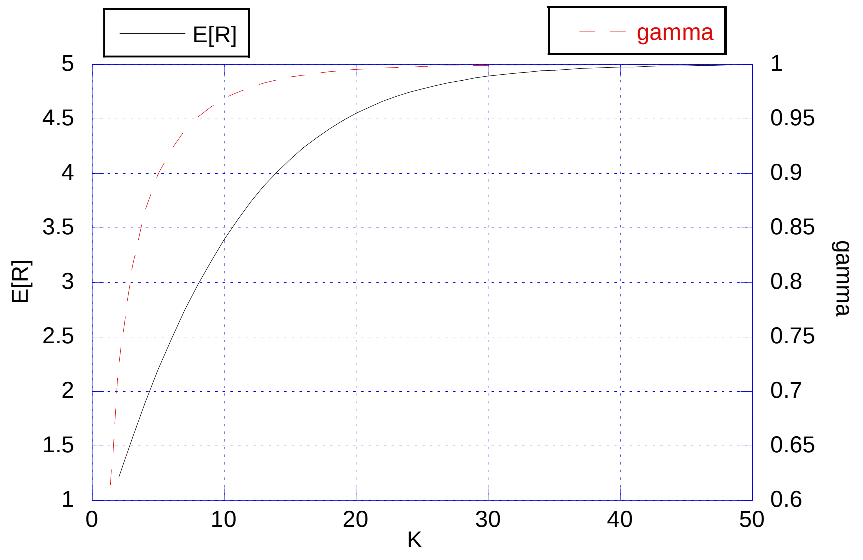
Again, $\gamma(K)$ is an increasing sequence and $\gamma(\infty) = \lim_{K \to +\infty} \gamma(K) = \lambda$. It is also $\gamma(1) \approx 0.56 \cdot \lambda$.
However, queueing increases the **response time**. Let us compute $E[R]$ as a function of K.

$$E[R] = \frac{E[N]}{\gamma} = \frac{\frac{u}{1-u} - \frac{(K+1)u^{K+1}}{1-u^{K+1}}}{\lambda \cdot \frac{1-u^K}{1-u^{K+1}}} = \frac{u + K \cdot u^{K+2} - (K+1)u^{K+1}}{\lambda \cdot (1-u^K) \cdot (1-u)} = \frac{u}{\lambda \cdot (1-u)} - \frac{K \cdot u^{K+1}}{\lambda \cdot (1-u^K)}$$

One can clearly see that:

$$\lim_{K \to +\infty} E[R] = \frac{u}{\lambda \cdot (1-u)} = \frac{1}{\mu - \lambda}$$

If you plot both $E[R]$ and $\gamma(K)$, you get the following (in the example, it is $\lambda = 1, \mu = 1.2$):

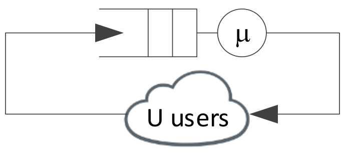


This is typical of queueing systems: when you add buffer space, the achieved utilization (or the throughput) **increases faster than the response time**. Both eventually reach their maximum value (i.e., the corresponding M/M/1's), but the utilization increases sooner. This means that **adding buffer space has a diminishing performance return**: the more you add buffer space, the smaller the return in utilization, and the higher the cost in response time. Of course, this discussion assumes that occasionally losing some input is not a problem. If this is not the case, one should add buffer space until the probability of losing a job is small enough, as shown before.

3.7 Systems with finite populations: M/M/1/*/U

It is often the case that a queueing system models a service center providing service to a **finite population of users**. Typical examples are:

- A **repair center** for broken machines. Once fixed, machines are put back in operation, and broken ones are queued for repair. The overall *population* of machines is a constant U.
- An **I/O device** being accessed by a finite number of processes.

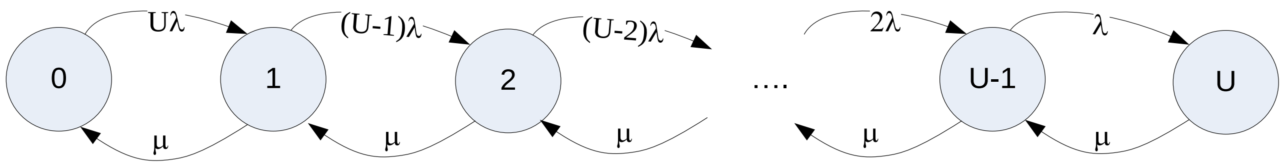


- A **network switch** having U input lines that **block** when they are being served.

Now, assume that users are **independent**, and that the time a user spends **outside** the system is exponentially distributed with a rate λ. Then, it follows that, if U users are the population and n users are inside the system at time t, then $U-n$ are outside the system and may be entering the system in the future. Therefore, the next arrival will occur when the **smallest (residual) "outside" time will have elapsed**. However, given that outside times are IID exponentials,

a) Residual outside times are distributed as outside times (memoryless property).
b) The minimum of $U-n$ IID exponentials is an exponential with a rate $(U-n)\cdot\lambda$.

Therefore, **the arrival rates are $\boldsymbol{\lambda_n=(U-n)\cdot\lambda}$.** On the other hand, the service rates are constant, $\mu_n=\mu$ (this is the case if we have only a constant-rate server), and the system will have $U+1$ states - hence, as long as its memory is at least equal to U, it does not really matter how large it is.
Having said this, it is quite straightforward to draw the CTMC:



The system is always stable (it has a finite number of states), and the equilibrium equations are:

<u>Global</u>

$$U\cdot\lambda\cdot p_0=\mu\cdot p_1$$
$$[(U-n)\cdot\lambda+\mu]\cdot p_n=\mu\cdot p_{n+1}+\big(U-(n-1)\big)\cdot\lambda\cdot p_{n-1},\quad 1\le n<U$$
$$\mu\cdot p_U=\lambda\cdot p_{U-1}$$

<u>Local</u>

$$(U-n)\cdot\lambda\cdot p_n=\mu\cdot p_{n+1}$$

From the local equilibrium equations, it is quite easy to obtain $p_n=\left(\frac{\lambda}{\mu}\right)^n\cdot\frac{U!}{(U-n)!}\cdot p_0$, $0\le n\le U$.

The system is always stable, and we get p_0 from the normalization condition:

$$p_0\cdot\sum_{n=0}^{U}\left(\frac{\lambda}{\mu}\right)^n\cdot\frac{U!}{(U-n)!}=1,\text{hence } p_0=\frac{1}{\sum_{n=0}^{U}\left(\frac{\lambda}{\mu}\right)^n\cdot\frac{U!}{(U-n)!}}$$

We can compute the performance indexes as an exercise:

$$E[N] = \sum_{n=0}^{U} n \cdot p_n = \sum_{n=0}^{U} \big(U - (U-n)\big) \cdot p_n$$

$$\begin{aligned}
&= U - \sum_{n=0}^{U-1} (U-n) \cdot \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{U!}{(U-n)!} \cdot p_0 \\
&= U - \sum_{n=0}^{U-1} \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{U!}{\big(U-(n+1)\big)!} \cdot p_0 \\
&= U - \frac{\mu}{\lambda} \cdot \sum_{n=1}^{U} \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{U!}{(U-n)!} \cdot p_0 \\
&= U - \frac{\mu}{\lambda} \cdot (1-p_0)
\end{aligned}$$

From this we get $E[N_q] = E[N] - (1-p_0) = U - \frac{\mu+\lambda}{\lambda}(1-p_0)$.

In order to apply Little's Law we need to compute:

$$\bar{\lambda} = \sum_{n=0}^{U} \lambda_n \cdot p_n = \sum_{n=0}^{U} (U-n) \cdot \lambda \cdot p_n = \lambda \cdot [U - E[N]] = \mu \cdot (1-p_0)$$

Hence, we get:

$$E[R] = \frac{E[N]}{\bar{\lambda}} = \frac{U}{\mu \cdot (1-p_0)} - \frac{1}{\lambda}, E[W] = \frac{E[N_q]}{\bar{\lambda}} = \frac{U}{\mu \cdot (1-p_0)} - \frac{1}{\lambda} - \frac{1}{\mu}$$

The last expression confirms that $E[R] = E[W] + E[t_s]$.

Finally, we have:

$$r_n = \frac{\lambda_n}{\bar{\lambda}} \cdot p_n = \frac{(U-n) \cdot \lambda}{\mu \cdot (1-p_0)} \cdot \left(\frac{\lambda}{\mu}\right)^n \cdot \frac{U!}{(U-n)!} \cdot p_0 = \frac{p_{n+1}}{1-p_0}, \ \ n < U$$

As usual, we can devise more complex systems with finite populations and C servers, etc. These only bring algebraic complications, and there is nothing interesting from a conceptual standpoint.

One might wonder how **finite-population systems** relate to **infinite-population systems** (i.e., all the others that we had discussed thus far), and why we normally assume a *constant* arrival rate in the latter. It is because, when U grows to infinity, all numbers $U - k$ are not dissimilar to U, hence the arrival rate is approximately constant. This is the same approximation through which we obtained a Poisson distribution from a binomial, when the number of trials (i.e., individuals in the population) is very large and the probability of success (i.e., that each single individual arrives in the system) is very small.

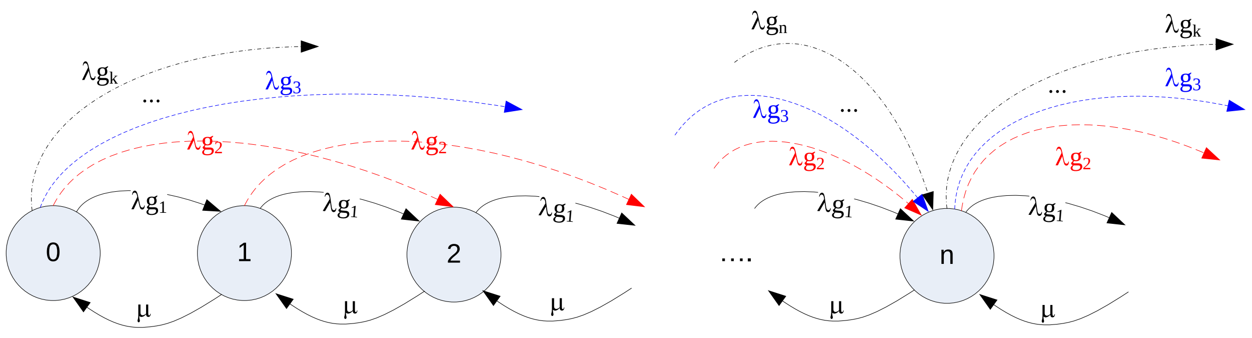
3.8 Systems with bulk arrivals

So far, we have only analyzed systems with **nearest-neighbor transitions**. However, there are practical cases when non-nearest-neighbor transitions may occur. Consider in fact the following examples:

- A processing plant where trucks arrive with exponential interarrival times, carrying a **random number of items** that must be processed individually. Thus, if we describe the system state using the number of items in the processing plant, one arrival implies a state jump of as many items as the truck contains.
- A network protocol sending **messages** to the underlying protocol. Messages may be arbitrarily long, and they may be **fragmented** and transmitted separately.

In all these cases, if we assume that arrivals are exponential, we retain the property that the only relevant parameter to describe the system state is the number of jobs in the system. However, one arrival will increase the number of queued jobs by more than one unit, so we will have **non-nearest-neighbor transitions** to the right in the CTMC.

How do we **write the CTMC** in this case? Call g the RV of the number of jobs per arrival, and let g_k be its PMF, i.e. $g_k = P\{g = k\}$. It is $g_0 = 0$, and (obviously) $\sum_{k=1}^{+\infty} g_k = 1$. This does not necessarily mean that the support of g must be infinite: if it is finite, we will have g_k starting from some index k^*. Assume that g_k is independent of the interarrival time and service time distribution. Then **the rate of the arc going from state i to state k is $\lambda \cdot g_{k-i}$.** This said, the CTMC **can** be drawn, though not too easily:



The above diagram is quite hard to follow, since:

- All states will have **as many outgoing arcs to the right as the support of RV g allows**, each one with a label $\lambda \cdot g_j, j \geq 1$. If the maximum number of jobs in a bulk arrival is infinite, there will be infinitely many outgoing arcs to the right from each state.

- Each state n will have **exactly n incoming arcs from the left**, labelled $\lambda \cdot g_{n-j}$, $0 \le j < n$ and coming from state j.
- Finally, we still have the service transition going from n to $n-1$ at a rate μ.

This may be **quite difficult** to do with pen and paper (you may want to write some code, lest you forget something).

In any case, you can still write **global** equilibrium equations (if you try **local ones**, you will surely end up forgetting some arc). These are simpler than one may think:

- With $n = 0$, $\sum_{i=1}^{+\infty} \lambda \cdot g_i \cdot p_0 = \mu \cdot p_1$, hence $\lambda \cdot p_0 = \mu \cdot p_1$ as usual.
- With $n \ge 1$,

$$\underbrace{\left[\sum_{i=1}^{+\infty} \lambda \cdot g_i + \mu\right] \cdot p_n}_{outgoing} = \underbrace{\mu \cdot p_{n+1} + \sum_{i=0}^{n-1} \lambda \cdot g_{n-i} \cdot p_i}_{incoming}$$

i.e.

$$(\lambda+\mu)\cdot p_n = \mu \cdot p_{n+1} + \lambda \cdot \sum_{i=0}^{n-1} g_{n-i} \cdot p_i$$

Computing the SS probabilities using the **direct method** (i.e., writing $p_n = f_n(p_0)$ and enforcing normalization) is probably too difficult in this case. When this is the case, the usual technique to get at least some performance metrics is to **switch to the PGFs**. We multiply each equation by z^n and sum everything up.

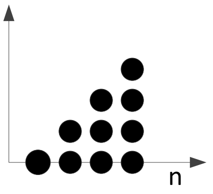
$$\begin{cases} \lambda \cdot z^0 \cdot p_0 = \mu \cdot z^0 \cdot p_1 \\ (\lambda+\mu)\cdot z^n \cdot p_n = \mu \cdot z^n \cdot p_{n+1} + \lambda \cdot z^n \cdot \sum_{i=0}^{n-1} g_{n-i} \cdot p_i \qquad n \ge 1 \end{cases}$$

$$(\lambda+\mu)\cdot \mathbf{P}(z) - \mu \cdot p_0 = \mu \cdot p_1 + \mu \cdot \sum_{n=1}^{+\infty} z^n \cdot p_{n+1} + \lambda \cdot \sum_{n=1}^{+\infty}\sum_{i=0}^{n-1} z^n \cdot g_{n-i} \cdot p_i$$

$$(\lambda+\mu)\cdot \mathbf{P}(z) - \mu \cdot p_0 = \cancel{\mu \cdot p_1} + \frac{\mu}{z}\cdot\left[\mathbf{P}(z) - p_0 - \cancel{z \cdot p_1}\right] + \lambda \cdot \sum_{i=0}^{+\infty}\sum_{n=i+1}^{+\infty} z^n \cdot g_{n-i} \cdot p_i$$

$$(\lambda+\mu)\cdot \mathbf{P}(z) - \mu \cdot p_0 = \frac{\mu}{z}\cdot\left[\mathbf{P}(z) - p_0\right] + \lambda \cdot \sum_{i=0}^{+\infty} z^i \cdot p_i \cdot \sum_{n=i+1}^{+\infty} z^{n-i} \cdot g_{n-i}$$

$$(\lambda+\mu)\cdot \mathbf{P}(z) - \mu \cdot p_0 = \frac{\mu}{z}\cdot\left[\mathbf{P}(z) - p_0\right] + \lambda \cdot \mathbf{P}(z) \cdot \mathbf{G}(z)$$



where $\mathbf{G}(z)$ is the PGF of $\boldsymbol{g}$.

Thus, we end up with:

$$\mathbf{P}(z) = \frac{\mu \cdot p_0 \cdot (1-z)}{\mu \cdot (1-z) - \lambda \cdot z \cdot [1 - \mathbf{G}(z)]}$$

The normalization condition is $\mathbf{P}(1) = 1$. Unfortunately, this leads to an undetermined expression, hence we must use De L'Hopital's rule to get to the bottom of it. We get:

$$\lim_{z\to 1^-}\mathbf{P}(z) = \left.\frac{-\mu\cdot p_0}{-\mu-\lambda\cdot[1-\mathbf{G}(z)]+\lambda\cdot z\cdot\mathbf{G}'(z)}\right|_{z=1} = \frac{-\mu\cdot p_0}{-\mu+\lambda\cdot\mathbf{G}'(1)} = \frac{p_0}{1-\frac{\lambda}{\mu}\cdot E[g]}$$

This is because a property of the PGF is that $\mathbf{G}'(1) = E[g]$. From the latter we get:

$$p_0 = 1-\frac{\lambda}{\mu}\cdot E[g]$$

p_0 is a probability, so it must be non-negative. This means that $\frac{\lambda}{\mu}\cdot E[g] < 1$, which is our stability condition. In fact, it is quite clear that $\rho = (1-p_0) = \frac{\lambda}{\mu}\cdot E[g]$, since $\lambda\cdot E[g]$ is the actual average rate of **job** arrivals. This yields the following expression for $\mathbf{P}(z)$:

$$\mathbf{P}(z) = \frac{\mu\cdot(1-\rho)\cdot(1-z)}{\mu\cdot(1-z)-\lambda\cdot z\cdot[1-\mathbf{G}(z)]}$$

As an exercise, let us instantiate the above in some simple cases:

<u>**Single arrivals**</u>: $g_i = \begin{cases}1 & i=1\\ 0 & i\neq 1\end{cases}$

Thus, we get $\mathbf{G}(z) = 1\cdot z^1 = z$.

Under the condition that $p_0 = 1-\frac{\lambda}{\mu}\cdot E[g] > 0$, i.e. $\rho = \frac{\lambda}{\mu}\cdot E[g] = \frac{\lambda}{\mu} < 1$, we get:

$$\mathbf{P}(z) = \frac{\mu\cdot(1-\rho)\cdot(1-z)}{\mu\cdot(1-z)-\lambda\cdot z\cdot[1-z]} = \frac{1-\rho}{1-\rho\cdot z}$$

We know that the latter is the PGF of the M/M/1 SS probabilities $p_n = (1-\rho)\cdot\rho^n$.

<u>**Constant-batch multiple arrivals**</u>: $g_i = \begin{cases}1 & i=b\\ 0 & i\neq b\end{cases}$

In this case $E[g] = b$, $\mathbf{G}(z) = 1\cdot z^b = z^b$, hence the stability condition is $\rho = \frac{\lambda}{\mu}\cdot b < 1$, which makes sense intuitively. Hence, we get (recall that $1-z^b = (1-z)\cdot\sum_{j=0}^{b-1} z^j$):

$$\mathbf{P}(z) = \frac{\mu\cdot(1-\rho)\cdot(1-z)}{\mu\cdot(1-z)-\lambda\cdot z\cdot[1-z^b]} = \frac{1-\rho}{1-\frac{\rho}{b}\cdot\sum_{i=1}^{b} z^i}$$

Thus, we get:

$$E[N] = \frac{d}{dz}\mathbf{P}(z)|_{z=1} = \frac{d}{dz}\left[\frac{1-\rho}{1-\frac{\rho}{b}\cdot\sum_{i=1}^{b} z^i}\right]_{z=1} = (1-\rho)\cdot\left.\frac{\frac{\rho}{b}\cdot\sum_{i=1}^{b} i\cdot z^{i-1}}{\left[1-\frac{\rho}{b}\cdot\sum_{i=1}^{b} z^i\right]^2}\right|_{z=1} = \frac{\rho\cdot(b+1)}{2\cdot(1-\rho)}$$

Note that, when $b=1$, we obtain the familiar formula of of M/M/1 systems: $E[N] = \frac{\rho}{1-\rho}$.

<u>**Geometric arrivals**</u>: $g_i = (1-\alpha)\cdot\alpha^{i-1},\quad i\geq 1$

Thus, we get $\mathbf{G}(z) = \sum_{k=1}^{+\infty} z^k \cdot g_k = \sum_{k=1}^{+\infty} z^k \cdot (1-\alpha) \cdot \alpha^{k-1} = \frac{(1-\alpha)\cdot z}{1-\alpha\cdot z}$

Under the condition that

$$p_0 = 1 - \frac{\lambda}{\mu} \cdot E[g] > 0, \text{i.e.}, \rho = \frac{\lambda}{\mu} \cdot \frac{1}{1-\alpha} < 1$$

In this case, we get:

$$\mathbf{P}(z) = \frac{\mu \cdot (1-\rho) \cdot (1-z)}{\mu \cdot (1-z) - \lambda \cdot z \cdot \left[1 - \frac{(1-\alpha)\cdot z}{1-\alpha \cdot z}\right]}$$

$$= \frac{\mu \cdot (1-\rho) \cdot (1-z) \cdot (1-\alpha \cdot z)}{\mu \cdot (1-z) \cdot (1-\alpha\cdot z) - \lambda \cdot z \cdot (1-z)}$$
$$= (1-\rho) \cdot \frac{1-\alpha\cdot z}{(1-\alpha \cdot z) - (1-\alpha)\cdot \rho \cdot z}$$
$$= (1-\rho) \cdot \frac{1-\alpha \cdot z}{1-(\alpha+\rho-\alpha\cdot\rho)\cdot z}$$

This is hard to anti-transform. However, we can still find some interesting data:

$$p_0 = \lim_{z\to 0} \mathbf{P}(z) = 1-\rho$$

$$p_1 = \frac{d}{dz}\mathbf{P}(z)\bigg|_{z=0}$$
$$= (1-\rho)\cdot\left[\frac{-\alpha \cdot [1-(\alpha+\rho-\alpha\cdot\rho)\cdot z] + (1-\alpha\cdot z)\cdot(\alpha+\rho-\alpha\cdot\rho)}{[1-(\alpha+\rho-\alpha\cdot\rho)\cdot z]^2}\right]_{z=0}$$
$$= \frac{(1-\rho)\cdot\rho\cdot(1-\alpha)}{[1-(\alpha+\rho-\alpha\cdot\rho)\cdot z]^2}\bigg|_{z=0}$$
$$= (1-\rho)\cdot\rho\cdot(1-\alpha)$$

$$E[N] = \frac{d}{dz}\mathbf{P}(z)|_{z=1} = \frac{(1-\rho)\cdot\rho\cdot(1-\alpha)}{[1-(\alpha+\rho-\alpha\cdot\rho)\cdot z]^2}\bigg|_{z=1} = \frac{(1-\rho)\cdot\rho\cdot(1-\alpha)}{[(1-\rho)\cdot(1-\alpha)]^2} = \frac{\rho}{(1-\rho)\cdot(1-\alpha)}$$

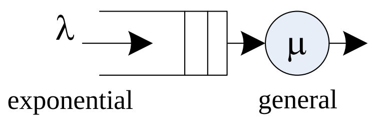
The same procedure can be used with systems having **bulk services**, i.e. where arcs going to the **right** may imply more than a single state jump. Computations tend to be nastier for these systems.

3.9 Systems with non-exponential service time distributions

So far, we have assumed that:

- Interarrivals are exponential;
- Service times are exponential.

And we have described a theory that allows one to find not only **mean values of the steady-state performance indexes**, but also – in most cases – their **distributions**. There are cases when interarrival and (more frequently) service times **cannot be fitted to an exponential** distribution. In these



cases, the analysis is made complex by the fact that **the number of jobs in the system** is not a sufficient characterization of its state anymore.

Despite this, some results can be found, for instance for the **M/G/1** system, i.e. the one with exponential interarrivals, **general service times** and one server. By "general" we mean that **any** distribution can be used (including the exponential, which would make this system an M/M/1, for which we have a direct method).

Let t_S be the RV that models the service times, and let us assume that $E[t_S] = 1/\mu$ and $Var(t_S)$ are known. If $\rho = \frac{\lambda}{\mu} < 1$, then the system is stable and **Pollaczek and Khinchin**'s formula states that:

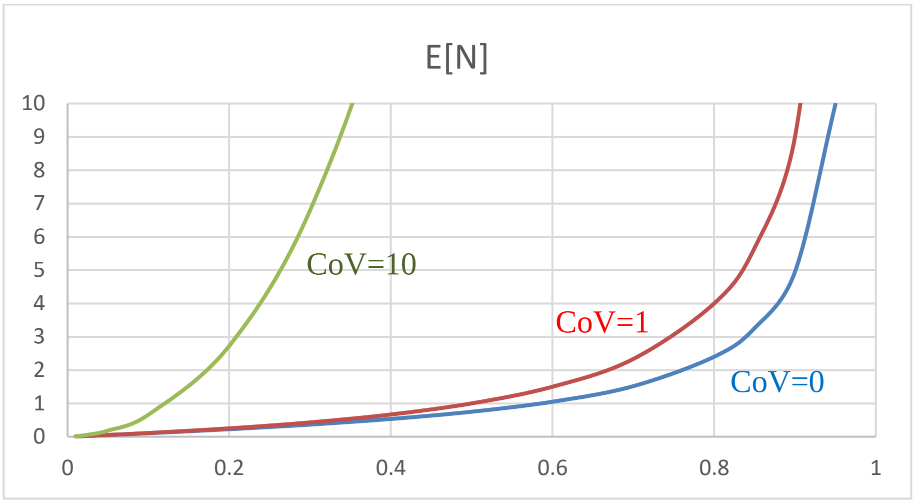
$$\boxed{E[N] = \rho + \frac{\rho^2 + \lambda^2 \cdot Var(t_S)}{2 \cdot (1 - \rho)} = \rho + \frac{\rho^2 \cdot [1 + CoV(t_S)^2]}{2 \cdot (1 - \rho)}}$$

Note the following:

- given $E[N]$, one can always compute the other three mean performance indexes (waiting time, response time and number of jobs in the queue), using Little's law and few other obvious tricks;
- when t_S is exponential, PK's formula yields Kleinrock's formula for M/M/1 systems.

The version of PK's formula that mentions explicitly the CoV is quite insightful, since we know that **the exponential's CoV is equal to 1**. This means that the mean number of jobs in the system will also depend on "how variable" service times are. If the service is **deterministic** (which we call an M/D/1 system), then it will be $Var(t_S) = CoV(t_S) = 0$, and we will have the smallest possible average number of jobs in a system.

PK's formula shows that the **variability** of the service time **increases the queue occupancy.** If that variability is very high (e.g., the service time distribution is fat- or heavy-tailed), then a system **may become congested even when $\rho \ll 1$**. This confirms that it is very important to pick the **right model** for the service times, otherwise you will end up thinking that your system has negligible queueing when in fact it does not.



PK's formula tells us something which has a strong physical significance: **queueing arises from variability (or randomness)**. In a system where both interarrival times and service times are constant (i.e., a D/D/1 system), there is **no queueing as long as $\rho \leq 1$** (yes, we can also afford equality here). If either or both the interarrival times and the service times are variable, then there will be queueing. Queueing arises from jobs "bunching up". If interarrival times are stochastic, there will be jobs that arrive **close to each other**, and this generates queueing. If service times are stochastic, there will occasionally be **a long service time**, during which many jobs will queue up. The higher the variability, the longer the queue will be on average. Because of Little's law, the same can be said of the response and waiting times.

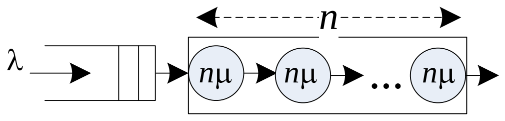
PK formula computes $E[N]$ exactly, but it does not allow you to compute **steady-state probabilities** in a closed form. For instance, you cannot compute $P\{N \geq 3\}$. If results like this are required, you have two choices: the first one is to analyze your M/G/1 system using the method of the **imbedded Markov chain**. This method yields **exact results (for the PGF)**, but it requires an entirely different mathematics, related to discrete-time processes (ours are continuous-time). If you are interested, just check any QT book.

An alternative method is that of **Phase-Type distributions** to obtain **approximate numerical** results, with a tunable trade-off between accuracy and complexity. With that method, you model the general service-time distribution using **compositions of exponential distributions**. This way, you can still write down a (more complex) CTMC, and you can solve it using numerical techniques (the so-called **Neuts' matrix-analytic method**). This method is well described in QT books too. In a few moments, we will present an interesting case of the application of this method, for which the maths we introduced so far is enough. First, we give another result.

3.9.1 M/G/∞ systems and insensitivity

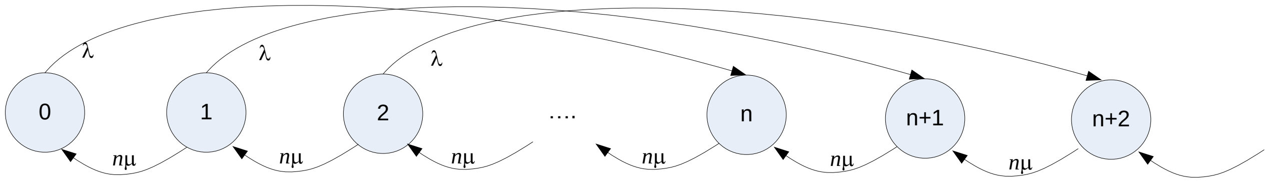
Another interesting result is the one for the **M/G/∞ system**. This is one where there are infinitely many identical servers, but the distribution of the service time is general. For this system, the **steady-state probabilities** are exactly **the same as the M/M/∞'s**. In other words, the SS probabilities for an **M/G/∞** system depend only on the **mean value of the service time**, and not on its particular distribution. Any distribution with the same mean will yield the same results. This is called **insensitivity property**, and such a property is a rare gem in queueing theory. Systems with insensitivity properties are usually found when **there is no queueing**. This is in fact one such case. In fact, if there are infinite servers, the variability of the service time does not really matter, since a long service time will not create queueing.

3.9.2 Exercise: M/E$_n$/1 system



Consider now a system where the service time distribution is an n-stage Erlang distribution, for which the service time is the sum of n IID exponentials, each with a rate $n \cdot \mu$. The Erlang distribution is a subclass of Phase-Type distributions. The mean service time will of course be $n \cdot 1/(n \cdot \mu) = 1/\mu$. This system can be modeled by a server that is constituted of ***n* serving stages**, such that **only one job may be in service at any time**. The next job will enter the first service stage after the current job leaves the last service stage, so that no two jobs are being served concurrently.

Assume that we count the number of **residual stages to be traversed** in order to clear the backlog, as a state characterization. In a possible trajectory, a job leaving a service stage would count as a unitary downward step. On the other hand, an arrival will count as **an upward step of *n* stages**, increasing by n the total number of service stages that need to be traversed. Therefore, we would obtain a CTMC which is, by all accounts, the same as the one of a **constant-batch bulk-arrival system**, n being in fact the batch length, where left-bound arcs have a rate $n \cdot \mu$. Once more, the number in the circles is not the number of **jobs** in the system, it is the total **number of stages** that must be traversed in order to clear the backlog.



For the latter, we already have formulas:

$$\mathbf{P}(z) = \frac{n \cdot \mu \cdot (1-\rho) \cdot (1-z)}{n \cdot \mu \cdot (1-z) - \lambda \cdot z \cdot [1-z^n]} = \frac{1-\rho}{1-\frac{\rho}{n} \cdot \sum_{i=1}^{n} z^i},$$

where $\rho = \frac{\lambda}{n \cdot \mu} \cdot n = \frac{\lambda}{\mu} < 1$. Calling S the number of stages, we get $E[S] = \frac{\rho \cdot (n+1)}{2 \cdot (1-\rho)}$.

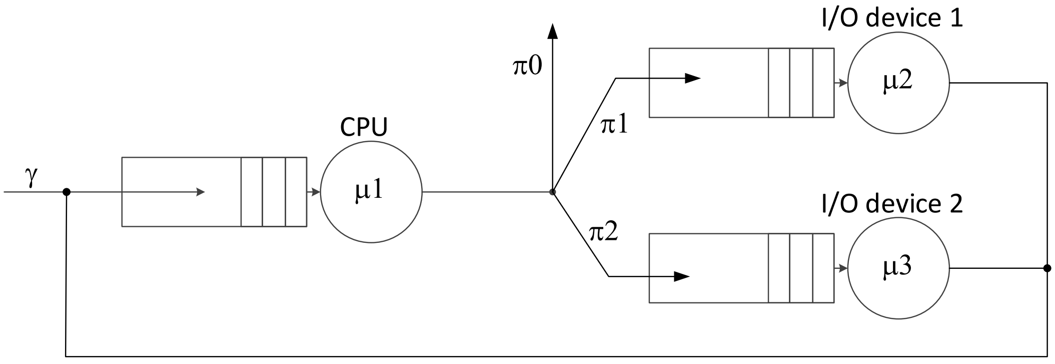
If one need to compute SS probabilities for the number of **stages** in the system (call it π_j), she can try either to antitransform the PGF, or to compute the first k probabilities by differentiating the PGF. For each number of stages j, the corresponding number of jobs is $N = \lceil j/n \rceil$, so one can obtain SS probabilities for the number of jobs as well. From the latter, it would not be too difficult to compute the **distribution of the response times** (the procedure would be similar to the one we used for the M/M/1). Also note that the mean number of jobs in the system can always be computed via PK's formula.

4 Queueing Networks

So far, we have examined queueing systems in isolation. Queueing Theory allows you to deal with systems composed by **several** queue+server subsystems, which are called **queueing networks**. Queueing networks can be either:

- **Open:** these do interact with the external environment, which injects jobs (**external arrivals**) and absorbs jobs leaving the QN. In these, the number of jobs in the system is **a variable**. Discovering that number (or, more correctly, its steady-state distribution) is in fact the main purpose of the analysis.
- **Closed**: there is **no interaction** with the external world. The number of jobs in the system is **constant** and it is an **input datum**. In these, we normally want to measure the **throughput**.

A typical example of a queueing network is the following (transactional server):



The server has **one** CPU and two I/O devices (e.g., disks). Jobs arrive from the outside, hence the QN is an **open** one. When they arrive, they first spend some random time on the CPU. That service time is exponentially distributed with a mean $1/\mu_1$. After that, they can:

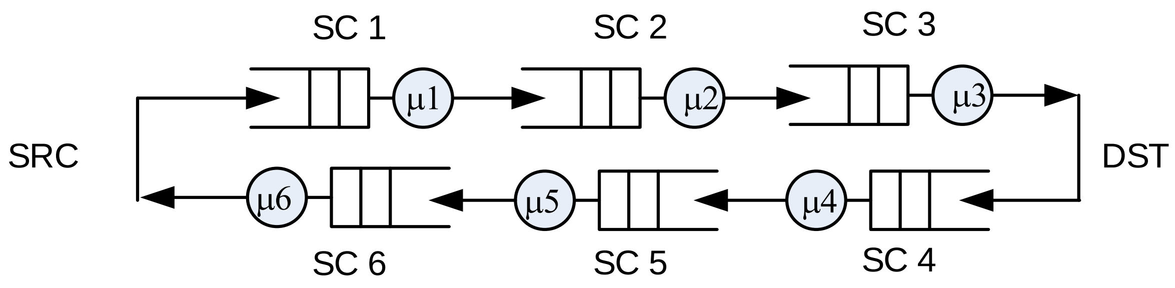
- Leave, with a probability π_0
- Queue at device 1, with probability π_1
- Queue at device 2, with probability π_2

And it is obviously $\pi_0 + \pi_1 + \pi_2 = 1$. After being processed at an I/O device, a job comes back to the CPU, and starts over again. Therefore, a job may pay **several visits** to the same service center before leaving the network.

In the above model, all SCs have been assumed to be M/M/1, but it does not have to be so. If you have – for instance – a multicore CPU, modeling it as an M/M/C would be more appropriate. We may want to add a **delay center** on the link coming back from the I/O devices to the CPU, to model some interaction with users (e.g., the need to press a key), etc.

Closed QNs are a bit more difficult to envisage. One way to picture a closed QN is to figure out an **open QN**, and then **connect its output to its input,** so that a job "leaving" the network is immediately fed back to the input. This models a situation when a system **admits a finite number of jobs simultaneously**, and as soon as one job is completed and leaves the network, another one replaces it. A typical case of a system modeled using a closed QN is a **multiprogrammed system**, where the number of processes is constant.

Another case is modeling **network paths**, e.g. to solve **flow-control** problems. For instance, a network path can be modeled by a chain of servers. The fact that flow control is in place is modeled by fixing the number of packets to the flow-control window size. In a closed QN like this, we want to know the **throughput**, i.e. how fast packets circulate.



How do we **describe** a QN? We need to know:

- The network **topology**, which can be represented by a **directed graph.** Let M be the number of SCs in the network.
- The rates of the **external arrivals at each SC**, at least those that do admit external arrivals, call them γ_i.
- The **service rates** at each SC μ_i, and the **number of servers** C_i.
- The **routing matrix Π**, whose entries $\pi_{i,j}$ are the probabilities that a job leaving SC i reaches SC j. If $\sum_{j=1}^{M} \pi_{i,j} < 1$, then $\pi_{i,0} = 1 - \sum_{j=1}^{M} \pi_{i,j}$ is the probability that a job will **leave the QN** after visiting SC i.

The **state** of the network at a given time t will be a **vector**[6] $\underline{n}(t) = [n_1(t), n_2(t), \dots n_M(t)]^T$, where each $n_i(t)$ represents the number of jobs at SC i at time t. If the network admits a steady state, then we will have SS probabilities associated to each vector $\underline{n} = [n_1, n_2, \dots n_M]^T$. In other words, we will define $p_{\underline{n}} = p(n_1, n_2, \dots, n_M) = P\{N_1 = n_1, N_2 = n_2, \dots, N_M = n_M\}$, the JPMF of RVs $N_1, N_2, \dots, N_M$, each one representing the number of jobs on SC i.

[6] It should be a *column* vector. However, writing column vectors takes up too much space, therefore we will often write them as *row* vectors instead and add a "transpose" symbol. This is not very elegant, but it makes for concise reading.

4.1 Characterizing the output of a service center

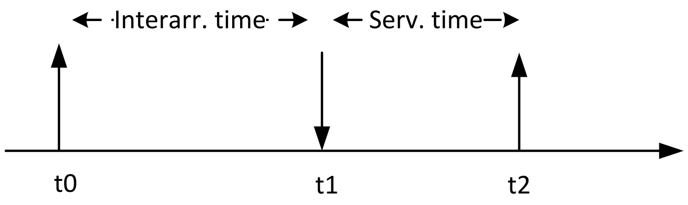
In a QN, the output of a SC constitutes **part of the input for another SC**. We have not discussed yet how to **characterize the output** of a SC.

Consider an M/M/1 SC at the steady state (but the result is more general, and holds for an M/M/C, even with $C \to \infty$), whose interarrival times are distributed as $F_A(t) = 1 - e^{-\lambda \cdot t}$, and whose service times are distributed as $F_B(t) = 1 - e^{-\mu \cdot t}$. We want to compute the distribution of the **inter-departure times** $F_D(t) = P\{D \le t\}$. We leverage the Total Probability law, and get:

$F_D(t) = P\{D \le t\} = P\{D \le t | n > 0\} \cdot P\{n > 0\} + P\{D \le t | n = 0\} \cdot P\{n = 0\}$.

Let us discuss the two terms separately:

- $P\{D \le t | n > 0\}$: when $n > 0$, the next job starts service as soon as the previous one leaves. Therefore, the inter-departure times are distributed **as the service times**: $P\{D \le t | n > 0\} = F_B(t)$.



- $P\{D \le t | n = 0\}$: when $n = 0$, a job departing at time t_0 leaves the system **empty**. Therefore, the inter-departure time, i.e. the time until the next job departs, consists of **two contributions**: one (residual) **interarrival time**, and one **service time**. However, residual interarrival times are distributed the same as interarrival times, because the exponential is memoryless. Thus, $P\{D \le t | n = 0\} = P\{A + B \le t\}$.

Furthermore, note that $P\{n > 0\} = \rho$ and $P\{n = 0\} = 1 - \rho$. Therefore, we have:

$F_D(t) = P\{D \le t\} = F_B(t) \cdot \rho + P\{A + B \le t\} \cdot (1 - \rho)$.

In order to write $P\{A + B \le t\}$ simply, we **can switch to the LST domain**. In fact, $\mathbf{L}_{A+B} = \mathbf{L}_A \cdot \mathbf{L}_B$, because of the convolution property and since A and B are independent.

Now, $\mathbf{L}_A = \frac{\lambda}{\lambda+s}$, and $\mathbf{L}_B = \frac{\mu}{\mu+s}$, hence:

$$\begin{aligned}
\mathbf{L}_D &= \frac{\mu}{\mu+s} \cdot \rho + \frac{\mu}{\mu+s} \cdot \frac{\lambda}{\lambda+s} \cdot (1-\rho) \\
&= \frac{\lambda}{\mu+s} + \frac{\cancel{\mu}}{\mu+s} \cdot \frac{\lambda}{\lambda+s} \cdot \frac{\mu-\lambda}{\cancel{\mu}} \\
&= \frac{\lambda}{\mu+s} \cdot \left(1 + \frac{\mu-\lambda}{\lambda+s}\right) \\
&= \frac{\lambda}{\cancel{\mu+s}} \cdot \frac{\cancel{\mu+s}}{\lambda+s} \\
&= \mathbf{L}_A
\end{aligned}$$

Since LSTs are univocal, this can only mean that $F_D(t) = F_A(t) = 1 - e^{-\lambda \cdot t}$. In other words, **the inter-departure times have the same distribution as the interarrival ones.** This is somewhat surprising, and should not be misconstrued. The above result does not mean that **interarrival times between two jobs are preserved at their departure**, which would be **false**. Only, that the inter-departure time of those two jobs will be drawn from the same distribution as the interarrival one. This result is **general**, and holds for M/M/C systems, for any value of C. Let us enounce it more formally:

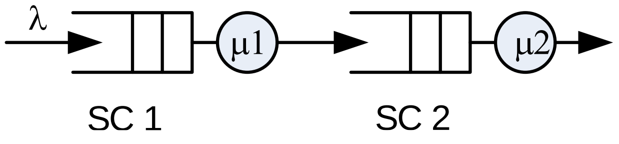
Burke's Theorem (a.k.a. "$M \rightarrow M$" property)

Given an M/M/C system in a steady state, whose average arrival rate is λ, then:

a) *The* ***departure process*** *is a Poisson one, with a rate λ*
b) *$\forall t$, the number of jobs in the system $n(t)$ is* ***independent*** *of the inter-departure times in $[0, t)$.*

■

Burke's theorem has a straightforward consequence. Take a **tandem** (or "linear") QN like the one in the figure, consisting of two M/M/1 SCs. In the latter, **the arrivals at SC 2** are **a Poisson process with a rate** λ, and $n_1(t)$ is independent of what happens downstream – and, in particular, it is independent of $n_2(t)$.



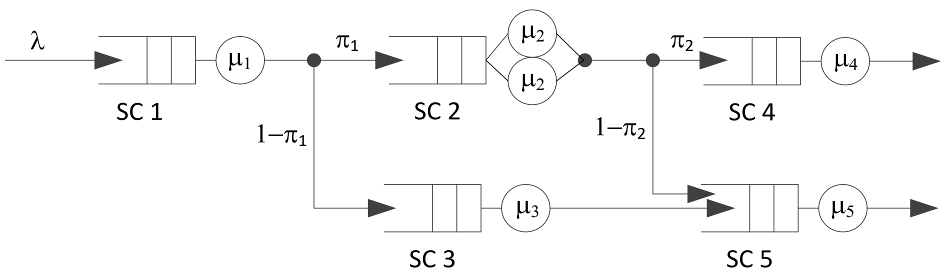
This means that $p_{\underline{n}} = p(n_1, n_2) = p_1(n_1) \cdot p_2(n_2) = [(1-\rho_1) \cdot {\rho_1}^{n_1}] \cdot [(1-\rho_2) \cdot {\rho_2}^{n_2}]$, where from now on $p_i(n_i)$ will denote "the probability that n_i jobs are at SC i", i.e. the PMF for node i, with a slight but necessary deviation from the usual notation. Of course, we still need to check *a posteriori* that $\rho_i < 1 \; \forall i$, otherwise no SS probabilities exist.

We say that such a network has a **product form**, i.e.:

we can write **joint, network-wide SS probabilities** as **the product of per-SC SS probabilities**.

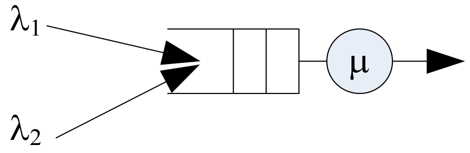
4.2 From Burke's theorem to queueing networks

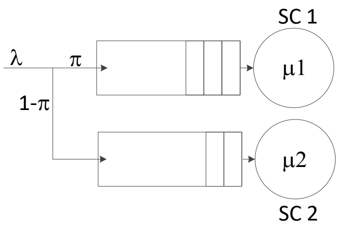
Burke's theorem has many important consequences. Take any **acyclic network** with **probabilistic routing**, where each SC is an M/M/C. **Acyclic** means that there is no feedback loop (for instance, part of the output of SC 5 being fed back to SC 2 would constitute a feedback loop).



We know that the output of SC 1 is a Poisson process with a rate λ (by Burke's Theorem). Can we characterize the input to SC 2 and SC 3?

This is relevant, since in a QN it may happen that two or more Poisson processes are **merged** into one to generate the input to a node. It may also happen that the output of a node is **split probabilistically** into two or more flows (through routing), which then form the input of two different SCs.





We can observe straightforwardly that **the superimposition of independent Poisson processes is a Poisson process**, with a rate equal to the sum of the rates. In fact, a Poisson process has exponential interarrival times. The next arrival will occur when the **shortest (residual) interarrival time** expires, but we know that, with independent exponentials:

a) the adjective “residual” is immaterial (exponentials are memoryless);
b) the minimum is still an exponential with a rate equal to the sum of the rates.

Therefore, the superimposition of two **independent** Poisson processes is itself a Poisson process, with a rate equal to the sum of the rates.

If, instead, a Poisson process is **split probabilistically**: arrivals are Poissonian with a rate λ, and they are routed to SC 1, with a probability π, or to SC 2, with a probability $1-\pi$. Then the arrivals at either SC are themselves **independent Poisson processes**, with rates $\lambda_1 = \lambda \cdot \pi$, $\lambda_2 = \lambda(1-\pi)$. This can be obviously generalized to a probabilistic splitting of a process into n processes, with probabilities $\pi_i\ s.t. \sum_{i=1}^{N} \pi_i = 1$.

Thus, coming back to the above example, we now know the following:

- by Burke’s theorem, the output process at SC 1 is Poisson with a rate λ;
- the probabilistic splitting of a Poisson process is still a Poisson process, hence arrivals at SC 2 are Poisson with a rate $\pi_1 \cdot \lambda$ and those to SC 3 are Poisson with a rate $(1-\pi_1) \cdot \lambda$;
- still by Burke’s theorem, the number of jobs at each SC is independent;
- the superimposition of independent Poisson processes is still a Poisson process, hence arrivals at SC 4 are Poisson with a rate $\pi_1 \cdot \pi_2 \cdot \lambda$ and those at SC 5 are Poisson with a rate $[\pi_1 \cdot (1-\pi_2) + (1-\pi_1)] \cdot \lambda = (1-\pi_1 \cdot \pi_2) \cdot \lambda$;

All the above SCs are M/M/C where the arrival and service rates are known, hence:

we can compute stability conditions for each of them, **in isolation;**

- we can compute SS probabilities for each of them **in isolation**, call them $p_i(n_i)$, under the respective stability conditions[7].

Burke's theorem guarantees that, if all the stability conditions are verified simultaneously, then:

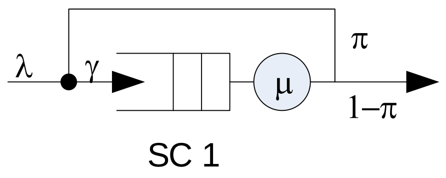
$$p_{\underline{n}} = p(n_1, n_2, n_3, n_4, n_5) = \prod_{i=1}^{5} p_i(n_i).$$

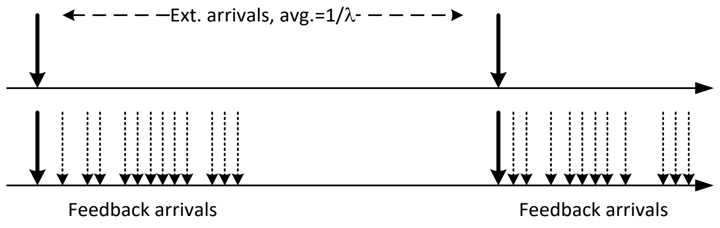
Therefore, all **acyclic networks** with probabilistic routing have a product form, and their inputs and outputs are Poisson processes.

Furthermore, we can easily compute $E[N_i]$, $E[R_i]$ (by applying Little's Law to each SC), $E[N] = \sum_{i=1}^{5} E[N_i]$, and $E[R] = E[N]/\lambda$. If required, we can also compute **distributions** of response times at the single service centers, etc.

Note that this holds if **queues are infinite**. Infinite queues are a requirement of Burke's Theorem.

4.2.1 Queueing networks with feedback loops



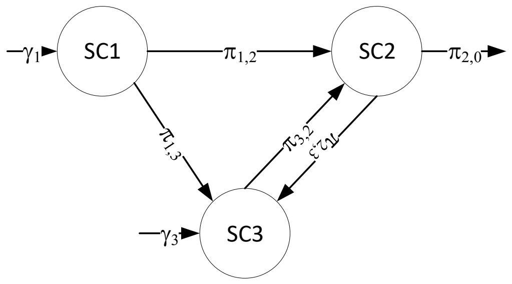


Let us consider instead an open queueing network with a **feedback loop** such as the one in the figure: jobs leave the network with a probability $1 - \pi$ and are sent back to SC 1 with a probability π. This is, of course, not a tandem network. We know that the superimposition of **independent** Poisson processes is a Poisson process. However, the external arrivals **and** the arrivals on the feedback loop are **by no means independent**. This can be shown through a simple example: if $\boldsymbol{\lambda} \ll \boldsymbol{\mu}$ and π is close to 1, then one external arrival will trigger a burst of arrivals at the feedback loop (on average $1/(1-\pi)$), whose average spacing is a service time $1/\mu$. The burst eventually dies out, and another one occurs after the next external arrival. Thus, arrivals on the feedback loop are triggered by an external arrival, hence the two processes are not independent. Moreover, the arrival process at SC 1 is **not a Poisson one**. What is surprising is that, despite the above, the **departure process is still a Poisson one**, with a rate $\gamma \cdot (1 - \pi)$. In fact, as long as the **external** arrivals are Poisson, then even QNs with **feedback** arcs, like the one above, **still admit a product form**.

[7] Thus far, we have denoted with $p_n(t)$ the probability of finding n jobs in a queueing system at time t. Since we now discuss queueing *networks* at the steady state, we need to alter the notation a bit: from now on we will write $p_i(n)$ to denote the SS probability of finding n jobs at SC i.

4.3 General results for open queueing networks

It is now time to lay down the hypotheses under which **an open network admits a product form**. We define an **open Jackson's network** as a **directed graph** whose nodes are M/M/C SCs, and whose arcs represent **routing** among SCs. Each arc has a weight equal to the routing probability: $\pi_{i,j}$ is the probability that a job leaving SC i will reach SC j. We call $\pi_{i,0}$ the probability that a job leaves after visiting SC i. It is, of course, $\sum_{j=0}^{M} \pi_{i,j} = 1,\ \forall i$. Moreover, $\pi_{i,j}$ must be **independent of the network's state**. In other words, the routing probability must not change based on how jobs are distributed in the network (which, for instance, excludes dynamic load balancing from this kind of modeling[8]). Finally, at each SC external arrivals are Poissonian with a rate γ_i. Vector $\underline{\boldsymbol{\gamma}} = [\gamma_1, \dots, \gamma_M]^T$ must not be identically null (otherwise the QN is not an open one). Summing up:



1) M M/M/C_i SCs. At each SC i, the C_i severs have a service rate μ_i;
2) Poisson external arrivals $\underline{\boldsymbol{\gamma}} = [\gamma_1, \dots, \gamma_M]^T$;
3) Markovian routing, i.e., the routing probabilities are **state-independent**;
4) arcs are traversed in zero time (the only delay is at the nodes).

The above four hypotheses define an **open Jackson's network**. OJNs do **admit a product form,** as per the following theorem.

<u>Jackson's Theorem</u>

In an OJN, under hypotheses 1-4 above, if $\rho_i = \frac{\lambda_i}{C_i \cdot \mu_i} < 1\ \ \forall i$, *then it is* $p_{\underline{n}} = p(n_1, \dots, n_M) = \prod_{i=1}^{M} p_i(n_i)$, *where:*

$$p_i(n_i) = \begin{cases} p_i(0) \cdot \dfrac{(C_i \cdot \rho_i)^{n_i}}{n_i!} & n_i \le C_i \\ p_i(0) \cdot \dfrac{C_i^{C_i} \cdot \rho_i^{n_i}}{C_i!} & n_i \ge C_i \end{cases}$$

are the SS probabilities of an M/M/C_i *system whose severs have a rate* μ_i. *If* $C_i = 1$ *(i.e., an* M/M/1 *system), the SS probabilities collapse to* $(1 - \rho_i) \cdot \rho_i^{n_i}$.

■

[8] By **dynamic load balancing**, we mean, for instance, the policy of routing a job to the downstream SC with the *smallest queue*. This would make routing probabilities dependent on the state of the downstream SCs.

The only missing tile in the above picture are **the arrival rates** λ_i, which are assumed to be known quantities in the above theorem. Obtaining the arrival rates is rather **straightforward**. A job arrives at SC i because:

a) it arrives from the outside, at a rate γ_i, or
b) it leaves SC j and is routed to SC i, according to routing probability $\pi_{j,i}$.

This yields the following relationship: $\lambda_i = \gamma_i + \sum_{j=1}^{M} \pi_{j,i} \cdot \lambda_j$, which holds for every SC i.

The above equality can be written in a matrix form, i.e.: $\underline{\boldsymbol{\lambda}} = \underline{\boldsymbol{\gamma}} + \underline{\boldsymbol{\Pi}}^{\mathbf{T}} \cdot \underline{\boldsymbol{\lambda}}$, where $\underline{\boldsymbol{\Pi}} = \{\pi_{i,j}\}$ is the routing matrix. This means that the arrival rates at each SC can be computed by solving the above matrix equation, i.e. $\underline{\boldsymbol{\lambda}} = \left(\underline{\mathbf{I}} - \underline{\boldsymbol{\Pi}}^{\mathbf{T}}\right)^{-1} \cdot \underline{\boldsymbol{\gamma}}$. The latter can be solved at least numerically using a spreadsheet software. In most cases, $\underline{\boldsymbol{\lambda}}$ can be obtained in a closed form. Note that, if you sum up all the $\lambda_i = \gamma_i + \sum_{j=1}^{M} \pi_{j,i} \cdot \lambda_j$ on index i, you get the following:

$$\sum_{i=1}^{M} \lambda_i = \sum_{i=1}^{M} \gamma_i + \sum_{i=1}^{M}\sum_{j=1}^{M} \pi_{j,i} \cdot \lambda_j$$

Changing the symbol for this index to avoid confusion

$$\sum_{i=1}^{M} \lambda_i = \sum_{i=1}^{M} \gamma_i + \sum_{j=1}^{M} \lambda_j \cdot \sum_{i=1}^{M} \pi_{j,i}$$

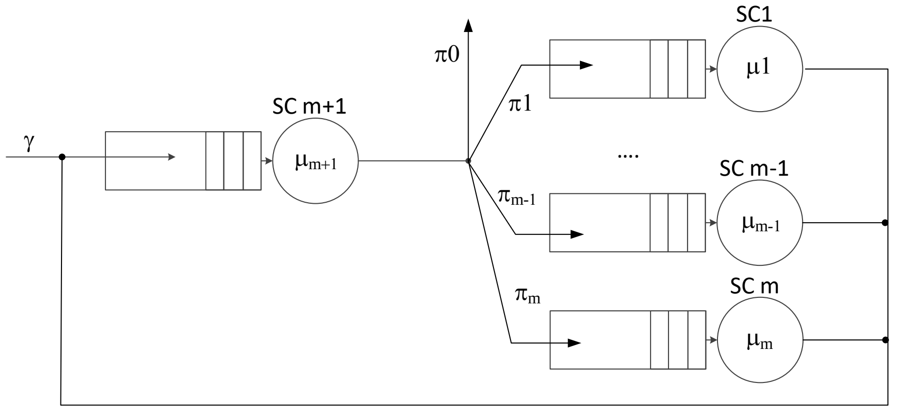
$$\sum_{j=1}^{M} \lambda_j \left(1 - \sum_{i=1}^{M} \pi_{j,i}\right) = \sum_{i=1}^{M} \gamma_i$$

$$\sum_{j=1}^{M} \lambda_j \cdot \pi_{j,0} = \sum_{i=1}^{M} \gamma_i$$

Which confirms that, at the steady state, the aggregate input and output rates for the QN must be the same. We instantiate the above procedure in a simple example. Writing I/O balance equations at each SC is in fact the quickest way to compute the arrival rate vector $\underline{\boldsymbol{\lambda}}$.

Example

Consider the following system, which consists of $m + 1$ M/M/1 SCs.



The routing matrix and the external arrivals are the following:

$$\underline{\mathbf{\Pi}} = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 1 \\ \pi_1 & \pi_2 & \dots & \pi_m & 0 \end{bmatrix}, \underline{\mathbf{\gamma}} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \\ \gamma \end{bmatrix}.$$

Rather than by inverting matrix $(\underline{\mathbf{I}} - \underline{\mathbf{\Pi}}^{\mathbf{T}})$, the above system can be solved faster by considering I/O balance: $\gamma = \lambda_{m+1} \cdot \pi_0$, hence $\lambda_{m+1} = \frac{\gamma}{\pi_0}$. Moreover, from the graph it is clear that:

$$\lambda_i = \lambda_{m+1} \cdot \pi_i = \frac{\gamma}{\pi_0} \cdot \pi_i, \quad 1 \le i \le m$$

Thus, the stability conditions are:

$$\begin{cases} \rho_i = \dfrac{\gamma}{\mu_i} \cdot \dfrac{\pi_i}{\pi_0} < 1 & 1 \le i \le m \\ \rho_{m+1} = \dfrac{\gamma}{\mu_{m+1} \cdot \pi_0} < 1 & \end{cases}$$

And, if the above conditions hold, then it is:

$$p_{\underline{n}} = p(n_1, n_2, \dots, \mathrm{n}_{m+1}) = \prod_{i=1}^{m+1} p_i(n_i) = \prod_{i=1}^{m+1} (1 - \rho_i) \cdot \rho_i^{n_i}$$

Given the above, we can easily compute some performance metrics:

$$E[N] = \sum_{i=1}^{m+1} E[N_i] = \sum_{i=1}^{m+1} \frac{\rho_i}{1 - \rho_i}$$

$$E[N_q] = \sum_{i=1}^{m+1} E\left[N_{q_i}\right] = \sum_{i=1}^{m+1} \frac{{\rho_i}^2}{1 - \rho_i}$$

■

If one wants to know which SC contributes to the overall response time how, some care must be observed. It stands to reason that the overall response time $E[R]$ depends on the SC's response times $E[R_i]$. By Little's law, we can also write $E[R] = E[N]/\gamma_{tot}$, where $\gamma_{tot} = \sum_{i=1}^{M} \gamma_i$, i.e.

$$E[R] = \frac{E[N]}{\gamma_{tot}} = \sum_{i=1}^{M} \frac{E[N_i]}{\gamma_{tot}} = \sum_{i=1}^{M} \frac{E[N_i]}{\lambda_i} \cdot \frac{\lambda_i}{\gamma_{tot}} = \sum_{i=1}^{M} E[R_i] \cdot \frac{\lambda_i}{\gamma_{tot}}$$

Why is there a multiplying coefficient λ_i/γ_{tot} in the last sum? Because a SC can be **visited never, once or more than once** while a job traverses the QN. Each SC's response time $\boldsymbol{R_i}$ must therefore be **multiplied** by the **mean number of visits to that SC**. Define V_i as the RV that counts the number of visits to SC i. Then, we define the **mean residence time at SC *i* as** $E[T_i] = E[R_i] \cdot E[V_i]$. The OJN's response time is the sum of the **residence times,** $E[R] = \sum_{i=1}^{M} E[T_i]$, which implies that:

$$E[V_i] = \frac{\lambda_i}{\gamma_{tot}}$$

The mean number of visits to a SC is equal to its **arrival rate** divided by the **total external arrival rate**. We can now continue the above example and compute the response times and the mean number of visits:

<u>Example (continued)</u>

We compute the OJN's response time and the individual residence times. We already know that

$$\lambda_i = \begin{cases} \gamma \cdot \frac{\pi_i}{\pi_0} & 1 \le i \le m \\ \frac{\gamma}{\pi_0} & i = m+1 \end{cases}, \quad \rho_i = \begin{cases} \frac{\gamma}{\mu_i} \cdot \frac{\pi_i}{\pi_0} < 1 & 1 \le i \le m \\ \frac{\gamma}{\mu_i \cdot \pi_0} < 1 & i = m+1 \end{cases}.$$

Thus, we can compute straightforwardly:

$$E[N_i] = \frac{\rho_i}{1-\rho_i} = \begin{cases} \frac{\gamma \cdot \pi_i}{\mu_i \cdot \pi_0 - \gamma \cdot \pi_i} & 1 \le i \le m \\ \frac{\gamma}{\mu_i \cdot \pi_0 - \gamma} & i = m+1 \end{cases}, \quad E[N] = \sum_{i=1}^{m+1} E[N_i] = \left(\sum_{i=1}^{m} \frac{\gamma \cdot \pi_i}{\mu_i \cdot \pi_0 - \gamma \cdot \pi_i} \right) + \frac{\gamma}{\mu_{m+1} \cdot \pi_0 - \gamma}.$$

Through Little's Law, we can compute the OJN's response time:

$$E[R] = \frac{E[N]}{\gamma_{tot}} = \left(\sum_{i=1}^{m} \frac{\pi_i}{\mu_i \cdot \pi_0 - \gamma \cdot \pi_i} \right) + \frac{1}{\mu_{m+1} \cdot \pi_0 - \gamma}$$

We can obtain the same result by computing the mean number of visits to each SC and their mean residence time:

$$E[V_i] = \frac{\lambda_i}{\gamma_{tot}} = \begin{cases} \frac{\pi_i}{\pi_0} & 1 \le i \le m \\ \frac{1}{\pi_0} & i = m+1 \end{cases}, \quad E[R_i] = \frac{1}{\lambda_i} \cdot \frac{\rho_i}{1-\rho_i} = \begin{cases} \frac{\pi_0}{\mu_i \cdot \pi_0 - \gamma \cdot \pi_i} & 1 \le i \le m \\ \frac{\pi_0}{\mu_i \cdot \pi_0 - \gamma} & i = m+1 \end{cases},$$

hence $E[T_i] = E[V_i] \cdot E[R_i] = \begin{cases} \frac{\pi_i}{\mu_i \cdot \pi_0 - \gamma \cdot \pi_i} & 1 \le i \le m \\ \frac{1}{\mu_i \cdot \pi_0 - \gamma} & i = m+1 \end{cases}.$

One can readily check that $E[R] = \sum_{i=1}^{m+1} E[T_i]$.

■

4.4 Closed Queueing Networks

We now move to analyzing **closed Jackson's networks**. The hypotheses we posit are the same as for OJNs, with a major difference: we assume that **no external arrivals/departures** occur, and that the number of jobs is a **constant *K*,** which is given as an input datum. Everything else stays the same (M/M/C SCs, Markovian routing, etc.).

Note that, if the number of jobs is fixed, the **state space $\mathcal{E}$ has a finite cardinality**, since there will be a finite number of ways to distribute K jobs among M SCs. The fact that it is finite does not imply **that it is small**. In fact, one could check that the following property holds:

$$|\mathcal{E}| = \binom{K+M-1}{M-1}$$

In fact, the above expression is the number of ways to insert $M-1$ "division marks" in a line of $K+M-1$ elements, i.e. all the possible ways to separate K items into M (possibly empty) subsets. The above property implies that $|\mathcal{E}| = O\left(K^{M-1}\right)$. In practical cases (few tens SCs and jobs) the state space is really huge.

Since CJNs are a variation of OJNs, we may attempt to use **the same solution procedure** as for OJNs, keeping in mind that we will have $\gamma_i = 0$ and $\pi_{i,0} = 0\ \forall i$. However, we soon face a major problem. If we attempt to compute the arrival rates at each SC, we get the following equation:

$\underline{\boldsymbol{\lambda}} = 0 + \underline{\mathbf{\Pi}}^{\mathbf{T}} \cdot \underline{\boldsymbol{\lambda}}$. This is a **homogeneous system**; hence it admits:

- no solution (except the null one, which is however unhelpful), if the routing matrix $\underline{\mathbf{I}} - \underline{\mathbf{\Pi}}^T$ has full rank;
- infinite solutions otherwise.

Luckily, our routing matrix **does not have full rank**, since $\sum_{j=1}^{M} \pi_{i,j} = 1\ \forall i$ (recall that $\pi_{i,0} = 0$), which leaves us with infinite solutions. In particular, if $\underline{\mathbf{e}} = [e_1, e_2, \dots, e_M]^T$ is one solution to $\underline{\boldsymbol{\lambda}} = 0 + \underline{\mathbf{\Pi}}^{\mathbf{T}} \cdot \underline{\boldsymbol{\lambda}}$, then $k \cdot \underline{\mathbf{e}}$, $k \in \mathbb{R}$, will be a solution as well. This means that we can only obtain solutions which depend on **a multiplying constant**, and we must find some way to get rid of that constant. Our problem is solved by the following theorem:

Gordon and Newell's Theorem

*In a CJN, take **any solution** to system $\underline{\boldsymbol{\lambda}} = 0 + \underline{\boldsymbol{\Pi}}^{\boldsymbol{T}} \cdot \underline{\boldsymbol{\lambda}}$, and call it $\underline{\mathbf{e}} = [e_1, e_2, \dots, e_M]^T$. Call $\rho_i = e_i/\mu_i$. Then, $p_{\underline{n}} = p(n_1, \dots, n_M) = \frac{1}{G(M,K)} \cdot \prod_{i=1}^{M} f_i(n_i)$, where:*

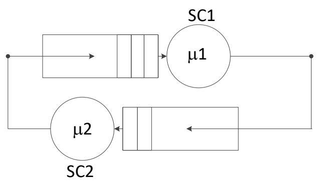
$$f_i(n_i) = \begin{cases} \dfrac{(C_i \cdot \rho_i)^{n_i}}{n_i!} & n_i \le C_i \\ \dfrac{C_i^{C_i} \cdot \rho_i^{n_i}}{C_i!} & n_i \ge C_i \end{cases}$$

And $G(M,K)$ is a normalizing constant such that $\sum_{\underline{n} \in E} p_{\underline{n}} = 1$, i.e., $G(M,K) = \sum_{\underline{n} \in \mathcal{E}} \left(\prod_{i=1}^{M} f_i(n_i)\right)$.

■

Note that expressions $f_i(n_i)$ are **almost** the SS probabilities of an M/M/C_i SC, the only thing missing being a multiplying factor $p_i(0)$. Of course, if a SC is an M/M/1, then $f_i(n_i) = \rho_i^{n_i}$ (this is a special case that we will develop further later on). Moreover, note that we have a **normalizing constant** $G(M,K)$, which is only fair given that we have chosen our initial solution to the routing equation arbitrarily. The fact that we can write $G(M,K)=\sum_{\underline{n}\in\mathcal{E}}\prod_{i=1}^{M}f_i(n_i)$ does not imply that this is the most efficient way to compute it. In fact, the above computation is prohibitive in most cases. Even if we give up computing $G(M,K)$ in a closed form and settle for a numerical solution, we incur the risk of **numerical instability**, since large powers and sums are involved. Luckily for us, there is an **efficient algorithm to compute** $G(M,K)$. This will be discussed in a minute, after giving an example of application of GN's theorem.

Example



Consider a CJN with two M/M/1 SCs. Its routing matrix is trivially $\underline{\mathbf{\Pi}}=\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$. Hence the solution to the routing equation is $e_1=e_2$. Thus, the arrival rates are the same, and they can be defined but for a multiplying constant. This implies that **we can set e_1 to any value**, and the solution will not be affected by it (the normalizing constant $G(M,K)$ will, but the SS probabilities will not). This means that **the smart thing to do is to choose e_1 so as to simplify computations**. In this case, a good choice would be $e_1=\mu_1$ ($e_1=\mu_2$ would do just as well). By doing so, we get $\rho_1=1$, $\rho_2=\mu_1/\mu_2$. By GN's theorem, we get $p(n_1,n_2)=\frac{1}{G(2,K)}\cdot 1^{n_1}\cdot\left(\frac{\mu_1}{\mu_2}\right)^{n_2}$. Having in mind that $n_1+n_2=K$, this can be rewritten as $p(K-n,n)=\frac{1}{G(2,K)}\cdot\left(\frac{\mu_1}{\mu_2}\right)^{n}$, n being the number of jobs at SC 2.

In this case it is quite easy to compute $G(M,K)$ directly from the normalization formula. In fact, it is: $G(2,K)=\sum_{\underline{n}\in\mathcal{E}}\left(\prod_{i=1}^{2}f_i(n_i)\right)=\sum_{n=0}^{K}\left(\frac{\mu_1}{\mu_2}\right)^{n}$. Assuming $\mu_1\neq\mu_2$, it is $G(2,K)=\frac{1-(\mu_1/\mu_2)^{K+1}}{1-\mu_1/\mu_2}$, thus:

$$p(K-n,n)=\frac{1-\mu_1/\mu_2}{1-(\mu_1/\mu_2)^{K+1}}\cdot\left(\frac{\mu_1}{\mu_2}\right)^{n}$$

This is intuitively clear: if $\mu_1>\mu_2$, then SC 1 is faster than SC 2, which means that one is more likely to find **many jobs at SC 2** (i.e., $p(K-n,n)$ should increase with n). Otherwise, if $\mu_1<\mu_2$ then SC 1 is slower than SC 2, which means that $p(K-n,n)$ should **decrease** with n. Finally, if $\mu_1=\mu_2$, it is $G(2,K)=K+1$, and each state is equally likely.

■

We are now left with two problems: first, we need a **computationally efficient** (and, possibly, numerically stable) way to compute $G(M,K)$. Second, we need to compute the performance indexes.

4.4.1 Buzen's convolution algorithm

This algorithm allows one to compute $G(M,K)$ efficiently. If **all SCs are M/M/1,** the algorithm completes in $O(M \cdot K)$ steps and only requires filling up a table. More involved computations are required if some SCs are **load-dependent** (i.e., they have more than one server). The general version of the algorithm can be found on QT textbooks – hereafter we will stick to the simplified one.

For a CJN of M/M/1 SCs, Buzen's algorithm is strikingly simple. Consider the definition $G(M,K) = \sum_{\underline{n}\in\mathcal{E}} \prod_{i=1}^{M} f_i(n_i)$, and **set apart all the vectors in $\mathcal{E}$ where** $n_M = 0$. We can write the above equality as:

$$G(M,K) = \sum_{\substack{\underline{n}\in\mathcal{E} \\ n_M=0}} \prod_{i=1}^{M} f_i(n_i) + \sum_{\substack{\underline{n}\in\mathcal{E} \\ n_M>0}} \prod_{i=1}^{M} f_i(n_i) = \sum_{\substack{\underline{n}\in\mathcal{E} \\ n_M=0}} \prod_{i=1}^{M} {\rho_i}^{n_i} + \sum_{\substack{\underline{n}\in\mathcal{E} \\ n_M>0}} \prod_{i=1}^{M} {\rho_i}^{n_i}$$

Now, in the first addendum, we can stop the product at $M-1$ (since the last factor will be $\rho_M^{n_M} = \rho_M^0 = 1$). On the other hand, in the second addendum the last term in the product can be written as $\rho_M^{n_M} = \rho_M \cdot \rho_M^{\widehat{n_M}}$, with $\widehat{n_M} \geq 0$, since $n_M > 0$. This leads to:

$$G(M,K) = \sum_{\substack{\underline{n}\in\mathcal{E} \\ n_M=0}} \prod_{i=1}^{M-1} {\rho_i}^{n_i} + \rho_M \cdot \sum_{\substack{\underline{n}\in\mathcal{E} \\ \widehat{n_M}\geq 0}} \left(\prod_{i=1}^{M} {\rho_i}^{n_i} \right)$$

One may observe that:

- the first addendum is the normalizing constant of a CJN with ***K* jobs circulating among $\boldsymbol{M-1}$ SCs** (the M^{th} one is in fact empty). In fact, the state space for that CJN is $\mathcal{E} \equiv \{(n_1, n_2, \ldots, n_{M-1}, 0) : n_i \geq 0, \sum_{i=1}^{M-1} n_i = K\}$. Thus, the first addendum is $G(M-1,K)$.
- In the second addendum, **we have "set apart" one factor $\boldsymbol{\rho_M}$**, i.e. one of the circulating jobs, "pinning" it at SC M. The sum is therefore **the normalizing constant of a CJN with $\boldsymbol{K-1}$ jobs circulating among *M* SCs** (the K^{th} job is in fact the one "pinned" on SC M). The state space for this CJN is $\mathcal{E} \equiv \{(n_1, n_2, \ldots, n_{M-1}, \widehat{n_M}) : n_i \geq 0,\ \widehat{n_M} \geq 0,\ (\sum_{i=1}^{M-1} n_i) + \widehat{n_M} = K-1\}$. Thus, we can replace that sum with $G(M,K-1)$.

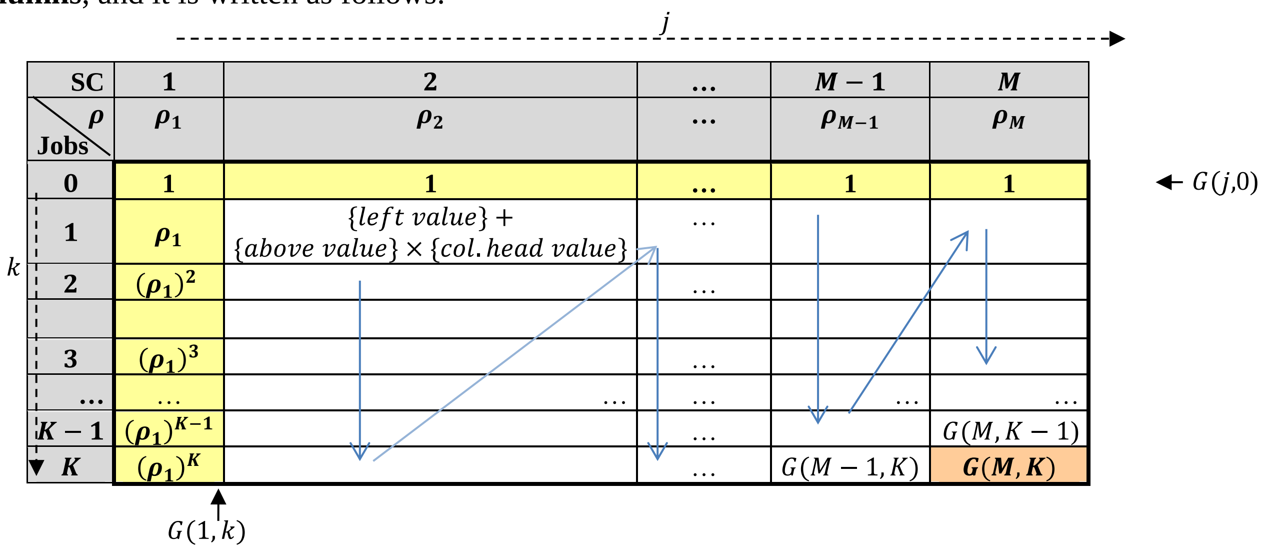
This yields the following **recursive formula**:

$$G(M,K) = G(M-1,K) + \rho_M \cdot G(M,K-1)$$

If we know the initial conditions for the above formula, we can compute $G(M,K)$ for arbitrary values of M and K. However, these initial conditions are quite straightforward. We need to compute:

- $G(1,k)$, i.e., the normalizing constant of a CJN with **one SC and k jobs**. But the state space $\mathcal{E}$ of that CJN includes **only one state**, i.e. k jobs at the one and only SC. Thus, it is $G(1,k) = \sum_{\underline{n}\in\mathcal{E}} \prod_{i=1}^{M} \rho_i^{n_i} = \rho_1^{\,k}, \forall k \geq 0$.
- $G(j,0)$, i.e., the normalizing constant of a CJN with **j SCs and 0 circulating jobs**. Again, for a CJN like this, the state space $\mathcal{E}$ includes only one state, i.e. the one where all the j SCs are empty. Thus, $G(j,0) = \sum_{\underline{n}\in\mathcal{E}} \prod_{i=1}^{j} \rho_i^{n_i} = \prod_{i=1}^{j} \rho_i^{\,0} = 1, \forall j \geq 0$.

Given the above initializations, we can run the algorithm with the help of a simple **table** (this is even quicker if you use a spreadsheet software). The table has **$K+1$ rows** (from 0 to K included) and **M columns**, and it is written as follows:



SC	1	2	…	$M-1$	M
Jobs \ ρ	ρ_1	ρ_2	…	ρ_{M-1}	ρ_M
0	1	1	…	1	1
1	ρ_1	$\{left\ value\} + \{above\ value\} \times \{col.\ head\ value\}$	…		
2	$(\rho_1)^2$		…		
3	$(\rho_1)^3$		…		
…	…	…	…	…	…
$K-1$	$(\rho_1)^{K-1}$		…		$G(M,K-1)$
K	$(\rho_1)^K$		…	$G(M-1,K)$	$G(M,K)$

Once the first row and column have been initialized, we start filling the table from the **top left corner,** going down first, and then right. In each cell, the only operation required is to **multiply the value above** by the one **at the heading of the column**, and add to this result the **value in the cell to the left**. At the end the required constant $G(M,K)$ will appear at the bottom-right corner. We will see later on that **other values** in the table will be useful as well. If the values in the above table get **numerically unstable**, then you may want to **scale up/down** the solution that you chose when you first solved the routing equation. A good choice is to select an initial solution such that the ρ_i are **around one** as much as possible. If the algorithm is run on a spreadsheet, that initial value can be written in a cell and modified at will to improve numerical stability, without having to redo all the computations.

4.4.2 Performance indexes in Closed Queueing Networks

Given $G(M,K)$, you can compute $p_{\underline{n}}$ for every vector $\underline{n} \in \mathcal{E}$. Thus, you can compute all the performance indexes. These are **amazingly simple**, and they involve ratios of $G(i,j)$ for some indexes i,j.

CDF, PMF and mean number of jobs at a single SC

We want to compute $F_i(j) = P\{N_i \leq j\}$ and $p_i(j) = P\{N_i = j\}$.

Let us first compute $P\{N_i \geq j\}$. In fact, this can be computed by "setting apart" j jobs at SC i, and repeating the same reasoning of Buzen's algorithm. It is:

$$P\{N_i \geq j\} = \sum_{\substack{\underline{n}\in\mathcal{E} \\ n_i \geq j}} p_{\underline{n}} = \frac{1}{G(M,K)} \cdot \sum_{\substack{\underline{n}\in\mathcal{E} \\ n_i \geq j}} \prod_{h=1}^{M} \rho_h{}^{n_h} =$$

$$= \frac{1}{G(M,K)} \cdot \sum_{\substack{\underline{n}\in\mathcal{E} \\ \hat{n}_i \geq 0}} \left(\prod_{\substack{h=1 \\ h\neq i}}^{M} \rho_h{}^{n_h} \cdot \rho_i^{\hat{n}_i + j} \right) = \frac{G(M,K-j)}{G(M,K)} \cdot \rho_i^j$$

Both $G(M,K-j)$ and $G(M,K)$ can be read in the last column of the table. From the above we get:

$$F_i(j) = P\{N_i \leq j\} = 1 - P\{N_i \geq j+1\} = 1 - \frac{G\big(M,K-(j+1)\big)}{G(M,K)} \cdot \rho_i^{j+1}$$

$$p_i(j) = P\{N_i \geq j\} - P\{N_i \geq j+1\} = \frac{\rho_i^j}{G(M,K)} \cdot \big[G(M,K-j) - \rho_i \cdot G\big(M,K-(j+1)\big)\big]$$

Note that, if we define $G(M,x) = 0$ when $x < 0$, we can also compute both the CDF and the PMF with $j = K$. From $p_i(j)$ we can compute $E[N_i]$:

$$E[N_i] = \sum_{j=1}^{K} j \cdot \frac{\rho_i^j}{G(M,K)} \cdot \big[G(M,K-j) - \rho_i \cdot G\big(M,K-(j+1)\big)\big]$$

$$= \frac{1}{G(M,K)} \cdot \left[\sum_{j=1}^{K} j \cdot \rho_i^j \cdot G(M,K-j) - \sum_{j=1}^{K-1} j \cdot \rho_i^{j+1} \cdot G\big(M,K-(j+1)\big) \right]$$

$$= \frac{1}{G(M,K)} \cdot \left[\sum_{j=1}^{K} j \cdot \rho_i^j \cdot G(M,K-j) - \sum_{j=1}^{K-1} (j+1) \cdot \rho_i^{j+1} \cdot G\big(M,K-(j+1)\big) + \sum_{j=1}^{K-1} 1 \cdot \rho_i^{j+1} \cdot G\big(M,K-(j+1)\big) \right]$$

$$= \frac{1}{G(M,K)} \cdot \left[\sum_{j=1}^{K} j \cdot \rho_i^j \cdot G(M,K-j) - \sum_{h=2}^{K} h \cdot \rho_i{}^h \cdot G(M,K-h) + \sum_{h=2}^{K} \rho_i{}^h \cdot G(M,K-h) \right]$$

$$= \frac{1}{G(M,K)} \cdot \left[\rho_i \cdot G(M,K-1) + \sum_{h=2}^{K} \rho_i{}^h \cdot G(M,K-h) \right]$$

$$= \frac{1}{G(M,K)} \cdot \left[\sum_{h=1}^{K} \rho_i{}^h \cdot G(M,K-h) \right]$$

$$= \sum_{j=1}^{K} P\{N_i \geq j\}$$

Joint probabilities at two or more SCs

Similarly, we can compute **joint probabilities for two or more SCs**, e.g. $P\{N_i \geq j, N_l \geq m\}$. The trick is always the same, i.e. to **set apart j jobs at SC i and m jobs at SC l.** The result is:

$$P\{N_i \geq j, N_l \geq m\} = \frac{G\big(M, K-(j+m)\big)}{G(M,K)} \cdot {\rho_i}^j \cdot {\rho_l}^m$$

Utilization

For a single SC, this can easily be computed as $U_i = P\{N_i \geq 1\} = \rho_i \cdot \frac{G(M,K-1)}{G(M,K)}$.

If we need the probability that **two or more SCs** are simultaneously busy, we easily get:

$$U_{i,l} = P\{N_i \geq 1, N_l \geq 1\} = \frac{G(M,K-2)}{G(M,K)} \cdot \rho_i \cdot \rho_l,$$

and so on.

Throughput of a SC

The throughput of a SC i is equal to μ_i when the latter is busy, hence:

$$\gamma_i = \mu_i \cdot U_i = \frac{G(M,K-1)}{G(M,K)} \cdot \mu_i \cdot \rho_i.$$

Response time of a SC

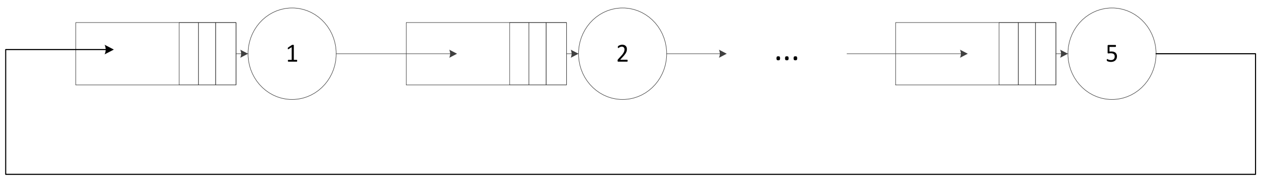
The response time can be computed through Little's law, as:

$$E[R_i] = E[N_i]/\gamma_i = \frac{\sum_{h=1}^{K} {\rho_i}^h \cdot G(M,K-h)}{e_i \cdot G(M,K-1)}.$$

Example

Consider a CJN of $M = 5$ **identical SCs** in a tandem, having $K = 8$ circulating jobs. Compute:

1) the mean number of jobs at a SC;
2) the mean response time of a SC, and the mean **circuit time.** Explain the result;
3) the probability that **all SCs** are simultaneously busy.



The routing matrix is:

$$\underline{\mathbf{\Pi}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Trying to solve $\underline{\boldsymbol{\lambda}} = \underline{\boldsymbol{\Pi}}^T \cdot \underline{\boldsymbol{\lambda}}$ obviously yields $e_i = k$. Thus, one may conveniently set $e_i = \mu$, which leads to $\rho_i = 1$. This means that the SS probabilities will be **uniform** (which is intuitively clear, given the symmetry of the CJN). In fact, it is:

$$p_{\underline{n}} = \frac{1}{G(M,K)} \cdot \prod_{i=1}^{M} 1^{n_i} = \frac{1}{G(M,K)}$$

Moreover, it is:

$$G(M,K) = \sum_{\underline{n}\in\mathcal{E}} \prod_{i=1}^{M} 1^{n_i} = |\mathcal{E}| = \binom{K+M-1}{M-1} = \binom{12}{4} = 495$$

since all states are equally likely. However, we will still run Buzen's convolution algorithm, both as a cross check *and* because we need intermediate values in the table to compute performance indexes:

SC	**1**	**2**	**3**	**4**	**5**
$\boldsymbol{\rho}$ / **jobs**	**1**	**1**	**1**	**1**	**1**
0	1	1	1	1	1
1	1	2	3	4	5
2	1	3	6	10	15
3	1	4	10	20	35
4	1	5	15	35	70
5	1	6	21	56	126
6	1	7	28	84	210
7	1	8	36	120	330
8	1	9	45	165	495

1) The mean number of jobs per SC is:

$$E[N_i] = \frac{1}{G(M,K)} \cdot \left[\sum_{h=1}^{K} \rho_i^{\,h} \cdot G(M,K-h)\right]$$
$$= \frac{1}{G(M,K)} \cdot \left[\sum_{h=1}^{K} G(M,K-h)\right]$$
$$= [\ldots] = \frac{8}{5}$$

The latter could have been obtained reasoning by symmetry.

2) The utilization of each of the SC is:

$$U_i = P\{N_i \geq 1\} = \rho_i \cdot \frac{G(M,K-1)}{G(M,K)} = \frac{330}{495} = \frac{2}{3}$$

Thus, the throughput is $\gamma_i = \frac{2}{3} \cdot \mu$, and the mean response time at a SC is:

$$E[R_i] = \frac{E[N_i]}{\gamma_i} = \frac{8}{5} \cdot \frac{3}{2\cdot\mu} = \frac{12}{5} \cdot E[t_s]$$

Thus, the mean circuit time is $M \cdot E[R_i] = 5 \cdot \frac{12}{5} \cdot E[t_s] = 12 \cdot E[t_s]$. This result has an intuitive explanation: each job has to be served ***M* times** in a full circuit. While traveling along the circuit, it will find ahead of itself **other $K-1$ jobs**. Therefore, the average cycle time should be $M+K-1$ times the average service time, which is in fact the above result.

3) The probability that **all SCs are busy** at the same time is:

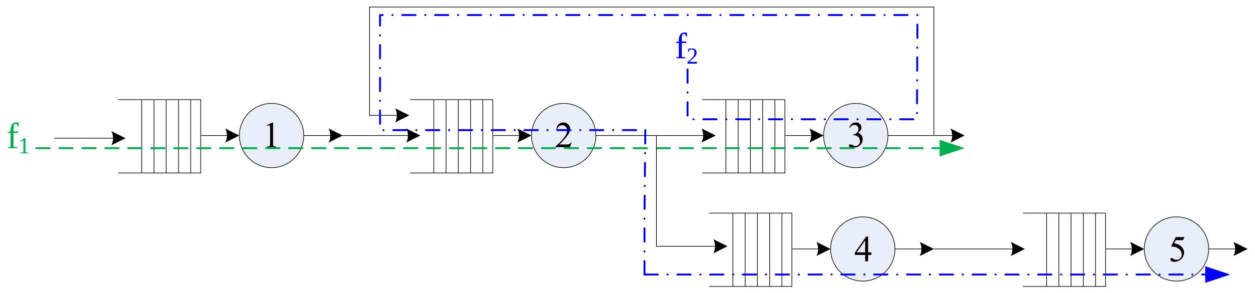
$$\frac{G(M,K-M)}{G(M,K)}\cdot 1\cdot 1\ldots\cdot 1=\frac{35}{495}=\frac{7}{99}$$

■

4.5 Classed queueing networks

The fact that OJNs and CJNs have **probabilistic routing** is somewhat unpleasant. Quite often, systems that we would like to model as QNs offer service to **several classes of jobs**, and routes are different depending **on the class**. For instance, you may have a **computer network** where several flows (each having its source and destination) traverse a number of routers. In that case, the routes for the flows will be different, but they will be **fixed**, and not probabilistic.

The same model can be used to represent a **manufacturing plant**, where each **product line** has a different **line of assembly**, which involves visiting several assembly points in a fixed order. One assembly point may provide service for different products simultaneously. We would model product lines as **classes of jobs**, assembly points as **service centers**, and the sequence of assembly points to be visited as a **fixed per-class route**.



For instance, we might have a QN where:

- Flow 1 traverses SCs 1, 2, 3 (call it *route R1*).
- Flow 2 traverses SCs 3, 2, 4, 5 (call it *route R2*).

In this case, there is no probabilistic routing, meaning that **all jobs** belonging to flow 1 will follow route R1, and all packets from flow 2 will traverse route R2. Jackson's hypothesis of Markovian routing does not hold anymore.

If, instead, we assume that **routing probabilities depend on the class of a job**, we can just assign different classes to flow 1 and flow 2, and write down two routing matrices (one per flow/class) that describe the routing:

$$\underline{\mathbf{\Pi_1}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \underline{\mathbf{\Pi_2}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Nothing prevents adding to this model (static) load balancing per class, e.g., f1's packets that leave node 2 going to **either node 4 or node 3**, with given probabilities.

4.5.1 Classed queueing systems in isolation

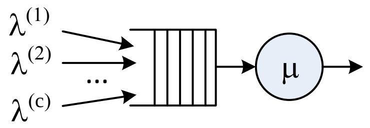
Let us take a look at classed systems in isolation first. Defining the **state of a SC** in a classed network is not so easy. In fact, the behavior of the SC is not determined anymore by the **number of jobs in the system**, but also by the **class of each and every job**. In fact, the state of a SC having n jobs is an n-tuple of class indicators:

$$S = \left(c^{(1)}, c^{(2)}, \dots c^{(n)}\right),$$

meaning that job 1 (the one at the head of the queue) is of class $c^{(1)}$, job 2 is of class $c^{(2)}$, etc. This implies that analyzing classed QNs is somewhat **notationally heavy**, although conceptually simple. We give the following results (without proof – please refer to any QT book if you need one).

Theorem 1 (classed M/M/1)

Take an M/M/1 *SC with* c *classes* $1 \dots c$. *Assume that the arrival processes of each class* j *are independent. Call* $\lambda^{(j)}$ *their arrival rate, and let* $\lambda = \sum_{j=1}^{c} \lambda^{(j)}$. *Call* $\rho = \lambda/\mu$. *If* $\rho < 1$, *then we have:*



$$p_S = \frac{\lambda^{(c^{(1)})} \cdot \lambda^{(c^{(2)})} \cdot \dots \lambda^{(c^{(n)})}}{\mu^n}(1-\rho)$$

■

In other words, to obtain the SS probability of a state S, you multiply the arrival rates **of the class that each job belongs to**, and everything else is unchanged with respect to the formula for an un-classed M/M/1. Note that:

- if you have only one class, the above formula collapses to the classical M/M/1's, and it only depends on the **overall number** of jobs;
- in a classed system, the probability **is the same for any two states with the same number of jobs per class, arranged in a different order** (the product at the numerator is in fact commutative). Therefore, it depends only on the **number of jobs of each class,** not on their position in the queue.

This last property allows us to find something useful:

Corollary 1 (classed M/M/1)

The probability that a classed M/M/1 SC has n jobs overall (whichever their class) is:

$$P\{n\ jobs\ at\ SC\} = \rho^n \cdot (1-\rho)$$

■

Proving the above is easy. In fact, it is:

$$P\{n\ jobs\ at\ SC\} = \sum_{c^{(1)}} \sum_{c^{(2)}} \dots \sum_{c^{(n)}} p_S = \frac{(1-\rho)}{\mu^n} \sum_{c^{(1)}} \lambda^{(c^{(1)})} \sum_{c^{(2)}} \lambda^{(c^{(2)})} \dots \sum_{c^{(n)}} \lambda^{(c^{(n)})} = \frac{(1-\rho)}{\mu^n} \cdot \lambda^n$$
$$= \rho^n \cdot (1-\rho)$$

But then, we do not even **need** a proof. If the arrival processes for each class are **independent**, all we need is to acknowledge that – for the sole purpose of counting the jobs inside the system – the result must be indistinguishable from an **unclassed** M/M/1 whose arrival rate is $\lambda = \sum_{j=1}^{c} \lambda^{(j)}$.

4.5.2 Open classed queueing networks

It turns out that **classed Open QNs** still admit a product form, i.e. the probability of a certain network state is the product of the probabilities of the states at the single SCs.

Theorem 2 (product form for classed OJNs)

In a classed OJN of M M/M/1 SCs, as long as $\rho_i < 1, 1 \le i \le M$, steady-state probabilities admit a product form:

$$p_{\{S_1,S_2,\dots S_M\}} = \prod_{i=1}^{M} p_{S_i}$$

Where p_{S_i} are those of Theorem 1. In order to compute $\lambda_i = \sum_{j=1}^{c} \lambda_i^{(j)}$, the arrival rate $\lambda_i^{(j)}$ of class-j jobs at SC i must be determined by solving c times the per-class versions of arrival-rate equations, i.e.,

$$\underline{\boldsymbol{\lambda}^{(\mathbf{j})}} = \underline{\boldsymbol{\gamma}^{(\mathbf{j})}} + \underline{\boldsymbol{\Pi}^{(\mathbf{j})}}^{\mathbf{T}} \cdot \underline{\boldsymbol{\lambda}^{(\mathbf{j})}}$$

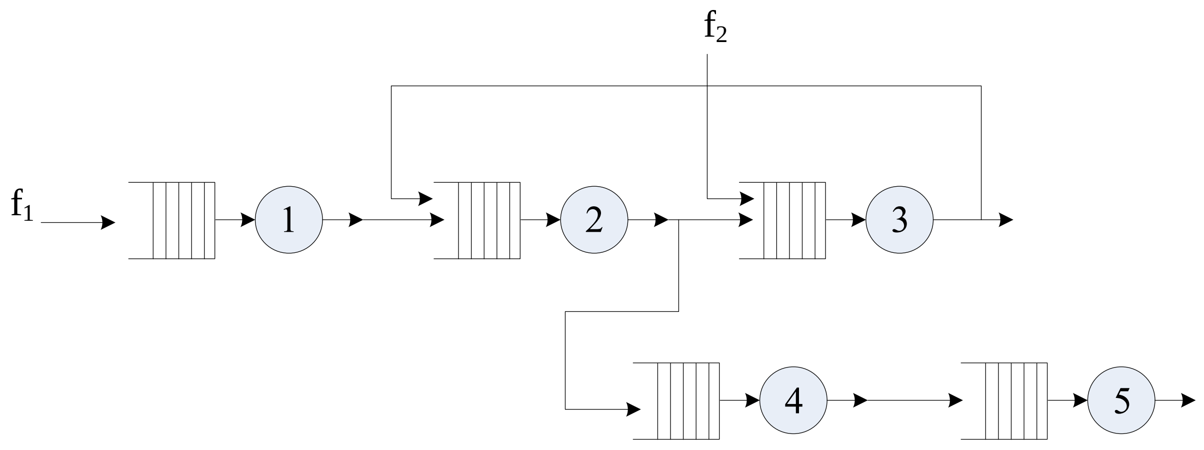
Moreover, the probability of finding $\underline{n} = (n_1, n_2, \dots n_M)$ jobs at the M SCs (again, whichever their class) has a product form too:

$$p_{\underline{n}} = \prod_{i=1}^{M} P\{n_i\ jobs\ at\ SC\ i\} = \prod_{i=1}^{M} \rho_i^{n_i} \cdot (1-\rho_i)$$

■

This means that the only added complication when dealing with **classed** OJNs is that you have to compute the arrival rates **for each class individually**, and then **aggregate them** into per-SC arrival rates. All things considered, the analysis of classed QNs is not conceptually more difficult – it just requires a few more computations.

4.5.3 Exercise – response times in a routed network



- Flow 1 traverses SCs 1, 2, 3 (call it *route R1*).
- Flow 2 traverses SCs 3, 2, 4, 5 (call it *route R2*).

Compute the **average end-to-end delay** for packets of flow 1 and flow 2.

Call $\gamma^{(1)}$ the arrival rate of flow/class 1, and $\gamma^{(2)}$ that of flow/class 2. We straightforwardly obtain the arrival rates for each class at each SC through visual inspection:

- Flow 1: $\lambda_1^{(1)} = \lambda_2^{(1)} = \lambda_3^{(1)} = \gamma^{(1)}, \lambda_4^{(1)} = \lambda_5^{(1)} = 0$
- Flow 2: $\lambda_3^{(2)} = \lambda_2^{(2)} = \lambda_4^{(2)} = \lambda_5^{(2)} = \gamma^{(2)}, \lambda_1^{(2)} = 0$

Thus, we can compute per-SC arrival rate values:

$$\lambda_1 = \lambda_1^{(1)} + \lambda_1^{(2)} = \gamma^{(1)}$$

$$\lambda_2 = \lambda_2^{(1)} + \lambda_2^{(2)} = \gamma^{(1)} + \gamma^{(2)}$$

$$\lambda_3 = \lambda_3^{(1)} + \lambda_3^{(2)} = \gamma^{(1)} + \gamma^{(2)}$$

$$\lambda_4 = \lambda_4^{(1)} + \lambda_4^{(2)} = \gamma^{(2)}$$

$$\lambda_5 = \lambda_5^{(1)} + \lambda_5^{(2)} = \gamma^{(2)}$$

Assuming that $\mu_i > \lambda_i \; \forall i$, the mean number of jobs at each SC is finite, hence the mean response time at each SC is (by Little's law):

$$E[R_i] = \frac{\rho_i}{1-\rho_i} \cdot \frac{1}{\lambda_i} = \frac{1}{\mu_i - \lambda_i}$$

Therefore:

- For flow 1, we get:

$$E[R] = \frac{1}{\mu_1 - \gamma^{(1)}} + \frac{1}{\mu_2 - (\gamma^{(1)} + \gamma^{(2)})} + \frac{1}{\mu_3 - (\gamma^{(1)} + \gamma^{(2)})};$$

- for flow 2, we get:

$$E[R] = \frac{1}{\mu_3 - (\gamma^{(1)} + \gamma^{(2)})} + \frac{1}{\mu_2 - (\gamma^{(1)} + \gamma^{(2)})} + \frac{1}{\mu_4 - \gamma^{(2)}} + \frac{1}{\mu_5 - \gamma^{(2)}}$$

The above result makes perfect sense intuitively. Note that you **cannot** obtain the same result modeling the above network as an unclassed OJN, since you would need to choose routing probabilities (namely, $\pi_{2,4}$ and $\pi_{2,3}$) arbitrarily. The probability that a packet leaving SC 2 is routed to SC 3 depends on its belonging to flow 1, hence on flow 1's sending rate compared to flow 2's. Therefore, routing would not be state-independent, and you would not be able to use Jackson's theorem.

4.6 Closing remarks on FCFS queueing networks

All the QNs we have discussed so far have service centers with **FCFS queueing**. In order to analyze FCFS QNs, we require that **service times are exponential**, otherwise we simply cannot perform the analysis (without exponential service times there is no product form).

FCFS queueing is typical of:

- **Computer networks**, where jobs are packets and they are transmitted atomically;
- **Human systems**, where jobs are in fact human beings, and they are served atomically too.

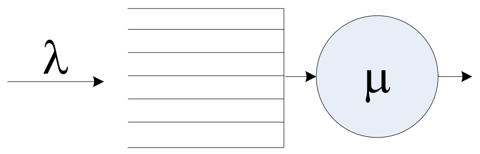
FCFS QNs (especially classed ones) are good models for the above systems. In a computer network, servers model the links between routers. If packets have **constant size**, the exponential service time hypothesis does not hold, yet the results we obtain through Markovian analysis (e.g., $E[R]$) can be proved to be **upper bounds** of those you would compute in the real systems. Therefore, QT may give you interesting results even when its hypotheses are not met.

Class-dependent routing is a nice modeling tool, since it allows to model per-flow network routing. The assumption that the service time depends on the **server**, and not on the **class**, is somewhat limiting. We would like to have **class-dependent service times**, but these models do not allow it.

Note that we have dealt with **open classed QNs** only. The product form extends to **closed classed QNs**, and to **mixed classed QNs** (i.e., those that are *open* w.r.t. one class and *closed* w.r.t. another). These generalizations can be found on most QT books.

5 Processor-sharing queueing systems

When modeling computer systems as queueing systems, we would like to model the tasks that run on a processor as jobs, and the CPU as a server. In this case, however, tasks **share the processor time**, often equally through (e.g.) round-robin scheduling. In this case, modeling things in terms of FCFS queueing is not a good idea. For K tasks sharing a processor with a service rate μ, it would be more reasonable to assume that **all jobs are served simultaneously at a rate** μ/K, where "simultaneous" service is an abstraction for a fast-paced time-sharing service, where the granularity of the time slices can be neglected.

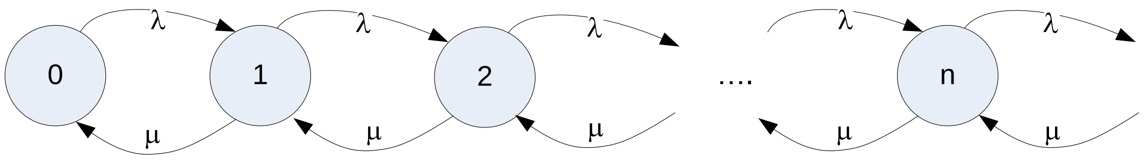


We call a system like this a **processor-sharing queueing system**. In Kendall's notation, we denote it as M/M/1/PS (if both arrivals and services are memoryless), as opposed to the M/M/1/FCFS, which we had only called M/M/1 thus far, omitting to specify the queueing discipline. Luckily, there are interesting results covering PS systems, including networks of PS systems (which do admit product forms). PS systems are in fact **easier to analyze** than FCFS systems – because **they do not have queueing**.

Let us model an M/M/1/PS **through a CTMC**. We can do this since interarrivals and service times are memoryless, hence the **number of jobs in the system** should be the only necessary state characterization.

- When there are **zero jobs**, the only thing that can happen is a new job arrival, with a rate λ. Clearly, arrivals occur at a rate λ, whatever the state. The only thing that requires some discussion is thus the **service rate.**
- When there is **one job**, it has the whole server to itself. There is no difference between this case and the one of a M/M/1/FCFS system holding just one job, hence the arc going to the left has a rate μ.
- When there are **two jobs**, it is as if each was served by a dedicated server of rate $\mu/2$. Note that when the second job joins the system, the first one has already been served for a while. However, its residual service time is still exponential, due to the memoryless property. Therefore, the arc going to the left is the one of an M/M/2/FCFS system when two jobs are in, i.e. it has a rate $2 \cdot \mu/2 = \mu$.
- When the system is in state n, the transition to $n-1$ will occur when the minimum among n (residual) service times expires. However, all service times are exponential with a rate μ/n, hence the arc going to the left has a rate $n \cdot \mu/n = \mu$.

The conclusion is that the CTMC is **the same as an M/M/1/FCFS'**. Hence, the SS probabilities are the same.

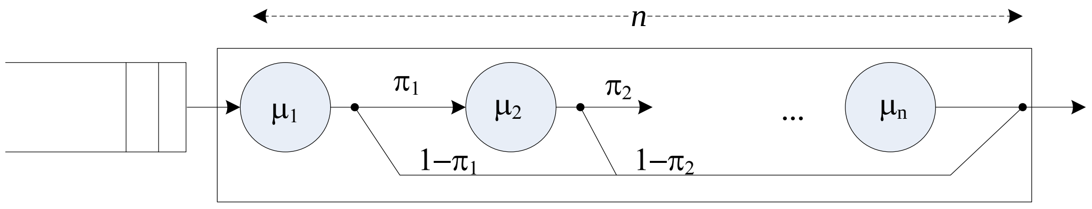


If $\rho < 1$, then $p_n = (1-\rho)\cdot\rho^n$, hence $E[N] = \frac{\rho}{1-\rho}$, $E[R] = \frac{E[N]}{\lambda} = \frac{1/\mu}{1-\rho} = \frac{1}{\mu-\lambda}$, exactly the same as an FCFS's. Note that $E[N_q], E[W]$ do not make sense here.

Recall that in a PS system there is no queueing. In fact, PS systems are (almost) **insensitive to the distribution of service times**. As long as service times are Coxian (see below), then the results are the same, and the only relevant parameter is the **mean service time $1/\mu$.** Therefore, all M/Cox/1/PS systems (which in practice means almost all M/G/1/PS systems) behave like the one above. Recall that **when there is no queueing**, the distribution of service times does not really matter (only its mean value does).

Coxian distributions are particular cases of Phase-Type distributions, which can accommodate pretty much everything, at least in an approximate way. They can be described as follows:

Imagine that the service includes **at most n stages**. Each of them takes an exponential time (possibly with different means), and all stages are independent of each other. After each stage j, you can either go to the next stage, with probability π_j, or leave the server, with probability $1-\pi_j$. A Coxian collapses to an Erlang if $\pi_j = 1\ \forall j$, and all stages are identical.



Coxian distributions can approximate distributions with:

- $CoV < 1$: e.g., an Erlang one, or even a deterministic one (which is the limit of an Erlang for $n \to \infty$, recall the CLT);

- $CoV > 1$, e.g., those with a large variability, even approximating heavy-tailed ones.

For instance, a (possibly crude) Coxian approximation of a distribution with $E[X] = m$ and $CoV[X] = c$, with $c \geq 0.25$ can be obtained by matching it to a 2-stage Coxian as follows:

$$\mu_1 = 2m, \quad \pi_1 = \frac{1}{2c^2}, \qquad \mu_2 = \mu_1 \cdot \pi_1$$

More involved computations are required to obtained better matches (e.g., matching the first three moments). If you want to know more about how to fit a generic distribution to a Coxian, check the literature. For our purpose, it is enough to say that most distributions can be approximated with a Coxian, and the approximation can be made arbitrarily close by increasing the number of stages (and, consequently, the complexity of the computations).

What are the SS probabilities in an M/Cox/1/PS system? They are surprisingly easy to compute. The first question to address is **what is the mean service time** for a job in such a system? This is an easy question:

$$E[t_s] = \frac{1}{\mu_1} + \pi_1 \cdot \frac{1}{\mu_2} + \pi_1 \cdot \pi_2 \cdot \frac{1}{\mu_3} + \cdots = \sum_{i=1}^{n} \frac{1}{\mu_i} \cdot \prod_{j=1}^{i-1} \pi_j$$

Then call $\rho = \lambda \cdot E[t_s]$, and the SS probabilities are simply:

$$p_n = \rho^n \cdot (1 - \rho),$$

which is almost too good to be true. They have the same shape as an M/M/1's, and the only difference is that you need to compute the mean service time considering the traversal of up to n stages. Note that this result does **only hold because there is no queueing in a PS system**. Queueing creates variability. An M/Cox/1/FCFS system would behave very differently (you can analyze it using matrix analytic methods and check for yourselves).

The last, very important thing about QNs of PS servers is the following:

Theorem[9]

Open/closed networks of M/Cox/1/PS systems, with or without classes, admit a product form, as long as external arrivals (if any) are Poissonian and routing (possibly per class) is Markovian.

■

The above thesis implies that we can study networks of PS systems, with **generic service times** (with a pinch of salt) as if they were OJNs or CJNs, as long as we use the mean value of whatever service time distribution we have at the nodes in place of $1/\mu$.

Over the last decades, more and more QNs have been found to admit product forms, including:

- other **service disciplines** (e.g., LCFS with *preemption* and *resume*);

[9] This thesis is part of the famous "BCMP theorem" (named after the initials of its inventors), published in 1975, which sets very general conditions under which QNs admits product forms. Its thesis is indeed more general than this one, but such generality often baffles first-time readers. Interested students may want to check the above paper for more details.

- QNs where jobs may **change class** from one SC to the other, with known probabilities;
- **load-dependent** and **class-dependent** service times at the nodes;
- **special forms** of state-dependent routing;
- special cases of **blocking** and **finite queues**.

Therefore, there is a good chance that any model you come up with can be accommodated in the above framework.

6 Exercises

6.1 Single-queue systems

6.1.1 Problem

A packet switching node is modeled via an M/M/1 queueing system. The link speed C is 1400 bits/s, and the mean packet length $E[M]$ is 10 bits. Under some load conditions, the mean number of packets in the system is $E[N] = 50$ at the steady state. Compute:

a) the packet arrival rate λ;
b) the mean response time $E[R]$;
c) the mean number of packets in the queue $E[N_q]$;
d) the mean queue waiting time $E[W]$;
e) the 95th percentile of the response time π_{95};
f) the link speed such that the above is equal to 1ms.

Solution

a) We know that $E[M] = 10$ and that $C = 1400$. This allows us to compute $\mu = C/E[M] = 140$. Furthermore, since we have $E[N] = 50$, this means that the system is **positive recurrent**, hence it is $E[N] = \rho/(1-\rho)$. Therefore, we obtain $\rho = E[N]/(1+E[N]) = 50/51$.
Note that it stands to reason that $\rho \sim 1$, since the mean number of packets is high.
Now, since $\rho = \lambda/\mu$, we obtain $\lambda = \mu \cdot \rho = \frac{C}{E[M]} \cdot \frac{E[N]}{1+E[N]} = 137.26$.

b) The average response time can be obtained by applying Little's Theorem to the system queue+server in the steady state. Therefore, it is $E[R] = E[N]/\lambda = 0.364s$.

c) The mean number of packets in the queue can be computed directly as: $E[N_q] = E[N] - (1-p_0) = E[N] - \rho = 49.02$, since the system has one server.

d) There are two ways to compute $E[W]$. The first one is by applying Little's theorem to the system "queue" in the steady state. This gets us: $E[W] = E[N_q]/\lambda = 0.357s$. Equivalently, one may leverage linearity of expectations and write $E[W] = E[R] - E[t_S]$, where t_S is the service time. It is $E[t_S] = b = 1/\mu$, i.e. $b = 1/\mu = 1/140 = 0.00714$. Then, $E[W] = E[R] - b$ yields the same result.

e) The distribution of the response time is exponential $F(y) = P\{R \le y\} = 1 - e^{-y/E[R]}$. In order to compute the 95th percentile, we need to solve $F(\pi_{95}) = 1 - e^{-\pi_{95}/E[R]} = 0.95$, hence:

$$\begin{aligned} &e^{-\pi_{95}/E[R]} = 0.05 \\ &-\pi_{95}/E[R] = \ln(0.05) \\ &\pi_{95} = -E[R]\ln(0.05) \\ &\pi_{95} \approx 3E[R] = 1.092s \end{aligned}$$

Note that the 95th percentile of the exponential is actually 2.996 times its mean value. Therefore, $\pi_{95} \approx 3E[R]$ is an accurate estimate.

f) The required equation is $\pi_{95} \approx 3E[R] = 10^{-3}$. Now,

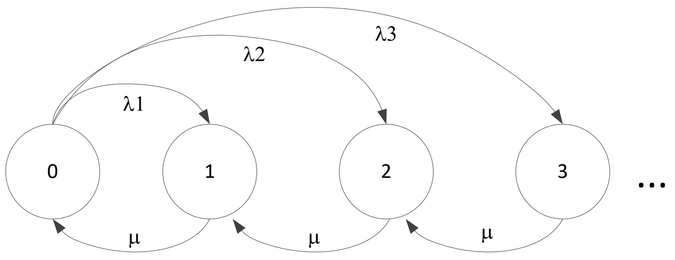
$$E[R] = \frac{E[N]}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{\lambda/\mu}{\lambda(1-\lambda/\mu)} = \frac{1}{\mu-\lambda} = \frac{1}{C/E[M]-\lambda}$$

Therefore, we have:

$$\begin{aligned} &\frac{3}{C/E[M]-\lambda} = 10^{-3} \\ &C = E[M]\cdot\left(3\cdot10^{3}+\lambda\right) \\ &C \approx 10\cdot(3000+137.26) = 31372b/s \end{aligned}$$

6.1.2 Problem

Assume that a packet switching node is modeled via an M/M/1 queueing system whose transition diagram is reported below. The state of the system is represented by the **number of packets** in the system, although value λ_i specifies the arrival rate of **messages** carrying i packets each.



a) Use the CTMC to provide a physical interpretation of the behavior of this system;
b) write down the global equilibrium equations and determine the stability condition.

Assuming that $\lambda_n = \lambda/n^3$, compute:

c) the steady-state probabilities;
d) the throughput;
e) the utilization and the mean service time.

Note: it may be useful to know that $\sum_{n=1}^{+\infty}\frac{1}{n^2} = \frac{\pi^2}{6} \approx 1.645$.

Solution

a) The physical interpretation is the following: the system has **bulk arrivals.** It accepts messages with possibly many packets in a burst. On receipt of a **message** (containing an arbitrary number of packets) the system **stops** accepting messages (**line blocking**) and starts serving the available packets, until it is empty again.

b) We can write down the global equilibrium equations at the steady state by circling each state and taking into account outgoing and incoming arcs.

$$p_0 \cdot \left(\sum_{n=1}^{+\infty} \lambda_n\right) = \mu \cdot p_1$$
$$\mu \cdot p_k = \mu \cdot p_{k+1} + \lambda_k \cdot p_0 \qquad (k > 1)$$

From the first equation, we get $p_1 = \frac{1}{\mu} \cdot \left(\sum_{n=1}^{+\infty} \lambda_n\right) \cdot p_0$.

From the second equation, we get:

$$\mu \cdot p_2 = \mu \cdot p_1 - \lambda_1 \cdot p_0$$
$$p_2 = p_1 - \frac{\lambda_1}{\mu} \cdot p_0 = \left[\frac{1}{\mu} \cdot \left(\sum_{n=1}^{+\infty} \lambda_n\right) - \frac{\lambda_1}{\mu}\right] \cdot p_0 = \frac{1}{\mu} \cdot \left(\sum_{n=2}^{+\infty} \lambda_n\right) \cdot p_0$$

And again:

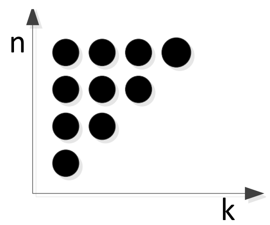
$$\mu \cdot p_3 = \mu \cdot p_2 - \lambda_2 \cdot p_0$$
$$p_3 = p_2 - \frac{\lambda_2}{\mu} \cdot p_0 = \left[\frac{1}{\mu} \cdot \left(\sum_{n=2}^{+\infty} \lambda_n\right) - \frac{\lambda_2}{\mu}\right] \cdot p_0 = \frac{1}{\mu} \cdot \left(\sum_{n=3}^{+\infty} \lambda_n\right) \cdot p_0$$

Thus it is easy to see that $p_k = \frac{1}{\mu} \cdot \left(\sum_{n=k}^{+\infty} \lambda_n\right) \cdot p_0$.

In order to compute the stability condition, we need to enforce that $\sum_{k=0}^{+\infty} p_k = 1$, which means that: $\left[1 + \frac{1}{\mu} \cdot \sum_{k=1}^{+\infty} \sum_{n=k}^{+\infty} \lambda_n\right] \cdot p_0 = 1$. This, in turn, holds if and only if $\sum_{k=1}^{+\infty} \sum_{n=k}^{+\infty} \lambda_n$ is limited. Let us see what the double summation yields.

You easily see that λ_1 appears once, λ_2 appears twice, … λ_j appears j times. Thus, the double summation can be rewritten as:

$$\sum_{k=1}^{+\infty} \sum_{n=k}^{+\infty} \lambda_n = \sum_{n=1}^{+\infty} n \cdot \lambda_n$$



Therefore, the stability condition is that the arrival rates are such that $\sum_{n=1}^{+\infty} n \cdot \lambda_n < +\infty$. The physical interpretation is that the *mean arrival rate of packets* should be finite.

The condition does not depend on the server speed, since the switch only accepts packets when it is empty.

c) In this case, we have $\lambda_n = \lambda/n^3$, which means that the stability condition is verified, since:

$$\sum_{n=1}^{+\infty} n \cdot \lambda_n = \sum_{n=1}^{+\infty} \frac{\lambda}{n^2} = \frac{\lambda \cdot \pi^2}{6}$$

Therefore, the steady-state probabilities are:

$$\left[1 + \frac{\lambda}{\mu} \cdot \frac{\pi^2}{6}\right] \cdot p_0 = 1$$

$$p_0 = \frac{1}{1 + u \cdot \pi^2/6}$$

Where $u = \lambda/\mu$. Furthermore, it is:

$$p_k = \frac{\lambda}{\mu} \cdot \left(\sum_{n=k}^{+\infty} \frac{1}{n^3}\right) \cdot p_0 = \frac{u}{1 + u \cdot \pi^2/6} \cdot \left(\sum_{n=k}^{+\infty} \frac{1}{n^3}\right)$$

d) The definition of throughput is the following: $\gamma = \sum_{n=1}^{+\infty} \mu_n \cdot p_n$, i.e. the mean value of the service rate. In our case, it is $\mu_n = \mu$, hence:

$$\gamma = \mu \cdot \sum_{n=1}^{+\infty} p_n = \mu \cdot (1 - p_0) = \mu \cdot \frac{u \cdot \pi^2/6}{1 + u \cdot \pi^2/6} = \frac{\lambda \cdot \pi^2/6}{1 + u \cdot \pi^2/6}$$

e) By definition, the utilization is the probability that the server is not idle. Therefore, we have

$$\rho = \sum_{n=1}^{+\infty} p_n = 1 - p_0 = \frac{u \cdot \pi^2/6}{1 + u \cdot \pi^2/6}$$

The mean service time could be obtained without any computations, since the service rate does not depend on the state. Therefore, it is surely $E[t_S] = 1/\mu$. In any case, the definition yields:

$$E[t_S] = \frac{\rho}{\gamma} = \frac{u \cdot \cancel{\pi^2/6}}{\cancel{1 + u \cdot \pi^2/6}} \cdot \frac{\cancel{1 + u \cdot \pi^2/6}}{\lambda \cdot \cancel{\pi^2/6}} = \frac{u}{\lambda} = \frac{1}{\mu}$$

6.1.3 Problem

Consider a switch of a packet network having N output links and M input links, with $M \geq N$. Assume that the output links can be modeled as exponential servers with identical rate μ, and the arrival processes on each input line are IID Poisson processes with a rate λ. When a packet arrives on the switch from an input line, the switch does the following:

- if there is at least one idle server, the packet goes immediately under service, blocking the input line it arrived on while it is being served;

- otherwise (there are no idle servers), the packet is discarded.

a) Model the switch as a birth-death process, compute the arrival rate and draw the CTMC;
b) compute the steady-state probabilities and the stability condition of the system;
c) compute the steady-state probability p_B that all the servers are busy (blocking probability);
d) compute the probability that an arriving packet finds all the servers busy (p_L, packet loss probability), and prove that $p_L(M) = p_B(M-1)$;
e) When $N = M$, prove that $p_n = \binom{N}{n} a^n \cdot (1-a)^{N-n}$, where $a = \frac{\lambda}{\lambda+\mu}$. Provide a physical interpretation for the result.

Note: it may be helpful to observe that $(M-k)\binom{M}{k} = M \cdot \binom{M-1}{k}$

Solution

The state of the switch can be characterized by the number of **busy servers n.**

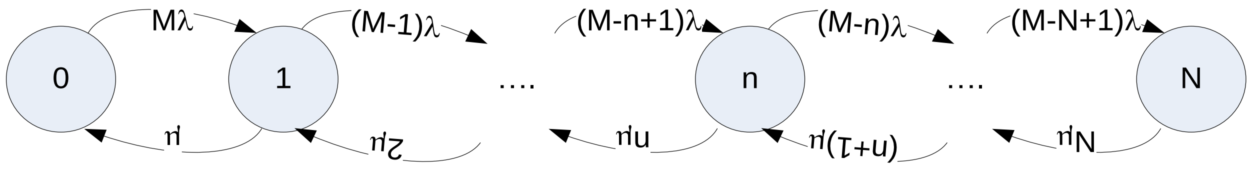
a) Let us start with the **arrival rates**.

- Consider an empty system (i.e., state 0), where all the M input links can send packets. The *overall* arrival rate can be computed by taking into account that the sum of M IID Poisson processes is itself a Poisson process with a mean value equal to $M \cdot \lambda$. Therefore, it is $\lambda_0 = M \cdot \lambda$.

- Let us now move to state 1. Now, **one of the processes is blocked**, since there is one packet in the system (hence one input link is blocked). Therefore, the aggregate arrival rate is $(M-1) \cdot \lambda$. Therefore, $\lambda_1 = (M-1) \cdot \lambda$.

- For a generic state $n < N$, it is $\lambda_n = (M-n) \cdot \lambda$.

- The system has $N+1$ states, since the switch drops packets when N servers are busy. This means that a non null aggregate arrival rate can still be derived, but no state transition is possible.



As far as **service rates** are concerned, they are proportional to the number of busy servers, thus it is $\mu_n = n \cdot \mu \quad 1 \le n \le N$.

b) Since the CTMC has only nearest-neighbor transitions, we can use the well-known formula to compute p_n as a function of p_0:

$$p_n = \prod_{i=0}^{n-1} \frac{\lambda_i}{\mu_{i+1}} p_0$$

By substituting $\lambda_n = (M-n)\cdot\lambda$ and $\mu_n = n\cdot\mu$ into the above we obtain:

$$\begin{aligned} p_n &= \frac{(M-n+1)\lambda\cdot\ldots(M-1)\lambda\cdot M\lambda}{n\cdot\mu\cdot(n-1)\cdot\mu\cdot\ldots\cdot 2\mu\cdot\mu}p_0 \\ &= \frac{M!/(M-n)!}{n!}\cdot\frac{\lambda^n}{\mu^n}p_0 \\ &= \binom{M}{n}\cdot\left(\frac{\lambda}{\mu}\right)^n\cdot p_0 \qquad 0\le n\le N \end{aligned}$$

Therefore, the normalization condition is the following: $p_0\cdot\sum_{n=0}^{N}\left[\binom{M}{n}\cdot\left(\frac{\lambda}{\mu}\right)^n\right]=1$.

Since the above is a finite sum, the system is positive recurrent. Therefore, we have:

$$\begin{cases} p_0 = \dfrac{1}{\sum_{n=0}^{N}\left[\binom{M}{n}\cdot\left(\frac{\lambda}{\mu}\right)^n\right]} \\ p_k = \binom{M}{k}\cdot\left(\dfrac{\lambda}{\mu}\right)^k\cdot p_0 \quad k\ge 1 \end{cases}$$

(note that the last expression also holds for $k=0$).

c) The probability that N servers are busy is:

$$p_B = p_N = \frac{\binom{M}{N}\cdot\left(\frac{\lambda}{\mu}\right)^N}{\sum_{n=0}^{N}\left[\binom{M}{n}\cdot\left(\frac{\lambda}{\mu}\right)^n\right]}$$

Note that this is the probability that **a random observer** finds the system in state N.

d) The solution to this point is **not** the same as the previous point's. The former requested the probability that N servers are busy **at any time**. This one is the probability that N servers are busy **when a packet arrives**. The two things are known to be different **when the arrival rates depend on the state of the system**, which is in fact the case in point.

The arrival-time steady-state probabilities (for a generic state n) can be computed as follows:

$$r_n = \frac{\lambda_n\cdot p_n}{\sum_{j=0}^{N}\lambda_j\cdot p_j} = \frac{(M-n)\cdot\lambda\cdot\binom{M}{n}\cdot\left(\frac{\lambda}{\mu}\right)^n\cdot p_0}{\sum_{j=0}^{N}\left\{(M-j)\lambda\cdot\binom{M}{j}\cdot\left(\frac{\lambda}{\mu}\right)^j\cdot p_0\right\}} = \frac{(M-n)\cdot\cancel{\lambda}\cdot\binom{M}{n}\cdot\left(\frac{\lambda}{\mu}\right)^n}{\sum_{j=0}^{N}\left[(M-j)\cancel{\lambda}\cdot\binom{M}{j}\cdot\left(\frac{\lambda}{\mu}\right)^j\right]}$$

Keeping into account that: $(M-n)\cdot\binom{M}{n} = M\cdot\binom{M-1}{n}$, we obtain:

$$r_n = \frac{\cancel{M}\cdot\binom{M-1}{n}\cdot\left(\frac{\lambda}{\mu}\right)^n}{\sum_{j=0}^{N}\left[\cancel{M}\cdot\binom{M-1}{j}\cdot\left(\frac{\lambda}{\mu}\right)^j\right]}$$

This means that the packet loss probability p_L is equal to r_N, i.e.:

$$p_L = \frac{\binom{M-1}{N} \cdot \left(\frac{\lambda}{\mu}\right)^N}{\sum_{j=0}^{N}\left[\binom{M-1}{j} \cdot \left(\frac{\lambda}{\mu}\right)^j\right]}$$

Note that it is $p_L(M) = p_B(M-1)$. The packet loss probability for a system with M input links is the blocking probability for a system with $M-1$ input links.

e) If $N = M$, we get the following:

$$p_k = \binom{N}{k} \cdot \left(\frac{\lambda}{\mu}\right)^k \cdot p_0 = \frac{\binom{N}{k} \cdot \left(\frac{\lambda}{\mu}\right)^k}{\sum_{n=0}^{N}\left[\binom{N}{n} \cdot \left(\frac{\lambda}{\mu}\right)^n\right]}$$

However, since $\sum_{n=0}^{N}\left[\binom{N}{n} \cdot \left(\frac{\lambda}{\mu}\right)^n\right] = \left(1+\frac{\lambda}{\mu}\right)^N$, we obtain $p_k = \frac{\binom{N}{k} \cdot \left(\frac{\lambda}{\mu}\right)^k}{\left(1+\frac{\lambda}{\mu}\right)^N}$.

If we define $a = \frac{\lambda}{\lambda+\mu} < 1$, we obtain:

$$a = \frac{\lambda/\mu}{1+\lambda/\mu}$$

$$\frac{\lambda}{\mu} = \frac{a}{1-a}$$

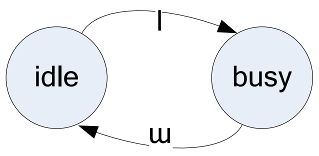
We substitute the latter in the former expression and obtain:

$$\left(\frac{\lambda}{\mu}\right)^k = \left(\frac{a}{1-a}\right)^k, \quad \left(1+\frac{\lambda}{\mu}\right)^N = \left(\frac{1}{1-a}\right)^N$$

$$p_k = \frac{\binom{N}{k} \cdot \left(\frac{\lambda}{\mu}\right)^k}{\left(1+\frac{\lambda}{\mu}\right)^N} = \frac{\binom{N}{k} \cdot \frac{a^k}{(1-a)^k}}{\frac{1}{(1-a)^N}} = \binom{N}{k} \cdot a^k \cdot (1-a)^{N-k}$$

Therefore, if the number of input links and servers is equal, the steady-state probabilities are those of a binomial RV.

This makes perfect sense. Consider a server in isolation. It oscillates between a *busy* and an *idle* state. The transition from idle to busy occurs at a rate λ, and those from busy to idle occur at a rate μ. This is a two-state birth-death process, whose steady-state probabilities are:



$$\begin{cases} p_{busy} = \frac{\lambda}{\lambda+\mu} = a \\ p_{idle} = \frac{\mu}{\lambda+\mu} = 1-a \end{cases}$$

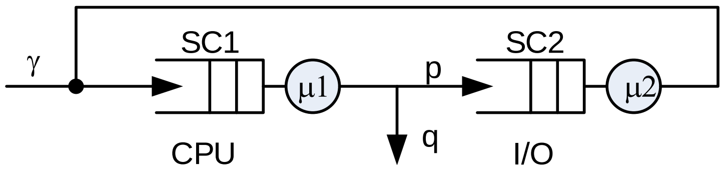
Since there are N such servers, and they are independent of each other, the number of busy servers is distributed as a *binomial RV* over N trials having success probability $p = a$.

6.2 Queueing networks

6.2.1 Problem (open queueing network)

Consider the computer system in the figure, and let $p + q = 1$. Compute:

1) the PMF of the number of jobs in the two service centers. Which of the two centers is the system bottleneck?
2) The mean number of jobs in the two service centers and in the system;
3) the mean response time of the system and the mean number of visits at the service centers.



Solution

1) This is an open Jackson's network, and its routing matrix is $\underline{\mathbf{\Pi}} = \begin{bmatrix} 0 & p \\ 1 & 0 \end{bmatrix}$. Arrivals are $\underline{\mathbf{\gamma}} = \begin{bmatrix} \gamma \\ 0 \end{bmatrix}$.

We could compute the arrival rate by solving equation $\underline{\boldsymbol{\lambda}} = \left(\underline{\mathbf{I}} - \underline{\mathbf{\Pi}}^T\right)^{-1} \cdot \boldsymbol{\gamma}$. It is easier if we observe that, since the input and output must balance, it must be $\gamma = \lambda_1 \cdot q$. Moreover, from the routing we get that $\lambda_2 = p \cdot \lambda_1$. Hence, we obtain $\lambda_1 = \gamma/q$, $\lambda_2 = p \cdot \gamma/q$.

Now that we have the arrival rates, we can compute the utilization of the SCs as $\rho_1 = \lambda_1/\mu_1 = \gamma/(\mu_1 \cdot q)$, and $\rho_2 = \lambda_2/\mu_2 = p \cdot \gamma/(\mu_2 \cdot q)$.

By Jackson's Theorem, this network admits a product form, which is:

$p(n_1, n_2) = [(1 - \rho_1) \cdot \rho_1{}^{n_1}] \cdot [(1 - \rho_2) \cdot \rho_2{}^{n_2}]$, as long as $\rho_i < 1, \quad 1 \le i \le 2$, i.e.:

$$p(n_1, n_2) = \left[\left(1 - \frac{\gamma}{\mu_1 \cdot q}\right) \cdot \left(\frac{\gamma}{\mu_1 \cdot q}\right)^{n_1}\right] \cdot \left[\left(1 - \frac{p \cdot \gamma}{\mu_2 \cdot q}\right) \cdot \left(\frac{p \cdot \gamma}{\mu_2 \cdot q}\right)^{n_2}\right]$$

The SC with the highest utilization is the system bottleneck. The condition by which $\rho_1 > \rho_2$ is $\gamma/(\mu_1 \cdot \cancel{q}) > p \cdot \gamma/(\mu_2 \cdot \cancel{q})$, i.e. $p \cdot \mu_1 < \mu_2$. Under the above inequality, the CPU is the bottleneck. Otherwise, the I/O subsystem is the bottleneck.

2) By Jackson's theorem we know that each SC behaves like an independent M/M/1 system, hence:

$$E[N_i] = \frac{\rho_i}{1 - \rho_i} = \begin{cases} \dfrac{\gamma}{q \cdot \mu_1 - \gamma} & i = 1 \\ \dfrac{p \cdot \gamma}{q \cdot \mu_2 - p \cdot \gamma} & i = 2 \end{cases}$$

Obviously, it is $E[N]=E[N_1]+E[N_2]$, since mean values are additive.

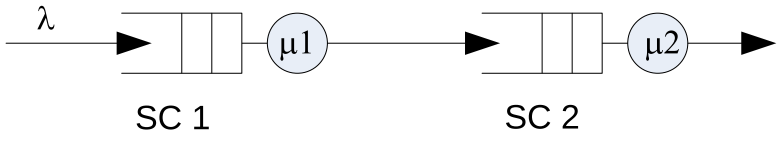
3) By Little's law, we get: $E[R]=E[N]/\gamma$, i.e.:

$$E[R]=\frac{1}{q\cdot\mu_1-\gamma}+\frac{p}{q\cdot\mu_2-p\cdot\gamma}$$

The average number of visits at each SC is $E[V_i]=\lambda_i/\gamma$, i.e. $E[V_1]=1/q$, $E[V_2]=p/q$. We can easily verify that $E[R]=E[T_1]+E[T_2]=E[V_1]\cdot E[N_1]/\lambda_1+E[V_2]\cdot E[N_2]/\lambda_2$. In fact, $E[V_i]/\lambda_i=1/\gamma$, and the computation is straightforward.

6.2.2 Problem (open queueing network)

Consider a tandem queueing network like the one in the figure. Assume that all SCs have infinite queues. Let $\mathcal{E}=\{(n_1,n_2)|n_i\geq 0\}$ be the state space for the system, where (n_1,n_2) denotes a state in which n_1 jobs are at SC 1 and n_2 are at SC 2.



1) Draw the CTMC for the above system, including at least all the states such that $n_1+n_2\leq 3$.
2) Write down the global equilibrium equations.
3) Verify Jackson's theorem in this case study, i.e. prove that the equilibrium equations have the following solution: $p_{\underline{n}}=p(n_1,n_2)=\left[(1-\rho_1)\cdot{\rho_1}^{n_1}\right]\cdot\left[(1-\rho_2)\cdot{\rho_2}^{n_2}\right]$, with $\rho_i=\lambda/\mu_i$.

Solution

The CTMC can be computed as follows:

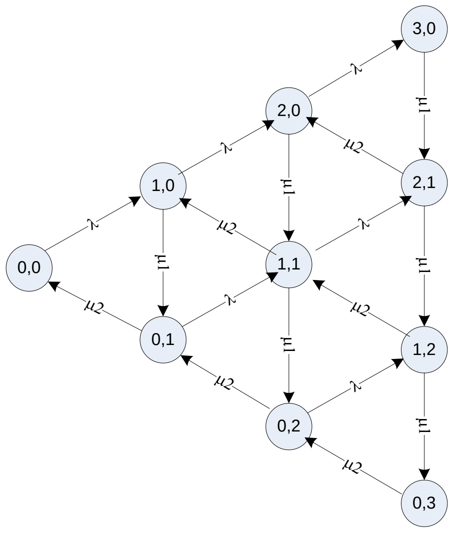
- starting from state $(0{,}0)$ (the system is empty), one arrival – which occurs at a rate λ – brings the system to state $(1{,}0)$.
- In state $(1{,}0)$, two things may happen:
  - one arrival at SC 1 (still at a rate λ), which brings the systems to state $(2{,}0)$;
  - one departure from SC 1, which occurs at a rate μ_1 and brings the system to state $(0{,}1)$.

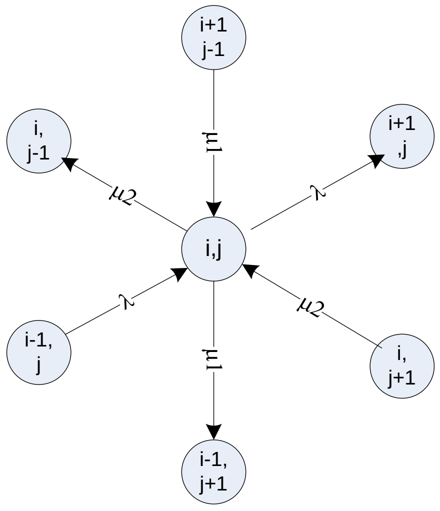
  The above behavior can be easily generalized to all the states $(j,0)$:
  - one arrival at SC 1 (still at a rate λ), which brings the systems in state $(j+1{,}0)$;
  - one departure from SC 1, which occurs at a rate μ_1 and brings the system to state $(j-1{,}1)$.
- In state $(0{,}1)$, again, two things may happen:
  - one arrival at SC 1 (at a rate λ), which brings the systems to state $(1{,}1)$;
  - one departure from SC 2, which occurs at a rate μ_2 and brings the system to state $(0{,}0)$.

The above behavior can be easily generalized to all the states $(0, j)$.

- In state (1,1), the following may happen:
  - one arrival at SC 1 (still at a rate λ), which brings the systems in state $(2,1)$;
  - one departure from SC 1, which occurs at a rate μ_1 and brings the system to state $(1,2)$;
  - one departure from SC 2, which occurs at a rate μ_2 and brings the system to state $(1,0)$.

This can be easily generalized to all states (i, j), where both i and j are non null. Therefore, a portion of the diagram is the one in the following figure. On the right, the figure displays the portion of the state diagram related to generic state (i, j), where both i and j are non null.





2) The global equilibrium equations can be easily inferred from the diagram on the right, with reference to generic state (i, j), assuming both i and j are non null.

$$p_{i,j} \cdot [\lambda + \mu_1 + \mu_2] = p_{i-1,j} \cdot \lambda + p_{i+1,j-1} \cdot \mu_1 + p_{i,j+1} \cdot \mu_2$$

If either or both i and j are null, then the above equation still holds, provided that we remove the missing states. To that aim, we can use the step function $\delta(i)$, which is 1 if $i > 0$ and 0 if $i = 0$, and write down a general expression, which holds for any state (i, j):

$$p_{i,j} \cdot [\lambda + \mu_1 \cdot \delta(i) + \mu_2 \cdot \delta(j)] = p_{i-1,j} \cdot \delta(i) \cdot \lambda + p_{i+1,j-1} \cdot \delta(j) \cdot \mu_1 + p_{i,j+1} \cdot \mu_2.$$

3) We just need to substitute the given expression for $p_{i,j}$ in the above equation, and check that equality holds:

$$\begin{aligned}
&[(1-\rho_1) \cdot {\rho_1}^i] \cdot [(1-\rho_2) \cdot {\rho_2}^j] \cdot [\lambda + \mu_1 \cdot \delta(i) + \mu_2 \cdot \delta(j)] = \\
&[(1-\rho_1) \cdot {\rho_1}^{i-1}] \cdot [(1-\rho_2) \cdot {\rho_2}^j] \cdot \delta(i) \cdot \lambda + \\
&[(1-\rho_1) \cdot {\rho_1}^{i+1}] \cdot [(1-\rho_2) \cdot {\rho_2}^{j-1}] \cdot \delta(j) \cdot \mu_1 + \\
&[(1-\rho_1) \cdot {\rho_1}^i] \cdot [(1-\rho_2) \cdot {\rho_2}^{j+1}] \mu_2
\end{aligned}$$

By multiplying and/or dividing each addendum by ρ_i as required, the above expression can be quickly rewritten as:

$$\begin{aligned}&[(1-\rho_1)\cdot\rho_1{}^i]\cdot[(1-\rho_2)\cdot\rho_2{}^j]\cdot[\lambda+\mu_1\cdot\delta(i)+\mu_2\cdot\delta(j)] =\\ &[(1-\rho_1)\cdot\rho_1{}^i]\cdot[(1-\rho_2)\cdot\rho_2{}^j]\cdot\delta(i)\cdot\lambda/\rho_1 +\\ &[(1-\rho_1)\cdot\rho_1{}^i]\cdot[(1-\rho_2)\cdot\rho_2{}^j]\cdot\delta(j)\cdot\mu_1\cdot\rho_1/\rho_2 +\\ &[(1-\rho_1)\cdot\rho_1{}^i]\cdot[(1-\rho_2)\cdot\rho_2{}^j]\mu_2\cdot\rho_2\end{aligned}$$

From the above, we obtain:

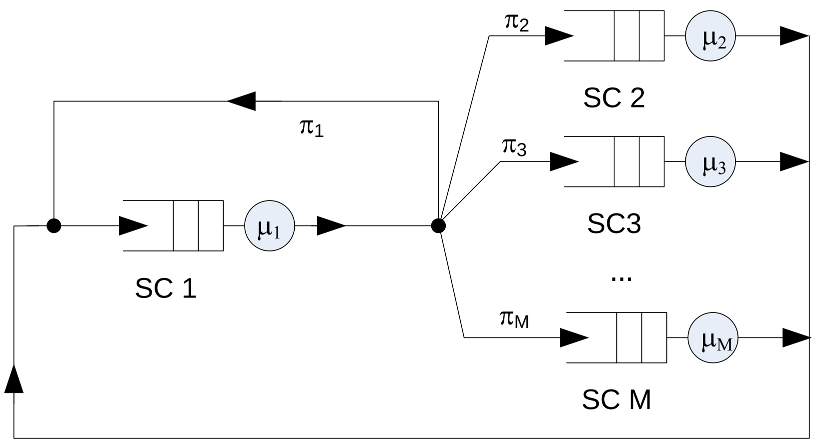
$$\lambda+\mu_1\cdot\delta(i)+\mu_2\cdot\delta(j)=\delta(i)\cdot\lambda/\rho_1+\delta(j)\cdot\mu_1\cdot\rho_1/\rho_2+\mu_2\cdot\rho_2$$

By substituting $\rho_i=\lambda/\mu_i$ into the above, we get:

$$\begin{aligned}&\lambda+\mu_1\cdot\delta(i)+\mu_2\cdot\delta(j)=\delta(i)\cdot\cancel{\lambda}\cdot\frac{\mu_1}{\cancel{\lambda}}+\delta(j)\cdot\cancel{\mu_1}\cdot\frac{\cancel{\lambda}}{\cancel{\mu_1}}\cdot\frac{\mu_2}{\cancel{\lambda}}+\cancel{\mu_2}\cdot\frac{\lambda}{\cancel{\mu_2}}\\ &\lambda+\mu_1\cdot\delta(i)+\mu_2\cdot\delta(j)=\mu_1\cdot\delta(i)+\mu_2\cdot\delta(j)+\lambda\end{aligned}$$

Hence equality holds.

6.2.3 Problem (closed queueing network)



Consider the *central-server* closed queueing network in the figure. Let K be the number of jobs in the system, and let μ_i be the service rate at each SC. Assume that all SCs are M/M/1.

1) Compute the steady-state probabilities in their general form.
2) Specialize the computation for $\pi_j=1/M$, $\mu_j=\mu$. Explain the result.
3) Run Buzen's algorithm with $M=5, K=10$ and compute the normalizing constant and the steady-state probabilities.
4) Compute the utilization and the throughput of each server in the above case.

Solution

The routing matrix is the following:

$$\underline{\mathbf{\Pi}} = \begin{bmatrix} \pi_1 & \pi_2 & \dots & \pi_{M-1} & \pi_M \\ 1 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & \dots & 0 & 0 \end{bmatrix}$$

Hence the linear system to be solved is the following:

$$\begin{cases} \lambda_1 = \pi_1 \cdot \lambda_1 + \lambda_2 + \ldots + \lambda_M \\ \lambda_2 = \pi_2 \cdot \lambda_1 \\ \lambda_3 = \pi_3 \cdot \lambda_1 \\ \dots \\ \lambda_M = \pi_M \cdot \lambda_1 \end{cases}$$

We can easily observe that the equations of this linear system are not independent, since the first one boils down to $\lambda_1 = \pi_1 \cdot \lambda_1 + \pi_2 \cdot \lambda_1 + \ldots + \pi_M \cdot \lambda_1 = \lambda_1 \cdot \sum_{i=1}^{M} \pi_i = \lambda_1$.

One solution is thus $\lambda_1 = \mu_1$, $\lambda_i = \pi_i \cdot \mu_1$ for $2 \le i \le M$. Thus, $\mathbf{e} = [\mu_1, \pi_2 \cdot \mu_1, \ldots, \pi_M \cdot \mu_1]^T$. From this we can compute the values for ρ_i: $\underline{\boldsymbol{\rho}} = [1, \pi_2 \cdot \mu_1/\mu_2, \ldots, \pi_M \cdot \mu_1/\mu_M]^T$.

Therefore, by Gordon and Newell's Theorem, we have:

$$p_{\underline{n}} = p(n_1, n_2, \ldots, n_M) = \frac{1}{G(M,K)} \cdot \prod_{i=2}^{M} \left(\frac{\pi_i \cdot \mu_1}{\mu_i} \right)^{n_i}$$

2) With $\pi_j = 1/M$, $\mu_j = \mu$, the above computation becomes:

$p_{\underline{n}} = p(n_1, n_2, \ldots, n_M) = \frac{1}{G(M,K)} \cdot \prod_{i=2}^{M} \left(\frac{1}{M^{n_i}} \right) = \frac{1}{G(M,K)} \cdot \frac{1}{M^{K-n_1}}$.

The result should not surprise us. In fact, it tells us that *every state* having n_1 jobs at SC 1 has the same probability. In fact, from the above data we infer that all the other SCs are *equivalent*, since they have the same service rate and the same routing probability. Therefore, $p(n_1, n_2, \ldots, n_M)|_{n_1}$ should be uniform, which it is.

3) We run Buzen's algorithm with the help of a spreadsheet, with the following initialization:

- row 0: $G(j, 0) = 1, \ 1 \le j \le M$
- column 1: $G(1, j) = \rho_1{}^j = 1, \ 1 \le j \le K$

The final result is in the table, and it is equal to 2.44. Note that, had we chosen $\lambda_1 = 10\mu_1$, the required constant would have been around $G(M, K) \approx 2.44 \cdot 10^{10}$. This is to remind us that, depending on the initial choice, numerical instability may arise. Since powers are involved, it is preferable that the value of the ρs be kept as close as possible to 1.

Thus, the steady-state probabilities are: $p_{\underline{n}} = p(n_1, n_2, \ldots, n_5) = \frac{1}{2.44} \cdot \frac{1}{5^{10-n_1}}$.

One may wonder why the constant $G(M, K)$ exhibits a negligible dependence on the number of jobs K. This is again a consequence of our initial choice: since $\rho_j = 1/M < 1$, then it makes sense that numbers do not grow that much while moving down in a column.

SC	1	2	3	4	5
ρ / jobs	1	1/5	1/5	1/5	1/5
0	1	1	1	1	1
1	1	1.20	1.40	1.60	1.80
2	1	1.24	1.52	1.84	2.20
3	1	1.25	1.55	1.92	2.36
4	1	1.25	1.56	1.94	2.42
5	1	1.25	1.56	1.95	2.43
6	1	1.25	1.56	1.95	2.44
7	1	1.25	1.56	1.95	2.44
8	1	1.25	1.56	1.95	2.44
9	1	1.25	1.56	1.95	2.44
10	1	1.25	1.56	1.95	**2.44**

The mean number of jobs at each SC can be computed through a spreadsheet software again:

$$E[N_i] = \frac{1}{G(M,K)} \cdot \sum_{j=1}^{K} \rho_i{}^j \cdot G(M,K-j)$$

For $i = 1$ it is $E[N_1] = \frac{\sum_{j=1}^{K} G(M,K-j)}{G(M,K)} \approx 9 = K - 1$, whereas for $i > 1$ we have $E[N_i] = \frac{\sum_{j=1}^{K} \frac{G(M,K-j)}{5^j}}{G(M,K)} \approx 0.25 = \frac{1}{M-1}$. Of course, $\sum_{i=1}^{M} E[N_i] = K$.

4) The utilization is $\rho_i \cdot \frac{G(M,K-1)}{G(M,K)} \approx \rho_i$. This makes sense: as the number of jobs increases, the probability that SC 1 is idle should go to zero (every job passes through SC 1 in a round). Moreover, the utilization of SCs 2 to M tends to $\pi_i \cdot \mu_1/\mu_i = 1/M$, and this makes sense as well. The throughput is:

$$X_i \approx \mu_i \cdot \rho_i = \begin{cases} \mu & i = 1 \\ \pi_i \cdot \mu = \dfrac{\mu}{M} & 2 \le i \le M' \end{cases}$$

which again makes sense, given that the routing probabilities are uniform. This said, we can compute $E[R_i]$ as:

$$E[R_1] = \frac{E[N_1]}{\mu} = 9\frac{1}{\mu}$$

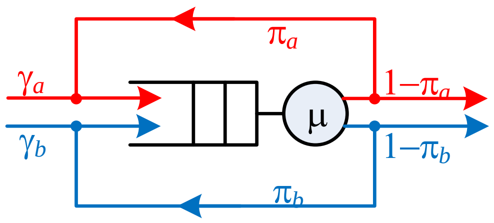
$$E[R_i] = \frac{E[N_i]}{\mu} = \frac{1}{M-1} \cdot \frac{M}{\mu} = \frac{5}{4\mu}, \quad 2 \le i \le M$$

6.2.4 Problem (classed open queueing network)

Consider a system serving two clients, a and b. Client a (b) issues service requests independently, with exponential interarrival times, at a rate γ_a (γ_b). Service requests are queued FCFS, and the queue is infinite. Each service request may require more than one passage through the system: on average, requests by client a (b) require m_a (m_b) such passages. This is achieved by sending requests that leave the server back to the queue with a certain probability (to be computed). The server is exponential with a rate μ.

1) Model this system as a queueing network and compute the routing probabilities and the arrival rates;
2) Compute the stability condition as a function of the system parameters;
3) Compute the mean number of service requests in the system;
4) Compute the joint probability to have n_a requests of client a *and* n_b requests of client b in the system. Explain your findings;

Solution



1) The system can be modeled as a classed OJN, where the two clients are two classes. There is only one SC, that has a rate μ, and the routing matrixes are $\mathbf{\Pi_a} = [\pi_a]$, $\mathbf{\Pi_b} = [\pi_b]$. The number of visits to the SC is a geometric RV, where $P\{V_i = k\} = \pi_i^{k-1} \cdot (1-\pi_i)$, hence $m_i = E[V_i] = \frac{1}{1-\pi_i}$ and $\pi_i = 1 - \frac{1}{m_i}$.

   Using input-output balance, one finds that $\gamma_i = (1-\pi_i) \cdot \lambda_i$, hence $\lambda_i = \gamma_i \cdot m_i$.
2) The total ingress rate to the system is therefore $\lambda_a + \lambda_b = \gamma_a \cdot m_a + \gamma_b \cdot m_b$, hence stability reads $\gamma_a \cdot m_a + \gamma_b \cdot m_b < \mu$.
3) Under the above conditions, define $\rho = \frac{\gamma_a \cdot m_a + \gamma_b \cdot m_b}{\mu}$, and it is

$$P\{n_a + n_b = n\} = \rho^n \cdot (1-\rho)$$

4) In a classed system, the probability to be in *any* state S having $n_a + n_b$ service requests is equal to:

$$P_S = \frac{\lambda_a^{n_a} \cdot \lambda_b^{n_b}}{\mu^{n_a+n_b}} \cdot (1-\rho)$$

   There are $\binom{n_a+n_b}{n_a}$ such states S, their number being the number of possible subsets of n_a items in a set of $n_a + n_b$.

   Call $\boldsymbol{S} = \{S | n_a\ jobs\ of\ class\ a\ and\ n_b\ jobs\ of\ class\ b\}$. We obtain:

$$P_{\boldsymbol{S}} = \binom{n_a + n_b}{n_a} \frac{\lambda_a^{n_a} \cdot \lambda_b^{n_b}}{\mu^{n_a+n_b}} \cdot (1-\rho)$$

$$= \binom{n_a + n_b}{n_a} \frac{\lambda_a^{n_a} \cdot \lambda_b^{n_b}}{(\lambda_a + \lambda_b)^{n_a+n_b}} \cdot \rho^{n_a+n_b}(1-\rho)$$

$$= \left[\binom{n_a + n_b}{n_a} \left(\frac{\lambda_a}{\lambda_a + \lambda_b}\right)^{n_a} \cdot \left(\frac{\lambda_b}{\lambda_a + \lambda_b}\right)^{n_b}\right] \cdot \rho^{n_a+n_b}(1-\rho)$$

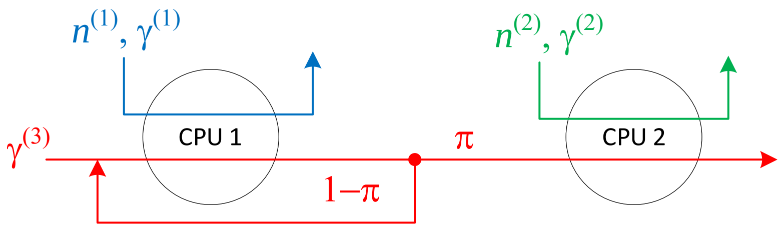
The first term in the above expression is the probability of having n_a successes in a repeated trial of $n_a + n_b$ trials, whereas the second term is the SS probability to have $n_a + n_b$ jobs in the system at all.

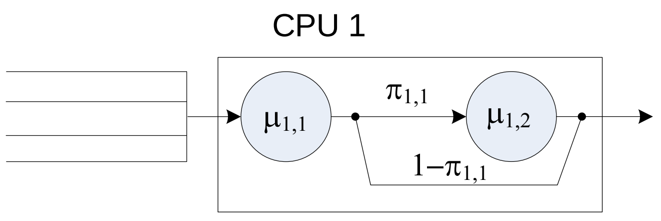
The alert reader can check that $\sum_{n_a=0}^{+\infty} \sum_{n_b=0}^{+\infty} P_S = 1$. This is obtained by rewriting the above double sum as $\sum_{n=0}^{+\infty} \sum_{n_a=0}^{n} P_S$, where $n = n_a + n_b$.

6.2.5 Problem (classed open QN of PS systems)

A processing system has two CPUs (CPU1 and CPU2), which are shared by a number of flows. Tasks are scheduled round-robin on both CPUs. CPU 1 serves $n^{(1)} + 1$ flows. One flow (pictured in red in the figure below) has an external arrival rate $\gamma^{(3)}$, whereas the other $n^{(1)}$ flows have an arrival rate $\gamma^{(1)}$. Similarly, CPU 2 serves $n^{(2)} + 1$ flows, $n^{(2)}$ of which have an external arrival rate $\gamma^{(2)}$. We are interested in the performance of tasks belonging to the *red flow*. The latter is served at least once at CPU 1, and then is routed to CPU 2 (or back to CPU 1) with a probability π $(1-\pi)$.

The distribution of service times at CPU 2 is an Erlang with 5 equal stages, each one with a rate μ_2. The distribution of service times at CPU 1 is a 2-stage Coxian as in the figure below.





1) Compute the mean service time at CPU 2, and the parameters of CPU 1's server that allow one to experience a service time with a mean m and a CoV χ (≥ 0.25).
2) Compute the routing matrix for the red flow and the overall arrival rates at each CPU.
3) Find the stability condition for the above system
4) Compute the mean number of jobs at CPU 1 and CPU 2

Solution

1) The service time at CPU 2 is the sum of 5 IID exponentials with a rate μ_2. Therefore, it is $E[t_{S2}] = 5/\mu_2$. As far as CPU 1 is concerned, it is $E[t_{S1}] = m$ (obviously), and:
$\mu_{1,1} = 2m$, $\pi_{1,1} = 1/2\chi^2$, $\mu_{1,2} = m/\chi^2$.

2) The routing matrix for the red flow is

$$\underline{\mathbf{\Pi}}^{(\mathbf{3})} = \begin{bmatrix} 1-\pi & \pi \\ 0 & 0 \end{bmatrix}$$

As far as I/O balance is concerned, we observe that:

- $\gamma^{(3)} = \lambda_2^{(3)}$ and $\gamma^{(3)} = \lambda_1^{(3)} \cdot \pi$, hence $\lambda_1^{(3)} = \frac{\gamma^{(3)}}{\pi}$.
- At CPU 1, we have a further $\lambda_1^{(1)} = n^{(1)} \cdot \gamma^{(1)}$ of input
- At CPU 2, we have a further $\lambda_2^{(2)} = n^{(2)} \cdot \gamma^{(2)}$ of input

Thus, we have:

- $\lambda_1 = \lambda_1^{(1)} + \lambda_1^{(3)} = n^{(1)} \cdot \gamma^{(1)} + \frac{\gamma^{(3)}}{\pi}$
- $\lambda_2 = \lambda_2^{(2)} + \lambda_2^{(3)} = n^{(2)} \cdot \gamma^{(2)} + \gamma^{(3)}$

3) The stability conditions are the following:

- At CPU 1, we have $\lambda_1 \cdot E[t_{S1}] < 1$, i.e. $\left(n^{(1)} \cdot \gamma^{(1)} + \frac{\gamma^{(3)}}{\pi}\right) \cdot m < 1$
- At CPU 2, we have $\lambda_2 \cdot E[t_{S2}] < 1$, i.e. $5 \cdot \frac{n^{(2)} \cdot \gamma^{(2)} + \gamma^{(3)}}{\mu_2} < 1$

4) Under the above conditions, the mean number of jobs at each CPU is the following:

$$E[N_1] = \frac{\left(n^{(1)} \cdot \gamma^{(1)} + \frac{\gamma^{(3)}}{\pi}\right) \cdot m}{1 - \left(n^{(1)} \cdot \gamma^{(1)} + \frac{\gamma^{(3)}}{\pi}\right) \cdot m}$$

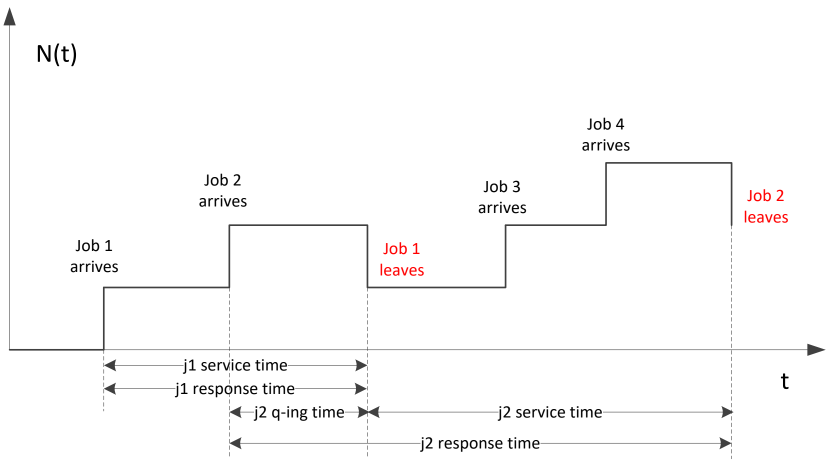
$$E[N_2] = \frac{5 \cdot \frac{n^{(2)} \cdot \gamma^{(2)} + \gamma^{(3)}}{\mu_2}}{1 - 5 \cdot \frac{n^{(2)} \cdot \gamma^{(2)} + \gamma^{(3)}}{\mu_2}} = \frac{5 \cdot \left(n^{(2)} \cdot \gamma^{(2)} + \gamma^{(3)}\right)}{\mu_2 - 5 \cdot (n^{(2)} \cdot \gamma^{(2)} + \gamma^{(3)})}$$

7 Appendix

7.1 Stochastic processes

Take a system consisting of a FCFS queue and a server. We need to characterize the **state** of this system at a given time t. The way we characterize its state depends **on what we want to observe**. For instance, we may be interested in:

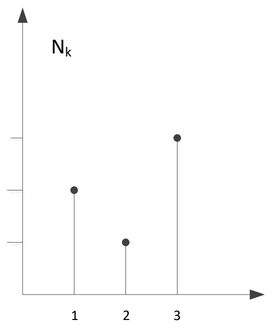
1) The **number of jobs in the system** at time t (also called the **backlog** at that time)



$N(t)$ is a **discrete** quantity (it is an integer), which is a function of a **continuous parameter (time)**. The above is a **trajectory** (or realization, or sample path), which depends on the **interarrival times** of the customers at the queue, as well as on the **service times** (or service demands). Given different interarrival and service times, the trajectory is going to be different.

2) The **number of jobs that a departing job leaves behind.**

This is a **discrete variable N_k**, which is a function of a **discrete parameter** (the job number). Again, the one in the figure is **a trajectory**, and different trajectories are possible. For instance, if **interarrival times** were generally shorter, then the trajectory would be **higher**. The same would occur if the service times were **longer**, all else being equal.

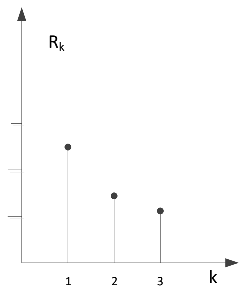


Since interarrival and service times are normally **random variables**, we should be a little more precise when we use terms such as "higher". We mean "higher" **in a stochastic sense**: if the interarrival times are **shorter** (in a stochastic sense: shorter interarrivals times have a higher probability), then

the trajectory will be higher (in a stochastic sense: higher trajectories will have a higher probability). Of course, lower, flatter trajectories will still be possible, but they will occur less frequently.
Note that you can always **compute this trajectory $\boldsymbol{N_k}$ given the first state characterization** $N(t)$. In fact, it is $\boldsymbol{N_k = N\left(t_k^{D^+}\right)}$. The **vice versa**, instead, does not hold - since you cannot create information that you do not possess. We say that the process $\boldsymbol{N_k}$ **is embedded into** $N(t)$.

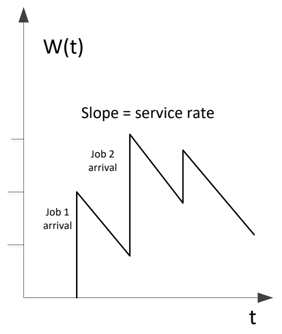
3) The **time it takes for job $\boldsymbol{k}$ to leave the system**

The above time is called *response time* of job *k*. This is a continuous-space variable $\boldsymbol{R_k}$, whose parameter is discrete.



4) The **time it takes to clear the backlog at time $\boldsymbol{t}$**

The backlog at time *t* is the amount of work it takes to serve all the jobs which are in the system at time t. This is a continuous-space variable $\boldsymbol{W(t)}$, whose parameter is a continuous time. A trajectory is going to look like this:

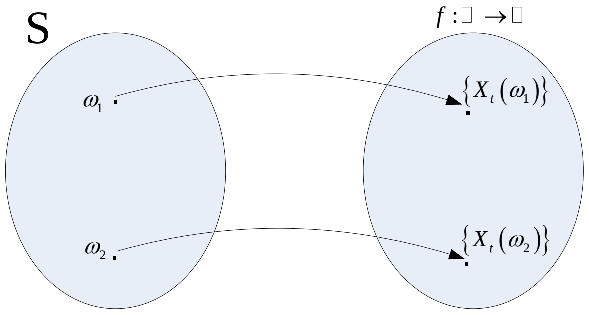


To summarize, we are talking about **random trajectories**, which can be any combination of

- Discrete/continuous in the **state space** (i.e., in the ordinates)
- Discrete/continuous in the **parameter** (i.e., in the abscissas)

The parameter is often (albeit confunsingly) called **time**, even when it is not (e.g., case 2 and 3). Thus, we talk about discrete-space/continuous-time trajectories, etc.
We need a **mathematical framework** for these entities. We extend the definition of **random variable,** which we provided time ago, to functions of time.

A **random variable** is a function from a sample space to **real numbers.** We extend this definition, and define a **random process** (or *stochastic process*), as a function from a sample space to a **space of real-valued (discrete or continuous) functions of (discrete or continuous) time**.

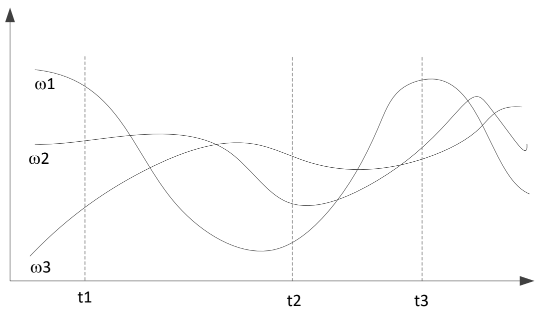


For instance, given an **outcome of a random experiment** ω_1, a process is the mapping from that outcome to a **function of time** (e.g., a real-valued function of continuous time). The outcome of the experiment can be, in this case, a suitably large number of interarrival times and service times for the jobs in the system.

Given that outcome, the trajectory X_t is perfectly deterministic. The fact that makes these trajectories aleatory is the fact that **the outcomes are aleatory**. Other common names for trajectories are **sample paths**, or **realizations** of the process. Notation X_t is used with continuous times as well.

Consider now several trajectories of the same process (cont. space/cont. time as an example):

Fix a value for the parameter, e.g. t_1. Now X_{t_1} is a **random variable**, with some distribution. So is X_t for any other value t of the parameter. The same obviously happens if the parameter is discrete, and if the state-space is discrete.



Suppose that you want to characterize the above process from a stochastic point of view. It stands to reason that you should need the **joint CDF** for all the values of the parameter. More specifically,

$$\forall n \in \mathbb{N}, \forall \{t_1, t_2, \dots, t_n\} \in \mathbb{R}^n | 0 < t_1 < t_2 < \dots t_n,$$
$$F_X(x_1, x_2, \dots x_n) = P\{X_{t_1} \le x_1, X_{t_2} \le x_2, \dots X_{t_n} \le x_n\}$$

For each n-tuple of parameters (and for each value of n), you would need the JCDF. This is, of course, unachievable in practice, **unless** some more hypotheses are brought on the scene.

One **gross simplification** would be to assume that the RVs X_t are **IID**. In this case, you would only need to know $F_X(x)$, and then any joint characterization could be obtained by leveraging independence. Processes like these are called **independent processes**, and unfortunately they are unsuitable for our purposes. We have already observed that, in several real systems, trajectories exhibit a **positive correlation**, so points in the trajectories are never independent. We need to strike a **better trade-off** between **analytical tractability and modeling effectiveness**.

A very good compromise can be found by using a modeling that the Russian mathematician Markov obtained at the beginning of the 20th century.

7.1.1 Markov processes

Consider, for instance, a discrete-state process. We can characterize it using its JPMF (which is equivalent to using the JCDF, since one can be obtained from the other).

$\{X_t, t \in T\}$ is known if $p(x_1, x_2, \dots x_n) = P\{X_{t_1} = x_1, X_{t_2} = x_2, \dots, X_{t_{n-1}} = x_{n-1}, X_{t_n} = x_n\}$ is known for every n and n-tuple of values $x_1, x_2, \dots x_n$.

Using a purely algebraic manipulation, we can write the above probability as follows:

$$P\{X_{t_1} = x_1, X_{t_2} = x_2, \dots, X_{t_{n-1}} = x_{n-1}, X_{t_n} = x_n\} =$$
$$P\{X_{t_n} = x_n | X_{t_1} = x_1, X_{t_2} = x_2, \dots, X_{t_{n-1}} = x_{n-1}\} \cdot P\{X_{t_1} = x_1, X_{t_2} = x_2, \dots, X_{t_{n-1}} = x_{n-1}\}$$

Of course, we can iterate the reasoning using the last available instant every time:

$$P\{X_{t_1} = x_1, \dots, X_{t_n} = x_n\} =$$
$$P\{X_{t_n} = x_n | X_{t_1} = x_1, \dots, X_{t_{n-1}} = x_{n-1}\} \cdot P\{X_{t_1} = x_1, \dots, X_{t_{n-1}} = x_{n-1}\} =$$
$$P\{X_{t_n} = x_n | X_{t_1} = x_1, \dots, X_{t_{n-1}} = x_{n-1}\} \cdot P\{X_{t_{n-1}} = x_{n-1} | X_{t_1} = x_1, \dots, X_{t_{n-2}} = x_{n-2}\} \cdot P\{X_{t_1} = x_1, \dots, X_{t_{n-2}} = x_{n-2}\}$$

Until we get to the final expression:

$$P\{X_{t_1} = x_1, \dots, X_{t_n} = x_n\} =$$
$$P\{X_{t_n} = x_n | X_{t_1} = x_1, \dots, X_{t_{n-1}} = x_{n-1}\} \cdot P\{X_{t_{n-1}} = x_{n-1} | X_{t_1} = x_1, \dots, X_{t_{n-2}} = x_{n-2}\} \cdot \dots$$
$$\dots \cdot P\{X_{t_2} = x_2 | X_{t_1} = x_1\} \cdot P\{X_{t_1} = x_1\}$$

This can always be written, whatever the process. The **Markov property** is the following:

$$P\{X_{t_n} = x_n | X_{t_1} = x_1, \dots, X_{t_{n-1}} = x_{n-1}\} = P\{X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}\}$$

In other words, the state of a process at time t_n is entirely determined by the state at time t_{n-1}, and it is conditionally independent of any state before t_{n-1}. Knowledge of what happens at times t_{n-j}, $j > 1$, does not yield any further information. Quoting a famous phrase, "the **future** of a process is conditionally **independent of the past**, once the **present is known**".

The above property implies that the JPMF can be written as follows:

$$P\{X_{t_1} = x_1, \dots, X_{t_n} = x_n\} = \prod_{j=2}^{n} P\left\{X_{t_j} = x_j | X_{t_{j-1}} = x_{j-1}\right\} \cdot P\{X_{t_1} = x_1\}$$

We call a **Markov process** one that possesses the Markov property. For a MP, the **current state** is enough to determine (in a stochastic sense) the **future state**, and knowledge of the past does not yield more information. Note – incidentally – that IID processes are a special case of Markov processes, since

$$P\{X_{t_n} = x_n | X_{t_1} = x_1, \dots, X_{t_{n-1}} = x_{n-1}\} = P\{X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}\} = P\{X_{t_n} = x_n\}$$

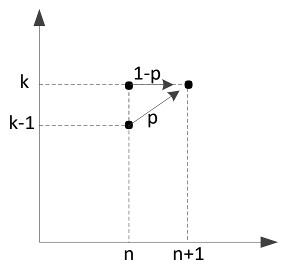
7.1.2 Example: Bernoulli process

This is a discrete-space/discrete-time process, which is obtained as follows: suppose you have a slotted-time system, where a customer **may or may not arrive** at any timeslot n, $n > 0$. At any timeslot, a new customer arrives with probability p. The sequence of IID Bernoullian RVs $\{X_n, n > 0\}$ is one way to describe the process.

A **second way** to characterize the above process is to consider the **number of arrived customers** by time n, call it N_n. If X_n are IID Bernoullian, then N_n is their sum, hence is a **binomial** with parameters n and p: $P\{N_n = k\} = p_n(k) = \binom{n}{k} p^k \cdot (1-p)^{n-k}$. It is $N_0 = 0$. We can see N_n as a **process** (discrete-space/discrete-time), whose trajectories can only be **increasing**. We call processes like these **counting processes, or arrival processes.**

Now, let us consider a given trajectory at time n, and let us predict its future state at time $n+1$. $P\{N_{n+1} = k\}$ can be obtained using Total Probability as follows:

$$P\{N_{n+1} = k\} = \sum_{j=0}^{+\infty} P\{N_{n+1} = k | N_n = j\} \cdot P\{N_n = j\}$$
$$= p \cdot P\{N_n = k-1\} + (1-p) \cdot P\{N_n = k\}$$



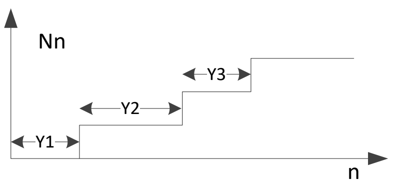
Since all the other conditional terms in the sum are null.

You will have noticed that there is **no further information** to be gained by knowing $P\{N_{n-1} = j\}$ (or of any less recent time instant), if you **already know $P\{N_n = j\}$**. Therefore, $\{N_n, n > 0\}$ is a **Markov process**. All the information you need in order to predict a future state is summarized in the **present state**, and knowing the past does not bring any further insight.

Note also that, for this process, the conditional probabilities are **independent** of the particular time n at which they are computed: $P\{N_{n+1} = k | N_n = j\} = P\{N_{m+1} = k | N_m = j\}$.

A **third way** of describing the same phenomenon is to **measure its interarrival times**. Call Y_1 the time at which the first customer arrives. Y_1 is a **geometric** RV: $P\{Y_1 = k\} = p \cdot (1-p)^{k-1}$.

Now, call Y_2 the **inter-arrival time** between customers 1 and 2 (we can see Y_1 as the interarrival time between the start of the system, at time $t = 0$, and the arrival time of the first customer). In this case as well it is $P\{Y_2 = k\} = p \cdot (1-p)^{k-1}$, and this holds for every interarrival time Y_j. Therefore, the sequence of interarrival time is a sequence of **IID Geometric RVs**.



We have discussed three equivalent ways to characterize the above process.

1) A **sequence of IID Bernoulli RVs** $\{X_n, n > 0\}$, denoting the probability that a new customer arrives at time n;
2) A **counting process** $\{N_n, n > 0\}$, denoting the number of customers arrived by time n. This is a Markov process.
3) A sequence of IID geometric RVs, modeling the **interarrival times**.

7.1.3 Example: Poisson process

Let us discuss how to model an arrival process in **continuous time.** This is made slightly trickier by the fact that, strictly speaking, the probability that something happens (e.g., a customer arrives) **at time $\boldsymbol{t}$** is zero. Therefore, we need to exert some care when we define the counting process and inter-arrival times. We define a **Poisson process** with a parameter λ as follows:

Consider the **counting process** $\{N(t), t \geq 0\}$ of customer arrivals by time t in a system. This is a Poisson process if, in a **small interval** $[t, t + \Delta t)$, the following happens:

1) The probability of **one arrival** in $[t, t + \Delta t)$, i.e., $P\{N(t + \Delta t) - N(t) = 1\}$, is equal to $\lambda \cdot \Delta t + o(\Delta t)$. The term $o(\Delta t)$ groups terms that go to zero faster than Δt, i.e., $\lim_{\Delta t \to 0} o(\Delta t)/\Delta t = 0$.
2) The probability of **zero arrivals** in $[t, t + \Delta t)$, i.e., $P\{N(t + \Delta t) - N(t) = 0\}$, is equal to $1 - \lambda \cdot \Delta t + o(\Delta t)$
3) The probability of **two or more arrivals** in $[t, t + \Delta t)$, i.e., $P\{N(t + \Delta t) - N(t) > 1\}$, is $o(\Delta t)$.
4) The process has **independent increments**: $\forall \{t_1, t_2, \dots, t_n\} \in \mathbb{R}^n | t_1 < t_2 < \dots t_n$ the **increments $\boldsymbol{N(t_{j+1}) - N(t_j)}$ are independent** quantities.

Let us see what follows from the above properties. Take an **arbitrarily large time interval** $[t, t + T)$, and let us compute $P\{N(t + T) - N(t) = k\}$. We can divide that interval into m **small intervals** whose width is $\Delta t = T/m$, where – by properties 1 and 2, you will have **either one or zero arrivals**, barring higher-order infinitesimals $o(\Delta t)$. Therefore, you can compute $P\{N(t + T) - N(t) = k\}$ using the insight of a **Bernoulli process** with $p = \lambda \cdot \Delta t = \lambda \cdot T/m$. You can do this because – by property 4 – what happens in $[t + jT/m, t + (j + 1)\,T/m)$ is independent of what happens outside that interval. This said:

$$P\{N(t + T) - N(t) = k\} = \binom{m}{k} \left(\frac{\lambda T}{m}\right)^k \left(1 - \frac{\lambda T}{m}\right)^{m-k}$$

However, we know from the theory that, if m is large, we can approximate this expression with:

$$\binom{m}{k}\left(\frac{\lambda T}{m}\right)^k\left(1-\frac{\lambda T}{m}\right)^{m-k} = \frac{m\cdot(m-1)\cdot\ldots\cdot(m-k+1)}{k!}\cdot\left(\frac{\lambda T}{m}\right)^k\cdot\frac{(1-\lambda T/m)^m}{(1-\lambda T/m)^k}$$
$$= \frac{m\cdot(m-1)\cdot\ldots\cdot(m-k+1)}{m^k}\cdot\frac{(\lambda T)^k}{k!}\cdot\frac{(1-\lambda T/m)^m}{(1-\lambda T/m)^k}$$
$$\simeq 1\cdot\frac{(\lambda T)^k}{k!}\cdot\frac{e^{-\lambda T}}{1}$$

Therefore, $P\{N(t+T)-N(t)=k\}=\frac{(\lambda T)^k}{k!}\cdot e^{-\lambda T}$. The number of arrivals (*increments*) in a time interval of a length T is a Poisson RV with a mean λT. This means that λ is the number of arrivals per unit of time, i.e. the **arrival rate** of the customers.

Still because of property 4, this result does **not depend on the instant *t*,** but only of the width of the time interval T. For this reason, we can (and will henceforth) write it down as $p_k(T)=\frac{(\lambda T)^k}{k!}\cdot e^{-\lambda T}$.

What about the **interarrival times** of a Poisson process? Assume, as usual, that $N(0)=0$, and call S_n the time at which the n-th customer arrives. It is easy to observe that these two events are in fact the same: $\{S_n\le t\}\equiv\{N(t)\ge n\}$. In fact, the n-th arrival occurs by t if and only if the counting process is at least equal to n at time t. We can use this to compute the CDF of the first arrival time.

$$F_1(t)=P\{S_1\le t\}=P\{N(t)\ge 1\}=1-p_0(t)=1-\frac{(\lambda t)^0}{0!}\cdot e^{-\lambda t}=1-e^{-\lambda t}$$

This means that the CDF of the first arrival time is **exponential**, with a mean $1/\lambda$. However, we soon realize that the above property holds for **all the <u>inter</u>-arrival times**. In fact:

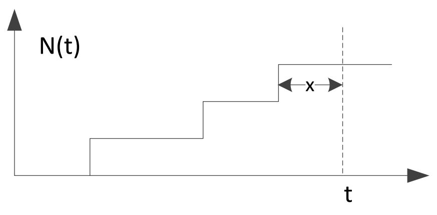
$$P\{S_{k+1}-S_k\le t\}=P\{N(S_k+t)-N(S_k)\ge 1\}=P\{N(t)\ge 1\}=1-e^{-\lambda t},$$

where the last passage is due to independent increments. Therefore:

In a Poisson process (and **only** in a Poisson process), interarrival times are exponentially distributed, hence **memoryless**. Saying "Poisson process" or "exponential interarrivals" is exactly the same.

This states that the **only thing** that we need to describe the state of the process at time t is $N(t)$. We do not need to know, for instance, how far in the past the last arrival has occurred, because the probability that the **next arrival** will be within a given time interval is not influenced by the past, due to the memoryless property of the exponential distribution. In other words, **a Poisson process is a Markov process**.

Compare this with a different arrival process, e.g., one where interarrivals are **constant**. We cannot describe the state of the process using $N(t)$ **only**: we also need to know x, the continuous RV describing how far in the past the last arrival occurred. With a Poisson process, we do not need that piece of information.



7.1.4 Properties of Poisson processes

Take an interval $[0, t]$, and assume that $N(t) = 1$, i.e. we know that one arrival has occurred by time t. We want to understand what the distribution is of the arrival time **given that one arrival occurred**. In other words, we want to compute $P\{S_1 \le s | N(t) = 1\}$, with $0 \le s \le t$. This is equal to:

$$P\{S_1 \le s | N(t) = 1\} = \frac{P\{S_1 \le s, N(t) = 1\}}{P\{N(t) = 1\}} = \frac{P\{N(s) = 1, N(t) - N(s) = 0\}}{P\{N(t) = 1\}}$$

Because of the independent increments property, we get:

$$\frac{P\{N(s) = 1, N(t) - N(s) = 0\}}{P\{N(t) = 1\}} = \frac{P\{N(s) = 1\} \cdot P\{N(t) - N(s) = 0\}}{P\{N(t) = 1\}}$$

Since arrivals are Poissonian, we easily obtain:

$$\frac{e^{-\lambda s} \cdot \frac{\lambda s}{1!} \cdot e^{-\lambda(t-s)} \cdot \frac{\lambda^0 \cdot (t-s)^0}{0!}}{e^{-\lambda t} \cdot \frac{\lambda t}{1!}} = \frac{s}{t}$$

In other words, the **conditional** distribution of arrival times is **uniform**.

Now we want to derive the distribution of the ***n*-th arrival time**, call it S_n. Recall that $\{S_n \le t\} \equiv \{N(t) \ge n\}$. Therefore,

$$P\{S_n \le t\} = P\{N(t) \ge n\} = 1 - P\{N(t) \le n - 1\} = 1 - \sum_{k=0}^{n-1} e^{-\lambda t} \frac{(\lambda t)^k}{k!}$$

This distribution is called **Erlang distribution** with ***n*** **stages**.

The Erlang distribution is quite common. Assume that a system has n jobs queued, and the service times are exponentially distributed with a rate λ. Assume that the arrivals are blocked, and compute the time it takes to empty the system. It is clear that this is an n-stage Erlang distribution as well.

The above expression is the CDF of the Erlang distribution, $F_n(t)$. The PDF $f_n(t)$ can be computed mechanically, by deriving the above expression. This is cumbersome, so we use a simple trick instead. It is $f_n(t) = \frac{d}{dt} F_n(t)$. However, $f_n(t)dt \simeq \int_t^{t+dt} f_n(y)dy$, and the latter is the probability that the n-th arrival occurs in $[t, t + dt)$. For this to happen, it must be that:

- $n - 1$ arrivals have occurred **before** time t;
- the last arrival occurs in the **small** interval $[t, t + dt)$.

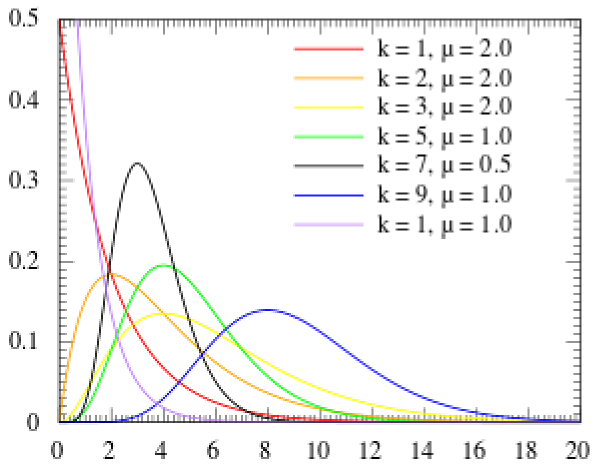
The two probabilities can be multiplied because of the independent increments, hence we get:

$$f_n(t)dt = e^{-\lambda t} \cdot \frac{(\lambda t)^{n-1}}{(n-1)!} \cdot \lambda dt$$

Therefore, it is: $f_n(t) = e^{-\lambda t} \cdot \lambda \cdot \frac{(\lambda t)^{n-1}}{(n-1)!}$.

Let us take a look at what an Erlang distribution looks like:

For $n = 1$ it is an exponential. This is clear both intuitively and from the formulas. When n>1 it starts **peaking** and then goes down. When n gets large, due to the CLT, it looks like a Normal.



We can compute $E[S_n]$ and $Var(S_n)$ leveraging **additivity** of mean values and **independence** (recall that the Erlang is the sum of n independent exponentials). Therefore, we get $E[S_n] = \frac{n}{\lambda}$, $Var(S_n) = \frac{n}{\lambda^2}$. From these we obtain $CoV(S_n) = \frac{\sqrt{Var(S_n)}}{E[S_n]} = \frac{1}{\sqrt{n}}$. In other words, the CoV of an n-stage Erlang is **smaller than one** (and, specifically, it is smaller than an exponential's), and it gets smaller with n (which is again a consequence of the CLT).

7.2 *Formal derivation of Chapman-Kolmogorov equations*

In this appendix we use the insight of Poisson processes to derive formally Chapman-Kolmogorov's equations.

7.2.1 M/M/1 system

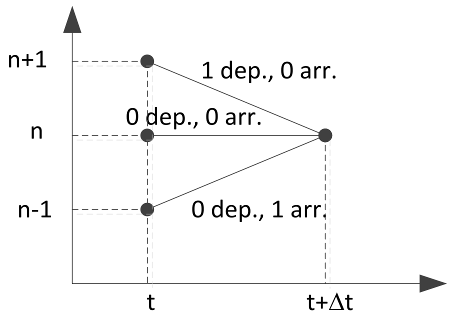
We report here the full derivation of CK equations for an M/M/1 system

We will now compute the **PMF of the number of jobs in the system** at time t. Call $p_n(t)$ the probability that there are n jobs at time t, i.e. $p_n(t) = P\{N(t) = n\}$.

The procedure is as follows: **we write down $p_n(t + \Delta t)$ using Total Probability**, and then we let $\Delta t \to 0$.

The expression is: $p_n(t + \Delta t) = \sum_{k=0}^{+\infty} P\{N(t + \Delta t) = n | N(t) = k\} \cdot p_k(t)$, which is an infinite sum. Luckily, we can **neglect all but three terms** in that sum.

Let us assume first that $n \geq 1$ (the case $n = 0$ needs special care and will be dealt with separately later on), and let us figure out what the possible trajectories are that lead to point $(t + \Delta t, n)$ in the Cartesian plane.



a) The most obvious trajectory is one where $N(t) = n$, and there are **zero arrivals/departures** in $[t, t + \Delta t]$. The probability that zero arrivals/departures occur in a **small** interval $[t, t + \Delta t]$ (recall that we let $\Delta t \to 0$) is:

$$[1 - \lambda \cdot \Delta t + o(\Delta t)] \cdot [1 - \mu \cdot \Delta t + o(\Delta t)] =$$
$$1 - (\lambda + \mu) \cdot \Delta t + o(\Delta t)$$

Of course, there are infinite possible trajectories such that $N(t) = n$ **and** $N(t + \Delta t) = n$: for instance, one where there is **one arrival and one departure** in $[t, t + \Delta t]$. However, the probability that there is one arrival and one departure is $[\lambda \cdot \Delta t + o(\Delta t)] \cdot [\mu \cdot \Delta t + o(\Delta t)] = o(\Delta t)$, hence it is negligible. We quickly see that the **only** possible trajectory such that $N(t) = n$ **and** $N(t + \Delta t) = n$ whose probability is non-negligible is the one where **zero arrivals and zero departures occur**. All trajectories where two or more events are required have a negligible $o(\Delta t)$ probability.

b) Another possibility is that $N(t) = n + 1$, and **one departure occurs** in $[t, t + \Delta t]$. The probability of zero arrivals/one departure in $[t, t + \Delta t]$ is:

$[1 - \lambda \cdot \Delta t + o(\Delta t)] \cdot [\mu \cdot \Delta t + o(\Delta t)] = \mu \cdot \Delta t + o(\Delta t)$.

Again, all **other** trajectories such that $N(t) = n + 1$ **and** $N(t + \Delta t) = n$ require at least two events in $[t, t + \Delta t]$, hence their probability is $o(\Delta t)$.

c) Another possibility is that $N(t) = n - 1$, and **one arrival occurs** in $[t, t + \Delta t]$. The probability of zero departures/one arrival in $[t, t + \Delta t]$ is:

$[\lambda \cdot \Delta t + o(\Delta t)] \cdot [1 - \mu \cdot \Delta t + o(\Delta t)] = \lambda \cdot \Delta t + o(\Delta t)$.

Again, all **other** trajectories such that $N(t) = n - 1$ **and** $N(t + \Delta t) = n$ require at least two events in $[t, t + \Delta t]$, hence their probability is $o(\Delta t)$.

Any other trajectory, such as those with $N(t) = n \pm j$, $j \geq 2$, requires at least two events to occur, hence has $o(\Delta t)$ probability. The only non-negligible terms in the infinite sum are those listed at points a-c above. Therefore, we can write:

$$\begin{aligned} p_n(t+\Delta t) &= \sum_{k=0}^{+\infty} P\{N(t+\Delta t)=n|N(t)=k\}\cdot p_k(t) \\ &= [1-(\lambda+\mu)\cdot\Delta t+o(\Delta t)]\cdot p_n(t)+ \\ &\quad [\mu\cdot\Delta t+o(\Delta t)]\cdot p_{n+1}(t)+ \\ &\quad [\lambda\cdot\Delta t+o(\Delta t)]\cdot p_{n-1}(t) \end{aligned}$$

The latter quickly becomes:

$$\frac{p_n(t+\Delta t)-p_n(t)}{\Delta t} = \left[-(\lambda+\mu)+\frac{o(\Delta t)}{\Delta t}\right]\cdot p_n(t)+\left[\mu+\frac{o(\Delta t)}{\Delta t}\right]\cdot p_{n+1}(t)+\left[\lambda+\frac{o(\Delta t)}{\Delta t}\right]\cdot p_{n-1}(t)$$

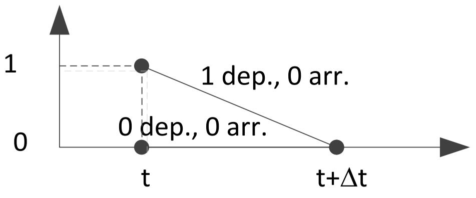
And, if we let $\Delta t \to 0$, we obtain (recall that $o(\Delta t)$ terms go to zero **faster than Δt**):

$$\frac{d}{dt}p_n(t) = -(\lambda+\mu)\cdot p_n(t)+\mu\cdot p_{n+1}(t)+\lambda\cdot p_{n-1}(t)$$

The above is called **Chapman-Kolmogorov equation**.

We still have to deal with the case $n=0$, which we had set apart. In this case, in fact, only **two trajectories are possible**:

a) The one where $N(t)=n=0$, and there are **zero arrivals/zero departures** in $[t,t+\Delta t]$. The probability that this occurs is $[1-\lambda\cdot\Delta t+o(\Delta t)]\cdot 1 = 1-\lambda\cdot\Delta t+o(\Delta t)$. This is because one cannot have departures when there are no jobs, so "zero departures" is the certain event in this case.
b) The one where $N(t)=n+1=1$, and there are **zero arrivals/one departure** in $[t,t+\Delta t]$, with a a probability: $\mu\cdot\Delta t+o(\Delta t)$.



The third trajectory, in fact (the one that involves one arrival and zero departures) cannot occur. Therefore, we can repeat the same computations and obtain:

$$\begin{aligned} p_0(t+\Delta t) &= \sum_{k=0}^{+\infty} P\{N(t+\Delta t)=0|N(t)=k\}\cdot p_k(t) \\ &= [1-\lambda\cdot\Delta t+o(\Delta t)]\cdot p_0(t)+\quad[\mu\cdot\Delta t+o(\Delta t)]\cdot p_1(t) \end{aligned}$$

Which leads to:

$$\frac{d}{dt}p_0(t) = -\lambda\cdot p_0(t)+\mu\cdot p_1(t)$$

This way, we obtain a **system of differential equations** that describes $p_n(t)$ for each time t.

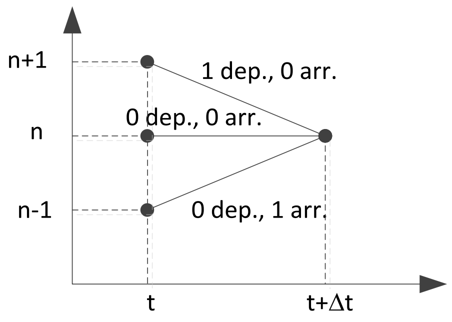
$$\begin{cases} \frac{d}{dt}p_n(t) = -(\lambda+\mu)\cdot p_n(t) + \mu\cdot p_{n+1}(t) + \lambda\cdot p_{n-1}(t) & n>0 \\ \frac{d}{dt}p_0(t) = -\lambda\cdot p_0(t) + \mu\cdot p_1(t) & n=0 \end{cases}$$

7.2.2 M/M/2 system

Like we did for the M/M/1 system, we write down Chapman-Kolmogorov's equations and compute $p_n(t)$. This time we need to distinguish **three cases**: $n \geq 2, n = 1, n = 0$.

Case $n \geq 2$:

Let us figure out what the possible trajectories are that lead to point $(t+\Delta t, n)$ in the Cartesian plane. As usual, there are three such trajectories:



a) one where $N(t) = n$, and there are **zero arrivals/departures** in $[t, t+\Delta t]$, whose probability is:

   $\left[(1-\lambda\cdot\Delta t + o(\Delta t))\cdot(1-\mu\cdot\Delta t + o(\Delta t))^2\right] =$
   $1-(\lambda+2\mu)\cdot\Delta t + o(\Delta t)$

   The square is due to the fact that there are **two independent busy servers**, hence the probability that neither of them outputs a job is the product of two identical terms.

b) one where $N(t) = n+1$, and **one departure occurs** in $[t, t+\Delta t]$, whose probability is:

   $[1-\lambda\cdot\Delta t + o(\Delta t)]\cdot\{[\mu\cdot\Delta t + o(\Delta t)]\cdot[1-\mu\cdot\Delta t + o(\Delta t)] + [1-\mu\cdot\Delta t + o(\Delta t)]\cdot[\mu\cdot\Delta t + o(\Delta t)]\}$
   $= 2\mu\cdot\Delta t + o(\Delta t)$

   In this case, in fact, **one** server has a departure, and the other hasn't. However, both events (departure, no departure) and (no departure, departure), which are disjoint and with the same probability, must be considered, hence the resulting term has a multiplying factor 2.

c) Another one $N(t) = n-1$, and **one arrival occurs** in $[t, t+\Delta t]$, whose probability is:

   $[\lambda\cdot\Delta t + o(\Delta t)]\cdot[1-\mu\cdot\Delta t + o(\Delta t)]^2 = \lambda\cdot\Delta t + o(\Delta t).$

   Again, mind the square, since there are two busy servers.

Any other trajectory, such as those with $N(t) = n \pm j, j \geq 2$, requires at least two events to occur, hence has $o(\Delta t)$ probability. The only non-negligible terms in the infinite sum are those listed at points a-c above. Therefore, we can write:

$$\begin{aligned} p_n(t + \Delta t) &= \sum_{k=0}^{+\infty} P\{N(t + \Delta t) = n | N(t) = k\} \cdot p_k(t) \\ &= [1 - (\lambda + 2\mu) \cdot \Delta t + o(\Delta t)] \cdot p_n(t) + \\ &\quad [2\mu \cdot \Delta t + o(\Delta t)] \cdot p_{n+1}(t) + \\ &\quad [\lambda \cdot \Delta t + o(\Delta t)] \cdot p_{n-1}(t) \end{aligned}$$

And, if we let $\Delta t \to 0$, we obtain:

$$\frac{d}{dt} p_n(t) = -(\lambda + 2\mu) \cdot p_n(t) + 2\mu \cdot p_{n+1}(t) + \lambda \cdot p_{n-1}(t)$$

Case $n = 1$

In this case, **all three trajectories are possible, but probabilities are different**. In fact, only **one server is busy**, whereas the other is idle.

a) one where $N(t) = n = 1$, and there are **zero arrivals/departures** in $[t, t + \Delta t]$, whose probability is:

$$\left[\left(1 - \lambda \cdot \Delta t + o(\Delta t)\right) \cdot \left(1 - \mu \cdot \Delta t + o(\Delta t)\right)\right] \cdot 1 = \\ 1 - (\lambda + \mu) \cdot \Delta t + o(\Delta t)$$

There is no square this time, since one of the servers is idle, so the fact that it will have no departures is the **certain** event.

b) one where $N(t) = n + 1 = 2$, and **one departure occurs** in $[t, t + \Delta t]$, whose probability is:

$$[1 - \lambda \cdot \Delta t + o(\Delta t)] \cdot \{2 \cdot [\mu \cdot \Delta t + o(\Delta t)] \cdot [1 - \mu \cdot \Delta t + o(\Delta t)]\} = \\ 2\mu \cdot \Delta t + o(\Delta t)$$

c) Another one $N(t) = n - 1 = 0$, and **one arrival occurs** in $[t, t + \Delta t]$, whose probability is $\lambda \cdot \Delta t + o(\Delta t)$. Note that there are no busy servers in this case.

From the above, we get

$$\frac{d}{dt} p_1(t) = -(\lambda + \mu) \cdot p_1(t) + 2\mu \cdot p_2(t) + \lambda \cdot p_0(t)$$

Case $n = 0$

In this case, only **two trajectories are possible**:

a) The one where $N(t) = n = 0$, and there are **zero arrivals/zero departures** in $[t, t + \Delta t]$. The probability that this occurs is $[1 - \lambda \cdot \Delta t + o(\Delta t)] \cdot 1 = 1 - \lambda \cdot \Delta t + o(\Delta t)$. This is because one cannot have departures when there are no jobs, so “zero departures” is the certain event in this case.

b) The one where $N(t) = n + 1 = 1$, and there are **zero arrivals/one departure** in $[t, t + \Delta t]$, with a a probability: $\mu \cdot \Delta t + o(\Delta t)$ (note that in this case there is one busy server and one idle server).

Thus, we have:

$$\frac{d}{dt} p_0(t) = -\lambda \cdot p_0(t) + \mu \cdot p_1(t)$$

We thus obtain the usual **system of differential equations** that describes $p_n(t)$ for each time t.

$$\begin{cases} \frac{d}{dt} p_n(t) = -(\lambda + 2\mu) \cdot p_n(t) + 2\mu \cdot p_{n+1}(t) + \lambda \cdot p_{n-1}(t) & n > 1 \\ \frac{d}{dt} p_1(t) = -(\lambda + \mu) \cdot p_1(t) + 2\mu \cdot p_2(t) + \lambda \cdot p_0(t) & n = 1 \\ \frac{d}{dt} p_0(t) = -\lambda \cdot p_0(t) + \mu \cdot p_1(t) & n = 0 \end{cases}$$

7.3 *Useful mathematical series*

Source: Wikipedia

7.3.1 Sums of powers

$$\sum_{k=1}^{m} k = \frac{m(m+1)}{2}$$

$$\sum_{k=1}^{m} k^2 = \frac{m(m+1)(2m+1)}{6} = \frac{m^3}{3} + \frac{m^2}{2} + \frac{m}{6}$$

$$\sum_{k=1}^{m} k^3 = \left[\frac{m(m+1)}{2}\right]^2 = \frac{m^4}{4} + \frac{m^3}{2} + \frac{m^2}{4}$$

$$\zeta(2) = \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

$$\zeta(4) = \sum_{k=1}^{\infty} \frac{1}{k^4} = \frac{\pi^4}{90}$$

$$\zeta(6) = \sum_{k=1}^{\infty} \frac{1}{k^6} = \frac{\pi^6}{945}$$

7.3.2 Power series

$$\sum_{k=0}^{n} z^k = \frac{1-z^{n+1}}{1-z}$$

$$\sum_{k=1}^{n} kz^k = z\frac{1-(n+1)z^n + nz^{n+1}}{(1-z)^2}$$

$$\sum_{k=1}^{n} k^2z^k = z\frac{1+z-(n+1)^2z^n + (2n^2+2n-1)z^{n+1} - n^2z^{n+2}}{(1-z)^3}$$

7.3.3 Exponential functions

$$\sum_{k=0}^{\infty} \frac{z^k}{k!} = e^z$$

$$\sum_{k=0}^{\infty} k\frac{z^k}{k!} = ze^z$$

$$\sum_{k=0}^{\infty} k^2\frac{z^k}{k!} = (z+z^2)e^z$$

$$\sum_{k=0}^{\infty} k^3\frac{z^k}{k!} = (z+3z^2+z^3)e^z$$

$$\sum_{k=0}^{\infty} k^4\frac{z^k}{k!} = (z+7z^2+6z^3+z^4)e^z$$

7.3.4 Binomial coefficients

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0, \text{ where } n > 0$$

$$\sum_{k=0}^{n} \binom{k}{m} = \binom{n+1}{m+1}$$

$$(1+z)^\alpha = \sum_{k=0}^{\infty} \binom{\alpha}{k} z^k, |z| < 1$$

$$\sum_{k=0}^{\infty} \binom{\alpha+k-1}{k} z^k = \frac{1}{(1-z)^\alpha}, |z| < 1$$

$$\sum_{k=0}^{\infty} \frac{1}{k+1}\binom{2k}{k} z^k = \frac{1-\sqrt{1-4z}}{2z}, |z| \le \frac{1}{4}$$

$$\sum_{k=0}^{\infty} \binom{2k}{k} z^k = \frac{1}{\sqrt{1-4z}}, |z| < \frac{1}{4}$$