"Sapienza" University of Rome
Faculty of Information Engineering, Informatics and Statistics
Department of Computer Science

Computer Network Performance

Lecture notes integrated with the book "Performance Modeling and Design of Computer Systems", M. Harchol-Balter

Author
Simone Bianco

December 3, 2025

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Computer Network Performance* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/Exyss/university-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **bianco.simone@outlook.it**
- LinkedIn: **Simone Bianco**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

Networks and Probability

Licence:

1 Queueing theory

1.1 Introduction to performance evaluation

The modern digital infrastructure is based on networks formed of hundres of interconnected computer systems. To manage the ingoing and outgoing traffic, *queuing* is essential. **Queueing theory** is the study of what happens when many *jobs* compete for limited resources, often resulting in queues and delays due to the system not being capable of serving every incoming job at once. At its core, this theory seeks to explain why queues form, how they behave and what can be done to minimize or eliminate them.

Consider a simple example: a computer system, such as a web server, handling just one job. The job arrives, consumes some resources (like CPU and I/O), and then departs. Since there are no other jobs in the system, it's easy to predict exactly when it will finish: there's no waiting time and no queue. However, resources are often shared among many jobs, leading to contention and delays.

The first goal of queueing theory is **predicting** the system's performances that we're interested in. The second goal is to **improve design** in order to achieve a better performance. We'll start by giving some concrete examples of performance prediction and system analysis. First, we'll give some definitions.

Definition 1.1: Open and closed system

A system is said to be open if there is at at least one way for jobs to exit the system. If no way exists, the system is said to be closed.

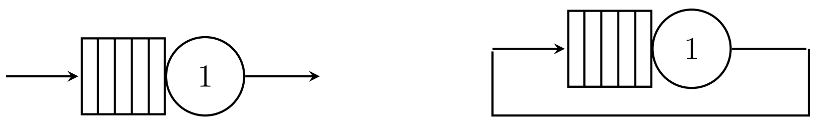


Figure 1.1: An open system and a closed system, both with one server and its queue.

We observe that closed systems may or may not contain a "thinking center", i.e. a subsystem of terminals that does some work before sending jobs to the system. Closed systems that don't contain such thinking center are called *batch systems*, while the others are called *interactive systems*.

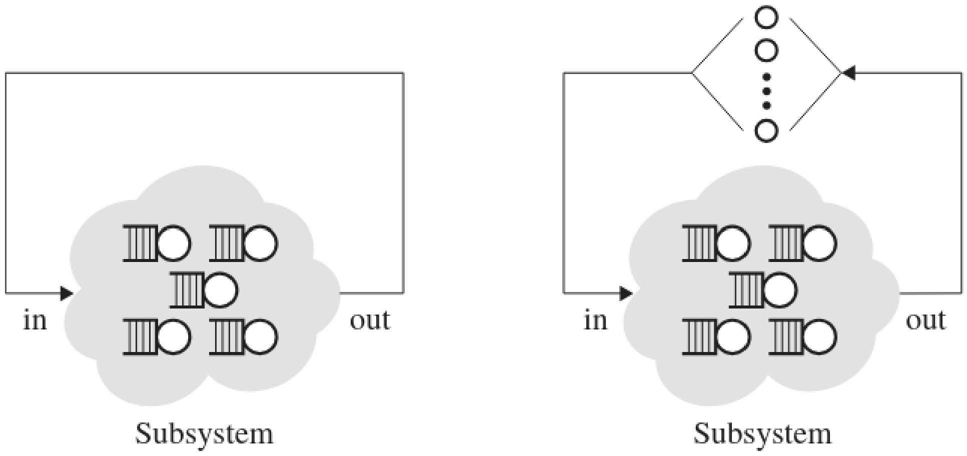


Figure 1.2: A batch system (left) and an interactive system (right)

In order to analyze and predict the behavior of a queueing system, we must first define a few key performance metrics. These metrics describe how jobs arrive, how quickly they are served and how efficiently the system operates overall. We start by defining three main parameters.

The **arrival rate**, typically denoted by λ and measured in jobs per unit of time, represents how frequently new jobs enter the system. For example, if an average of 10 jobs arrive every second, then $\lambda = 10$ jobs/second. In most queueing models, job arrivals are assumed to follow a random process – often a Poisson process – meaning that the times between successive arrivals are independent and exponentially distributed. The arrival rate thus characterizes the workload intensity imposed on the system.

The **service rate**, typically denoted by μ, measures how quickly the system can process jobs. Specifically, it represents the average number of jobs that can be completed per unit of time by a single server. For instance, if a server can complete 5 jobs per second on average, then $\mu = 5$ jobs/second.

The **throughput**, typically denoted by X, indicates the actual rate at which jobs are completed and leave the system. If a server processes 6 jobs per second on average, then $X = 6$ jobs/second.

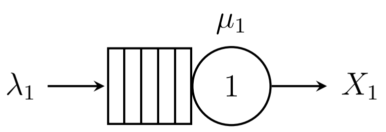


Figure 1.3: A device and its parameters.

To formally define these three metrics, we introduce three **elementary measures** based on elapsed time. Given a time instant t and a device i, let:

1. $A_i(t)$ denote the total number of jobs arrived at device i at time t
2. $C_i(t)$ denote the total number of jobs completed by device i at time t
3. $B_i(t)$ denote the total number of seconds the device i was busy working on jobs at time t

For the arrival rate and the throughput, we can express there two metrics as the ratio between the number of jobs of interest over the total amount of time t that elapsed (as $t \to +\infty$).

Definition 1.2: Arrival rate and Throughput

Given a device i, we define the arrival rate λ_i and the throughput X_i of device i as:

$$\lambda_i = \lim_{t \to +\infty} \frac{A_i(t)}{t} \qquad X_i = \lim_{t \to +\infty} \frac{C_i(t)}{t}$$

For the service rate, instead, we must restrict our interest only to the portion of time for which the device was actually busy working (otherwise, we would also be accounting for the time spent by jobs waiting in the queue).

Definition 1.3: Service rate

Given a device i, we define the service rate μ_i of device i as:

$$\mu_i = \lim_{t \to +\infty} \frac{C_i(t)}{B_i(t)}$$

These three fundamental measures can also be defined at **global** level, i.e. for the whole system. In this case, they are usually denoted with λ, μ, X and they are defined as:

$$\lambda = \lim_{t \to +\infty} \frac{A(t)}{t} \qquad \mu = \lim_{t \to +\infty} \frac{C(t)}{B(t)} \qquad X = \lim_{t \to +\infty} \frac{C(t)}{t}$$

where $A(t)$ and $C(t)$ are the total number of arrived and completed jobs in the system, while $B(t)$ is the total number of seconds that the system was busy working (i.e. with at least one device working). From now on, we'll always assume that the symbols λ, μ and X refer to these three parameters. When we have a system with multiple servers, we'll use pedices to distinguish the servers' parameters (e.g. λ_1, μ_1, X_1 are the parameters of Server 1). When the subscript is omitted, we're usually referring to the global metric, i.e. the metric over the whole system.

1.2 System stability

In order for performance measures to actually mean something, the system must be working at full capacity (hence why we consider $t \to +\infty$). When the system is closed, this hypothesis is not required since even a single job is able to make the system always operational (no idle time). We observe that these three parameters are strictly related to each other and may act as a bottleneck:

- If $\lambda < \mu$, the system receives at most as many jobs than those that it can process, meaning that it is able to serve them all. Hence, we get that $X = \lambda$.
- If $\lambda \geq \mu$, the system receives more jobs than those that it can process, meaning that it will be forced to drop some of them. Hence, we get that $X = \mu$.

We observe that the relationship between the arrival rate and the service rate not only defines the value of the throughput, but it also defined the **stability** of the system. A system is said to be *stable* when there is no queue build-up any device of the system, i.e. the size of every queue of each device doesn't "explode", becoming too large (thus forcing the devices to drop incoming jobs). It's easy to see that a system is stable if and only if each device is able to process jobs faster than they arrive in the queue, meaning that $\lambda_i < \mu_i$.

Proposition 1.1: System stability

A system is stable if and only if for every device i it holds that $\lambda_i \leq \mu_i$.

Proof. We prove both directions. First, it's easy to see that if $\lambda_i \leq \mu_i$ holds for every i then each device is able to process every incoming job without queue build-up, concluding that the system is stable.

Let $A_i^{\text{queue}}(t)$ and $D_i^{\text{queue}}(t)$ respectively denote the number of jobs arrived to and departed from the queue of device i up to time t. Similarly, let $N_i^{\text{queue}}(t)$ be the number of jobs in the queue of device i at time t. We observe that:

$$N_i^{\text{queue}}(t) = A_i^{\text{queue}}(t) - D_i^{\text{queue}}(t)$$

By definition, we have that $A_i^{\text{queue}}(t) = A_i(t) = \lambda t$. Similarly, we have that $D_i^{\text{queue}}(t) \leq \mu t$ (due to the fact that some jobs may be dropped due to the queue being full). This implies that:

$$N_i^{\text{queue}}(t) = A_i^{\text{queue}}(t) - D_i^{\text{queue}}(t) \geq \lambda t - \mu t = (\lambda - \mu)t$$

By contrapositive, suppose that there is at least one device j such that $\lambda_j > \mu_j$. Then, as $t \to +\infty$ we get that $N_j^{\text{queue}}(t) = +\infty$, meaning that the system is unstable. $\square$

Clearly, we want to restrict our interest to systems that are stable since measuring the performance of an unstable system has no real use (every measure either goes to infinity or to zero as time elapses). Hence, from now on we'll $\underline{\text{assume to be working with stable systems}}$. Since $\lambda_i = \mu_i$ holds for every device in stable systems, we know that the throughput is limited by the arrival rate, thus $X_i = \lambda_i < \mu_i$. In closed systems, the equality $X_i = \lambda_i$ always holds (even in unstable conditions) due to necessary packet drops.

Proposition 1.2

Given a device i, it holds that:

- If the system is closed then $X_i = \lambda_i$
- If the system is open then $X_i = \lambda_i$ if and only if the system is stable

Each time we want to change some part of the system, we have to ensure that stability is preserved. For instance, consider the following system with probabilistic routing.

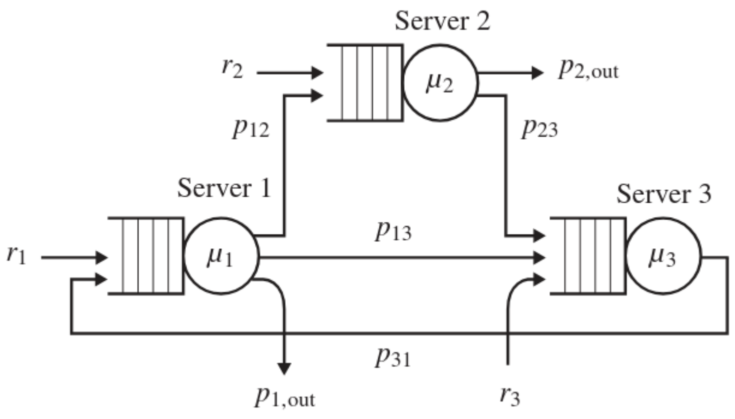


Figure 1.4: A system with probabilistic routing

Here, each device i receives external arrivals with rate r_i, but it also receives internal arrivals from some of the other servers. The sum of these two arrival rates corresponds to the total arrival rate λ_i. Each packet that is processed by device i is next routed to server j with probability $p_{i,j}$, where the device "out" represents the outside of the system.

Suppose that $\mu_1 = \mu_2 = \mu_3 = 10$ j/s (jobs/seconds) and that $r_2 = r_3 = 1$ j/s. The routing probabilities are set to the following values:

- $p_{1,2} = p_{2,out} = 0.8$
- $p_{2,3} = p_{1,3} = 0.2$
- $p_{3,1} = 1$

We want to find the maximum possible value for the external arrival rate r_1, preserving stability. First, we observe that:

$$\begin{cases} \lambda_1 = r_1 + p_{3,1}X_3 \\ \lambda_2 = r_2 + p_{1,2}X_1 \\ \lambda_3 = r_3 + p_{1,3}X_1 + p_{2,3}X_2 \end{cases}$$

Since we want stability to hold, we can assme that $X_i = \lambda_i$ holds for each device i. Substituting the probabilities, we obtain that:

$$\begin{cases} \lambda_1 = r_1 + \lambda_3 \\ \lambda_2 = r_2 + \frac{8}{10}\lambda_1 \\ \lambda_3 = r_3 + \frac{2}{10}\lambda_1 + p_{2,3}\lambda_2 \end{cases}$$

Solving the system in function of r_1, we obtain that:

$$\begin{cases} \lambda_1 = \frac{15}{8} + \frac{25}{16}r_1 \\ \lambda_2 = \frac{5}{4} + \frac{5}{4}r_1 \\ \lambda_3 = \frac{15}{8} + \frac{9}{16}r_1 \end{cases}$$

To impose stability, we know that $\lambda_i < \mu_i$ must hold for every device i. Therefore, we get that:

$$\begin{cases} 10 > \frac{15}{8} + \frac{25}{16}r_1 \\ 10 > \frac{5}{4} + \frac{5}{4}r_1 \\ 10 > \frac{15}{8} + \frac{9}{16}r_1 \end{cases}$$

Solving each inequality, we conclude that:

$$\begin{cases} r_1 < \frac{26}{5} \\ r_1 < 6 \\ r_1 < \frac{130}{9} \end{cases}$$

Thus, in order for the system to be stable the value of r_1 must be such that:

$$r_1 < \min\left(\frac{26}{5}, 6, \frac{130}{9}\right) = \frac{26}{5}$$

1.3 Utilization and time analysis

In the previous sections introduced rate metrics to analyze the speed of the system. But what about time metrics? In particular, we're interested in knowing performances such as how much is a device working, the time required by a job to be processed by the whole system and by a single device.

The first metric that we're going to introduce is the **utilization** of a device, written as ρ_i, which describes how busy a device is. Intuitively, this metric is represented as the ratio between the the time in which the device was busy working over the total elapsed time.

Definition 1.4: Utilization

Given a device i, we define the utilization ρ_i as:

$$\rho_i = \lim_{t \to +\infty} \frac{B_i(t)}{t}$$

Since the utilization is described as the percentage of time the device was busy working, it's easy to see that:

$$\begin{aligned}\rho_i &= \Pr[\text{device } i \text{ is busy}]\\ &= \Pr[N_i^{\text{server}} = 1]\\ &= 1 \cdot \Pr[N_i^{\text{server}} = 1] + 0 \cdot \Pr[N_i^{\text{server}} = 0]\\ &= \mathbb{E}[N_i^{\text{server}}]\end{aligned}$$

where N_i^{server} is the **server population** of device i, that being the number of jobs in the server of device i (thus not in the queue), which is at most 1.

The second metric that we're going to introduce is **service time**, typically written as S_i. Intuitively, this is the time spent by a device to process a job. By definition, the average service time of a device is nothing more than the inverse of its service rate.

Definition 1.5: Service time

Given a device i, we define the service time S_i of device i as the random variable such that:

$$\mathbb{E}[S_i] = \frac{1}{\mu_i}$$

Clearly, the above definition implies that:

$$\mathbb{E}[S_i] = \frac{1}{\mu_i} = \lim_{t \to +\infty} \frac{B_i(t)}{C_i(t)}$$

This allows us to formalize our first law, known as the **utilization law**.

Law 1.1: Utilization law

Given a device i, it holds that:

$$\rho_i = \mathbb{E}[S_i]X_i$$

Proof through definitions. Through easy algebraic manipulation we get that:

$$X_i = \frac{C_i(\tau)}{\tau} = \frac{C_i(\tau)}{B_i(\tau)} \cdot \frac{B_i(\tau)}{\tau} = \mu_i\rho_i$$

Thus, we conclude that $\rho_i = \mathbb{E}[S_i]X_i$. □

Proof through expected value. By definition, we have that:

$$\begin{aligned}X_i &= \mathbb{E}[\text{Completion rate of } i]\\ &= \mathbb{E}[\text{Completion rate of } i \mid i \text{ is busy}] \Pr[i \text{ is busy}]\\ &\quad + \mathbb{E}[\text{Completion rate of } i \mid i \text{ isn't busy}] \Pr[i \text{ isn't busy}]\\ &= \mu_i\rho_i + 0(1-\rho_i)\\ &= \mu_i\rho_i\end{aligned}$$

Thus, we conclude that $\rho_i = \mathbb{E}[S_i]X_i$. □

To avoid confusion, we remark that service time referes only to the time spent by the job inside the device itself, thus without considering the time spent by it inside its queue. This latter metric is known as **waiting time**, written as W_i. When considering both portion sof time, the metric is known as **response time**, the time spent by a single job be fully served by a device.

Definition 1.6: Response time

Given a device i, we define the response time R_i of device i as the random variable such that:

$$\mathbb{E}[R_i] = \mathbb{E}[W_i] + \mathbb{E}[S_i]$$

where W_i is the time spent by a job inside the queue of device i.

Consider a single device i. When the system is stable, we know that jobs are being processed faster than their arrival in the queue ($\mu_i > \lambda_i$). In this case, the value $\mu_i - \lambda_i$ represents how much faster the server can serve customers than they arrive, namely the *net service capacity*. It's easy to see that the inverse of such metric is exactly the average response time of the device (a formal proof will be given in Chapter 3).

Proposition 1.3

Consider a stable system. Given a device i, it holds that:

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

Through the above result, we also conclude that:

$$\mathbb{E}[W_i] = \mathbb{E}[R_i] - \mathbb{E}[S_i] = \frac{1}{\mu_i - \lambda_i} - \frac{1}{\mu_i} = \frac{\lambda_i}{\mu_i(\mu_i - \lambda_i)}$$

Since $X_i = \lambda_i$ holds, through the utilization law, we conclude that:

$$\mathbb{E}[W_i] = \frac{\lambda_i}{\mu_i(\mu_i - \lambda_i)} = \frac{X_i}{\mu_i(\mu_i - \lambda_i)} = \rho\,\mathbb{E}[R_i]$$

In other words, the average time spend by a time in the queue is given by the average time required by a job to be fully processed by the device and the workload of the device itself.

1.4 Little's law

We introduce two new global metrics: the system traversal time and the system response time. The **system traversal time** can be informally described as the time required by a job to go from "out" of the system to "out" of the system. The **system response time**, instead, can be informally described as the time required by a job to go from "into" the system to "out" of the system.

For open systems, it holds that $\mathbb{E}[T] = \mathbb{E}[R]$ since there is no difference from coming out of the system or into the system due to all jobs coming from the outside. For closed system, instead, the traversal time may also contain time required by the system to "think" (e.g. we're working with an interactive system). Thus, we have that $\mathbb{E}[T] = \mathbb{E}[R] + \mathbb{E}[Z]$, where Z is the **thinking time** of the system. If the closed system has no thinking center, we can simply set $\mathbb{E}[Z] = 0$.

Definition 1.7: Traversal time

The traversal time T of a system is the random variable such that:

- $\mathbb{E}[T] = \mathbb{E}[R]$ if the system is open
- $\mathbb{E}[T] = \mathbb{E}[R] + \mathbb{E}[Z]$ if the system is closed

To compute the value of $\mathbb{E}[R]$, we usually use Little's Law, one of the most powerful and elegant results in queueing theory and operations management. It provides a simple relationship between three fundamental performance measures of any stable system. Formally, it states that $\mathbb{E}[T] = \frac{\mathbb{E}[N]}{X}$, where N is the number of jobs in the system. In order for such law to hold, the system must be **ergodic**.

Definition 1.8: Ergodic system

A system is said to be ergodic if it has the following three properties:

- *Irreducibility*: from every state (defined as the population in the servers) the system can reach every other state and back.
- *Positive recurrence*: starting from a state, the probability to return in that same state over infinite time is 1.
- *Aperiodicity*: the system doesn't have a periodic behavior, meaning that it doesn't return in a specific state after a precise amount of time.

Law 1.2: Little's law

Given an ergodic system, it holds that:

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{X}$$

Proof. Fix a generic time instant t. Let $\mathcal{A}(t)$ denote the area used by the jobs up to time t (see figure Figure 1.5). Interpreting the graph as a continuous function, we get that:

$$\mathcal{A}(t) = \int_0^t N(s)\, ds$$

where $N(s)$ denotes the number of jobs in the system at time s. For each job j, let T_j denote the time required to complete job j. Moreover, let $J_A(t)$, $J_C(t)$ denote, respectively, the set of arrived and completed jobs at time t. We observe that:

$$\sum_{j \in J_C(t)} T_j \leq \mathcal{A}(t) \leq \sum_{j \in J_A(t)} T_j$$

Dividing the whole inequality by t we get that:

$$\frac{1}{t} \sum_{j \in J_C(t)} T_j \leq \frac{1}{t} \int_0^t N(s)\, ds \leq \frac{1}{t} \sum_{j \in J_A(t)} T_j$$

Through algebraic manipulation we get that:

$$\frac{C(t)}{t} \cdot \frac{\sum_{j \in J_C(t)} T_j}{C(t)} \leq \frac{\int_0^t N(s)\, ds}{t} \leq \frac{A(t)}{t} \cdot \frac{\sum_{j \in J_A(t)} T_j}{A(t)}$$

By taking the limit as $t \to +\infty$, we obtain that:

$$\begin{aligned} &\lim_{t \to +\infty} \frac{C(t)}{t} \cdot \frac{\sum_{j \in J_C(t)} T_j}{C(t)} \leq \lim_{t \to +\infty} \frac{\int_0^t N(s)\, ds}{t} \leq \lim_{t \to +\infty} \frac{A(t)}{t} \cdot \frac{\sum_{j \in J_A(t)} T_j}{A(t)} \\ \implies & X \cdot \lim_{t \to +\infty} \frac{\sum_{j \in J_C(t)} T_j}{C(t)} \leq \mathbb{E}[N] \leq X \cdot \lim_{t \to +\infty} \frac{\sum_{j \in J_A(t)} T_j}{A(t)} \end{aligned}$$

since $\lim_{t \to +\infty} \frac{C(t)}{t} = X$ and $\lim_{t \to +\infty} \frac{A(t)}{t} = \lambda = X$. Moreover, we observe that as $t \to +\infty$ the difference between arrival and completion becomes negligible. Thus, we get that:

$$\lim_{t \to +\infty} \frac{\sum_{j \in J_C(t)} T_j}{C(t)} = \mathbb{E}[T] = \lim_{t \to +\infty} \frac{\sum_{j \in J_A(t)} T_j}{A(t)}$$

concluding that $X\,\mathbb{E}[T] \leq \mathbb{E}[N] \leq X\,\mathbb{E}[T]$. □

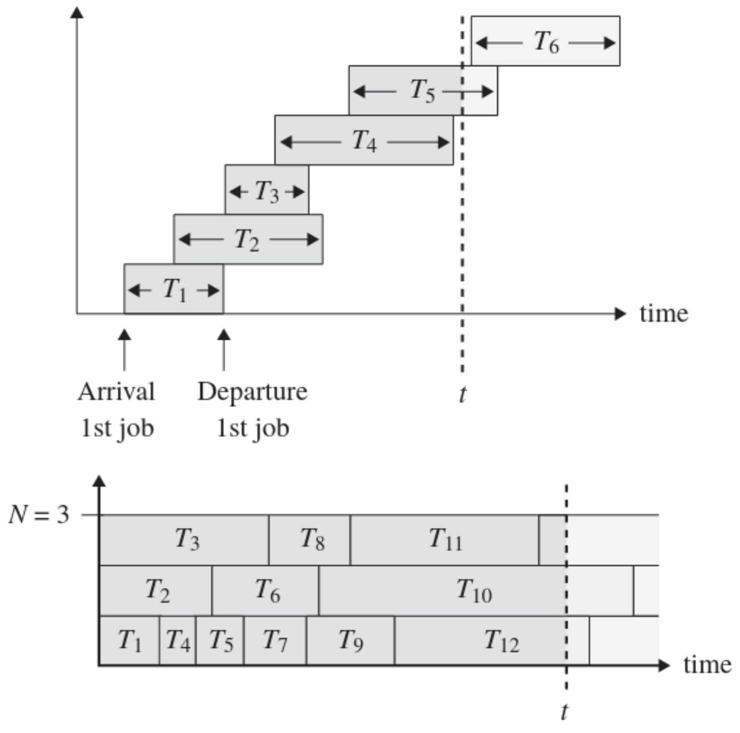


Figure 1.5: Graph of job system times in open system (above) and closed systems (below). The darker area corresponds to $\mathcal{A}(t)$.

After proving the law's correctness, we discuss its applications. First of all, we observe that the law can be made more explicit for open and closed systems by expanding the traversal time. For open systems we have that:

$$\mathbb{E}[R] = \mathbb{E}[T] = \frac{\mathbb{E}[N]}{X}$$

For closed systems, we also observe that $\mathbb{E}[N] = N$ since the number of jobs in the system is constant. Thus:

$$\mathbb{E}[R] = \frac{N}{X} - \mathbb{E}[Z]$$

The latter formula is also known as **response time law for closed systems**.

Nonetheless, the true power of the law comes from the fact that it can be applied not only to the whole system but also to sub-systems. The key idea here is that any sub-system can be viewed as a "box" with ingoing and outgoing jobs. Then, each box can be viewed as an open system on which the law can be applied.

For instance, consider the following interactive system with a central system deployed in the cloud (thus we don't know its interal connections).

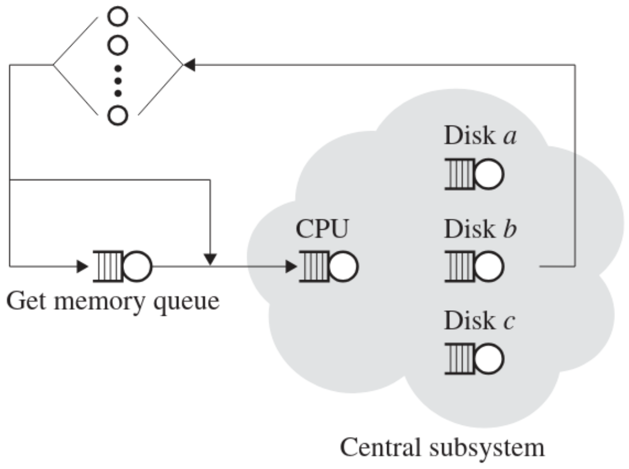


Figure 1.6: An interactive system with a cloud central system

The following information about the system is known:

- $\mathbb{E}[N] = 23$
- $\mathbb{E}[Z] = 21$ secs
- $X = 0.45$ jobs/secs
- $\mathbb{E}[N_{\text{mem}}] = 11.65$

We want to compute the response time of the central cloud system, namely R_{cloud}. With the known information, we can easily apply the response time law to the whole system, obtaining that:

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{X} - \mathbb{E}[Z] = 23 \cdot \frac{100}{45} - 21 = \frac{271}{9} \text{ sec}$$

Now, we use the real power of Little's law. Consider the following sub-system described by the red box. We refer to this sub-system as *mem-sys*.

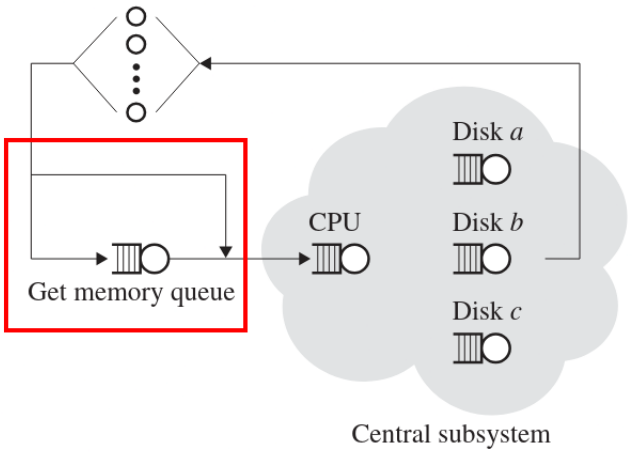


Figure 1.7: The mem-sys sub-system.

Similarly, the cloud center can also be viewed as a sub-system. We observe that

$$\mathbb{E}[R_{\text{cloud}}] = \mathbb{E}[T] - \mathbb{E}[R_{\text{mem-sys}}]$$

Since this mem-sys sub-system can be viewed as an open system, by Little's law we have that:

$$\mathbb{E}[R_{\text{mem-sys}}] = \frac{\mathbb{E}[N_{\text{mem-sys}}]}{X_{\text{mem-sys}}}$$

Now, we observe that $X_{\text{mem-sys}} = X$ since every job of the system enters to and exits from mem-sys. Moreover, we also have that $\mathbb{E}[N_{\text{mem-sys}}] = \mathbb{E}[N_{\text{mem}}]$, concluding that:

$$\mathbb{E}[R_{\text{mem-sys}}] = \frac{\mathbb{E}[N_{\text{mem-sys}}]}{X_{\text{mem-sys}}} = \frac{\mathbb{E}[N_{\text{mem}}]}{X} = \frac{1165}{100}\frac{100}{45} = \frac{233}{9} \text{ sec}$$

Therefore, we conclude that:

$$\mathbb{E}[R_{\text{cloud}}] = \mathbb{E}[T] - \mathbb{E}[R_{\text{mem-sys}}] = \frac{271}{9} - \frac{233}{9} = \frac{38}{9}$$

On a simpler case, Little's law can also be used to derive the utilization law. Given a system made of a single device, we can consider the sub-system composed only by the server. Since there is no queue in the system, we know that $\mathbb{E}[R_i^{\text{server}}] = \mathbb{E}[S_i^{\text{server}}]$. Then, byLittle's law we conclude that:

$$\mathbb{E}[S_i^{\text{server}}] = \mathbb{E}[R_i^{\text{server}}] = \frac{\mathbb{E}[N_i^{\text{server}}]}{X_i^{\text{server}}} = \frac{\rho}{X_i}$$

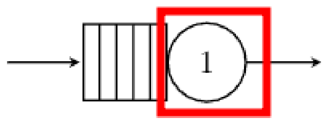


Figure 1.8: The sub-system considered to derive the utilization law

1.5 Forced flow law

Consider the following interactive system with a CPU device. The device has two outgoing links: one that goes back to the terminal center and one that loops back into the device itself. Each job exiting the device has $\frac{1}{2}$ probability of looping back and $\frac{1}{2}$ probability of going back to the terminal.

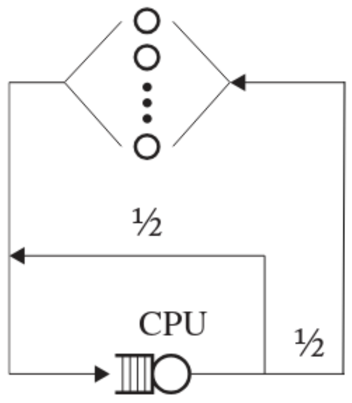


Figure 1.9: An interactive system with a fork with 0.5 probability of going back to the terminals.

Consider now the sub-system formed of the CPU device and the looping link (represented by the red box in Figure 1.10). Since every job of the system enters and exits from this sub-system, we know that the throughput of the latter must be equal to the system throughput. Thus, we have that $X_{\text{sub-sys}} = X$.

But what about the throughput of the sub-system formed only by the CPU device (represented by the green box in Figure 1.10)? We notice that every job that exits the CPU and goes back to the terminal center gets accounted both for the CPU throughput and the global system throughput. Since each exiting job has $\frac{1}{2}$ probability of going back to the terminal center, we can expect that $X = \frac{1}{2}X_i$. Indeed, this is the case!

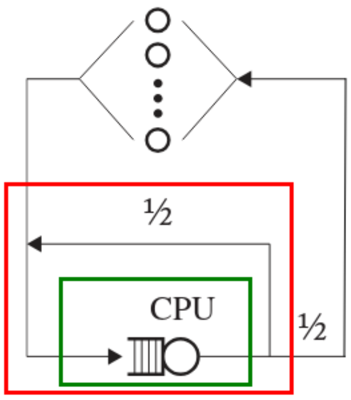


Figure 1.10: The two sub-systems.

In general, the relationship between the overall system throughput and the throughput of each individual service center within the system is described by the **forced flow law**, which formally states that $X_i = \mathbb{E}[V_i]X$. Here, V_i is the **visit ratio** of the device, that being the number of visits any job makes to the device.

Definition 1.9: Visit ratio

Given a device i, we define the visit ratio S_i of device i as:

$$V_i = \lim_{t \to +\infty} \frac{C_i(t)}{C(t)}$$

Law 1.3: Forced flow law

Given a device i, it holds that:

$$X_i = \mathbb{E}[V_i]X$$

Proof. Let $V_i^{(j)}$ be the number of visits that the j-th job makes to device i. Let $J_C(t)$ be the set of completed jobs at time t. We observe that:

$$C_i(t) = \sum_{j \in J_C(t)} V_i^{(j)}$$

Dividing the whole equality by t and rewriting the right-hand side, we get that:

$$\frac{C_i(t)}{t} = \frac{\sum_{j \in J_C(t)} V_i^{(j)}}{t} = \frac{\sum_{j \in J_C(t)} V_i^{(j)}}{C(t)} \cdot \frac{C(t)}{t}$$

By taking the limit as $t \to +\infty$, we obtain that:

$$\lim_{t \to +\infty} \frac{C_i(t)}{t} = \lim_{t \to +\infty} \frac{\sum_{j \in J_C(t)} V_i^{(j)}}{C(t)} \cdot \frac{C(t)}{t} \implies X_i = \mathbb{E}[V_i]X$$

$\square$

Computing the value of $\mathbb{E}[V_i]$ is not an easy task since it highly depends on the structure of the system and the probabilities of each link. For instance, consider the following generalization of the previous example.

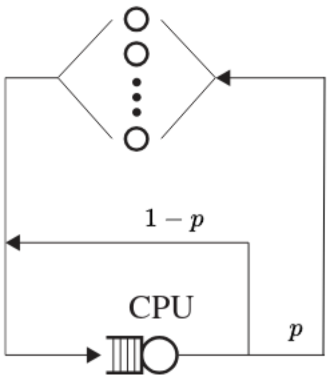


Figure 1.11: An interactive system with a fork with p probability of going back to the terminals.

First of all, we observe that $\Pr[V_i = k] = \Pr[\text{the job exits after looping } k-1 \text{ times}]$ since one visit is accounted when the job exits the loop. Moreover, probabilistically speaking, it is possible for a job to loop indefinitely in the system. Thus, we have that:

$$\mathbb{E}[V_i] = \sum_{k=0}^{+\infty} k \Pr[V_i = k] = \sum_{k=0}^{+\infty} k \Pr[\text{the job exits after looping } k-1 \text{ times}]$$

Now, it's easy to see that the event of looping $k-1$ times inside the device is described by a geometric process: in order to exit at the k-th iteration, the job must first loop for $k-1$ times, meaning that:

$$\Pr[\text{the job exits after looping } k-1 \text{ times}] = (1-p)^{k-1}p$$

Thus, we get that:

$$\mathbb{E}[V_i] = \sum_{k=0}^{+\infty} k(1-p)^k p$$

concluding that V_i is an r.v. with geometric distribution of parameter p. Identifying the distribution type of V_i allows us to immediately compute its expected value through known results. In particular, since $V_i \sim \text{Geo}(p)$, we immediately get that $\mathbb{E}[V_i] = \frac{1}{p}$. In other words, the averagae number of visits received by device i is the inverse of the probability of a job exiting the loop. Going back to our original example, we get that $X_i = 2X$ as expected by our observation.

The forced flow law also gives us an additional way to express the average system response time in terms of the average response time of the devices composing the system.

Proposition 1.4

Given a system with n devices, it holds that:

$$\mathbb{E}[R] = \sum_{i=1}^{n} \mathbb{E}[V_i]\,\mathbb{E}[R_i]$$

Proof. First, we observe that $\mathbb{E}[N] = \mathbb{E}[N_1] + \ldots + \mathbb{E}[N_n]$. By applying Little's law on each device and the whole system, we get that:

$$\mathbb{E}[R]X = \mathbb{E}[R_i]X_1 + \ldots + \mathbb{E}[R_n]X_n$$

By dividing both sides by X and by applying the forced flow law, we conclude that:

$$\mathbb{E}[R] = \mathbb{E}[V_1]\,\mathbb{E}[R_1] + \ldots + \mathbb{E}[V_n]\,\mathbb{E}[R_n]$$

□

1.6 Demand law and bottleneck law

The last fundamental metric that we're going to introduce is **device demand**, that being the effective load per job placed on a device. This metric is defined as the sum over all the single service time intervals per visit of a job.

Definition 1.10: Demand

Given a device i, the demand of i, written as D_i, is defined as:

$$D_i = \sum_{j=1}^{V_i} S_i^{(j)}$$

where $S_i^{(j)}$ is the service time of i for the j-th job.

Intuitively, even if a device is fast (small S_i), it may have an high visit count (high V_i), making the effective load higher. The relationship between these three metrics is described by the **demand law**.

Law 1.4: Demand law

Given a device i, it holds that:

$$\mathbb{E}[D_i] = \mathbb{E}[V_i]\,\mathbb{E}[S_i]$$

Proof. Before proving the result, we discuss its non-trivialness: even tought $S_i = \sum_{j=1}^{+\infty} S_i^{(j)}$, we cannot directly establish the result since V_i is a random variable – which may be dependent on other variables. In fact, an independence assumption will be made throughout the proof. First, we observe that:

$$\mathbb{E}[D_i] = \mathbb{E}\left[\sum_{j=1}^{V_i} S_i^{(j)}\right] = \sum_{n=0}^{+\infty} \mathbb{E}\left[\sum_{j=1}^{n} S_i^{(j)} \mid V_i = n\right] \Pr[V_i = n]$$

Here, we assume that the number of visits a job makes to device i is not affected by its service demand at the device (it's easy to see that this assumption is realistic in almost every case). Thus, we get that:

$$\begin{aligned}\mathbb{E}[D_i] &= \sum_{n=0}^{+\infty} \mathbb{E}\left[\sum_{j=1}^{n} S_i^{(j)}\right] \Pr[V_i = n] \\ &= \sum_{n=0}^{+\infty} n\,\mathbb{E}[S_i]\Pr[V_i = n] \\ &= \mathbb{E}[S_i]\,\mathbb{E}[V_i]\end{aligned}$$

□

Through the demand law we also get another formulation of demand in terms of elementary measures:

$$\mathbb{E}[D_i] = \mathbb{E}[V_i]\,\mathbb{E}[S_i] = \lim_{t\to+\infty} \frac{C_i(t)}{C(t)}\frac{B_i(t)}{C_i(t)} = \lim_{t\to+\infty} \frac{B_i(t)}{C(t)}$$

Out of all the metrics that we have discussed so far, demand is one of the most important ones to analyze in a system. This property is described by the **bottleneck law**

Law 1.5: Bottleneck law

Given a device i, it holds that:

$$\rho_i = \mathbb{E}[D_i]X$$

Proof. By applying the demand law, the forced flow law and the utilization law, we get that:

$$\mathbb{E}[D_i] = \mathbb{E}[V_i]\,\mathbb{E}[S_i] = \frac{\mathbb{E}[V_i]}{X}\,\mathbb{E}[S_i] = \frac{\rho_i}{X}$$

$\square$

The bottleneck law establishes a deep connection between the system throughput and the device with the highest expected demand. Fix any device i. Since $0 \leq \rho \leq 1$, through the bottleneck law we obtain that:

$$X = \frac{\rho_i}{\mathbb{E}[D_i]} \leq \frac{1}{\mathbb{E}[D_i]} \leq \frac{1}{D_{\max}}$$

where $D_{\max} = \max_i \mathbb{E}[D_i]$. This inequality establishes that the throughput of the system is "choked" by the device with the highest expected demand, referred to as the *bottleneck* (hence the name of the law). Through Little's law, we can also obtain a lower bound for the response time. For closed systems, we have that:

$$\mathbb{E}[R] = \frac{N}{X} - \mathbb{E}[Z] \geq ND_{\max} - \mathbb{E}[Z]$$

while for open systems we have that:

$$\mathbb{E}[R] = \frac{\mathbb{E}[N]}{X} \geq ND_{\max}$$

Another lower bound for the response time is given by the case of lowest possible congestion, i.e. when only one job is in the system. Let $R(n)$ denote the response time when n jobs are in the system. By definition, we have that $\mathbb{E}[R] = \mathbb{E}[R(N)]$. When only one job is in the system, we expect the job to pass through every device of the system on average. Thus, we conclude that:

$$\mathbb{E}[R] = \mathbb{E}[R(N)] \geq \mathbb{E}[R(1)] = D_{\text{tot}}$$

where $D_{\text{tot}} = \sum_i \mathbb{E}[D_i]$. Again, through Little's law this can be turned into another higher bound for the throughput.

$$X = \frac{N}{\mathbb{E}[R] + \mathbb{E}[Z]} \leq \frac{N}{D_{\text{tot}} + \mathbb{E}[Z]}$$

Law 1.6: Asymptotic bounds

For any system it holds that:

$$X \leq \min\left(\frac{\mathbb{E}[N]}{D_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D_{\max}}\right) \qquad R \geq \max\left(D_{\text{tot}}, \mathbb{E}[N]D_{\max} - \mathbb{E}[Z]\right)$$

Note: the above bounds are correct both for open and closed systems (in the former $\mathbb{E}[Z] = 0$, while in the latter $\mathbb{E}[N] = N$)

We observe that the above bounds are **tight** for small values of N only in closed systems since it is always pushing the throughput to the maximum ($X_i = \mu_i$), while in open systems this is not the case ($X_i = \lambda_i < \mu_i$).

We denote with N^* the average job population where the two metrics switch bound (e.g. X goes from the first bound to the second). For the throughput, this value is given by:

$$\frac{N_X^*}{D_{\text{tot}} + \mathbb{E}[Z]} = \frac{1}{D_{\max}} \implies N_X^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}}$$

For the response time, instead, the value is given by:

$$D_{\text{tot}} = N_R^* D_{\max} - \mathbb{E}[Z] \implies N_R^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}}$$

thus $N_X^* = N_R^*$. When the number of jobs N is below N^*, we know that both the system throughput and the response time are bounded by the total demand of the system. This case is known as **low population**. When $N > N^*$, instead, they are both bounded by $D_{\max}$. This case is known as **high population**.

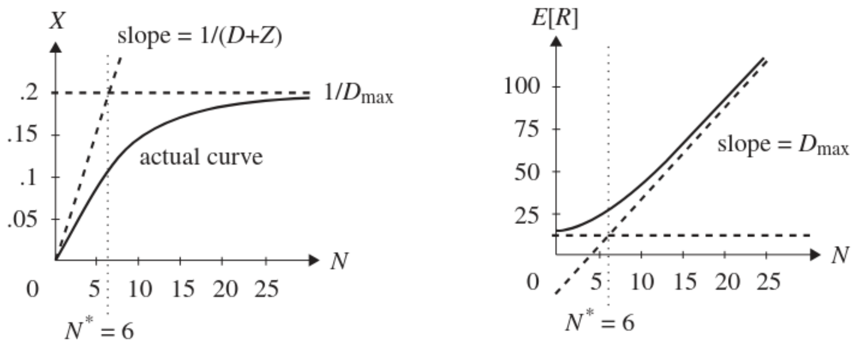


Figure 1.12: Performances of the system as functions of N

From this analysis, we get that reducing the demand of any device in low population will improve both metrics. When in high population, instead, we are forced to reduce the demand of the *bottleneck device* in order to improve both metrics. Through the demand law, we know that the demand of a device is given by the average number of visits it receives and the average service time. Thus, to change the demand we can either change the routing policy of the router (thus changing the number of visits for the device) or change the device itself (thus changing its service time).

For instance, consider the following interactive system connected to three cloud devices: a CPU and two disk called A and B. The known information is the following: $\mathbb{E}[Z] = 18$ secs, $\mathbb{E}[D_{\text{CPU}}] = 5$ secs, $\mathbb{E}[D_A] = 4$ secs, $\mathbb{E}[D_B] = 3$ secs and $\mathbb{E}[N] = 4$.

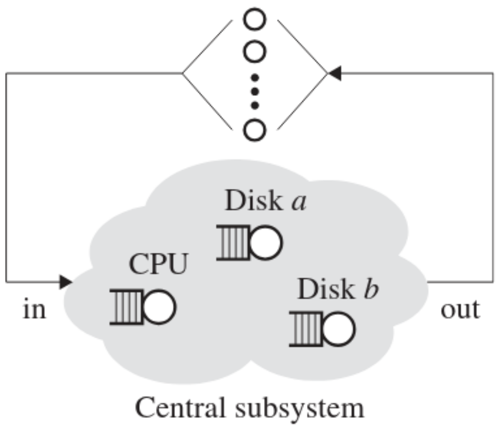


Figure 1.13: An interactive system with three cloud devices.

We're asked to find out the current maximum throughput, minimum response time and if replacing disk A with new disk that is twice as fast would improve such bounds. First, we compute the bounds:

$$X \leq \min\left(\frac{\mathbb{E}[N]}{D_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D_{\max}}\right) = \min\left(\frac{N}{30}, \frac{1}{5}\right)$$

$$R \geq \max\left(D_{\text{tot}}, \mathbb{E}[N]D_{\max} - \mathbb{E}[Z]\right) = \max\left(12, 5N - 18\right)$$

then the knee-point of the bounds:

$$N^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}} = \frac{12 + 18}{5} = 6$$

concluding that:

$$X \leq \begin{cases} \frac{1}{30}N & \text{if } N < 6 \\ 0.2 & \text{if } N \geq 6 \end{cases} \qquad R \geq \begin{cases} 12 & \text{if } N < 6 \\ 5N - 18 & \text{if } N \geq 6 \end{cases}$$

We observe that $D_{\max} = D_{\text{CPU}} = 5$, hence the CPU is the bottleneck of the system. However, we cannot directly conclude that changing disk A with a faster one wouldn't improve the system. In fact, since $\mathbb{E}[N] < N^*$, we're in low population, meaning that we have to reduce the total demand in order to improve the bounds. Hence, our change to disk A would effectively reduce the system. Since disk A' is twice as fast as disk A, it holds that $\mu_{A'} = 2\mu_A$ and thus that $\mathbb{E}[S_{A'}] = \frac{1}{2}\mathbb{E}[S_A]$.

$$\mathbb{E}[D_{A'}] = \mathbb{E}[S_{A'}]\,\mathbb{E}[V_{A'}] = \frac{1}{2}\mathbb{E}[S_A]\,\mathbb{E}[V_A] = \frac{1}{2}\mathbb{E}[D_A] = 2$$

Thus $D'_{\text{tot}} = 10$, improving the bound for low population:

$$X' \leq \min\left(\frac{\mathbb{E}[N]}{D'_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D'_{\max}}\right) = \left(\frac{N}{28}, \frac{1}{5}\right)$$

$$R' \geq \max\left(D'_{\text{tot}}, \mathbb{E}[N]D'_{\max} - \mathbb{E}[Z]\right) = \max\left(10, 5N - 18\right)$$

1.7 Load balancing

Consider an interactive system with a CPU, a fast disk and a slow disk. We don't know the structure of the system, but we know that $\mathbb{E}[Z] = 15$ secs and $N = 20$. At time instant τ, we registered the following data:

- $C(\tau) = 200$
- $C_{\text{CPU}}(\tau) = 200$ and $B_{\text{CPU}}(\tau) = 400$ sec
- $C_{\text{FD}}(\tau) = 20000$ and $B_{\text{CPU}}(\tau) = 600$ sec
- $C_{\text{SD}}(\tau) = 2000$ and $B_{\text{CPU}}(\tau) = 100$ sec

We want to know if replacing the CPU with a faster one would improve the performances of the system. First, we compute the demands of the various devices by using the registered data:

$$D_{\text{CPU}} = \frac{B_{\text{CPU}}(\tau)}{C(\tau)} = \frac{400}{200} = 2 \text{ sec}$$

$$D_{\text{FD}} = \frac{B_{\text{FD}}(\tau)}{C(\tau)} = \frac{600}{200} = 3 \text{ sec}$$

$$D_{\text{SD}} = \frac{B_{\text{SD}}(\tau)}{C(\tau)} = \frac{100}{200} = 0.5 \text{ sec}$$

obtaining $D_{\text{tot}} = 5.5$ sec and $D_{\max} = D_{\text{FD}}$, meaning that the CPU isn't the bottleneck of the system. Hence, changing the CPU with a faster one would improve the performances only if we're in low population (since it would reduce the value of D_{tot}). However, we observe that:

$$N^* = \frac{\mathbb{E}[Z] + D_{\text{tot}}}{D_{\max}} = \frac{15 + 5.5}{3} = \frac{20.5}{3} < 20 = N$$

hence we're in high population, concluding that this change wouldn't improve the system at all. Are there other ways to improve the system without changing the fast disk with a faster one? The answer is yes! In fact, we have two methods to improve the system:

change the bottleneck with a faster device (thus reducing the value of $S_{\max}$) or reduce the number of jobs that reach the devices (thus reducing the value of $V_{\max}$).

Clearly, we cannot force the system to send less jobs to the bottleneck since all the re-routed jobs would still need to be served. The solution here is to **balance the load** of the system, i.e. re-routing jobs from the bottleneck to other devices. For instance, in our example we could try to re-route jobs from the fast disk to the slow disk, preserving the total number of visits.

It's easy to see that this operation not only reduces D_{FD} but also increases D_{SD}. However, if we reduce the demand of the fast disk by too much, the slow disk may become the new bottleneck. Hence, the minimum value of the maximum demand is reached when both devices have the same demand.

With these two constraints, we can model the following system of equations to find out the new values of $\mathbb{E}[V'_{\text{FD}}]$ and $\mathbb{E}[V'_{\text{SD}}]$:

$$\begin{cases} \mathbb{E}[V'_{\text{FD}}] + \mathbb{E}[V'_{\text{SD}}] = \mathbb{E}[V_{\text{FD}}] + \mathbb{E}[V_{\text{SD}}] \\ \mathbb{E}[V'_{\text{FD}}]\,\mathbb{E}[S'_{\text{FD}}] = \mathbb{E}[V'_{\text{SD}}]\,\mathbb{E}[S'_{\text{SD}}] \end{cases} \implies \begin{cases} \mathbb{E}[V'_{\text{FD}}] + \mathbb{E}[V'_{\text{SD}}] = \frac{C_{\text{FD}}(\tau)}{C(\tau)} + \frac{C_{\text{SD}}(\tau)}{C(\tau)} \\ \mathbb{E}[V'_{\text{FD}}]\frac{B_{\text{FD}}(\tau)}{C_{\text{FD}}(\tau)} = \mathbb{E}[V'_{\text{SD}}]\frac{B_{\text{SD}}(\tau)}{C_{\text{SD}}(\tau)} \end{cases}$$

$$\implies \begin{cases} \mathbb{E}[V'_{\text{FD}}] + \mathbb{E}[V'_{\text{SD}}] = 110 \\ \frac{3}{100}\,\mathbb{E}[V'_{\text{FD}}] = \frac{5}{100}\,\mathbb{E}[V'_{\text{SD}}] \end{cases} \implies \begin{cases} \mathbb{E}[V'_{\text{FD}}] = \frac{275}{4} \\ \mathbb{E}[V'_{\text{SD}}] = \frac{165}{4} \end{cases}$$

With these new values, we obtain that:

$$D'_{\text{FD}} = \mathbb{E}[V'_{\text{FD}}]\,\mathbb{E}[S'_{\text{FD}}] = \frac{275}{4}\frac{3}{100} = \frac{33}{16} \approx 2.06$$

$$D'_{\text{SD}} = \mathbb{E}[V'_{\text{SD}}]\,\mathbb{E}[S'_{\text{SD}}] = \frac{165}{4}\frac{5}{100} = \frac{33}{16} \approx 2.06$$

concluding that we improved the performance of the system without spending money to buy a faster disk.

1.8 Solved exercises

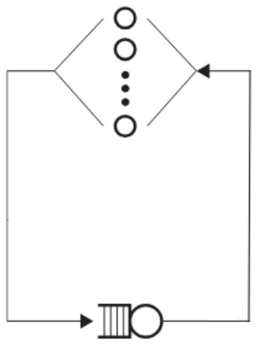
Problem 1.1

Consider a multi-tier system made of an application layer and a database layer. The first layer queries the second one, which then returns the asked data after retrieving it. We know that the system is running 400 simultaneous jobs and that the database layer manages 55 jobs/sec. Moreover, we know that the application layer requires an average think time of 150 ms per job.

- Model the systems and formalize the given parameters.
- Compute the response time of the database layer during peak hours.
- Does the response time improve if the application layer is replaced with one twice as fast? What's the reasoning behind this result?

Solution:

The system can be modeled as the following interactive system, where the terminal center corresponds to the application layer and the device corresponds to the database layer. The system's parameter are $N = 400$, $\mathbb{E}[Z] = 1.5$ msec and $\mu_{\text{DB}} = 55$ job/sec.



During peak hours, we know that the utilization of the the device is maximum, i.e. that $\rho_{\text{DB}} = 1$. Through the utilization law, we get that

$$X_{\text{DB}} = \frac{\rho_{\text{DB}}}{\mathbb{E}[S_{\text{DB}}]} = \mu_{\text{DB}}$$

Moreover, since we have only one device, we know that $X = X_{\text{DB}}$. By applying the response time law, we conclude that:

$$\mathbb{E}[R] = \frac{N}{X} - \mathbb{E}[Z] = \frac{400}{55} - 1510 = \frac{127}{22} \approx 5.77$$

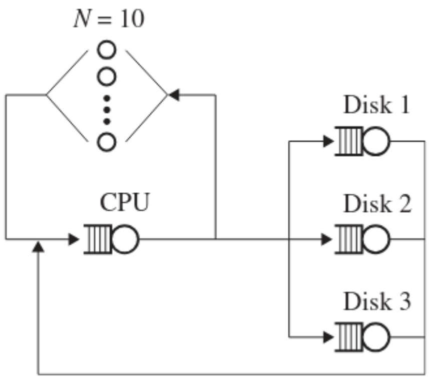
If we make the first layer twice as fast, i.e. $\mathbb{E}[Z'] = \frac{1}{2}\mathbb{E}[Z] = 0.75$ msec, we get that:

$$\mathbb{E}[R'] = \frac{N}{X} - \mathbb{E}[Z'] = \frac{400}{55} - 75100 = \frac{287}{44} \approx 6.52$$

Therefore, this change would worsen the response time. Intuitively, this comes from the fact that a faster application layer implies that jobs are being server more quickly to the database, increasing the average queue size and thus the response time.

Problem 1.2

Given the following system. We know that the throughput of disk 3 is 40 requests/sec, that the service time of an average request at disk 3 is 0.0225 sec and that average number of jobs in disk 3 is 4. Find the utilization and the waiting time of disk 3.



Solution:

Through the utilization law we know that:

$$\rho_3 = \mathbb{E}[S_3]X_3 = \frac{225}{10000}40 = \frac{9}{10}$$

The waiting time can be computed in two ways. The first method is to subtract the service time of disk 3 from its response time, which can be computed through Little's law:

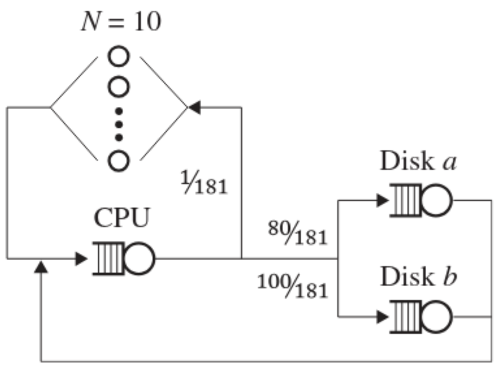
$$\mathbb{E}[W_3] = \mathbb{E}[R_3] - \mathbb{E}[S_3] = \frac{\mathbb{E}[N_3]}{X_3} - \mathbb{E}[S_3] = \frac{4}{40} - \frac{225}{10000} = \frac{1}{400}$$

The second method, instead, is to apply Little's law directly to the queue (as if it were a sub-system):

$$\mathbb{E}[W_3] = \mathbb{E}[R_3^{\text{queue}}] = \frac{\mathbb{E}[N_3^{\text{queue}}]}{X_3^{\text{queue}}} = \frac{\mathbb{E}[N_3] - \mathbb{E}[N_3^{\text{server}}]}{X_3^{\text{queue}}} = \frac{\mathbb{E}[N_3] - \mathbb{E}[4 - 0.9]}{40} = \frac{1}{400}$$

Problem 1.3

Given the following system, compute $\mathbb{E}[V_i]$ for each device.



Solution:

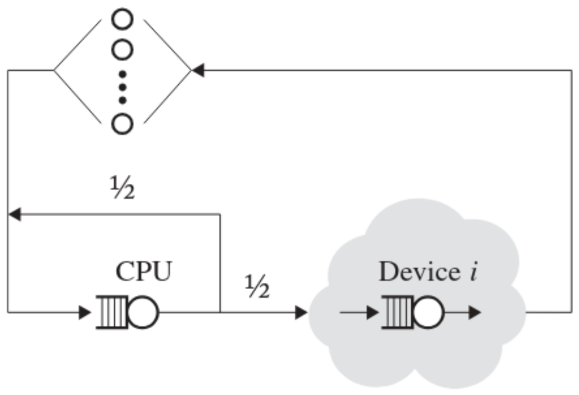
First, we notice that $\mathbb{E}[V_a] = \frac{80}{181}, \mathbb{E}[V_{\text{CPU}}]$ and $\mathbb{E}[V_b] = \frac{100}{181}\mathbb{E}[V_{\text{CPU}}]$. Thus computing $\mathbb{E}[V_{\text{CPU}}]$ gives us all the required values. Now, we observe that each job exiting the CPU can loop back into it by passing through one of the disk. This concludes that:

$$\begin{aligned}
\mathbb{E}[V_{\text{CPU}}] &= \sum_{k=0}^{+\infty} k \left(\frac{80}{181} + \frac{100}{181}\right)^{k-1} \frac{1}{181} \\
&= \sum_{k=0}^{+\infty} k \left(1 - \frac{180}{181}\right)^{k-1} \frac{1}{181} \\
&= 181
\end{aligned}$$

This concludes that $\mathbb{E}[V_{\text{CPU}}] = 181, \mathbb{E}[V_a] = 80$ and $\mathbb{E}[V_b] = 100$.

Problem 1.4

Consider the following interactive system with average think time of 5 seconds and average traversal time of 50 seconds. The CPU is known to be working at 50% capacity, while the device i is working at 30% capacity. Moreover, we know that the device's server processes each job with an average of 0.01 seconds and that it is visited 10 times per visit of the CPU. What is the value of $\mathbb{E}[N_{\text{CPU}}^{\text{queue}}]$ if the device is cloud sub-system processes an average of 20 jobs?



Solution:

First, we formalize the known information by setting $\mathbb{E}[Z] = 5$ sec, $\mathbb{E}[T] = 50$, $\rho_{\text{CPU}} = 0.5$, $\rho_i = 0.3$, $\mathbb{E}[S_i] = 0.01$ sec, $\mathbb{E}[V_i] = 10\,\mathbb{E}[V_{\text{CPU}}]$ and $\mathbb{E}[N_{\text{cloud}}] = 20$. Then, we observe that:

$$\mathbb{E}[N_{\text{CPU}}^{\text{queue}}] = \mathbb{E}[N_{\text{CPU}}] - \mathbb{E}[N_{\text{CPU}}^{\text{server}}] = \mathbb{E}[N_{\text{CPU}}] - \rho_{\text{CPU}} = \mathbb{E}[N_{\text{CPU}}] - \frac{1}{2}$$

Let *CPU-sys* be the sub-system formed by the CPU device and the backwards link. Since $\mathbb{E}[N_{\text{CPU}}] = \mathbb{E}[N_{\text{CPU-sys}}]$ and $X = X_{\text{CPU-sys}}$, by Little's law we have that:

$$\mathbb{E}[N_{\text{CPU}}] = \mathbb{E}[N_{\text{CPU-sys}}] = \mathbb{E}[T_{\text{CPU-sys}}]X_{\text{CPU-sys}} = \mathbb{E}[T_{\text{CPU-sys}}]X$$

Thus we need to compute both $\mathbb{E}[T_{\text{CPU-sys}}]$ and X. By the bottleneck law and the demand law, we know that:

$$X = \frac{\rho_i}{D_i} = \frac{\rho_i}{\mathbb{E}[V_i]\,\mathbb{E}[S_i]} = \frac{0.3}{10\,\mathbb{E}[V_{\text{CPU}}] \cdot 0.01}$$

Since $V_{\text{CPU}} = \text{Geo}(0.5)$, we know that $\mathbb{E}[V_{\text{CPU}}] = 2$, concluding that $X = 1.5$ jobs/sec. To compute $\mathbb{E}[T_{\text{CPU-sys}}]$, we observe that:

$$\mathbb{E}[T_{\text{CPU-sys}}] = \mathbb{E}[T] - \mathbb{E}[Z] - \mathbb{E}[T_{\text{cloud}}] = 50 - 5 - \frac{\mathbb{E}[N_{\text{cloud}}]}{X_{\text{cloud}}} = 45 - \frac{20}{1.5} = \frac{95}{3}$$

Putting everything together, we conclude that:

$$\mathbb{E}[N_{\text{CPU}}^{\text{queue}}] = \mathbb{E}[T_{\text{CPU-sys}}]X - \frac{1}{2} = \frac{95}{3}\frac{15}{10} - \frac{1}{2} = 47$$

Problem 1.5

Consider an interactive system with a CPU and a disk with high utilization. It is known that $N = 50$ and $\mathbb{E}[Z] = 10$ sec. At time instant τ, the following data is registered:

- $C(\tau) = 100$
- $C_{\text{CPU}_1}(\tau) = 300$ and $B_{\text{CPU}_1}(\tau) = 600$ sec
- $C_{\text{disk}_1}(\tau) = 400$ and $B_{\text{CPU}_1}(\tau) = 1200$ sec

To increase the throughput, we're proposed the following approaches:

1. Add a new CPU two times as fast to the system along with the old one and split their load optimally
2. Add a new disk three times as fast to the system along with the old one and split their load optimally

Assuming that the new devices have the same cost, which solution is the best? What is the maximum throughput of the best approach?

Solution:

We start by computing the metrics of the two intial devices:

$$\mathbb{E}[S_{\text{CPU}_1}] = \frac{B_{\text{CPU}_1}(\tau)}{C_{\text{CPU}_1}(\tau)} = \frac{600}{300} = 2 \text{ sec} \qquad \mathbb{E}[S_{\text{disk}_1}] = \frac{B_{\text{disk}_1}(\tau)}{C_{\text{disk}_1}(\tau)} = \frac{1200}{400} = 3 \text{ sec}$$

$$\mathbb{E}[V_{\text{CPU}_1}] = \frac{C_{\text{CPU}_1}(\tau)}{C(\tau)} = \frac{300}{100} = 3 \qquad \mathbb{E}[V_{\text{disk}_1}] = \frac{C_{\text{disk}_1}(\tau)}{C(\tau)} = \frac{400}{100} = 4$$

$$\mathbb{E}[D_{\text{CPU}_1}] = \mathbb{E}[V_{\text{CPU}_1}]\,\mathbb{E}[S_{\text{CPU}_1}] = 6 \qquad \mathbb{E}[D_{\text{disk}_1}] = \mathbb{E}[V_{\text{disk}_1}]\,\mathbb{E}[S_{\text{disk}_1}] = 12$$

Then, we compute the knee-point of the asymptotic bounds:

$$N^* = \frac{D_{\text{tot}} + \mathbb{E}[Z]}{D_{\max}} = \frac{18+10}{12} = \frac{7}{3}$$

Since $N^* < N$ and $D_{\max} = \mathbb{E}[D_{\text{disk}_1}]$, we conclude that the first approach wouldn't change since it doesn't improve the demand of the bottleneck. Hence, the second approach is the best one. Since the new disk is three times as fast as the old disk, we know that $\mathbb{E}[S'_{\text{disk}_2}] = \frac{1}{3}\,\mathbb{E}[S_{\text{disk}_1}] = 1$ sec.

To balance the load of the two disks, we solve the following system of equations:

$$\begin{cases} \mathbb{E}[V'_{\text{disk}_1}] + \mathbb{E}[V'_{\text{disk}_2}] = \mathbb{E}[V_{\text{disk}_1}] \\ \mathbb{E}[V'_{\text{disk}_1}]\,\mathbb{E}[S'_{\text{disk}_1}] = \mathbb{E}[V'_{\text{disk}_2}]\,\mathbb{E}[S'_{\text{disk}_2}] \end{cases} \implies \begin{cases} \mathbb{E}[V'_{\text{disk}_1}] = 1 \\ \mathbb{E}[V'_{\text{disk}_2}] = 3 \end{cases}$$

Next, we compute the new demands of the disks:

$$\mathbb{E}[D'_{\text{disk}_1}] = \mathbb{E}[V'_{\text{disk}_1}]\,\mathbb{E}[S'_{\text{disk}_1}] = 3 \qquad \mathbb{E}[D'_{\text{disk}_2}] = \mathbb{E}[V'_{\text{disk}_2}]\,\mathbb{E}[S'_{\text{disk}_2}] = 3$$

concluding that the CPU is the new bottleneck and that:

$$X' \leq \min\left(\frac{N}{D'_{\text{tot}} + \mathbb{E}[Z]}, \frac{1}{D'_{\max}}\right) = \left(\frac{N}{22}, \frac{1}{6}\right)$$

Problem 1.6

In the world of power distribution, one reasonable approximation is that the power that is allocated to a machine is proportional to the speed at which that machine can run. In this problem we assume that, if a machine is allocated power w, then that machine processes jobs at speed w jobs/sec.

Consider a closed batch system with two servers and N users, where N is assumed to be high. Assume that each job, with probability p, is routed to server 1 for processing and, with probability $1 - p$, is routed to server 2. You are given a total power budget W , which you need to distribute between the two machines. You can choose any way of dividing the power budget W between the two machines, and you can also choose any value you want for p, the routing probability.

1. What choice for dividing W and for picking p will maximize the throughput of your system?
2. Suppose that N was small. Would your answer still be the same? If so, explain why. If not, derive the optimal strategy.

Solution:

The problem asks us to find the optimal values of μ_1 and p under the constraints $W = \mu_1 + \mu_2$, $V_i = p$ and $V_2 = 1 - p$. We start by rewriting every metric in terms of μ_1:

$$\mathbb{E}[S_1] = \frac{1}{\mu_1} \qquad \mathbb{E}[S_2] = \frac{1}{W - \mu_1}$$

$$\mathbb{E}[D_1] = \mathbb{E}[V_1]\,\mathbb{E}[S_1] = \frac{p}{\mu_1} \qquad \mathbb{E}[D_2] = \mathbb{E}[V_2]\,\mathbb{E}[S_2] = \frac{1-p}{W - \mu_1}$$

When N is high, we know that X is bounded by $X \leq 1/D_{\max}$. Thus, the choice of μ_1 and p must minimize the value of $D_{\max}$. We easily observe that this is achieved when $\mathbb{E}[D_1] = \mathbb{E}[D_2]$.

$$\frac{p}{\mu_1} = \frac{1-p}{W - \mu_1} \implies Wp - \mu_1 p = \mu_1 - \mu_1 p \implies \mu_1 = Wp$$

Therefore, we conclude that μ_1 and p must be such that $\mu_1 = Wp$ (giving us infinite solutions). Moreover, we observe that when this is true we also have that:

$$\mathbb{E}[D_1] = \frac{p}{\mu_1} = \frac{p}{Wp} = \frac{1}{W}$$

$$\mathbb{E}[D_2] = \frac{1-p}{W - \mu_1} = \frac{1-p}{W - Wp} = \frac{1}{W}$$

When N is low, instead, we know that X is bounded by $X \leq N/D_{\text{tot}}$ (notice that $\mathbb{E}[Z] = 0$ since we're in a batch system). Thus, the choice of W and p must minimize the value of D_{tot}. We observe that:

$$D_{\text{tot}} = \mathbb{E}[D_1] + \mathbb{E}[D_2] = \frac{p}{\mu_1} + \frac{1-p}{W - \mu_1}$$

Through some algebraic manipulation, we can rewrite D_{tot} as a linear function of p.

$$D_{\text{tot}} = \left(\frac{1}{\mu_1} - \frac{1}{W - \mu_1}\right) p + \frac{1}{W - \mu_1}$$

Since D_{tot} is a linear function of p, we know that it reaches its minimum value when p is as low as possible or as high as possible, depending on the positivity of the angular coefficient (recall that $0 \leq p \leq 1$ since it is a probability). When $\mu_1 \leq W/2$, the angular coefficient is non-negative, thus the minimum value is reached when $p = 0$, concluding that:

$$D_{\text{tot}_{\text{min}}} = \left(\frac{1}{\mu_1} - \frac{1}{W - \mu_1}\right) \cdot 0 + \frac{1}{W - \mu_1} = \frac{1}{W - \mu_1}$$

which is minimized when $\mu_1 = 0$. When $\mu_1 > W/2$, instead, the angular coefficient is negative, thus the minimum value is reached when $p = 1$, concluding that:

$$D_{\text{tot}_{\text{min}}} = \left(\frac{1}{\mu_1} - \frac{1}{W - \mu_1}\right) \cdot 1 + \frac{1}{W - \mu_1} = \frac{1}{\mu_1}$$

which is minimized when $\mu_1 = W$. Therefore, we conclude that W and p must be such that either $p = 0$ and $\mu_1 = 0$ or when $p = 1$ and $\mu_1 = W$.

2 Markov chain analysis

2.1 Discrete-time Markov chains

Let's recap what we've seen in the previous chapter. In closed systems, we can approximate and bound the throughput X and the expected response time $E[R]$. These approximations do not depend on the distribution of job service times, but they do require the system to be closed. When the multiprogramming level N is either much larger than N^* or equal to 1, we obtain tight bounds on both X and $E[R]$. For intermediate values of N, however, we can only estimate these quantities.

For open systems, instead, our analytical capabilities are still limited. Even for a single queue, if we knew $E[N]$, we could compute $E[T]$, but we currently lack a general method for determining $E[N]$. Markov chain techniques offer a way to derive these performance metrics and much more. They allow us to determine not only the mean number of jobs, $E[N_i]$, at server i in a queueing network, but also the entire distribution of the number of jobs at that server.

To discuss about Markov chains, we first have to formalize the concept of **device state**, defined as the current population in the device. For instance, consider again the following simple open system.

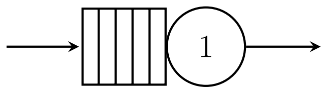


Figure 2.1: A simple open system with a single device.

Based on the size M_Q of the queue, this device could be in different states. For instance, if $M_Q = 3$, the device can be in 5 different states:

0. The device is idle and the queue is empty (0 jobs in the device)
1. The device is busy and the queue is empty (1 job in the device)

2. The device is busy and the queue contains one job (2 jobs in the device)
3. The device is busy and the queue contains two jobs (3 jobs in the device)
4. The device is busy and the queue contains three jobs (4 jobs in the device)

In general, for a finite queue we'll always have $M_Q + 2$ states. If the queue has infinite size, we'll have an infinite number of states. After defining the population states of the device, we can discuss the idea of *transitioning* from a state to another.

Suppose that we know that with probability 0.5 a new job will enter the device after a fixed time interval. Suppose that we also know that with probability 0.7 a job will exit the device. Naively, we could think that this is enough to define a model with proper transitions froma state to another. However, these probabilities do not suffice. For instance, suppose that a job enters the device when another one is exiting it. Intuitively, the population of the device will be the same as before. Therefore, we have to be more precise: we must talk about probabilities of *increasing* and *decreasing* the population of the device.

Consider now the following example. We have an open system composed by a single device with queue size $M_Q = 3$. We know that the probability of having a population increase is 0.3, while the probability of having a population decrease is 0.4. Is this information enough to model a correct system? The answer is yes!

For instance, suppose that we're in state 2 (one job in the device, one in the queue). We know that with probability 0.3 we'll transition to state 3 (one job in the device, two in the queue) and with probability 0.4 we'll transition to state 1 (one job in the device, no jobs in the queue). However, this information allows us to infer that with probability $1 - 0.3 - 0.4 = 0.3$ we'll stay in the same state ($1 - 0.3 = 0.7$ and $1 - 0.4 = 0.6$ for states 0 and 5).

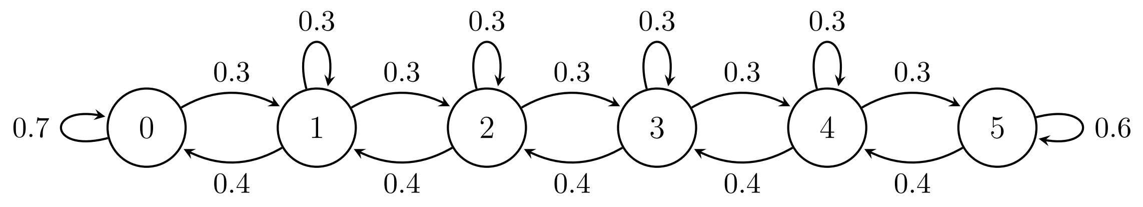


Figure 2.2: A Markov chain describing the previous example.

This is due to the fact that the sum of outgoing probabilities of each state must be equal to 1. This property follows in a natural way from the fact that from each state we <u>must</u> move to a another state (which could be itself) since the system must "act" on each time interval.

After giving the intuition behind how Markov chains work, we're ready to formally define them through **stochastic processes**, i.e. sequences $X_1, \dots, X_n$ of random variables.

Definition 2.1: Discrete-time Markov Chain

We define a discrete-time Markov chain (DTMC) as an infinite stochastic process $X_1, X_2, \dots$ such that:

- $\mathcal{I}$ is the state space
- $\forall n \in \mathbb{N}$, X_n is the random variable ranging over $\mathcal{I}$ and describing the state of the system at (discrete) time step n.
- The Markovian property holds for each state, meaning that $\forall n \in \mathbb{N}$, $\forall j, i \in \mathcal{I}$ and $\forall i_{n-1}, \dots, i_0 \in \mathcal{I}$ it holds that:

$$\Pr[X_{n+1} = j \mid X_n = i, X_{n-1} = t_{n-1}, \dots, X_0 = t_0] = \Pr[X_{n+1} = j \mid X_n = i]$$

We observe that the Markovian property states that the probability that the conditional distribution of any future state X_{n+1}, given past states $X_0, X_1, \dots, X_{n-1}$ and the present state X_n, is independent of past states and depends only on the present state X_n. This key property models the fact that from each state i of the graph we can move to its adjacent states independently of how we got to state i itself.

We also observe that the Markovian property implies that the probability of transitioning from state i to state j is completely independent of the time step n. In other words, $\forall n \in \mathbb{N}$ it holds that:

$$\Pr[X_{n+1} = j \mid X_n = i] = \Pr[X_n = j \mid X_{n-1} = i] = \dots = \Pr[X_1 = j \mid X_0 = i]$$

This allows us to work with the probabilities of each transition in a more concise way by defining a **probability transition matrix** associated with each DMTC.

Definition 2.2: Probability transition matrix

Given a DMTC M with state space $\mathcal{I}$, we define the probability transition matrix (PTM) of M as the $|\mathcal{I}| \times |\mathcal{I}|$ matrix P whose (i, j)-th entry $P_{i,j}$ is such that:

$$P_{i,j} = \Pr[X_{n+1} = j \mid X_n = i]$$

$$\begin{array}{c|cccccc|}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
0 & 0.7 & 0.3 & 0 & 0 & 0 & 0 \\
1 & 0.4 & 0.3 & 0.3 & 0 & 0 & 0 \\
2 & 0 & 0.4 & 0.3 & 0.3 & 0 & 0 \\
3 & 0 & 0 & 0.4 & 0.3 & 0.3 & 0 \\
4 & 0 & 0 & 0 & 0.4 & 0.3 & 0.3 \\
5 & 0 & 0 & 0 & 0 & 0.4 & 0.6
\end{array}$$

Figure 2.3: The probability transition matrix of the previous example.

By our previous observation, we know that the probabilities of the outgoing edges of each state must sum up 1. This is equivalent to saying that the rows of each probability transition matrix must sum up to 1, i.e. $\forall i \in \mathcal{I}$ it must hold that:

$$\sum_{j \in \mathcal{I}} P_{i,j} = 1$$

To get familiar with how to model problems through Markov chains, we'll give an examples. Consider a facility with some machines, where each machine is either working or in the repair center. If the machine is working today, then there is a 95% chance that it will be working tomorrow. If it is in the repair center today, then there is a 40% chance that it will be working tomorrow. We are interested in questions like "what fraction of time does my machine spend in the repair shop?"

To model this problem, two states suffice: one for when the machine is working and one for when it is broken, respectively labeled as W and B. From the given information, we know that $P_{W,W} = 0.95$ and $P_{B,W} = 0.4$. Since the outgoing probabilities must sum up to 1 for every state, we can infer that $P_{W,B} = 1 - P_{W,W} = 1 - 0.95 = 0.05$ and $P_{B,B} = 1 - P_{B,W} = 1 - 0.6 = 0.4$.

$$\begin{tikzcd} W \arrow[loop left, "0.95"] \arrow[r, bend left, "0.05"] & B \arrow[l, bend left, "0.4"] \arrow[loop right, "0.6"] \end{tikzcd}$$

$$\begin{array}{c|c} & \begin{matrix} W & B \end{matrix} \\ \begin{matrix} W \\ B \end{matrix} & \begin{bmatrix} 0.95 & 0.05 \\ 0.4 & 0.6 \end{bmatrix} \end{array}$$

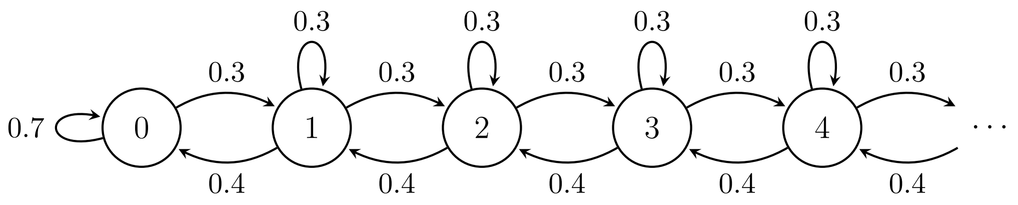
Figure 2.4: Model for the repair facility problem.

Suppose now that after the machine remains broken for 4 days, the machine is replaced with a new machine. Can we still use the same model? The answer is <u>no</u>: the Markovian property explicitely affirms that the next state depends <u>only</u> on the current state. In other words, Markov chains are **memoryless** by their very definition. To solve this issue, we must "force" memory in the model by duplicating states. In our example, we must replace state B with four states B_1, B_2, B_3, B_4, where, intuitively, state B_i corresponds to the machine being broken for i days in a row.

$$\begin{tikzcd} W \arrow[loop left, "0.95"] \arrow[r, bend left, "0.05"] & B_1 \arrow[l, bend left, "0.4"] \arrow[r, bend left, "0.6"] & B_2 \arrow[ll, bend left, "0.4"] \arrow[r, bend left, "0.6"] & B_3 \arrow[lll, bend left, "0.4"] \arrow[r, bend left, "0.6"] & B_4 \arrow[llll, bend left, "0.4"] \end{tikzcd}$$

Figure 2.5: Markov chain for the repair facility problem with the new rule.

In the previous chapter, we have discussed how in some cases it might be useful to consider the queue size as (countably) infinite. We observe that this is no issue: the definition of Markov chain doesn't restrict us to a finite number of states. In fact, we can even consider a countably infinite number of states.



$$
\begin{array}{c|cccccc}
 & 0 & 1 & 2 & 3 & 4 & \cdots \\
\hline
0 & 0.7 & 0.3 & 0 & 0 & 0 & \cdots \\
1 & 0.4 & 0.3 & 0.3 & 0 & 0 & \cdots \\
2 & 0 & 0.4 & 0.3 & 0.3 & 0 & \cdots \\
3 & 0 & 0 & 0.4 & 0.3 & 0.3 & \cdots \\
4 & 0 & 0 & 0 & 0.4 & 0.3 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}
$$

Figure 2.6: A Markov chain with an infinite number of states and its PTM.

2.2 Powers of the transition probabilities matrix

An absent-minded professor has two umbrellas that she uses when commuting from home to office and back. If it rains and an umbrella is available in her location, she takes it. If it is not raining, she always forgets to take an umbrella. Suppose that it rains with probability p each time she commutes, independently of prior commutes. Our eventual goal is to determine the fraction of commutes during which the professor gets wet.

First of all, we start by modeling the problem through a Markov chain. We observe that three states $0, 1, 2$ suffice: each state i tracks the number of umbrellas available to the professor at the current location, regardless of what this current location is. To model the probabilities of each state, we have to carefully consider each situation:

- If our current location has no umbellas, both umbrellas must be in the other location, which will also be our next location.
- If our current location has two umbellas and it rains, we'll take an umbrella with us. Hence, one umbrella will be in the next location (the one that we're going to).
- If our current location has two umbellas and it doesn't rain, we won't take an umbrella with us. Hence, no umbrellas will be in the next location.
- If our current location has one umbrella and it rains, we'll take one umbrella with us. Hence, two umbrellas will be in the next location.

- If our current location has one umbrella and it doesn't rain, we won't take an umbrella with us. Hence, one umbrella will be in the next location.

Hence, we can model the problem as follows.

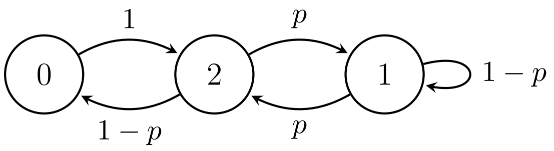


Figure 2.7: Markov chain for the umbrella problem.

Consider now the probability transition matrix P associated with the DTMC.

$$P = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 1 \\ 1 & 0 & 1-p & p \\ 2 & 1-p & p & 0 \end{array}$$

Figure 2.8: PTM for the umbrella problem.

The problem's request is to compute the probability that the professor gets wet. From our model, we know that:

$$\Pr[\text{prof. gets wet at step } n] = \Pr[X_n = 0] \Pr[\text{it rains}] = \pi_0^{(n)} p$$

where $\pi_0^{(n)} = \Pr[X_n = 0]$. To answer the request, we simply have to take the limit as $n \to +\infty$:

$$\lim_{n \to +\infty} \Pr[\text{prof. gets wet at step } n] = \lim_{n \to +\infty} \pi_0^{(n)} p = \pi_0 p$$

where π_0 is the value of $\pi_0^{(n)}$ as $n \to +\infty$. But how can be compute π_0, yet alone π_0^n? Through total probability, for each state $j \in \{0, 1, 2\}$ we get that:

$$\begin{aligned} \pi_j^{(n)} &= \sum_{i=0}^{2} \Pr[X_n = j \mid X_{n-1} = i] \\ &= \sum_{i=0}^{2} P_{j,i} \pi_i^{(n-1)} \end{aligned}$$

Our equation is therefore structured in a recursive way. Consider now the row vector $\pi^{(n)} = \begin{bmatrix} \pi_0^{(n)} & \pi_1^{(n)} & \pi_2^{(n)} \end{bmatrix}$. From our previous observation, we get that:

$$\pi^{(n)} = \begin{bmatrix} \pi_0^{(n-1)} & \pi_1^{(n-1)} & \pi_2^{(n-1)} \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1-p & p \\ 1-p & p & 0 \end{bmatrix} = \pi^{(n-1)} P$$

By unrolling the recursive equation, we get that $\pi^{(n)} = \pi^{(0)} P^n$. To give an intuition behind what P^n represents, consider the matrix P^2:

$$P^2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1-p & p \\ 1-p & p & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1-p & p \\ 1-p & p & 0 \end{bmatrix} = \begin{bmatrix} 1-p & p & 0 \\ p-p^2 & 2p^2-2p+1 & p-p^2 \\ 0 & p-p^2 & p^2-p+1 \end{bmatrix}$$

Here, each entry $P^2_{i,j}$ – short form of $(P^2)_{i,j}$ – is given by:

$$P^2_{i,j} = P_{i,0}P_{0,j} + P_{i,1}P_{1,j} + P_{i,2}P_{2,j}$$

In other words, $P^2_{i,j}$ is obtained by summing the probabilities of each *path* with length 2 over the Markov chain. In a similar fashion, for each n we have that:

$$P^n_{i,j} = P_{i,0}P^{n-1}_{0,j} + P_{i,1}P^{n-1}_{1,j} + P_{i,2}P^{n-1}_{2,j}$$

Therefore, we can conclude that P^n represents the n**-step probability transition matrix**, that being the matrix whose entries $P^n_{j,i}$ represent the probability of going to state j from state i in after n (discrete) time steps, that being $P^n_{j,i} = \Pr[X_n = j \mid X_0 = i]$.

Definition 2.3: n-step probability transition matrix

Given a DMTC M with state space $\mathcal{I}$, we define the n-step probability transition matrix of M as the n-th power P^n of its PTM P. In particular, for each $n > 1$ and each $i, j \in \mathcal{I}$ it holds that:

$$P^n_{j,i} = \sum_{k \in \mathcal{I}} P^{n-1}_{j,k} P_{k,j}$$

To make things easier, we set $p = 0.4$, obtaining the following values for P and P^2:

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0.6 & 0.4 \\ 0.6 & 0.4 & 0 \end{bmatrix} \qquad P^2 = \begin{bmatrix} 0.6 & 0.4 & 0 \\ 0.24 & 0.52 & 0.24 \\ 0 & 0.24 & 0.76 \end{bmatrix}$$

Let's compute some more powers (truncating values up to the third decimal):

$$P^5 \approx \begin{bmatrix} 0.057 & 0.307 & 0.635 \\ 0.184 & 0.377 & 0.438 \\ 0.381 & 0.438 & 0.180 \end{bmatrix} \qquad P^{10} \approx \begin{bmatrix} 0.302 & 0.412 & 0.285 \\ 0.247 & 0.391 & 0.361 \\ 0.171 & 0.361 & 0.466 \end{bmatrix}$$

$$P^{20} \approx \begin{bmatrix} 0.242 & 0.389 & 0.368 \\ 0.233 & 0.385 & 0.380 \\ 0.221 & 0.380 & 0.397 \end{bmatrix} \qquad P^{30} \approx \begin{bmatrix} 0.232 & 0.385 & 0.382 \\ 0.231 & 0.384 & 0.384 \\ 0.229 & 0.384 & 0.386 \end{bmatrix}$$

Therefore, we can conclude that as $n \to +\infty$ it holds that:

$$P^\infty = \lim_{n\to+\infty} P^n = \begin{bmatrix} 0.230 & 0.385 & 0.385 \\ 0.230 & 0.385 & 0.385 \\ 0.230 & 0.385 & 0.385 \end{bmatrix}$$

Now, let's go back to computing the vector $\pi = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 \end{bmatrix}$ for the umbrella problem. First, we may assume that $\pi^{(0)}$ is uniform, i.e. $\pi^{(0)} = \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}$, since each state has the same probability of being the initial one. By taking the limit of the recursive equation, we get that:

$$\begin{aligned} \pi &= \lim_{n\to+\infty} \pi^{(0)} P^n \\ &= \pi^{(0)} P^\infty \\ &= \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix} \begin{bmatrix} 0.230 & 0.385 & 0.385 \\ 0.230 & 0.385 & 0.385 \\ 0.230 & 0.385 & 0.385 \end{bmatrix} \\ &= \begin{bmatrix} 0.23 & 0.385 & 0.385 \end{bmatrix} \end{aligned}$$

Therefore, we conclude that each i-th entry of each j-th column of the matrix P^∞ converges to j-th entry of π, regardless of the initial state. We refer to the vector π as the **limiting distribution** of the DTMC.

Definition 2.4: Limiting distribution

Given a DMTC M with state space $\mathcal{I}$, we define the limiting distribution of M as the row vector π with $|\mathcal{I}|$ entries such that:

- $\pi_j = \lim_{n\to+\infty} P^n_{j,i}$ for each pair of states $i, j \in \mathcal{I}$, where P is the PTM of M.
- $\sum_{j\in\mathcal{I}} \pi_j = 1$

Each entry π_j is referred to as the limiting probability of being in state j.

After computing the limiting distribution π, we can finally answer the initial question of the problem:

$$\lim_{n\to+\infty} \Pr[\text{prof. gets wet at step } n] = \pi_0 p = 0.23 \cdot 0.4 = 0.092$$

concluding that the professor gets wet on any given day is almost 10%.

As defined, π is a limit. Yet it is not at all obvious that π represents a distribution (meaning that $\sum_{j\in\mathcal{I}} \pi_j = 1$), although this turns out to be easy to see understand thanks to the previous example.

More importantly, it is also not obvious that the limit π even exists! For instance, consider the following Markov chain.

$$\begin{tikzcd} 0 \arrow[r, bend left, "1"] & 1 \arrow[l, bend left, "1"] \end{tikzcd} \qquad \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

Figure 2.9: A Markov chain without limiting distribution.

By computing P^2, P^3 and P^4, we observe that:

$$P^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad P^3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = P \qquad P^4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = P^2$$

This implies that for each $k \in \mathbb{N}$ it holds that $P^{2k} = P^2$ and $P^{2k+1} = P^1$ and therefore that the matrix $P^\infty = \lim_{n \to +\infty} P^n$ doesn't converge! In next sections, we'll see ways to determine if such limit exists.

2.3 Stationary distribution

In the previous section we found out how the limiting distribution π can be computed (when it exists) by solving the equation $\pi = \pi_0 P^\infty$, which is equivalent to setting $\pi_j = P^\infty_{j,i}$ for each $j \in \mathcal{I}$.

However, computing P^∞ is clearly a tedious task, making this process impracticable. To fix this issue, we require a faster way to compute π. Let's go back to the original recursive equation of π. We saw how for each n it holds that:

$$\pi^{(n)} = \pi^{(n-1)} P$$

By taking the limit as $n \to +\infty$, we get that:

$$\pi = \lim_{n \to +\infty} \pi^{(n)} = \lim_{n \to +\infty} \pi^{(n-1)} P = \pi P$$

In other words, the PTM doesn't change the values of the limit distribution. Distributions with such property as said to be **stationary distributions** for the DTMC.

Definition 2.5: Stationary distribution

Given a DMTC M with state space $\mathcal{I}$, a vector v of $|\mathcal{I}|$ entries is said to be a stationary distribution of M if:

- $v = vP$, where P is the PTM of M
- $\sum_{j \in \mathcal{I}} v_j = 1$

We have already concluded that the limiting distribution (when it exists) is a stationary distribution. However, a stronger result can be proven: every stationary distribution must be equal to the limiting distribution!

Theorem 2.1

Given a DMTC M (with finite or countably infinite states), if M admits a limiting distribution π then it is the unique stationary distribution.

Proof. We prove the theorem only for DMTCs with a finite number of states. We already know that (when it exists) π is a stationary distribution, hence it suffices to prove that it is the unique stationary distribution. Let v be a stationary distribution of M and suppose that the limiting distribution π exists.. Since v is stationary, for each $k \in \mathcal{I}$ we have that:

$$v_k = \Pr[X_n = k] = \ldots = \Pr[X_0 = k]$$

Moreover, since v_k contains constant values, we have that $v = \lim\limits_{n \to +\infty} v$.

Now, fix any $j \in \mathcal{I}$. We observe that:

$$\begin{aligned} v_j &= \Pr[X_n = j] \\ &= \sum_{i \in \mathcal{I}} \Pr[X_n = j \mid X_0 = i] \Pr[X_0 = i] \\ &= \sum_{i \in \mathcal{I}} P^n_{j,i} v_i \end{aligned}$$

Therefore, by taking the limit as $n \to +\infty$ we have that:

$$v_j = \lim_{n \to +\infty} v_j = \lim_{n \to +\infty} \sum_{i \in \mathcal{I}} P^n_{j,i} v_i = \sum_{i \in \mathcal{I}} \lim_{n \to +\infty} P^n_{j,i} v_i = \sum_{i \in \mathcal{I}} \pi_j v_i = \pi_j \sum_{i \in \mathcal{I}} v_i = \pi_j$$

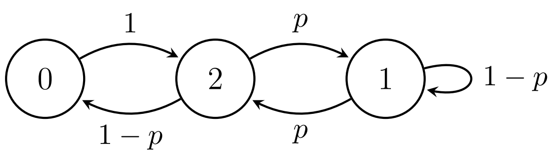
concluding that $v = \pi$.

Note: we observe that with a infinite number of states we cannot simply take a limit from the outside of an infinite sum to its inside since the sum might not converge. Thus, a proof for a countably infinite number of states requires to be more precise. □

The above theorem is a fundamental result in *Markov theory*, allowing us to compute limiting distributions (assuming it exists) simply by solving the following system, called **stationary equations**:

$$\begin{cases} \pi = \pi P \\ \sum_{j \in \mathcal{I}} \pi_j = 1 \end{cases}$$

For instance consider again the umbrella problem shown in the below picture, with a generic value p and its associated PTM P (Figure 2.8).



To compute the limiting distribution, all we have to do is to solve the following system of linear equations:

$$\begin{cases} \pi_0 &= (1-p)\pi_2 \\ \pi_1 &= (1-p)\pi_1 + p\pi_2 \\ \pi_2 &= \pi_0 + p\pi_1 \\ \pi_0 + \pi_1 + \pi_2 &= 1 \end{cases}$$

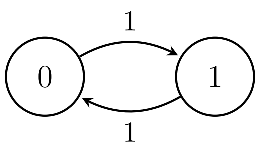
which yields the following solutions:

$$\pi_0 = \frac{1-p}{3-p} \qquad \pi_1 = \frac{1}{3-p} \qquad \pi_2 = \frac{1}{3-p}$$

By setting $p = 0.4$, we get the same solution to the umbrella problem, i.e. $\pi_0 p = \frac{24}{260} \approx 0.092$, further proving the truthfulness of the theorem.

Clearly, this method can only be used only if we're sure that the limiting distribution exists since otherwise it is not guaranteed that every stationary distribution is the limiting one.

For instance, consider again the counterexample shown in Figure 2.9.



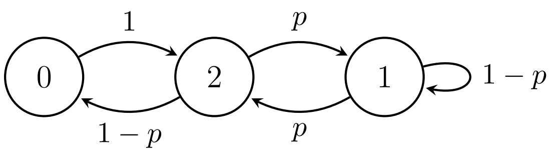
We've already proven that the limiting distribution for this DMTC doesn't exist. However, we can still find a stationary distribution by solving the associated system.

$$\begin{cases} \pi_0 &= \pi_1 \\ \pi_1 &= \pi_0 \\ \pi_0 + \pi_1 &= 1 \end{cases}$$

which yields the solutions $\pi_0 = \pi_1 = \frac{1}{2}$. Therefore, a stationary distribution may exist even when the limiting distribution doesn't exist.

2.4 The barrier technique

Consider again the umbrella problem and its stationary equations.



$$\begin{cases} \pi_0 &= (1-p)\pi_2 \\ \pi_1 &= (1-p)\pi_1 + p\pi_2 \\ \pi_2 &= \pi_0 + p\pi_1 \\ \pi_0 + \pi_1 + \pi_2 &= 1 \end{cases}$$

We observe that the second stationary equation reveals a secret fact:

$$\pi_1 = (1-p)\pi_1 + p\pi_2 \implies p\pi_1 = p\pi_2$$

This is no coincidence! If we consider the ingoing and outgoing probabilities of each state as “flows”, it’s easy to see that these two quantities must be equal. Therefore, the outgoing flow of state 2, that being $\pi_2 p$ must be equal to its ingoing flow, that being $\pi_1 p$, concluding that $p\pi_2 = p\pi_1$. This property is referred to as the **conservation of flows** between two states. Visually, it appears as if we placed a “barrier” between the two transitions $(2,1)$ and $(1,2)$, hence the name *barrier technique.*

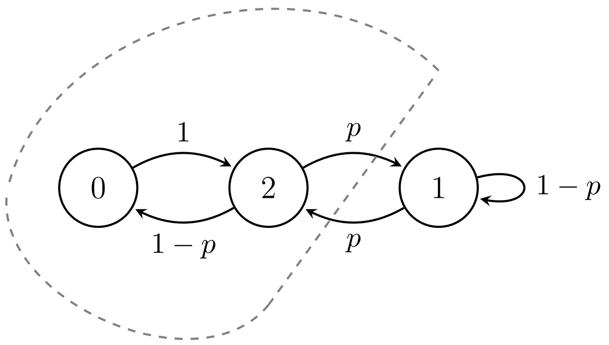
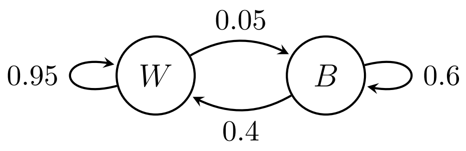


Figure 2.10: Visual representation of the barrier technique.

The barrier technique can be used to solve stability equations in a faster way by allowing us to consider simpler equations instead of the ones given by $\pi = \pi P$. For instance, consider again the Markov chain of the repair facility problem shown below.



To find the limiting probability in the usual way, we solve the following system.

$$\begin{cases} \pi_0 & = 0.95 \cdot \pi_0 + 0.4 \cdot \pi_1 \\ \pi_1 & = 0.05 \cdot \pi_0 + 0.6 \cdot \pi_1 \\ \pi_0 + \pi_1 & = 1 \end{cases}$$

By solving the system, we get that $\pi_0 = 0.875$ and $\pi_1 = 0.125$. Now, we'll find these same results by applying the barrier technique. Since the two transitions (W, B) and (B, W) exists, we know that the ingoing flow must be equal to the outgoing flow, allowing us to apply the barrier technique.

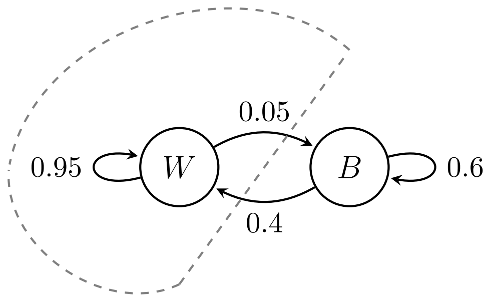
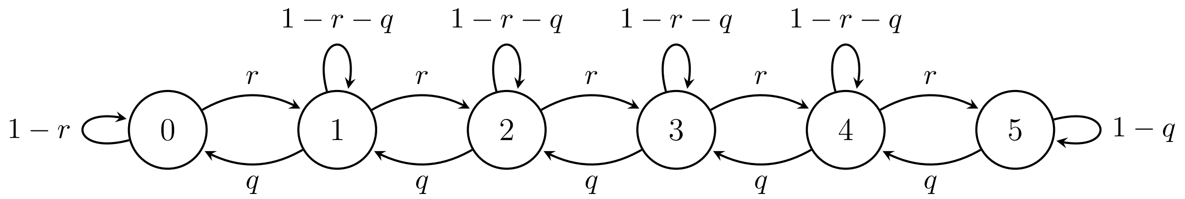


Figure 2.11: Application of the barrier technique on the repair facility problem.

Therefore, we get that $0.05 \cdot \pi_W = 0.4 \cdot \pi_B$ must hold. Moreover, we know that π must be a distribution, i.e. the sum of all of its entries must be equal to 1. This gives us a new smaller system of equations from which we can immediately conclude that $\pi_0 = 0.875$ and $\pi_1 = 0.125$:

$$\begin{cases} 0.05 \cdot \pi_W & = 0.4 \cdot \pi_B \\ \pi_0 + \pi_1 & = 1 \end{cases}$$

For bigger chains, the barrier technique allows us to reduce the number of computations by an extreme amount. For instance, consider the following Markov chain.



By applying the barrier technique on each pair of states $(i-1, i)$, with $i \in [5]$, we obtain:

$$r\pi_{i-1} = \pi_i q \implies \pi_i = \frac{r}{q}\pi_{i-1} \qquad \forall i \in [5]$$

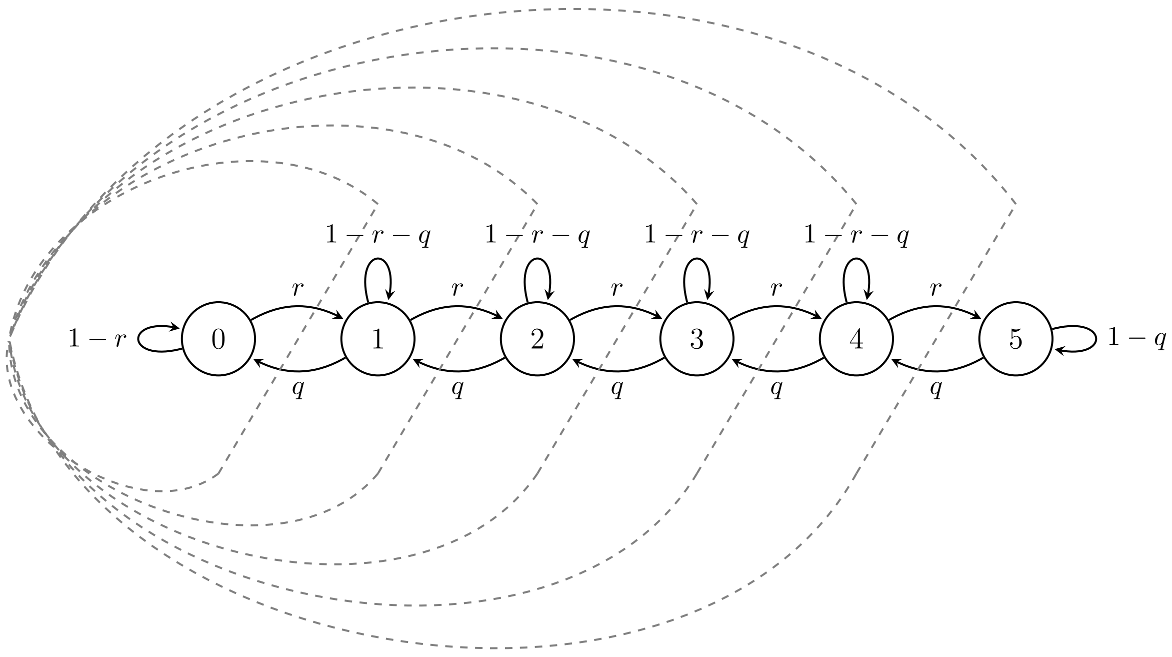


Figure 2.12: Graphical representation of multiple applications of the barrier technique. Each barrier is independent from the others, meaning that the inside and outside of a barrier should be considered without the other barriers.

By adding the condition $\sum_{i=0}^{5} \pi_i = 1$, we get the following system of equations:

$$\begin{cases} \pi_i &= \frac{r}{q}\pi_{i-1} \qquad \forall i \in [5] \\ \sum_{i=0}^{5} \pi_i &= 1 \end{cases}$$

Now, let $\rho = \frac{r}{q}$. By unrolling the recursive equation up to π_0, we get that:

$$\begin{cases} \pi_i &= \rho^i \pi_0 \qquad \forall i \in [5] \\ \sum_{i=0}^{5} \pi_i &= 1 \end{cases}$$

By substituting the unrolled equation into the sum, we get that:

$$1 = \sum_{i=0}^{5} \pi_i = \sum_{i=0}^{5} \rho^i \pi_0 = \pi_0 \sum_{i=0}^{5} \rho^i = \pi_0 \cdot \frac{1-\rho^6}{1-\rho}$$

where the last step is due to the finite geometric sum:

$$\sum_{i=0}^{n} x^i = \frac{1-x^{n+1}}{1-x}$$

Therefore, we get that:

$$\begin{cases} \pi_i & = \rho^i \pi_0 \qquad\qquad \forall i \in [5] \\ \rho_0 & = \dfrac{1-\rho}{1-\rho^6} \end{cases}$$

and thus that:

$$\pi_i = \frac{\rho^i(1-\rho)}{1-\rho^6} \qquad \forall i \in \{0, \dots, 5\}$$

2.5 Existence of limiting distributions

In the previous chapter we introduced performance analysis through Markov chains, observing that it may not always be true that a limiting distribution exists. In this section, we'll discuss how **ergodicity** ensures the existence of such limit. We already introduced the concept of ergodicity in the first chapter, stating that a system is ergodic when it is irreducible, positive recurrent and aperiodic. Now, we'll give the corresponding definitions for this three properties in the case of Markov chains.

We start with the simplest one: **aperiodicity**. We said that "the system is aperiodic if it never returns in a specific state after a precise amount of time". To understand how this applies to Markov chains, consider the following example.

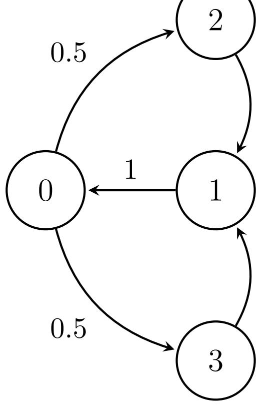

$$\begin{tikzcd} & 2 \arrow[d, bend left, "1"] \\ 0 \arrow[ur, bend left, "0.5"] \arrow[dr, bend right, "0.5"'] & 1 \arrow[l, "1"'] \\ & 3 \arrow[u, bend right, "1"'] \end{tikzcd}$$

Figure 2.13: A periodic Markov chain.

It's easy to see that this chain is periodic: if we start from state 0, it will return to this state every 3 time steps, independently of the transitions that it took. Now, observe what happens to the 3-step PTM:

$$P = \begin{bmatrix} 0 & 0 & 0.5 & 0.5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \implies P^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

We notice that the main diagonal of the matrix is filled with non-zero entries, meaning that every state has a non-zero probability of going back to itself in 3 steps. In particular, states 0 and 1 will surely go back to themselves every 3 steps. This allows us to define the idea of *state period*.

Definition 2.6: State period

Given a DMTC M with state space $\mathcal{I}$ and PTM P, we define the period of state $i \in \mathcal{I}$ as the GCD of the set $\{n \in \mathbb{N} \mid P^n_{i,i} > 0\}$. A state is said to be aperiodic if its period is 1. The chain M is said to be aperiodic if every state is aperiodic.

After defining periodicity, we make some observations on this definition. First of all, we have to clarify things: we said that a state is aperiodic if it has period 1, but shouldn't value set to zero in order to be aperiodic? Let's take a step back to the initial definition of aperiodicity: "the system is aperiodic if it never returns in a specific state after a precise amount of time". If the period of a state i is 1, we have two possibilities:

- The GCD of the set $\{n \in \mathbb{N} \mid P^n_{i,i} > 0\}$ is equal to 1 because each value of the set is bigger than 1 and pair-wise coprime. This coprimality guarantees that not every value of the set is a multiple of a specific number, breaking periodicity.
- The GCD of the set $\{n \in \mathbb{N} \mid P^n_{i,i} > 0\}$ is equal to 1 because 1 is inside the set. This can happen only if state i has a loop in the chain with non-zero probability. This guarantees that it has a probability of returning to iself every single time step, thus everything that may happen in time step i is may also happen in time step $i + 1$, breaking periodicity.

After clarifying the definition of aperiodicity, let's consider some practical examples. Consider again the Markov chain of the umbrella problem, shown below.

$$\begin{tikzcd} 0 \arrow[r, bend left, "1"] & 2 \arrow[l, bend left, "1-p"] \arrow[r, bend left, "p"] & 1 \arrow[l, bend left, "p"] \arrow[loop right, "1-p"] \end{tikzcd}$$

Let $p = 0.4$. Then, the state 1 has a loop with non-zero probability, concluding immediately that its period is 1. To compute the period of states 0 and 2, we take some powers of P:

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0.6 & 0.4 \\ 0.6 & 0.4 & 0 \end{bmatrix} \qquad P^2 \approx \begin{bmatrix} 0.6 & 0.4 & 0 \\ 0.24 & 0.52 & 0.24 \\ 0 & 0.24 & 0.76 \end{bmatrix} \qquad P^3 \approx \begin{bmatrix} 0 & 0.24 & 0.76 \\ 0.14 & 0.41 & 0.45 \\ 0.46 & 0.45 & 0.10 \end{bmatrix}$$

$$P_4 \approx \begin{bmatrix} 0.46 & 0.45 & 0.10 \\ 0.27 & 0.42 & 0.31 \\ 0.06 & 0.31 & 0.63 \end{bmatrix} \qquad P_5 = \begin{bmatrix} 0.06 & 0.31 & 0.63 \\ 0.19 & 0.38 & 0.44 \\ 0.38 & 0.44 & 0.18 \end{bmatrix} \qquad P_6 \approx \begin{bmatrix} 0.38 & 0.44 & 0.18 \\ 0.26 & 0.40 & 0.33 \\ 0.11 & 0.33 & 0.56 \end{bmatrix}$$

From these few powers, it already seems that for every $n \in \mathbb{N}$ and every $i \in \{0, 1, 2\}$ it holds that $P^n_{i,i} > 0$, concluding that the period of every state is 1 and thus concluding that the chain is aperiodic. This is no coincidence: the small loop of state 1 is sufficient to make the whole chain aperiodic. First, we give some definition.

Definition 2.7: Accessibility and communication

Consider a DMTC M with state space $\mathcal{I}$. Given a pair $i, j \in \mathcal{I}$, we say that state j is accessible from state i if $P^n_{i,j} > 0$ for some $n \in \mathbb{N}$. Similarly, we say that states i and j communicate if i is accessible from j and vice versa.

The above definitions and the previous observation make the following conclusion pretty obvious: if a state is aperiodic and another state communicates with it, the latter state can use the aperiodicity of the former state to break its own periodicity.

Proposition 2.1

Consider a DMTC M with state space $\mathcal{I}$. If state $i \in \mathcal{I}$ is aperiodic, every state $j \in \mathcal{I}$ that communicates with i is also aperiodic.

Proof. Omitted. □

After understanding aperiodicity, let's move to **irreducibility**. We said that "a system is irreducible if from every state the system can reach every other state and back". After defining the concept of communication, the definition of irreducible Markov chain is straight forward.

Definition 2.8: Irreducible Markov chain

A Markov chain is said to be irreducible if each pair of states communicate with each other.

This leaves us with defining **positive recurrence**. For now, we defer the discussion about positive recurrence. In fact, for finite-state DTMC, irreducibility and aperiodicity is all we need to guarantee that a limiting distribution exists!

Theorem 2.2: Existence of lim. distibution (finite case)

Consider a DMTC M with finite state space $\mathcal{I}$ and PTM P. If M is irreducible and aperiodic then $\lim\limits_{n \to +\infty} P^n$ exists and corresponds to a matrix L whose rows are all equal to the limiting distribution of M.

Proof. Fix $j \in [|\mathcal{I}|]$. Let e_j be the column vector whose entries are all set to 0 except for the j-th entry, which is set to 1. We observe that for each $n \in \mathbb{N}$ the product $P^n e_j$ corresponds to the j-th column of P^n.

Fix $n \in \mathbb{N}$. Let M_n and m_n be, respectively, the maximum and minimum value of any component of $P^n e_j$. Similarly, let s be the smallest element of the whole matrix P.

Note: we observe that these values are guaranteed to exist only on a finite state Markov chain, making this proof invalid for infinite state ones.

Claim: $M_n - m_n \leq (1-2s)(M_{n-1} - m_{n-1})$

Proof of the claim. Let $y = P^{n-1}e_j$. First, we observe that the largest possible value is obtained if all but one of the elements of y are M_{n-1}, with the remaining one being m_{n-1}, where m_{n-1} is weighted by the smallest value s. This concludes that:

$$M_n \leq (1-s)M_{n-1} + sm_{n-1}$$

With a symmetrical argument, we can conclude that:

$$m_n \geq (1-s)m_{n-1} + sM_{n-1}$$

Summing the two inequalities, we get that:

$$M_n - \leq (1-s)M_{n-1} + sm_{n-1} - (1-s)m_{n-1} - sM_{n-1} = (1-2s)(M_{n-1} - m_{n-1})$$

□

Due to irreducibility and aperiodicity of P, there must be a value $n_0 \in \mathbb{N}$ such that P^{n_0} has no null-entries. Let $P' = P^{n_0}$. We observe that the same argument used for the claim also holds for P', implying that $M'_n - m'_n \leq (1-2s')(M'_{n-1} - m'_{n-1})$.

By unrolling the recursive inequality, we get that:

$$M'_n - m'_n \leq (1-2s')^{n-1}(M'_1 - m'_1)$$

Now, we observe that $s' \leq 0.5$ must hold since each row of P must sum up to 1. Therefore, we conclude that:

$$0 \leq \lim_{n \to +\infty} M'_n - m'_n \leq \lim_{n \to +\infty} (1-2s')^{n-1}(M'_1 - m'_1) = 0$$

Thus, each column $(P')^n e_j$ will eventually reach a point where all entries are equal to each other, concluding that $\lim\limits_{n \to +\infty} (P')^n$ converges to a matrix L whose rows are all equal to each other, meaning that each row will be equal to the of limiting distribution of P. □

The above theorem implicitely states that positive recurrence is not needed for finite state chains. However, we're actually missing a piece of the puzzle: every irreducible finite state chain is already positive recurrent! In other words, we're actually already implicitely using positive recurrence in the theorem! To convince ourselves, recall that we said that "a system is positive recurrent if starting from a state, the probability to return in that same state over infinite time is 1". Then, for a finite state chain, irreducibility already implies that starting from a state we will eventually come back to it in a finite amount of time!

Then why do when do we also need positive recurrence? The answer is simple: for infinite chains! To formally define positive recurrence, we first have to define a more neutral concept, i.e. *recurrence*.

Definition 2.9: Recurrency and transiency

Consider a DMTC M with state space $\mathcal{I}$. Given a state $i \in \mathcal{I}$, let f_i be the probability of, eventually, returning to i starting from i, i.e.:

$$f_i = \Pr\left[\bigcup_{n=1}^{+\infty} X_n = i \mid X_0 = i\right]$$

We say that state i is recurrent if $f_i = 1$, otherwise we say that it is transient. We say that a Markov chain is recurrent if every state is recurrent. We say that a Markov chain is transient if every state is transient.

Proposition 2.2

With probability 1, the number of visits to a recurrent state is infinite. With probability 1, the number of visits to a transient state is finite.

Proof. If a state j is recurrent, then starting in state j, with probability 1 we will visit j again. Thus, repeating this argument, we see that with probability 1 state j will be visited an infinite number of times. In contrast, if state j is transient, then every time we visit state j, there is some probability $(1 - f_j)$ that we will never again visit j. Thus, with probability 1 state j will be visited a finite number of times. □

Proposition 2.3

Consider a DMTC M with state space $\mathcal{I}$. If state $i \in \mathcal{I}$ is recurrent, every state $j \in \mathcal{I}$ that communicates with i is also recurrent. Similarly, If state $i \in \mathcal{I}$ is transient, every state $j \in \mathcal{I}$ that communicates with i is also transient.

Proof. Omitted. □

Definition 2.10: Positive recurrence and null recurrence

Consider a DMTC M with state space $\mathcal{I}$. Given a recurrent state $i \in \mathcal{I}$, let μ_i be the expected time taken to return to state i when starting from i, i.e.:

$$\mu_i = \sum_{n=1}^{+\infty} n \Pr[X_n = i \mid X_0 = i]$$

We say that state i is positive recurrent if $\mu_i < +\infty$, otherwise we say that it is null recurrent.

The above proposition implies that in an irreducible Markov chain states are either all recurrent or transient. Without diving into mathematical details, we give a general form of the previous theorem that holds for every ergodic Markov chain.

Theorem 2.3: Ergodic theorem for Markov chains

In every ergodic Markov chain the limiting distribution exists.

Proof. Omitted. $\square$

2.6 The PageRank algorithm [Todo???]

2.7 Solved exercises

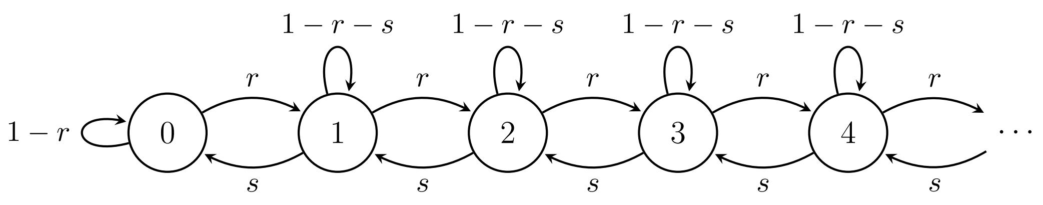
Problem 2.1

Consider an open system formed by a single server with an unbounded queue. Suppose that, at every time step, with probabiliy p_a a new job arrives to the server, while old jobs depart with probability p_d. Note that during a time step, we might have both an arrival and a transmission, or neither.

Assuming the limiting probability exists, find the values of $\mathbb{E}[N]$, $\mathbb{E}[N^{\text{server}}]$ and $\mathbb{E}[N^{\text{queue}}]$ without using the barrier technique.

Solution:

Using the given data, we can compute the probabilitites r and s of having a population increase or decrease. A population increase happens when a new job arrives and no job departs. Similarly, a population decrease happens when a no job arrives and a job departs. Thus, we get that $r = p_a(1 - p_d)$ and $s = (1 - p_a)p_d$. We construct a DTMC with an infinite set of states $\mathcal{I} = \mathbb{N}$ using the found probabilities.



Assuming the limiting probability exists, we consider the following stationary equations:

$$\begin{cases} \pi_0 &= (1-r)\pi_0 + s\pi_1 \\ \pi_i &= r\pi_{i-1} + (1-r-s)\pi_i + s\pi_{i+1} & \forall i > 0 \\ \sum_{i=0}^{+\infty} \pi_i &= 1 \end{cases}$$

Claim: $\forall i > 0$ it holds that $\pi_i = \frac{r}{s}\pi_{i-1}$.

Proof of the claim. We prove by induction that forall $i > 0$ it holds that $\pi_i = \frac{r}{s}\pi_{i-1}$. By simplifying the first stationary equation, we get that $\pi_1 = \frac{r}{s}\pi_0$, proving the base case. Assuming the equality holds for i, we prove that the i-th equation implies the equality for $i + 1$. In particular, we observe that $r\pi_{i-1} = s\pi_i$ is implied by the inductive hypothesis. Therefore, by substituting this in the i-th equation we get that:

$$\pi_i = s\pi_i + (1-r-s)\pi_i + s\pi_{i+1} \implies \pi_{i+1} = \frac{r}{s}\pi_i$$

□

Through the claim we get that $\pi_i = \rho^i \pi_0$ holds $\forall i \in \mathbb{N}$, where $\rho = \frac{r}{s}$. By substituting this result in the last stationary equation, we get that:

$$1 = \sum_{i=0}^{+\infty} \rho^i \pi_0 = \pi_0 \sum_{i=0}^{+\infty} \rho^i = \pi_0 \cdot \frac{1}{1-\rho}$$

where the last step is due to the closed form of the geometric series, concluding that $\pi_0 = 1 - \rho$ and thus that $\pi_i = \rho^i(1-\rho)$ holds for all $i \in \mathbb{N}$ Finally, we compute $\mathbb{E}[N]$ using the definition of expected value:

$$\mathbb{E}[N] = \sum_{i=0}^{+\infty} i \Pr[N = i] = \sum_{i=0}^{+\infty} i\pi_i = \sum_{i=0}^{+\infty} i\rho^i(1-\rho) = \rho \sum_{i=0}^{+\infty} i\rho^{i-1}(1-\rho) = \rho\frac{1}{1-\rho}$$

where the last step is due to the closed form of the expected value of a geometric random variable, concluding that $\mathbb{E}[N] = \frac{\rho}{1-\rho}$. Similarly, we compute $\mathbb{E}[N^{\text{server}}]$ as:

$$\mathbb{E}[N^{\text{server}}] = 0 \cdot \Pr[N^{\text{server}} = 0] + 1 \cdot \Pr[N^{\text{server}} \neq 0] = (1 - \pi_0) = \rho$$

and $\mathbb{E}[N^{\text{queue}}]$ through the previous values:

$$\mathbb{E}[N^{\text{queue}}] = \mathbb{E}[N] - \mathbb{E}[N^{\text{server}}] = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho}$$

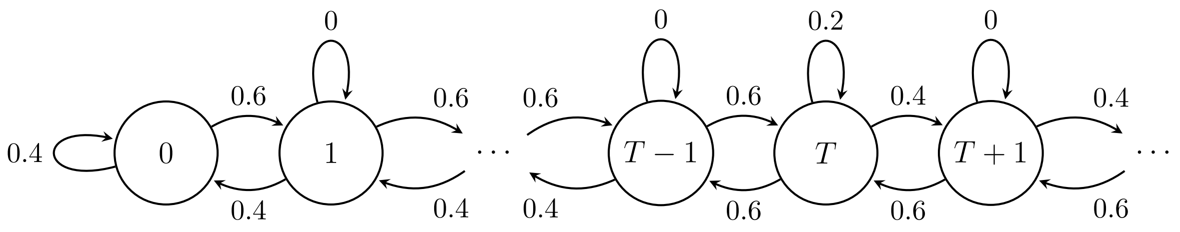
Problem 2.2

We define a threshold queue with parameter T as follows. When the number of jobs is less than T, the number of jobs decreases by 1 with probability 0.4 and increases by 1 with probability 0.6 at each time step. When the number of jobs increases to T or more, the reverse is true: the number of jobs increases by 1 with probability 0.4 and decreases by 1 with probability 0.6.

Assuming that the limiting probabilities exist, use the stationary equations to derive the limiting probability distribution as a function of T, for arbitrary threshold T.

Solution:

We start by drawing the DTMC describing the problem.



Assuming the limiting probability exists, we apply the barrier technique of each pair of adjacent states. We observe that for the first T states it holds that $0.6 \cdot \pi_{i-1} = 0.4 \cdot \pi_i$, while for the other states it holds that $0.4 \cdot \pi_{j-1} = 0.6 \cdot \pi_j$. Therefore, we get the following system:

$$\begin{cases} \pi_i &= \frac{3}{2}\pi_{i-1} \qquad \forall i \in \{1, \dots, T\} \\ \pi_j &= \frac{2}{3}\pi_{j-1} \qquad \forall j \in \mathbb{N} - \{1, \dots, T\} \\ \sum\limits_{i=0}^{+\infty} \pi_i &= 1 \end{cases}$$

By unrolling the recursive equations, we get that:

$$\pi_k = \begin{cases} \left(\dfrac{3}{2}\right)^k \pi_0 & \text{if } k \leq T \\ \left(\dfrac{2}{3}\right)^{k-T} \left(\dfrac{3}{2}\right)^T \pi_0 & \text{if } k > T \end{cases}$$

Using the last equation, through algebraic manipulation we get that:

$$\begin{aligned} 1 &= \sum_{i=0}^{+\infty} \pi_i \\ &= \sum_{i=0}^{T} \pi_i + \sum_{j=T+1}^{+\infty} \pi_j \\ &= \sum_{i=0}^{T} \left(\frac{3}{2}\right)^i \pi_0 + \sum_{j=T+1}^{+\infty} \left(\frac{2}{3}\right)^{j-T} \left(\frac{3}{2}\right)^T \pi_0 \\ &= \pi_0 \left(\sum_{i=0}^{T} \left(\frac{3}{2}\right)^i + \left(\frac{3}{2}\right)^T \sum_{j=T+1}^{+\infty} \left(\frac{2}{3}\right)^{j-T} \right) \\ &= \pi_0 \left(\frac{1 - \left(\frac{3}{2}\right)^{T+1}}{1 - \frac{3}{2}} + \left(\frac{3}{2}\right)^T \sum_{h=0}^{+\infty} \left(\frac{2}{3}\right)^{h+1} \right) \\ &= \pi_0 \left(\frac{1 - \left(\frac{3}{2}\right)^{T+1}}{1 - \frac{3}{2}} + \left(\frac{3}{2}\right)^{T-1} \sum_{h=0}^{+\infty} \left(\frac{2}{3}\right)^{h} \right) \\ &= \pi_0 \left(\frac{1 - \left(\frac{3}{2}\right)^{T+1}}{1 - \frac{3}{2}} + \left(\frac{3}{2}\right)^{T-1} \cdot \frac{1}{1 - \frac{2}{3}} \right) \end{aligned}$$

By letting $\rho = \frac{2}{3}$, we conclude that:

$$\pi_0 = \left(\frac{1 - \rho^{-(T+1)}}{1 - \rho^{-1}} + \rho^{-(T-1)} \cdot \frac{1}{1 - \rho} \right)^{-1}$$

and thus that:

$$\pi_k = \begin{cases} \rho^{-k} \left(\frac{1-\rho^{-(T+1)}}{1-\rho^{-1}} + \rho^{-(T-1)} \cdot \frac{1}{1-\rho} \right)^{-1} & \text{if } k \leq T \\ \rho^{k-2T} \left(\frac{1-\rho^{-(T+1)}}{1-\rho^{-1}} + \rho^{-(T-1)} \cdot \frac{1}{1-\rho} \right)^{-1} & \text{if } k > T \end{cases}$$

3

Analysis of continuous-time systems

3.1 The Poisson arrival process

Before transitioning from discrete-time Markov chains to continuous-time Markov chains in order to use them as a tool to analyze real-world systems, we have to give make some assumptions that are good enough to describe real world systems. In particular, we'll define the so called **Poisson arrival process**.

Informally, we recall that Poisson distributions are used to describe a the number of successes in a stochastic process whose successes happen very rarely and independently. More formally, we recall that each Poisson variable $X \sim \text{Poiss}(\lambda)$ corresponds to a binomial random variable $X' \sim \text{Bin}(n, p)$ where $\lambda = np, n \to +\infty$ and $p \to 0$.

We recall that binomial distributions represent the number of successes in a Bernoulli process, i.e. a sequence $X_1, \dots, X_n$ of i.i.d. Bernoulli variables. In Bernoulli processes, the arrivals can occur only at positive integer multiples of some given increment size (often taken to be 1). It's easy to see that such process could also be characterized by the sequence $T_1, T_2, \dots$ of **interarrival times**, where T_i represents the number of failures (i.e. non-arrivals) between the $(i-1)$-th success and the i-th success (i.e. the i-th arrival). In particular, we notice that these interarrival times are geometrically distributed i.i.d. random variables: by definition, the geometric distribution represents the number of failures before a success.

A Poisson arrival process is, in many ways, is the continuous-time version of the Bernoulli process, resulting in a simple and widely used stochastic process for modeling the times at which arrivals enter a system. Before formally defining Poisson arrival processes, we need to state some definitions.

Definition 3.1: Event sequence

An event sequence is a continuous time stochastic process described by a function $N : \mathbb{R}^+ \to \mathbb{N}$ such that $N(t)$ is the number of events that occurred up to time t.

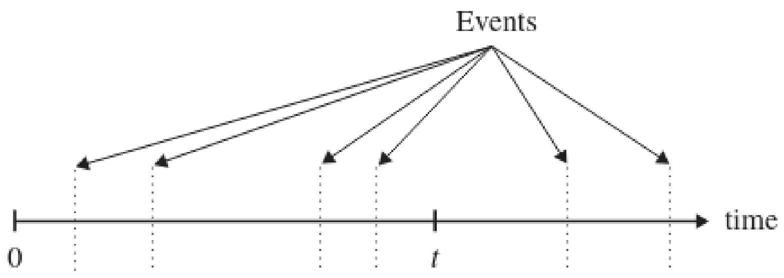


Figure 3.1: An event sequence where $N(t) = 4$.

We say that a sequence of events has **independent incrementals** if for all $t_0, \dots, t_n$, with $n \in \mathbb{N}$, the random variables $X_1, \dots, X_n$, defined as $X_i = N(t_i) - N(t_{i-1})$ for all $i \in [n]$, are all independent.

Moreover, we say that a sequence of events has **stationary increments** if forall $t \in \mathbb{R}^+$ it holds that the random variable $X_{t,s}$, defined as $X_{t,s} = N(t-s) - N(s)$, has the same distribution for all $s \in \mathbb{R}^+$.

Definition 3.2: Poisson arrival process

A poisson arrival process with rate λ is an event sequence such that:

- $N(0) = 0$
- The sequence has independent increments
- The number of events in any interval of length t is Poisson distributed with parameter $t\lambda$. In other words, for all $t, s \in \mathbb{R}^+$ it holds that:

$$\Pr[N(t+s) - N(s) = k] = \frac{e^{-\lambda t}(\lambda t)^k}{k!}$$

  with $k \in \mathbb{N}$.

We observe that the third condition is just a specific case of stationary increments. Observe that the assumption of stationary and independent increments is equivalent to asserting that, at any point in time, the process probabilistically restarts itself: the process from any point onward is independent of all that occurred previously (by independent increments) and also has the same distribution as the original process (by stationary increments). In other words, the interarrival times $T_1, T_2, \dots$ of a Poisson process are **memoryless**.

We recall that a random variable X is said to be *memoryless* when:

$$\Pr[X > t + s \mid X > s] = \Pr[X > s]$$

Now, we also recall that there are only two random variables with this property: the geometric distribution (in the discrete case) and the exponential distribution (in the continuous case). Since we're in a continuous time process, we conclude that interarrival times must be exponentially distributed. In fact, we observe that the cumulative distribution function (CDF) F_{T_i} for each $i \in [n]$ is equal to:

$$\begin{aligned}
F_{T_i}(t) &= \Pr[T_i \leq t] \\
&= 1 - \Pr[T_i > t] \\
&= 1 - \Pr[N(t + T_{i-1}) - N(T_{i-1}) = 0] \\
&= 1 - e^{-\lambda t}
\end{aligned}$$

which is exactly the CDF of an exponential distribution. Of course, the converse can also be proven, but it requires a way more formal proof.

Theorem 3.1: Poisson arrival process (2° def.)

A Poisson process with rate λ is a sequence of events such that $N(0) = 0$ and the interarrival times $T_1, T_2, \dots$ are i.i.d. exponential random variables of parameter λ.

Proof. Omitted. □

The Poisson process is the most widely used model for arrivals into a system for two main reasons. First, the Markovian properties of the Poisson process make it analytically tractable. Second, many phenomena that are the aggregate effect of a large number of individuals behave in a Poisson fashion. For instance, in communications networks, such as the telephone system, it is a good model for the sequence of times at which telephone calls are originated. Although the calls of a single user do not look like a Poisson process, the aggregate over many users does.

3.2 Continuous-time Markov Chains

After defining the Poisson process, we're ready to jump to the continuous analogous, i.e. **continuous-time Markov chains (CTMC)**. We start by giving a formal definition, similar to the discrete case.

Definition 3.3: Continuous-time Markov Chain

We define a continuous-time Markov chain (CTMC) as an continuous stochastic process $\{X(t) \mid t \geq 0\}$ such that:

- $\mathcal{I}$ is the state space
- $\forall t \geq 0$, $X(t)$ is the random variable ranging over $\mathcal{I}$ and describing the state of the system at (continuous) time step t.
- The Markovian property holds for each state, meaning that $\forall t, s \in \mathbb{N}$, $\forall j, i \in \mathcal{I}$ and $\forall x : \mathbb{R}^+ \to \mathcal{I}$ it holds that:

  $$\Pr[X(t+s) = j \mid X(s) = i, X(u) = x(u), 0 \leq u \leq s] = \Pr[X(t+s) = j \mid X(s) = i]$$

It's easy to see that through this definition we cannot easily establish a graphical model to represent CTMC due to the absence of discrete time. We'll build an intuition through the following discussion.

Let τ_i be the time until the CTMC leaves state i, given that the CTMC is currently in state i. By the Markovian and stationary properties of the CTMC, the probability that the CTMC leaves state i in the next t seconds is independent of how long the CTMC has already been in state i. In other words, we have that:

$$\Pr[\tau_i > t + s \mid \tau_i > s] = \Pr[\tau_i > t]$$

which corresponds to the *continuous memorylessness property*. Therefore, this argument allows us to conclude that a CTMC is a continuous stochastic process with the property that every time it enters state i, the following hold:

1. The amount of time τ_i that the process spends in state i before making a transition is exponentially distributed with some rate ν_i, i.e. $\tau_i \sim \text{Exp}(\nu_i)$.
2. When the process leaves state i, it will next enter state j with some probability $p_{i,j}$ independent of the time spent at state i, where $\sum_{j\in\mathcal{I}} p_{i,j} = 1$

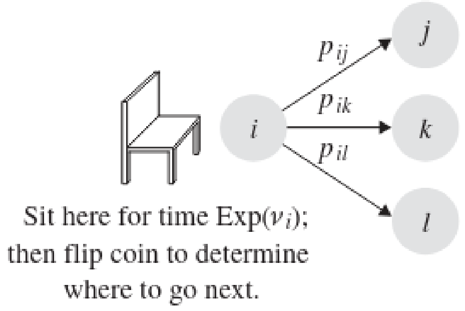

p_{ij}
p_{ik}
p_{il}
i
j
k
l
Sit here for time Exp(ν_i); then flip coin to determine where to go next.

Figure 3.2: First way to view CTMCs.

Observe that p_{ij}, the probability that when we leave i we next go to state j, is a constant: it is independent of time t (by stationarity) and it is independent of the time τ_i spent in state i (by Markovian property). Now, consider the moment just before we leave state i. At this moment, the time we have spent in state i is irrelevant (again, by Markovian property). All that is relevant is that we are at state i at this moment s. The particular time s is irrelevant as well (by stationarity).

This gives us another way to view CTMCs. Let $X_j \sim \text{Exp}(\nu_i p_{ij})$ represent the time to transition from i to j, for all $i \neq j$. Let $\tau_i = \min_{j\in\mathcal{I}}(X_j)$ be the time until the CTMCs leaves state i. Then, the next state m is given by $m = \arg\min_{j\in\mathcal{I}}(X_j)$, i.e. the state that minimizes the time waited until the CTMC leaves state i.

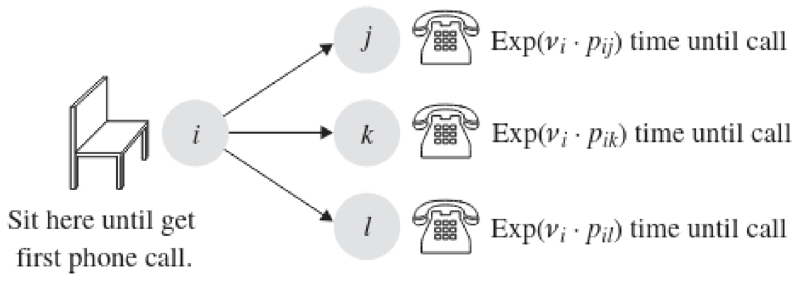

Sit here until get first phone call.
i
j
k
l
Exp($\nu_i \cdot p_{ij}$) time until call
Exp($\nu_i \cdot p_{ik}$) time until call
Exp($\nu_i \cdot p_{il}$) time until call

Figure 3.3: Second way to view CTMCs.

We won't formally prove the equivalence between the two views, as the above discussion suffices. Now, conside the following open system with continuous arrival and service rates.

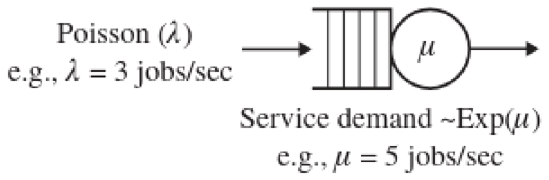


Figure 3.4: An open system with continuous arrival and service rates.

Using the second way to view CTMCs, we can model the problem through the following diagram.

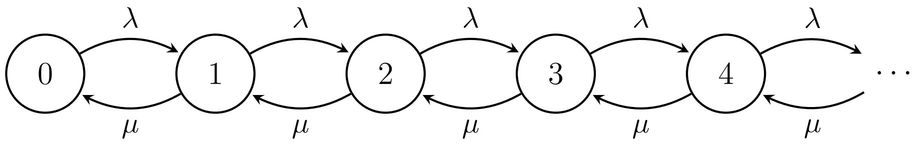


Figure 3.5: The continuous-time Markov chain of the previous example.

In particular, we observe that loops are not required: in this context, transitions represent the time waited before the state changes, not the probability of changing state! A natural question rises: do we require a brand new method to solve CTMCs? The answer, to some extent, is no: the idea is to **uniformize and discretize time** in order to study continuous-time processes using the descrete-time methods that we have already learned.

To achieve this, we can define a very small time interval δ, called *observation time*, and observe the state assumed by the continuous-time process every δ seconds. Then, given the *observation rate* $\Delta = 1/\delta$, we get that:

$$p_{\text{increase}} = \frac{\lambda}{\Delta} \qquad p_{\text{decrease}} = \frac{\mu}{\Delta}$$

We notice that in order for this idea to make sense, the observation rate Δ must be many times higher than any rate of the CTMC, i.e. $\Delta >> \lambda, \mu$. After defining such probabilities, we can define a DTMC "equivalent" to the origianl CTMC. This DTMC is referred to as **Embedded Discrete-time Markov chain (EDTMC)**.

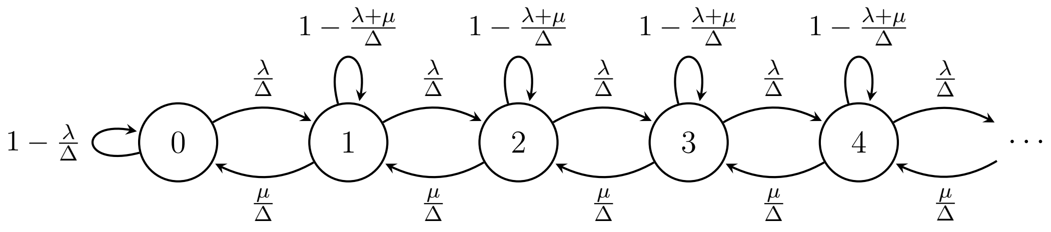


Figure 3.6: The embedded discrete-time Markov chain of the previous example.

Now, we can apply the barrier technique to compute the limiting distribution π of the EDTMC. In particular, we observe that for any pair of adjacent states we have that:

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

giving us the following system of stationary equations:

$$\begin{cases} \lambda\pi_{i-1} & = \mu\pi_i \qquad\qquad \forall i \in \mathbb{N} - \{0\} \\ \sum\limits_{i=0}^{+\infty} \pi_i & = 1 \end{cases}$$

In other words, the value chosen for Δ (or δ) doesn't influence the solutions! This is the fundamental idea that allows us to solve CTMCs by directly applying the **barrier technique** to the CTMC, without passing through the EDTMC.

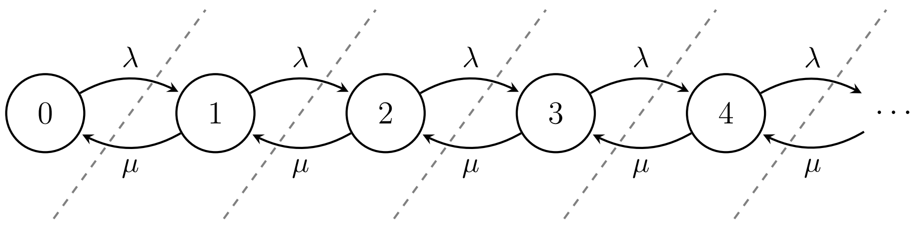


Figure 3.7: Application of the barrier technique to a continuous-time Markov chain.

3.3 Kendall's notation and common devices

Up until now, we have strictly focused on devices formed of an infinite queue and server. Now, we'll discuss devices in a more general way. First of all, we introduce a *notation* that will allow us to talk about devices in a more coincise way. In particular, we'll use **Kendall's notation**, that is the notation M/M/k/M_Q where:

- The first M stands for *memoryless arrivals*
- The second M stands for *memoryless departures*
- The parameter k stands for the number of servers connected to unique queue
- The parameter k stands for the size of the unique queue

Systems with infinite queue size are denoted with M/M/k/∞, or simply M/M/k. When the device doesn't have a queue, it is denoted with M/M/k/0.

3.3.1 Analysis of M/M/1/M_Q devices

The "standard" device that we have considered up until now is a device with memoryless arrivals, memoryless departures, one server and an infinte queue. Using Kendall's notation, this device is described as M/M/1.

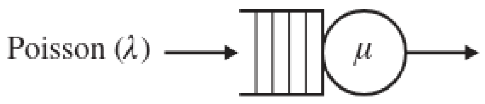


Figure 3.8: The M/M/1 queueing system.

To show that CTMCs are a good analytical tool for network systems, we first have to show that every measure that we already discussed in the first chapter is also valid in this context, i.e. it can be obtained through a simple Markov chain analysis instead of applying the definition.

Consider a M/M/1 queueing system with Poisson arrivals of rate λ and exponential service rate μ. We can model this system with the following CTMC:

$$\begin{tikzcd}
0 \arrow[r, bend left, "\lambda"] & 1 \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & 2 \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & 3 \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & 4 \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & \cdots \arrow[l, bend left, "\mu"]
\end{tikzcd}$$

Figure 3.9: The CTMC of a M/M/1 queueing system.

As always, we start by computing the limiting distribution. Using the barrier technique, we obtain the following stationary equations:

$$\begin{cases} \pi_i &= \frac{\lambda}{\mu}\pi_{i-1} \qquad\qquad \forall i \in \mathbb{N} - \{0\} \\ \sum\limits_{i=0}^{+\infty} \pi_i &= 1 \end{cases}$$

By unrolling the recursive equation we get that:

$$\pi_i = \left(\frac{\lambda}{\mu}\right)^i \pi_0 \qquad \forall i \in \mathbb{N}$$

Let $\rho = \lambda/\mu$. By substituting each π_i inside the last equation we get that:

$$1 = \sum_{i=0}^{+\infty} \pi_i = \sum_{i=0}^{+\infty} \rho^i \pi_0 = \pi_0 \frac{1}{1-\rho}$$

which concludes that $\pi_0 = 1 - \rho$. Therefore, it holds that $\pi_i = \rho^i(1-\rho)$ for all $i \in \mathbb{N}$. After computing the limiting distribution, we can focus on computing other performance measures.

First of all, we observe that ρ is still equal to the utilization of the device. In fact, since π_0 equals to the probability of the device being idle due to no jobs being inside of it, it should make sense that $\pi_0 = 1-\rho$. Moreover, we also observe that this implies that $\rho < 1$ must hold in order to have $\pi_0 > 0$, which also implies that $\lambda < \mu$ must hold, deriving the **stability condition**.

Moving on, we focus on performance metrics defined through expected values. Up until now, we have treated these values as nothing more than "a definition" in order to compute them. Now, we can finally use our new tools to compute them in the proper way. Let's start with $\mathbb{E}[N]$. By definition, we have that:

$$\mathbb{E}[N] = \sum_{i=0}^{+\infty} i \Pr[N = i] = \sum_{i=0}^{+\infty} i\pi_i$$

since π_i represents the probability of being in state i when the system is in steady mode. Hence, we get that:

$$\mathbb{E}[N] = \sum_{i=0}^{+\infty} i\rho^i(1-\rho) = \rho \sum_{i=0}^{+\infty} i\rho^{i-1}(1-\rho) = \frac{\rho}{1-\rho}$$

We observe that $\rho < 0.5$ (or even $\rho < 0.6$), the expected number of customers in the system hardly goes up. However after that point, it goes up a lot. Moreover, the impact of increasing ρ from 0.8 to 0.9 is much greater than the impact of increasing ρ from 0.7 to 0.8.

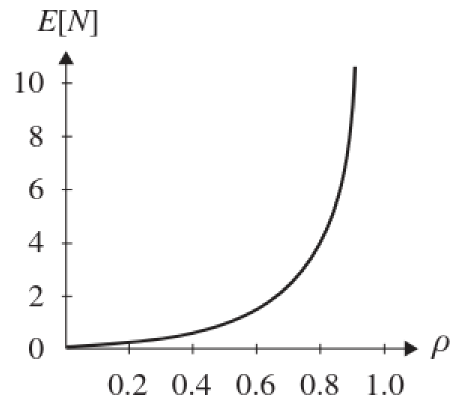


Figure 3.10: The expected population in an M/M/1 system as the value of ρ increases.

What about the expected population in the server, i.e. $\mathbb{E}[N^{\text{server}}]$. Using the formal definition of expected value we get that:

$$\mathbb{E}[N^{\text{server}}] = 0 \cdot \Pr[N^{\text{server}} = 0] + 1 \cdot \Pr[N^{\text{server}} = 1] = 0 \cdot \pi_0 + 1 \cdot \left(\sum_{i=1}^{+\infty} \pi_i \right) = 1 - \pi_0 = \rho$$

which is exactly the same result that we found in the first chapter. As a corollary, we also get that:

$$\mathbb{E}[N^{\text{queue}}] = \mathbb{E}[N] - \mathbb{E}[N^{\text{server}}] = \frac{\rho}{1-\rho} - \rho = \frac{\rho^2}{1-\rho}$$

What about the device throughput? Since we found out that $\lambda < \mu$ must hold even in this setup, we expect the throughput to be equal to $X = \lambda$. This is indeed true also in this setup. Let x_i be the arrival rate when in state i. Then, we have that:

$$X = \sum_{i=0}^{+\infty} x_i \pi_i = \sum_{i=0}^{+\infty} \lambda \pi_i = \lambda \sum_{i=0}^{+\infty} \pi_i = \lambda$$

Since everything that enters a system must also exit, we can also compute the throughput through the service rate. Let s_i be the service rate when in state i. Then, we have that:

$$X = \sum_{i=0}^{+\infty} s_i \pi_i = 0 \cdot \pi_0 + \sum_{i=1}^{+\infty} \mu \pi_i = \mu(1 - \pi_0) = \mu\rho$$

which is exactly the **utilization law**! Furthermore, by applying Little's law we get that:

$$\mathbb{E}[R] = \frac{\mathbb{E}[N]}{X} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\rho(1-\rho)} = \frac{1}{\mu - \lambda}$$

which again corresponds to what we already derived through informal arguments. As a bonus, we also get the closed formula for the expected time waited inside the queue:

$$\mathbb{E}[W] = \mathbb{E}[R] - \mathbb{E}[S] = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda}$$

Suppose now that the size of the queue is finite, meaning that we're considering a M/M/1/M$_Q$ device with Poisson arrivals of rate λ and exponential service rate μ. We observe that this device can have up to $M_Q + 1$ jobs inside it: 1 in the server and M$_Q$ in the queue. Therefore, we can represent this device with the following CTMC.

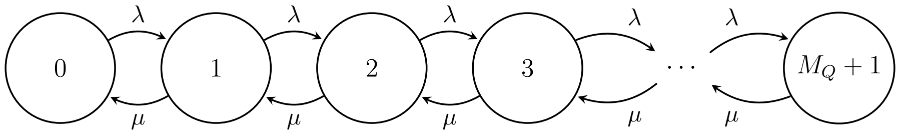


Figure 3.11: The CTMC of a M/M/1/M$_Q$ queueing system.

Since the number of states is now finite, we expect that performances must change! Again, we start by finding the limiting distribution. Let $m = M_Q + 1$. We have that:

$$\begin{cases} \pi_i &= \frac{\lambda}{\mu}\pi_{i-1} \qquad \forall i \in \{1, \dots, m\} \\ \sum\limits_{i=0}^{m} \pi_i &= 1 \end{cases}$$

By unrolling the recursive equation we get that:

$$\pi_i = \left(\frac{\lambda}{\mu}\right)^i \pi_0 \qquad \forall i \in \{0, \dots, m\}$$

Let $\rho = \lambda/\mu$. By substituting each π_i inside the last equation we get that:

$$1 = \sum_{i=0}^{m} \pi_i = \sum_{i=0}^{m} \rho^i \pi_0 = \pi_0 \frac{1 - \rho^{m+1}}{1 - \rho}$$

concluding that:

$$\pi_0 = \frac{1 - \rho}{1 - \rho^{m+1}}$$

and therefore that:

$$\pi_i = \frac{\rho^i (1 - \rho)}{1 - \rho^{m+1}} \qquad \forall i \in 0, \dots, m$$

Is the utilization of the device still equal to ρ? The answer is <u>no</u>. To distinguish it from the value ρ, we'll denote the utilization as U, which is given by:

$$U = \Pr[\text{device is working}] = 1 - \pi_0 = 1 - \frac{1 - \rho}{1 - \rho^{m+1}} = \frac{\rho(1 - \rho^m)}{1 - \rho^{m+1}}$$

All other performance metrics that are defined through expected values will also naturally change due to the sum being finite. For instance, we have that:

$$\begin{aligned}
\mathbb{E}[N] &= \sum_{i=0}^{m} i\pi_i \\
&= \sum_{i=0}^{m} i\frac{\rho^i(1-\rho)}{1-\rho^{m+1}} \\
&= \frac{\rho}{1-\rho^{m+1}} \sum_{i=0}^{m} i\rho^{i-1}(1-\rho) \\
&= \frac{\rho}{(1-\rho^{m+1})(1-\rho)}
\end{aligned}$$

To compute the throughput of a device with finite queue, we must consider that packets may indeed be *dropped*: if the buffer is full and another job arrives, it must be dropped. Let λ_{drop} be the drop rate, that being the value such that $X = \lambda - \lambda_{\text{drop}}$. We observe that, for a generic M/M/1/M$_Q$ system, packets get dropped only when the queue reaches its full capacity, i.e. when the population in the system reaches $M_Q + 1$. In other words, we have that:

$$\lambda_{\text{drop}} = \lambda \cdot \Pr[\text{drop a packet}] = \lambda\pi_{M_Q+1} = \frac{\lambda\rho^m(1-\rho)}{1-\rho^{m+1}}$$

Other performance metrics can be computed in the same way by applying Little's law. In particular, we observe that as $m \to +\infty$ every performance metric of the M/M/1/M$_Q$ device tends to become equal to the one of the M/M/1/∞ device.

3.3.2 Analysis of M/M/k/M$_Q$ devices

After showing that CTMC analysis is sound for M/M/1/M$_Q$ devices we move to more complex structures, that being M/M/k/M$_Q$ devices, also called *multi-server systems.* We assume to be working with Poisson arrivals of rate λ and that every server of the device shares the same exponential service rate μ.

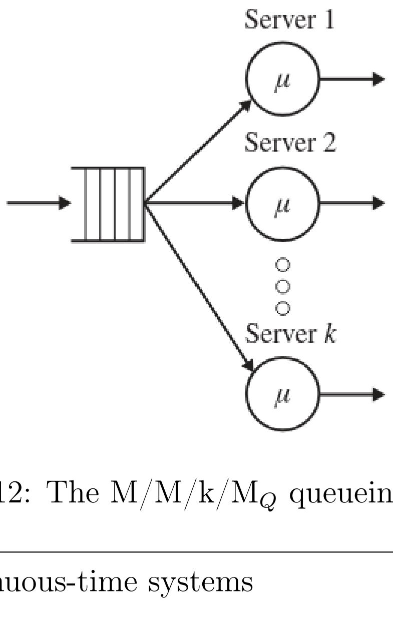


Figure 3.12: The M/M/k/M$_Q$ queueing system.

If we try to model the system through its population as done previously, a natural question arises: should we consider the population of each server? Since each server has the exact same service rate, this is not required: we don't care which server contains a job and which doesn't, we only care about how many of them are busy. This allows us to consider the ensamble of servers as a big box with at most k slots, for a total of $M_Q + k$ jobs in the system: k jobs in the servers and M_Q jobs in the queue.

However, this commodity comes at a price: considering the servers as a unique box implies that more jobs may depart from the system based on the number of servers that are working. For instance, if only one job is in the system we know that it can depart with rate μ since only one server of the ensable is working, but if two jobs are in the system then they can depart with rate 2μ since two servers are working, and so on. When all k servers are busy, meaning that we're in state $i \in \{k, \dots, M_Q + k\}$, the total service rate is fixed to $k\mu$ since no more server can be working. Therefore, we get the following Markov chain.

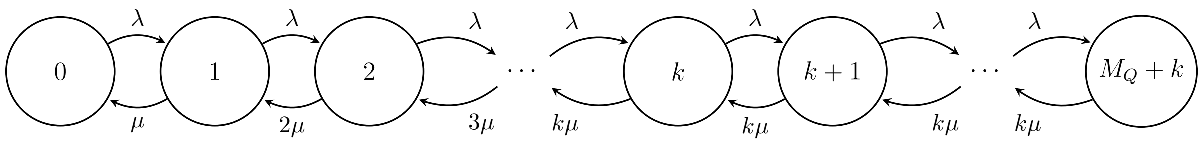


Figure 3.13: The CTMC of a M/M/k/M$_Q$ queueing system.

To find the limiting distribution, we can still use the barrier technique, obtaining the following stationary equations:

$$\begin{cases} \pi_i = \frac{\lambda}{i\mu}\pi_{i-1} & \forall i \in \{1, \dots, k-1\} \\ \pi_i = \frac{\lambda}{k\mu}\pi_{i-1} & \forall i \in \{k, \dots, m\} \\ \sum_{i=0}^{m} \pi_i = 1 \end{cases}$$

where $m = M_Q + k$. Let $\rho = \lambda/\mu$. By unrolling the recursive equation we get that:

$$\pi_i = \begin{cases} \dfrac{1}{i!}\rho^i \pi_0 & \text{if } i \in \{1, \dots, k-1\} \\ \dfrac{1}{k! \cdot k^{i-k}}\rho^i \pi_0 & \text{if } i \in \{k, \dots, m\} \end{cases}$$

By substituting each equation in the last one we obtain that:

$$\begin{aligned}
1 &= \sum_{i=0}^{m} \pi_i \\
&= \sum_{i=0}^{k-1} \pi_i + \sum_{i=k}^{m} \pi_i \\
&= \sum_{i=0}^{k-1} \frac{1}{i!}\rho^i \pi_0 + \sum_{i=k}^{m} \frac{1}{k! \cdot k^{i-k}} \rho^i \pi_0 \\
&= \pi_0 \left(\sum_{i=0}^{k-1} \frac{\rho^i}{i!} + \frac{1}{k!} \sum_{i=k}^{m} \frac{1}{k^{i-k}} \rho^i \right) \\
&= \pi_0 \left(\sum_{i=0}^{k-1} \frac{\rho^i}{i!} + \frac{1}{k!\rho^k} \sum_{h=0}^{m-k} \left(\frac{\rho}{k}\right)^h \right) \\
&= \pi_0 \left(\sum_{i=0}^{k-1} \frac{\rho^i}{i!} + \frac{1 - \theta^{m-k+1}}{k!\rho^k(1-\theta)} \right)
\end{aligned}$$

where $\theta = \rho/k$. Therefore, we have that:

$$\pi_0 = \left(\sum_{i=0}^{k-1} \frac{\rho^i}{i!} + \frac{1 - \theta^{m-k+1}}{k!\rho^k(1-\theta)} \right)^{-1}$$

Sadly, this cannot be expressed in a more coincise way – actually it could, but it requires advanced calculus functions. After finding out the limited distribution, we can compute each performance metric proceeding as we did for M/M/1/M$_Q$ devices.

3.4 Tandem systems

After analyzing systems composed of a single device with one or multiple server, we start to investigate networks with multiple devices. In particular, consider the *tandem* of M/M/1/0 devices represented in the figure below.

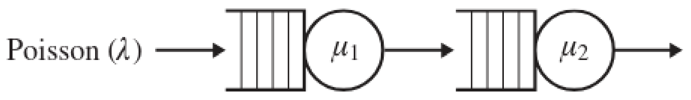


Figure 3.14: A tandem of M/M/1/0 devices.

We observe that the two servers have no queue. However, the wiring of the network forces the first server to act as a queue of size of one for the second server. In particular, if the latter is busy and the first server has completed its job, the job will sit inside the first server until the second server becomes free.

This forces us to consider the population of both devices simultaneously, meaning that the state space of the system must consider both. To achieve this, we can define a *vectorialized* state space $\{(0,0),(1,0),(0,1),(1,1)\}$ where the first entry of each pair represents the state of the first device and the second entry does so accordingly for the second state. The idea behind this vectorialized state space is simple: from state $(0,0)$ we jump to state $(1,0)$ when the first server receives a job, from state $(1,0)$ we jump to state $(0,1)$ when the first server completes a job, and so on. However, we observe that we need an additional state to model the fact that server one may be acting as a queue for the second server. We represent this state as $(Q,1)$, obtaining the state space $\mathcal{I} = \{(n_1,n_2) \mid n_1 \in \{0,1,Q\}, n_2 \in \{0,1\}\}$.

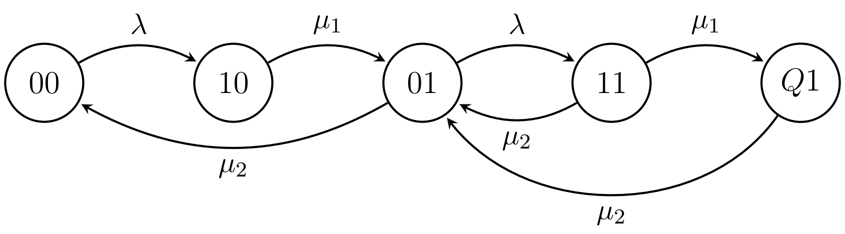


Figure 3.15: The CTMC of a tandem of M/M/1/0 devices.

Even thought this chain doesn't appear in the usual form, we can still apply the barrier technique to find out the limiting probability of this chain. Differently from previous cases, we consider a barrier that encapsulates a single state. For instance, by applying the barrier technique as shown in the below figure, we get that:

$$(\lambda+\mu_2)\pi_{0,1} = \mu_1\pi_{1,0} + \mu_2(\pi_{1,0}+\pi_{Q,1})$$

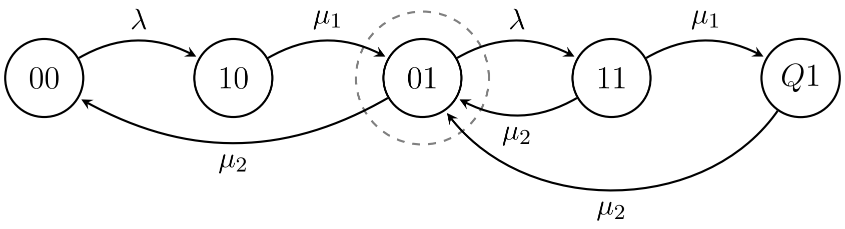


Figure 3.16: Application of the barrier technique on state $(0,1)$.

We derive the stationary equations proceeding in the same way on all states:

$$\begin{cases} \lambda\pi_{0,0} &= \mu_2\pi_{0,1} \\ \mu_1\pi_{1,0} &= \lambda\pi_{0,0} \\ (\lambda+\mu_2)\pi_{0,1} &= \mu_1\pi_{1,0} + \mu_2(\pi_{1,0}+\pi_{Q,1}) \\ (\mu_1+\mu_2)\pi_{1,1} &= \lambda\pi_{0,1} \\ \mu_2\pi_{Q,1} &= \mu_1\pi_{1,1} \\ \sum_{i\in\mathcal{I}}\pi_i &= 1 \end{cases}$$