SAPIENZA
UNIVERSITÀ DI ROMA

"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

October 25, 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

Introduction to queueing theory

1.1 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 it's queue.

To evaluate the performance of this system, we use three main parameters:

- **Arrival rate** (λ): the average number of jobs entering the system per second, corresponding to the workload or demand placed on the system.
- **Service rate** (μ): the processing power of the CPU, measured as the average number of jobs the CPU can complete per second.
- **Throughput** (X): the average number of jobs per second that are processed by the system.

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).

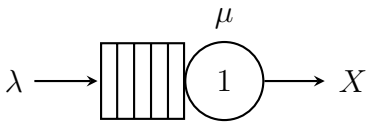

Figure 1.2: An open system and its parameters.

In order for these measures to actually mean something, we'll always assume that the system is under the heaviest load possible (meaning that it is full). 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). These three parameters are strictly related to each other and may act as a bottleneck:

- If $\lambda \leq \mu$, the server 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 $\lambda > \mu$, the server receives more jobs than those that it can process, meaning that it will be forced to drop some of them. Hence, we get that $X = \mu$.

To measure the time required by the system to process jobs, i.e. receive them and process them, three additional parameters are used:

- **Service time** (S): the average number of seconds spent by a job inside the CPU in order to be processed
- **Waiting time** (W): the average number of seconds spent by a job inside the queue before being processed
- **Response time** (R): the sum of service time and waiting time

By definition of service rate and service time, it's easy to see that $S = \frac{1}{\mu}$.

2

Idk this has to be organized

Definition 2.1: Service time

Given a device i, the service time of i is the random variable S_i whose expected value is defined as:

$$\mathbb{E}[S_i] = \frac{B_i(\tau)}{C_i(\tau)}$$

where τ is a generic time instant.

Definition 2.2: Average service rate

Given a device i, the average service rate of i, written as μ_i, is defined as:

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

where τ is a generic time instant.

We observe that $\mu_i = \frac{1}{\mathbb{E}[S_i]}$.

Law 2.1: Stability condition

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

Definition 2.3: Utilization

Given a device i, the utilization of i, written as ρ_i, is defined as:

$$\rho_i = \frac{B_i(\tau)}{\tau}$$

where τ is a generic time instant.

We observe that by definition $\Pr[\text{Device } i \text{ is busy}] = \rho_i$.

Definition 2.4: Throughput

Given a device i, the average throughput of i, written as X_i, is defined as:

$$X_i = \frac{C_i(\tau)}{\tau}$$

where τ is a generic time instant.

The average throughput of the whole system is written as X.

Law 2.2: Utilization law

Given a device i, it holds that:

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

Algebraic proof. 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 $\rho_i = \mathbb{E}[S_i]X_i$. □

Probabilistic proof. 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 $\rho_i = \mathbb{E}[S_i]X_i$. □

Definition 2.5: 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 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 2.3: Little's law

Consider an ergodic system. Let N be the random variable describing the number of jobs in the system. Then, it holds that:

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

where T is the time spent by one job in the system.

Proof. Fix a generic time instant t. Let $\mathcal{A}(t)$ denote the area used by the jobs up to time t. 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:

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

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

We observe that in open systems it holds that $T = R$ since there is no interactive system, concluding that:

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

For closed systems, instead, we can make two observations. First, we observe that $\mathbb{E}[N] = N$ since the number of jobs in the system is constant. Then, we observe that $T = R + Z$, concluding that:

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

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

Law 2.4: 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:

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

□

Definition 2.6: 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.

Law 2.5: 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:

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

□

Law 2.6: 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:

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

□

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 2.7: 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**.

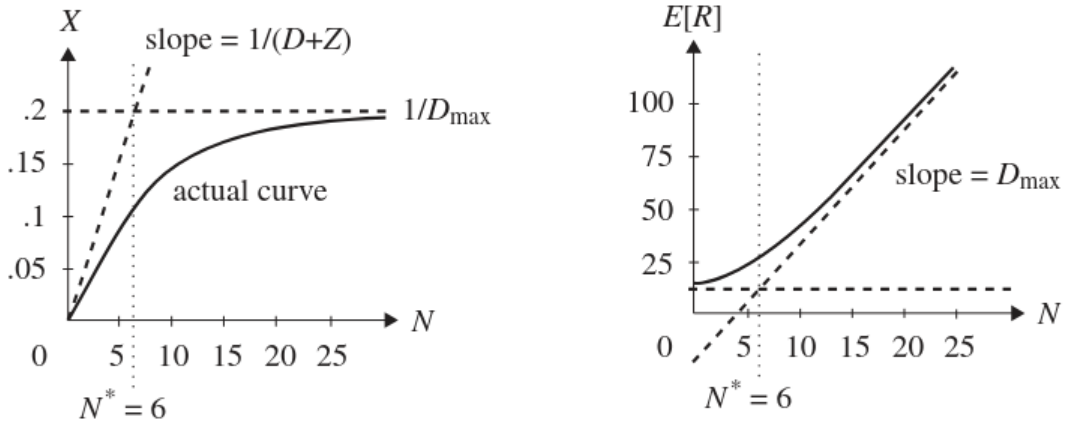

Figure 2.1: Performances of the system as a function of N

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).