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 4, 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).

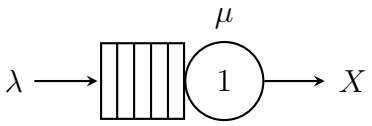

Figure 1.2: An open system and its parameters.

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 $S = \frac{1}{\mu}$.