1 Introduction

Quelli che s'innamoran di pratica sanza scienza son come il nocchiere, ch'entra in navilio sanza timone o bussola, che mai ha certezza dove si vada.[1]

Leonardo da Vinci

1.1 Network Traffic Engineering: What, Why, How

Engineering is the application of scientific principles and results to the design and optimization of machine, processes, systems. The typical approach of engineers consists of understanding objectives and requirements, abstracting a model of the system to be designed, defining a solution approach and testing for its suitability, i.e., checking if relevant performance metrics meet the prescribed requirements.

A key point is the ability of deriving a simplified model from a description of the system or function to be designed. The model should be simple enough to lend itself to analysis and provide understanding of performance trade-offs, yet it should not miss any feature having significant impact on the relevant performance indicators.

Optimization of the model is a second key step. This can be often stated as a constrained optimization problem, where constraints come from performance requirements, costs, physical limits of the system.

The entire modeling and design process can be conceived as a double loop (see Figure 1.1). First, comparison with simulations or experimental measurements leads to the refinement of the model, so that it can reliably match the relevant dynamics of the system to be modeled. Once the model is assessed, it is used to

1 "Those who are fond of practice without science are like the helmsman of a ship without rudder or compass, so that he's never sure of where he's heading."

Network Traffic Engineering: Stochastic Models and Applications, First Edition. Andrea Baiocchi.
© 2020 John Wiley & Sons, Inc. Published 2020 by John Wiley & Sons, Inc.

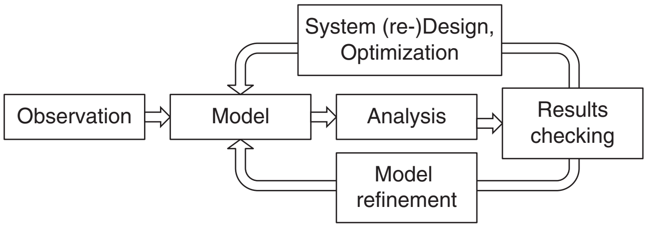

Figure 1.1 Scheme of system modeling and design process: from system observation and description, to model definition and refinement, based on comparison with simulations or experimental measurements (lower loop), then model usage for system dimensioning and optimization, according to an iterative refinement process based on performance results checking (upper loop).

refine the system design and to pursue its optimization, according to the results of the analysis, leading to new (hopefully better) performance results.

The very concise sketch of the engineering approach to problem solving is a general one. Traffic engineering refers to the design and optimization of a class of systems and processes: *networked service systems*.

Let us examine the keywords one by one.

Service system is an abstraction of any physical or logical function under Quality of Service (QoS) constraints. This is where the essence of *service* is.

Networked refers to the fact that multiple interconnected systems carry out the assigned task(s). For that purpose, "traffic" moves from one service system to another one, according to the topology of the interconnection and subject to the capacity of the network. We use terms in an informal way in this introductory section. So, by *network capacity* we mean the capability of the network to transfer resources (e.g., information, goods, vehicles) depending on the kind of service, hence network, we are considering (e.g., communication network, logistic network, transportation network) to provide service to users' demand.

Traffic can be defined informally as the stochastic process describing the users' service demand, as regards both time of demand submission to the system (arrival) and duration of service. Users of the service system (e.g., applications, persons, machines) require the service system to carry out its tasks to meet their service demand. Times when service demand is submitted to the system as well as the amount of work required to meet the specific demand can be characterized as random variables. Hence, traffic engineering is intimately connected with probability and stochastic processes theory and its applications, a prominent position being reserved to *queueing theory*. That is the preferential "language" of traffic engineering, even if also other mathematical tools are often used (e.g., fluid approximation theory, optimization theory, game theory, to mention a few).

Performance evaluation is at the heart of traffic engineering. A service system encompasses three major aspects: (i) users' traffic demand; (ii) serving capability and resources provided by the system; and (iii) QoS constraints. The aim of traffic engineering is the design of the service system to meet the expected users' demand under the prescribed quality constraints. Minimization of cost, both capital expenditure to set up the system and operational costs, is of paramount importance to the system provider. This is usually in conflict with meeting an assigned level of QoS, which is instead of primary relevance to the system users. Trading off costs for QoS, given the users' demand, is the core "business' of traffic engineering. Conversely, estimating the admissible demand for the desired level of QoS, given the available resources and the way the system is designed, is another key task of traffic engineering, leading to the definition of algorithms and procedures to rule the access of users to the system resources and to manage those resources (priority, scheduling, flow control, congestion control, multiple access).

The reason why such a discipline has been developed and has grown as a recognized field is that no 'free" resource is given in any service system. Hence, rational design of what resources to use, how much of them, and how to use them, still providing a "useful" service (i.e., meeting a specified QoS level) is key to making design of service systems viable from a technical-economic point of view. This is why the design of service systems calls for suitable quantitative methods, able to provide predictions of key performance indicators.

Since traffic engineering is based on system modeling and abstractions, it has long been recognized that many different technical fields give rise to networked service systems that lend themselves to common models, independent of details of the specific technology or application field. Mathematical tools have been developed that can be used across many different application areas to a very large spectrum of systems. To mention some of them, communication networks, computing systems, transportation networks, logistic networks, power grid networks, production processes, all can be cast into the service system abstraction and therefore be designed by resorting to network traffic engineering tools. Each example application domain is itself a highly structured and complex system, encompassing a huge variety of physical resources and processing logic (we could refer to them as "hardware" and "software," borrowing a classic terminology of information technologies).

Networked service systems can be modeled and analyzed by using different approaches. More in-depth, analysis, dimensioning, and optimization of service systems can be faced along three main lines:

1. Analytical models
2. Simulations
3. Experiments

Analytical models provide a mathematical description of the system that yields to tractable analysis (closed formulas) or, more often, to numerical investigation. This is the most powerful approach for a quick and nontrivial understanding of the performance trade-offs, to gauge stability margins of the system, to assess the impact of key system parameters on performance, to provide a setting for stating optimization problems. While producing an effective analytical model requires hard study and a bit of talent to strike the best balance between simplified assumptions and a representative model, the time and computational effort required to use an analytical model make it the least costly among the three approaches listed above. The real difficulty of an analytical model is not really in solving the model once stated (books are there just to provide a guide for that purpose). It is rather the ability "to make things simpler, but not easier," to say it with Albert Einstein's words. The art of modeling consists in making all sort of assumptions leading to the simplest model that still captures the aspects that are decisive to give a sensible answer to questions on the system. A fluid model that disregards completely the discrete nature of packet traffic in a communication network, such as the Internet, can be perfectly acceptable when we set out to study algorithms for congestion control, whereas it is definitely inadequate if we are interested in characterizing the delay jitter of a packet voice multiplexer or the collision probability of a random access protocol.

Among analytical models, a major role is played by stochastic process theory and queueing theory. The most useful class of stochastic process for service system analysis relates to Markov chains. The success of stochastic process theory and queueing theory as tools for network traffic engineering motivates the space devoted to them in this book.

Analytical models provide answers to basic questions in a quick and lightweight way. We can gain valuable insight on the system performance cheaply. When it comes to assessing second-order effects or we need to relax assumptions on the system model in a way that does not yield to analytical tractability any more, computer simulation is often a valid approach.

Computer simulation for network traffic engineering purposes amounts to defining a detailed operational model of the system and reproducing all processes involved in this detailed model by means of a computer program. This is obviously not a real-life system; it is, rather, a virtual image of how a simplified version of the real-life system would work. The limits of this approach reside only in the limits of the coding language and of the available computational resources. Extremely complex models can be simulated. As a matter of example, simulating a vehicular traffic management applications implies modeling these features:

- The road map. This includes meta-data describing road lanes, directions, signals, and surrounding environment (buildings, trees, tunnels).

- Vehicle mobility of different types of vehicles (cars, trucks, buses, motorcycles, bicycles, etc.), possibly also pedestrians. This includes modeling every useful detail of the mobility mechanics (end-to-end flow vehicles, routes, motion laws, overtaking, reaction times).
- Communication equipment on board vehicles, e.g., cellular transponders or other equipment for local communications (WiFi, vehicle-to-vehicle communication devices). This means modeling all communication architecture layers, from the physical layer to the application layer, with all relevant details of protocols of evert layer, modeling radio propagation among devices, modeling telecommunications traffic generation processes.
- Incidents that change the mobility environment.
- The logic of the application for vehicular traffic management, including the relevant message exchange among vehicles and the fixed network infrastructure.
- Feedback on vehicle mobility due to the information acquired through the traffic management application, according to the logic of the application.

Clearly, the software implementing all of these aspects must be highly complicated. A possible simulation framework able to support this kind of modeling is provided by VEINS (`https://veins.car2x.org`), a software package that integrates a module for the simulation of urban mobility (SUMO), a module to simulate communication network protocol stack (OMNET++), and all the logic required to develop simulation software of customized application logic, to import roadmaps and any other meta-data required to parametrize the simulation experiments.

Even this brief example suggests how powerful simulation can be. On the down side, setting up a simulation software requires a relatively high software coding skill level and possibly extensive training on specialized software packages (dedicated simulation software packages have been developed in many application fields of science and engineering, e.g, communication networks, transportation systems, power distribution). Mastering one of those specialized softwares typically requires several weeks up to months. Another drawback of the simulation-based approach is the limited flexibility of the model (making modifications can be very costly in terms of person-time effort). The computational burden of simulation could also be a problem, often making difficult to obtain a quick answer to what-if questions. The availability of extremely large processing resources in the public cloud and the ever-decreasing cost of computing power relax the boundary of feasibility continuously, but the bottom line is that developing a simulation model and running simulation experiments is worth it when one already has a quite firm vision of how the system could be designed and needs more stringent answers to a number of detailed issues before delving into the actual realization of the system.

The last word on system performance is real-life experiments on a possibly scaled-down prototype. Here we need not make any assumptions or simplifications on reality. However, development time and required skill level, as well as material cost, can grow to a significantly higher level than with simulations. At the same time, an experimental setting is even less flexible and easy to use than simulations. This is by far the less desirable alternative when it comes to characterizing the performance of a system, providing input for its design, and optimizing algorithms. It is however, the only way to give evidence that: (i) the proposed service system, or at least key parts of it, can be actually realized in a viable way; and (ii) key assumptions made in the development of the analytical or simulation model are backed up by experimental data. Both points are an unavoidable step in the (long) process leading from an idea to a successful product, whether it be a physical system or a process. Many technology success stories start with a theoretical idea, first proved by means of analytical models that point to possibly large gains of exciting breakthroughs, *provided that* some assumptions hold. Simulations and ultimately experiments give evidence of whether the theory is well grounded and promising, given the feasible technology context and the application opportunities.

1.2 The Art of Modeling

Analysis and design of service systems go through an abstraction process that leads from real-life systems or processes to a simplified formal description that yields to mathematical description. This abstraction process is more of an art than a science, since there is no single way of doing it, nor is the resulting model unique. The decision on what part of the original system is good to simplify, i.e., the list of assumptions, and the choice of the mathematical tool to be used are to a large extent a subjective decision-making process, highly dependent on the background, competence, and experience of the person(s) doing the job. A second crucial point is the application context and the very purpose we set out to develop a model, i.e., the questions for which we are seeking an answer. In a sense, a good model is one that leads as straightforwardly as possible (i.e., with the least effort) to a satisfactory answer (i.e., within the degree of accuracy we need) to the specific questions that matter for the problem at hand. There are obviously general techniques (e.g., Markov chain theory, queueing theory), established results, best practices. Yet, it is not infrequent that a useful model needs “customization,” depending on the application context, the purpose of the modeling, the amount of time, computational resources, and skill available to those that have to provide results.

Modeling is the approach taken to answer questions arising in some application context and involving ultimately the investment of workforce effort and the use of physical or logical valuable resources. The practical problems that trigger traffic engineering modeling are the essential motivation for it and push continuously the development of new tools and theories. As for many other applied science branches, sometimes traffic engineering theories are studied for themselves, i.e., extensions and generalization are investigated even beyond the specific questions that promoted initially the development of the model. Nevertheless, traffic engineering cannot be merely a theoretical investigation of mathematical tools. The art of developing a useful model from a description of a system, to answer questions on its performance and design, is a fundamental ingredient of traffic engineering. Besides a solid understanding of models, it is therefore highly recommended to be confronted with several application examples. The art of modeling is probably best grasped through a learn-by-experience approach. Let us then give two simplified yet meaningful examples of the kind of issues arising in this process.

Consider a store selling some specific kind of goods, e.g., vegetables and fruit. It can be thought of as a service system, where "users" are people coming to buy products sold at the store and the service provided by the system is the possibility of finding a selection of products with some specified quality (variety of products, freshness, packaging standards, bio-compatible production chain, special characteristics, like gluten-free). A number of questions could be posed on the system, such as in which part of the town should the shop be located, which kind of installation should be employed, how many people should be hired to run it, with which kind of skill and tasks, what is the best shop opening schedule, where and at what prices products sold at the store are best procured, and what selling prices should be applied. The kind of model to be developed depends in a crucial way on what the questions we seek an answer for are.

As a matter of example, let us ask how many employees we should hire for the store. That depends primarily on the volume of demand (how many people come to the store during a regular working day, how much vegetables and fruit they buy). It also depends on the store installation, whether it is a big store, with a reserved parking lot, a large building for goods exhibition, and large warehouses; a street shop, with a relatively small warehouse and limited space for exposing goods; or even a market stall. The economic viability of the proposed solution is the target of the modeling, under constraints on customer satisfaction, e.g., the average amount of time that a customer has to wait before being served during peak hours. The amount of people coming to the shop could vary during the day, on weekends with respect to working days, or on a seasonal basis. It depends on the local density of residential population and on how many competitors are located nearby. It is clear that many of the variables describing customers and their habits can be characterized only as random variables, with parameters that can only be predicted or measured within some level of accuracy. The complexity of the model should therefore be tuned to the accuracy of the knowledge of customer demand (there is no benefit in adopting a sophisticated customer arrival model requiring several parameters to be tuned, if we have only partial or inaccurate data to fit the model: better to use a very simple parsimonious model, with as few parameters as possible). The specific question we pose (how many employees to hire) could be answered by defining a queueing model of the store and applying known results to size the number of "servers" to meet the waiting time requirement. We could also try to state an optimization problem, e.g., what is the optimal number of employees, given a model of customer impatience (if we hire a lot of people, customer satisfaction will be excellent, we will attract a lot of customers, but the serving basin is anyway limited, so expenses for personnel would eventually exceed the growth of income; conversely, limiting the number of employees makes us lose customers and potential revenue, and that could possibly bring the store to shrink its customer basis to a level too small to survive).

As a second example, let us consider a road crossing. "Service" here consists of vehicles switching from one road to another one through the crossing. "Customers" are vehicles. The "server" is the crossing area, with all its features (e.g., traffic lights, number of lanes, roundabout or cross-shaped intersection). Depending on the way the crossing is used, it could be modeled as a single server or a set of multiple servers (e.g., in case multiple vehicles could engage the crossing simultaneously). The quality of service can be measured by the time a vehicle has to wait before being able to access the crossing and by the probability that a collision (accident) occurs at the crossing. A trade-off exists between the two. That is, if we introduce traffic lights at the crossing, we expect to reduce the probability of accidents, but also to increase the time that a vehicle has to wait before it can engage the intersection. Assume we have to optimize the green times of the crossing traffic lights to minimize the average vehicle waiting time. We could consider a first-order approximation of the system as a single server system with an infinite waiting line. This model applies to a single lane on a single road arriving at the crossing. It disregards interaction of vehicles engaging the crossing, i.e., inter-dependencies among the serving capacities of the servers representing the roads converging to the crossing. The simple model does not account either for vehicle mechanics (acceleration, speed, vehicle size and length) and for human reaction times, traffic light overhead times (dead times when switching from red to green and vice versa). A refined model could be made up of a network of queues, one for each road (or maybe, each lane) arriving at the crossing. A refined model, accounting for those details and including sophisticated statistical modeling of vehicle arrivals could be set up and analyzed by means of simulations. This is still a model, simulations being virtual processes that mimic real-life situation, still with a number of simplifications. The last word would be to construct a possibly scaled-down, real system, where experiments with real vehicles are run. This last approach is extremely costly and time-consuming. Moreover, it does not lend itself to stress the system at high traffic levels, since high traffic entails involving a large number of vehicles, making experiments unfeasible. This is why analytical modeling or simulation are extremely useful tools to understand performance trade-offs, design algorithms and optimize real-life service systems.

In the following we discuss in some detail an example. Starting from the description of a real-life issue, we derive a model and develop mathematical analysis and simulations. The assumptions required to derive the model are discussed as well as the lesson learned from performance results.

Before delving into the detailed development of a model, it is worth noting the relationship between modeling and machine learning. Machine learning encompasses a broad field of theories, algorithms, and applications that has been growing over the last several decades, leveraging on the progress of information and communication technologies. The last decade has witnessed an impressive growth of the application range of machine learning, boosted by the ever-increasing availability of computational power, of big data, and by breakthroughs in algorithm implementation (e.g., deep learning networks[2] [28,67]). Since this book is entirely devoted to modeling tools and examples of their applications to networked service systems, it is useful to take a quick look at an approach that might be alternative to modeling (as intended here).

A basic view of supervised machine learning may be stated as a problem of function identification (see Figure 1.2). As a matter of example, let us consider a set of "objects." An object is associated with a label y and is described through a set of features x. We can think of the features x as a vector of $\mathbb{R}^n$ (generalizations to qualitative features, belonging to non metric spaces, are possible). We assume that a functional relationship exists between x and y, i.e., it is $y = f(x)$ for a suitable (unknown) function $f(\cdot)$.

We aim at identifying the function $f(\cdot)$ (at least a good approximation of it) with a data-driven approach. We assume therefore that we are assigned a set of

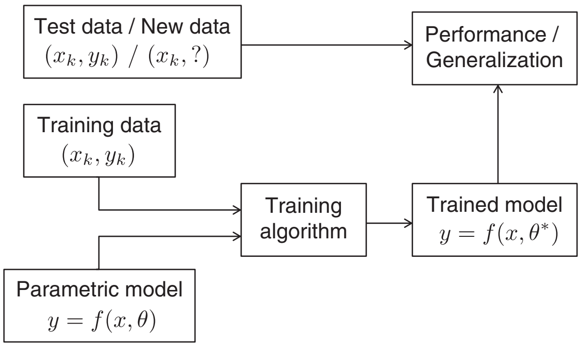

Figure 1.2 Illustration of a basic concept of supervised machine learning.

2 Recent works point out at apparently fundamental limits of current algorithms, e.g., see [75].

couples $(x_k, y_k), k \in \mathcal{I}$ (the so-called "ground truth"). We choose a parametric set of models for these data, say $f(x, \theta), \theta \in \mathcal{S}$. We split the available data into two sets: the training set $\mathcal{I}_{\text{train}}$, used to select one model within the chosen family, and a test set $\mathcal{I}_{\text{test}}$, used to evaluate the performance of the selected model.

Using the training data $(x_k, y_k), k \in \mathcal{I}_{\text{train}}$, and a suitable *learning algorithm* (training algorithm), we synthesize the "best" possible model, say $f(x, \theta^*)$ (typically, one that minimizes the error with respect to the ground truth in the training set).

Using the test data $(x_k, y_k), k \in \mathcal{I}_{\text{test}}$, we assess the performance of the synthesized function $f(x, \theta^*)$.

A key point is the *generalization* capability of the model $f(x, \theta^*)$, i.e., the ability of the selected model to yield the right value y when fed with a previously unseen input x (i.e., the ability to reconstruct a *new* couple, not belonging to the training set).

Even this very brief description of the machine learning approach highlights its key aspects: data-driven modeling, generalization capability. In a sense, this is a black-box approach. We do not start from a functional description of the system producing the objects, from which we try to find a model able to predict the output y for any given input x. Rather, we collect a (possibly large) set of *examples* by "running" the system (input-output couples), then approximate the functional relationship that we assume to exist between the input and the output data. We are therefore not required to *understand* the laws governing the internal working of the system that produces the objects. We are giving up to insight into the system. On the other hand, we are able to define a "working" model, that provides us (hopefully) useful answers, even for unmanageably complex systems, for which it is too hard to derive useful models in the "traditional" sense[3] .

Both approaches, modeling and machine learning, have their strong and weak points, both have their use cases. If we are able to state a model of the relationship between system input and output, say $y = \hat{f}(x)$, and to define an efficient algorithm to evaluate $\hat{f}(x)$ for any interesting x, there is no need to resort to machine learning. If we cannot collect enough data, in the form of couples (x_k, y_k), machine learning is not applicable, either. So what are use cases for machine learning?

- When we are not able to state a model.
- When we manage to state a model, but it is unfeasible to "solve" it, that is, to use it for deriving predictions on the studied system.
- When we have a model and feasible algorithms, but we can achieve a significant computational complexity reduction resorting to machine learning algorithms.
- When we can collect data in the form of couples (x_k, y_k) at a reasonable cost.

3 Note however that the machine learning approach cannot forget about a good grasp on the system to be modeled, even if modeling is data-driven. In fact, assuming that there exists a functional relationship between the input x and the output y is already calling for some field-expert knowledge on the specific system.

The first three items point at cases where machine learning offers a viable or preferable alternative, whereas "traditional" modeling could be at a deadlock or too hard. The last point is a precondition for machine learning to be a practical alternative.

We do not advocate either approach. Both have merits and should be considered when confronted with a real problem. "Integrated" solutions are also possible, that use both approaches to build a composite model[4] .

In the rest of this book we present tools and examples oriented to developing modeling skills. It is important to bear in mind that other approaches, besides "traditional" modeling, exist for designing and optimizing service systems. Which way to go depends on available skills and time, on design objectives, on technological opportunities.

1.3 An Example: Delay Equalization

Let us consider a streaming application in the Internet. Streaming is used for audio/video retrieval and play-out. The content is usually available in servers located in data centers (the "cloud"), possibly replicated in temporary cache memories, close to potential users (content delivery networks). The user has a play-out application (often embedded into a web browser), essentially consisting of a decoder and a graphical user interface. The encoded audio/video data is downloaded from the server to feed the decoder. Smooth play-out requires feeding the decoder with audio/video data according to exactly the same timing as produced by the encoder. This implies in turn that data delivery delay through the network should be constant and no piece of information should be lost.

In the real Internet occasional packet loss is possible (typically a few percent of packets are lost in moderately congested links). In the application example at hand, packet loss can be concealed by redundant coding, since human perception can be deceived up to a certain amount of missing information in the reconstruction of the audio/video streaming. More importantly however, packets sent through the Internet suffer variable delays: packet belonging to the same end-to-end flow, even if they follow the same network path, encounter different levels of node congestion. The inevitable result is that delays of successive packets are different and there is *no way* to avoid this impairment (unless changing the fundamental principles of data transfer through the Internet).

The contrast between application requirement (constant delay) and network operation, resulting in variable delays, can be reconciled by introducing a *delay equalization buffer* in front of the decoder. Delay equalization is performed by

4 To make a simple example, one might say that channel estimation and adaptive equalization in telecommunication receivers is a form of machine learning, embedded into a system that is designed and optimized based on mathematical models of the signal, the channel impairments, and the algorithms applied at the receiver.

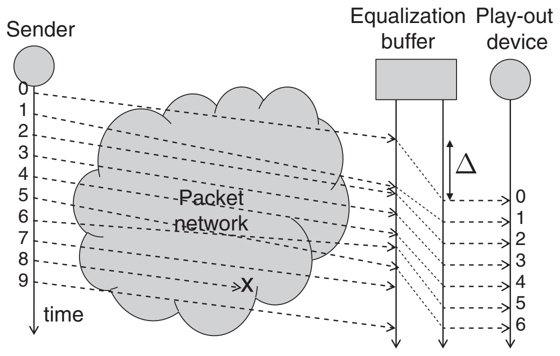

Figure 1.3 Illustration of the delay equalization of a streaming flow sent through a packet network. The 'x' denotes packet loss. The slope of the dashed arrows through the network corresponds to the delay suffered by packets. The initial play-out delay Δ is shown: once play-out starts, frames are read from the buffer at the same rate as they are produced by the sender.

imposing an *additional* delay to each arriving packet, so that the sum of the delay suffered by the packet in the network plus the equalization delay equals a fixed delay, *same* for all packets (hence the name of the algorithm: equalization has the same root as "equal").

Figure 1.3 illustrates the system components for this example: the server sending audio/video data, the intermediate network, the delay equalization buffer, and the decoder. The detailed explanation of the delay equalization algorithm is developed in Section 1.3.1.

In the following we pose performance questions, unveiling issues with the equalization algorithm (starvation, buffer overflow). We derive models that provide a quantitative tool to gain insight into the delay equalization algorithm and to dimension its parameters.

1.3.1 Model Setting

We assume that the encoder produces fixed length data frames at a fixed rate. Let L be the length of a frame and T the time interval between the emission of two consecutive frames. We assume that a frame fits into a single packet, i.e., there is a one-to-one correspondence between application-level data units (frames) and network-level data units (packets). In general, multiple packets are required to carry a single frame. Capturing this feature requires however a significantly more complex model (see Problem 1.1).

The data flow throughput sustained through the network is $\Lambda = L/T$. We assume that a transport capacity at least equal to Λ is available through the network.

Let $t_k = t_0 + kT,\ k \geq 0$ be the time when the k-th frame is released by the encoder and sent into the network. The network introduces a variable delay, say D_k for frame k. Occasionally, a frame can get lost and be never delivered at the destination. Out-of-order delivery is a possible outcome as well. In normal operating conditions, frame loss and out-of-order delivery are actually sporadic events: they could typically affect less than a few percent of the frames. For the time being we neglect frame loss and out-of-order delivery. Those issues are reconsidered in Section 1.3.3.

We also assume that the D_k's are independent, identically distributed (i.i.d.) random variables, admitting a probability density function (PDF) $f_D(x)$, defined for $x > 0$. Let also $G_D(x) = \mathcal{P}(D > x)$ be the complementary cumulative distribution function (CCDF) of the random variable D. Assuming that the D_k's have a same probability distribution amounts to require that the stochastic processes that cause delays inside the network are stationary during the data transfer. This can well be the case if the time dynamics of the network traffic (i.e., the time scale over which the average network link loads have a significant variation) is much bigger than the duration of the audio/video data transfer. As for statistical independence among the D_k's, this is a reasonable assumption, if the time dynamics of buffers within routers along the end-to-end path of the frames is smaller than the inter-frame time T. If that is true, the queue states sampled by two consecutive frames are weakly correlated. For a buffer of size B_r and a link of capacity C_r, the time scale of the buffer queue is roughly of the order of B_r/C_r. As a matter of example, with $B_r = 10$ Mbyte and $C_r = 10$ Gbit/s, we have $B_r/C_r = 8 \cdot 10^7$ bit$/10^{10}$ bit/s $= 8$ ms. If $T > 8$ ms, it is plausible that the delays encountered by two consecutive frames be negligibly correlated, hence the independence assumption is reasonable.

Play out at the receiver is started after a delay Δ from the reception of the 0-th frame (see Figure 1.3). After that, frames are consumed, one frame every T seconds, to feed the decoder at the receiver. Variable delays can be compensated by storing the incoming frames in an equalization buffer temporarily. Two issues must be faced then: (i) *starvation*, i.e., not finding the expected frame in the buffer at the time it must be used by the decoder; and (ii) *buffer overflow*, i.e., an arriving frame must be stored until play-out, but there is no more space left in the equalization buffer. We consider these two issues in the rest of this section, by dimensioning properly the initial delay Δ of the play-out and the equalization buffer size B.

1.3.2 Analysis by Equations

To keep the analysis manageable, we resort to approximations, a customary approach in system modeling. We assume the equalization buffer is infinite, i.e., we neglect the overflow issue when dimensioning the initial delay Δ.

If we apply an initial delay Δ, the k-th frame is expected at the decoder for play-out at time $t_{\text{out},k} = t_0 + D_0 + \Delta + kT,\ k \geq 0$. Starvation of the decoder is triggered by either of two possible events: (i) the k-th frame was late; or (ii) the k-th frame arrived in time, but it found a full buffer and was dropped. According to our approximation (infinite buffer size), the second event is ruled out. Hence, starvation is equivalent to the first event, i.e., it occurs if and only if $t_k + D_k > t_{\text{out},k}$, that is $D_0 + \Delta < D_k$. Denoting the probability of starvation with S, we get:

$$S = \mathcal{P}(D_k > \Delta + D_0) = \int_0^{\infty} f_D(x) G_D(x + \Delta) dx \tag{1.1}$$

For example, if $f_D(x) = \mu \exp(-\mu x)$ for $x > 0$, i.e., the delay through the network has negative exponential probability distribution with mean $\mathrm{E}[D] = 1/\mu$, we get

$$S = \frac{1}{2} e^{-\mu\Delta} = \frac{1}{2} e^{-\Delta/\mathrm{E}[D]} \tag{1.2}$$

If we require that $S < \epsilon_S$, then it must be $\Delta > \mathrm{E}[D] |\log(2\epsilon_S)|$.

Let us now relax the assumption of an infinite buffer and consider a buffer of size B. We can upper bound the probability of overflow of the buffer of size B with the probability that the occupancy level of the infinite buffer exceeds B.

Let $\tilde{N}(t)$ and $N(t)$ be the number of frames stored in the finite buffer and in the infinite buffer at time t, respectively. Let $\tilde{P}$ denote the overflow probability of a buffer of size B, i.e., the joint probability of the events $\{\tilde{N}(t) = B\}$ and $\mathcal{A}(t) = \{\text{a frame arrives at time } t\}$. Since it is[5] $\tilde{N}(t) \leq N(t)$ for all t, the event $\{\tilde{N}(t) = B\} \& \mathcal{A}(t)$ implies the event $N(t) > B$. Hence $\tilde{P} = \mathcal{P}(\text{buffer overflow}) \leq \mathcal{P}(N(t) > B)$, i.e., the probability that the buffer content of the infinite buffer exceeds the threshold B is an upper bound of the overflow probability of the finite buffer.

Let N_k be the number of frames stored in the buffer at the time immediately preceding play-out of the k-th frame, i.e., $N_k \equiv N(t_{\text{out},k-})$. Let $N = B/L$ (we assume B is an integer multiple of the fixed frame size). We have $\tilde{P} \leq \mathcal{P}(N_k > N) \equiv P$.

In the following we evaluate the probability distribution of N_k and use its tail to dimension the equalization buffer size. This is an example of how bounds and approximations help deriving performance results.

It is $N_k > n$ if and only if the arrival time of the $(k + n)$-th frame is less then the time $t_{\text{out},k}$, that is $t_{k+n} + D_{k+n} < t_{\text{out},k} = t_0 + D_0 + \Delta + kT$. Therefore, by letting $Q(n) = \mathcal{P}(N_k > n)$, we have

$$Q(n) = \mathcal{P}(t_0 + (k + n)T + D_{k+n} < t_0 + D_0 + \Delta + kT) = \mathcal{P}(D_{k+n} < D_0 + \Delta - nT) \tag{1.3}$$

5 This is also true if we account for packet loss event in the network.

Then

$$\begin{aligned} Q(n) &= \int_0^\infty f_D(x)\mathcal{P}(D_{k+n} < x + \Delta - nT)dx \\ &= \int_{\max\{0,nT-\Delta\}}^\infty f_D(x)[1 - G_D(x + \Delta - nT)]dx \\ &= G_D(\max\{0, nT - \Delta\}) - \int_{\max\{0,nT-\Delta\}}^\infty f_D(x)G_D(x + \Delta - nT)dx \end{aligned} \tag{1.4}$$

For example, with negative exponential network delays, we obtain

$$Q(n) = \begin{cases} \frac{1}{2}e^{-\mu(nT-\Delta)} & nT \geq \Delta, \\ 1 - \frac{1}{2}e^{\mu(nT-\Delta)} & nT \leq \Delta. \end{cases} \tag{1.5}$$

The upper bound P of the overflow probability we are looking for is then $P = Q(N)|_{N=B/L}$. The requirement $\tilde{P} \leq \epsilon_B$ on the overflow probability of the equalization buffer is guaranteed by imposing that $P \leq \epsilon_B$.

Let us assume that $NT \geq \Delta$. From eq. (1.5) we have $Q(N) = e^{-\mu(NT-\Delta)}/2$, with $\mu = 1/\mathrm{E}[D]$. Imposing $Q(N)|_{N=B/L} \leq \epsilon_B$, we find

$$B \geq L\frac{\Delta + \mathrm{E}[D]|\log(2\epsilon_B)|}{T} \tag{1.6}$$

Substituting the expression found for the initial delay Δ and recalling that the end-to-end throughput of the audio/video packet flow is $\Lambda = L/T$, we get

$$B \geq \Lambda\mathrm{E}[D](|\log(2\epsilon_S) + \log(2\epsilon_B)|) \quad \Rightarrow \quad \frac{B}{\Lambda\mathrm{E}[D]} \geq |\log(4\epsilon_B\epsilon_S)| \tag{1.7}$$

which gives the minimum required buffer size B as a function of the quantity $\Lambda \cdot \mathrm{E}[D]$, the so called bandwidth-delay product (BDP). The BDP is a key parameter in many networking problems. The dimensioning criterion of the buffer size B in eq. (1.7) exemplifies in a clear way the role of the quality of service constraints (the parameters ϵ_B and ϵ_S) and of key system parameters (the BDP in this case).

For $\epsilon_B = \epsilon_S = 10^{-3}$, we have $B \approx 12.43 \cdot (\Lambda \cdot \mathrm{E}[D])$. With a buffer of 256 kbytes we can face a mean network delay of about 41.2 ms for a throughput of 4 Mbit/s.

The expression in eqs. (1.2) and (1.7) hold for a negative exponential distribution of network delays. The analytical model developed above can in fact be used with a general distribution of network delays, to dimension the initial delay Δ and the buffer size B. The analytical formulas found for a general network delay distribution (eqs. (1.1) and (1.4)) can be used, at least numerically, to evaluate the starvation probability S and the upper bound P of the overflow probability for given values of the model parameters.

In the following we show the numerical results obtained by estimating the probability distribution of the network delay from a sample of measured round trip times (RTTs). The RTT trace has been collected between a host in a WiFi access

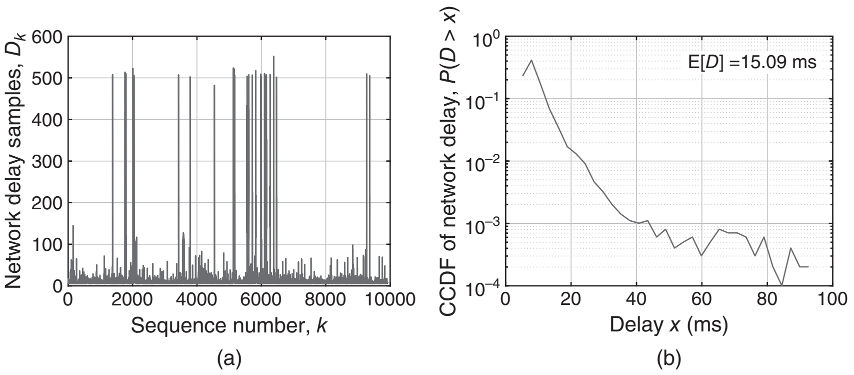

Figure 1.4 Left plot: Sequence of RTT values measured between a host in a WiFi access network and a server on public Internet. The sequence has been collected using `ping` with an interval of 0.1 *s* between consecutive message sending times. Right plot: empirical CCDF of the network delay D based on the RTT sequence shown in the left plot.

network and a server on the public internet, both located in Italy. The sequence of collected samples is shown in Figure 1.4(a). Figure 1.4(a) suggests that the network crossed by the packets exhibits a relatively large "random" variability of the delays, on top of which there are occasional delay spikes, according to an apparently random pattern. This hints to occasional heavy congestion phenomena, or to some recurring high-priority task, carried out by the target server used to collect RTT values, that causes a large delay of the `echo_reply` message.

Figure 1.4(b) illustrates the empirical CCDF of network delay. The estimated mean delay is 15.1 ms, while the estimated standard deviation is 43.2 ms. The bulk of probability is around the mean, yet there is a rather long tail that can hardly be estimated, given the available number of samples. The variability of network delay is evident also from the high value of the ratio of the standard deviation to the mean[6] .

The resulting numerical values of the starvation probability and of the overflow probability are shown in Figure 1.5(a) as a function of Δ and in Figure 1.5(b) as a function of B, respectively. We have assumed $L = 1400$ bytes and $T = 33\ ms$ (30 video frames per second).

The values of the initial delay and of the buffer size that meet the performance requirements $\epsilon_S = \epsilon_B = 10^{-2}$ are $\Delta_{\text{req}} = 220.7$ ms and $B = 19.6$ kbytes, respectively. The effect of the slow decay of the CCDF of network delays appears in the slow decay part of the curve of the starvation probability S as a function of Δ.

6 We have selected an extreme case, exhibiting a rarely seen large variability, to make numerical results more interesting in the chosen model setting. For the same reason, we have directly used the sequence of measured RTTs as representative of end-to-end network delays, without halving them.

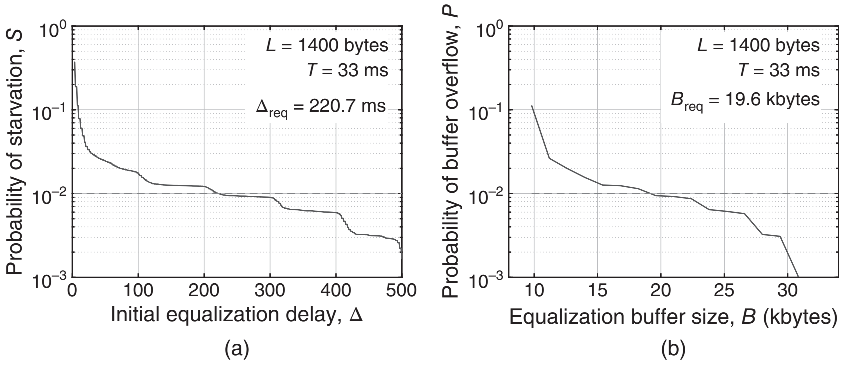

Figure 1.5 Left plot: starvation probability as a function of the initial delay Δ, assuming an infinite size equalization buffer. The minimum initial delay that meets the starvation probability requirement (dashed line) is $\Delta_{\text{req}} = 220.7$ ms. Right plot: buffer overflow probability as a function of the equalization buffer size B. The minimum buffer size that meets the packet loss probability requirement (dashed line) is $B_{\text{req}} = 19.6$ kbytes.

Even if numerical results are obtained by estimating the probability density function of the network delays from real data, still there are a number of assumptions underlying the model. Packet delays are assumed to be i.i.d. random variables, the equalization buffer is assumed to be infinite, out-of-order packet delivery has been neglected. To the cost of setting up a detailed simulation code, we can remove all of these assumptions and check results, which we do in the next section.

1.3.3 Analysis by Simulation

We can investigate the performance metrics of the delay equalization buffer by means of simulations. At the cost of developing the simulation model, coding it, and bearing the computational cost of running simulations (typically much more expensive than evaluating the analytical model), we gain the possibility to relax the assumptions we have made in the derivation of the analytical results of the previous section. Specifically, we can evaluate the frame loss probability (FLP) with the given buffer size B and initial equalization delay Δ, without the need of assuming an infinite buffer or resorting to upper bounds. As a consequence, starvation at time $t_{\text{out},k}$ occurs at the output of the buffer for two possible causes: (i) late arrival, i.e., frame k has not arrived yet; (ii) loss, i.e., frame k had already arrived at the input of the buffer, but it was dropped because of a full buffer at the time it arrived. The FLP is defined as the probability of starvation, whichever the cause.

Given the values of T and L, the simulation of the equalization buffer depends only on the parameters Δ and B, i.e., the initial delay and the buffer size.

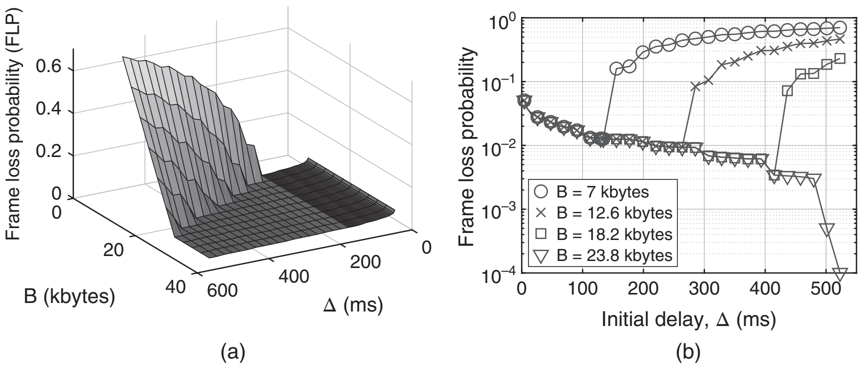

Figure 1.6 Simulations of the equalization buffer. Left plot: frame loss probability as a function of the initial delay Δ and the buffer size *B*. Right plot: frame loss probability as a function of the initial delay Δ for four different values of the buffer size *B*.

We account for the effect of the network still using the RTT experimental trace. The buffer size is converted into the maximum number of packets it can contain, $N_{\max} = \lfloor B/L \rfloor$.

Figure 1.6(a) shows the 3-D plot of the FLP, as a function of Δ and *B*. As expected, the FLP decreases, eventually going to 0 as the buffer size is increased[7] . It might appear counterintuitive that the FLP increases sharply with Δ for a given value of *B*. This is especially evident for the smaller sizes of the buffer.

The reason for this behavior is that FLP is the sum of two components. Some frames get lost because they arrive at the buffer *after* due time for play-out; other frames are lost because they are dropped due to buffer overflow. The first component is dominant at low Δ values (up to the order of a few standard deviations of the network delay) then fades away quickly as Δ grows. The second component of FLP is dominant when the buffer is small compared to the amount of frames that can arrive during the initial delay Δ. Since frames arrive at an average rate of $1/T$, on the average $\Delta \cdot L/T$ bytes arrive, before play out can start. If *B* is less than this quantity, frame loss is massive.

A clearer picture of the phenomenon can be appreciated by plotting FLP as a function of Δ for some values of *B*, as done in Figure 1.6(b). All curves have an initial common behavior, independent of *B*, that corresponds to the operation region dominated by late arrivals. This is the behavior correctly predicted by the analytical model. Then, as Δ grows up further, the subsequent behavior breaks up into

7 The FLP can actually hit 0, since this is a data-driven simulation, where packet delays are taken from a file of 10000 measured delays, hence there is a maximum delay and correspondingly a maximum initial delay beyond which no frame loss occurs, for a large enough buffer size. There are exactly 0 lost frames as soon as $\Delta > \Delta_{\max} = \max_k \{D_k\} - D_1 \approx 533$ ms and for $B > \Delta_{\max} L/T \approx 22.6$ kbytes.

different branches as a function of B. The FLP grows sharply with Δ, when the frame loss due to buffer overflows becomes dominant. The bigger B, the wider is the favorable dimensioning interval, where we achieve a low FLP value.

The comparison between Figure1.5(b) and Figure 1.6(b) gives a striking visual evidence of the gap between analytical model predictions and simulation results. Due to assumptions required to make the model tractable, the analytical curve of Figure 1.5(b) captures correctly the lower branch of the FLP curve resulting from simulations, but it misses completely the sudden increase of FLP when Δ grows beyond a threshold depending on B.

1.3.4 Takeaways

Consider a typical network traffic engineering problem, delay equalization. The highlights are as follows:

1. Defining an analytical model entails major simplifications and assumptions, the stronger the simpler the obtained results and potentially more insightful.
2. Analytical models may lose relevant effects or hold only for limited range of system parameters. Care must be taken when drawing conclusions on the basis of analytical models. They provide invaluable help in guiding the setup of more detailed evaluation tools (e.g., simulations or measurements), but they could miss some phenomena.
3. Simulations can be powerful, since they allow detailed modeling and enable us to relax a lot of assumptions. Still, it must be considered that simulation is based on models, Moreover, understanding the system dynamics by brute force simulation can turn into searching for a needle in a haystack.
4. The design and dimensioning tasks become much more effective when one has a good intuition and solid expectations on the system behavior.
5. Making illustrative graphs of the performance results, as well as visualizing the data, can help a lot. Hence, it is worth spending a significant fraction of the time allowance for performance evaluation on this task.

1.4 Outline of the Book

In this section we first give a concise account of the content of the next chapters. Then we discuss possible uses of this book for courses and self-learning. Finally, we introduce definitions and notation of general use throughout the book.

1.4.1 Plan

The book comprises ten chapters besides this one, plus an Appendix.

Chapter 2 is devoted to service system definition, the role of queueing models, and general properties of queues in equilibrium. A service system is an abstraction of a physical or a logical element providing a function (the service) to users according to a given level of quality, as measured by relevant performance metrics. The service system is deemed to be out of service if the quality of service constraints are violated. In network traffic engineering, examples of service systems can be a router, a web server, a virtual machine, a protocol, a (sub-)network, or a data center. The structural elements of a service system are defined in Section 2.1, and service demand is characterized in Section 2.2. Section 2.6 is devoted to the formal definition of the traffic process. General properties of service systems in statistical equilibrium are addressed. First, the notion of stationarity of a random process is discussed. Then, Little's law is stated and proved. The probability distributions of the state seen at different event epochs of a service system are characterized (Palm's probability distributions). Finally, the most common performance key indicators or metrics are introduced in Section 2.7.

The material in Chapter 3 is functional to applications to the modeling of networked service systems. The Poisson process is first introduced and characterized. Generalizations of the Poisson process are considered, namely nonhomogeneous and spatial Poisson processes as well as the Markov modulated Poisson process. Then, renewal processes are treated. Some operations on renewal process are analyzed in detail, namely excess variables and superposition. Finally, two special classes of processes with many applications in network and service system analysis are introduced: birth-death and branching processes.

Chapter 4 aims at a comprehensive account of the analysis of single server queues. The $M/G/1$ queueing system is analyzed in depth, with specific attention devoted to numerical evaluation of the probability distribution and to finite waiting line systems. An asymptotic approximation of the loss probability of the $M/G/1/K$ queue is presented as the queue size K grows. An account is given also of the $G/M/1$ model and of extensions to matrix-geometric models of single server queues, specifically the queues described by a quasi-birth-death (QBD) process. The general result known as Reich's formula, holding for any single server queue, closes the chapter. More results for the general $G/G/1$ queue can be obtained only via approximations and are dealt with in Chapter 8.

Chapter 5 focuses on queueing models with multiple parallel servers. The general $G/G/m$ model does not yield to closed-form analysis. Then, we address special, yet relevant, cases, mostly based on Poisson arrivals. We consider both loss- and wait-oriented systems. The first category is represented by the Erlang model, i.e., the $M/G/m/0$ queue, where there is no wait. This is of primary importance in many practical applications, e.g., in modeling cellular networks. We give several examples thereof. Then, we consider the $M/M/m$ queue, that is completely tractable and allows a good insight into the working of multi-server queues. We use this model to discuss a classic problem of service system, namely the comparison between separate versus shared queues. Finally, we analyze infinite server models. In spite of their seemingly only theoretical relevance, they have highly useful applications. We discuss the application of an infinite server model to the analysis of message propagation in a line network.

Differentiated treatment of traffic flows and sharing of a communication or processing resource are key issues in telecommunication networks as well as in many other networked service systems (transportation, computing, energy distribution, to mention few of them). We leverage on results of single server queueing, specifically on the $M/G/1$ queue, to derive models of priority queueing systems in Chapter 6. We address head-of-line, shortest job first, shortest remaining processing time, and preemptive policies. We use those results to understand the basic trade-offs of service policy differentiation. Ultimately, from a traffic engineering perspective, we aim at characterizing the impact of introducing prioritized service classes on performance perceived by different customers. The second part of the chapter is devoted to scheduling. Major examples of scheduling are introduced and analyzed, namely processor sharing, weighted fair queueing, credit-based fair queueing, least attained service. A specific attention is payed to weighted fair queueing, that has laid the conceptual ground on which one of the major attempts of providing quality of service in the Internet has been founded. Finally, we review optimal queueing disciplines for different classes of queueing system, where the classification is based on what information can be exploited by the service policy.

Chapter 7 is intended to provide a solid introduction to the vast topic of queueing networks. There is a large body of literature on queueing networks, given both the fundamental theoretical interest of the model by itself and its numerous applications to communications networking, cloud computing, transportation systems, manufacturing, inventory and storage management. We address first the Jackson-type model of a queueing network, considering both open and closed queueing networks. The general theory is discussed in detail. Optimization problems are defined, as well as extensive examples of use of those models applied mostly to communication and transportation networks. The famous Braess paradox is discussed as a highly instructive warning on how even apparently simple queueing network models can turn out to be deceptive to intuition. In addition, we introduce loss networks, since they are a completely different model with respect to most other queueing networks, so that they deserve an ad hoc treatment. Here too we devote significant space to application examples of the model. Finally, we discuss the stability of queueing networks, a topic of growing interest.

In Chapter 8 we review basic results and approaches for obtaining approximate results with more general models than those for which exact solutions are available. First, we consider G/G queues, both single server and multi-server.

We derive bounds and approximations for the mean system time. We also introduce an asymptotic bound for the waiting time probability distribution and the Gaussian approximation based on the Brownian motion process. Approximate analysis of the mean system time is extended to network of G/G queues. We then cover the fluid approximation, both as an asymptotic description of a properly scaled process and as a continuous state approximation of discrete systems. Within this framework, we address also stochastic fluid models. As an application example, we apply the fluid model to the performance evaluation of a packet multiplexer loaded with intermittent (On-Off) traffic sources.

The last three chapters form the third part of the book, devoted to application of modeling and performance evaluation tools to three broad fields of networked service systems: multiple access, congestion control and quality of service guarantees.

Models for Slotted ALOHA and carrier-sense multiple access (CSMA) are introduced in Chapter 9 as an application of the tools defined in the previous parts. Selected topics are discussed out of the huge existing literature. The main target is twofold: (i) grasp how the general analysis tools of the previous chapters can be applied to a specific technical context; (ii) give concrete examples of how the working of a system can be understood and performance trade-offs characterized by means of a model. Under this respect, multiple access systems are one of the major examples of the potential of Markov chains. We first look at Slotted ALOHA, devoting special attention to the stabilization of the protocol. Pure ALOHA is considered as well, specifically in the general case of variable length packets, which leads to a new, nonclassic analysis. CSMA is then examined in detail. We consider models able to describe a general multi-packet reception setting. Stabilization is investigated as well. The remaining part of the chapter is devoted to the famous WiFi MAC protocol, the CSMA/CA. We derive the saturation throughput, access delay performance and give a thorough discussion of the drawbacks of the binary exponential back-off mechanism, advocating an alternative back-off adaptation algorithm, the so-called idle sense. Finally, the fairness issue of WiFi is discussed and evaluated.

Congestion control is among the most important topics in network traffic engineering. In Chapter 10, we address specifically congestion control in the Internet, even if the considered models can be applied to other contexts, abstracting from technology details. We address closed-loop congestion control as realized by the Transmission Control Protocol (TCP). First, general ideas and definitions are laid out. Then, several variants of the TCP congestion control algorithms are reviewed (classic TCP, CUBIC, Vegas, DCTCP, BBR). As for the models, the fluid approximation is used to gain insight into the dynamics of a TCP connection. First, a simple constant capacity single bottleneck scenario is considered. Then, a variable capacity model is introduced, thus showing the usefulness of the fluid approach and at the same time, identifying a resonance phenomenon of TCP congestion control with the time scale of the bottleneck time-varying capacity. We consider then fluid models of multiple TCP connections sharing a same fixed capacity bottleneck link. We review models for classic TCP (with a drop-tail buffer and with a buffer running an Active Queue Management algorithm) and models for DCTCP. The fairness concept is explored and a general framework is introduced, based on Network Utility Maximization (NUM). Besides giving a general approach to the definition of fairness, NUM allows revisiting TCP congestion control, interpreting the classic TCP operations as a distributed, iterative algorithm for the solution of a global network optimization problem, namely the maximization of a social utility function under link capacity constraints. Finally, we review the main traffic engineering issues that TCP faces in current networking practice, highlighting state-of-the-art approaches to solve them and open problems.

Chapter 11 is devoted to models of traffic sources, sharing a common network resource, under strict quality of service (QoS) requirements. This framework is apt for so called inelastic or inflexible traffic sources, that require their throughput and delay to lie in suitable ranges to provide an effective service. We consider first the deterministic traffic theory. It is based on nontrivial *deterministic* bounds that describe the traffic source behavior and the service provided by network elements. The main result it provides is a kind of "system theory" that allows us to give worst-case end-to-end performance bounds for networked service systems and to dimension network elements that guarantee a prescribed level of quality of service. The down side is that performance bounds can sometimes be quite loose. Moreover, the stochastic nature of traffic is not canceled altogether, since we need to analyze and dimension devices (the "traffic shapers") that enforce deterministic bounds on the stochastic traffic flows offered to the network. We move then to stochastic models of the multiplexing of inelastic traffic sources. Here we introduce the concept of effective bandwidth and give some major results on the relevant theory. We show how effective bandwidth can be used to analyze and dimension a network of service elements, exploiting the stochastic variability of the offered traffic to reap the so-called multiplexing gain.

Finally, a primer on probability, random variables, and stochastic processes is presented in the Appendix. It gives essential definitions and properties to ease the reader of this book that needs a quick reference to refresh its background of probability and Markov chains.

1.4.2 Use

This book is meant as an advanced textbook, suitable for senior undergraduate, graduate, and PhD students. It can be consulted also by those who need a solid introduction to performance evaluation in an applied context. It aims to provide a self-contained source text to cover traffic theory, queueing theory, and their application to networked service systems.

The objective of the book is to provide a comprehensive guide to these topics:

- What traffic is, and how it is characterized and applied to networked systems.
- The performance evaluation tools from queueing theory, used to model, analyze, and dimension service systems to which traffic is offered.
- Applications of performance evaluation tools to major aspects of networked systems, selected for their interest both from theoretical and application points of view: multiple access, congestion control, quality of service.

The book can be a primary reference for classes in engineering, computer science, data science, and statistics, for students desiring to gain understanding of fundamentals of performance evaluation as well as aiming at consolidating a basic knowledge with more advanced material. The book assumes a basic knowledge of probability (a concise refresher is provided in the Appendix, tailored to topics required in the book) and programming (any language will do; what matters is having firmly understood the logic of programming). To appreciate fully several examples of the book, it is useful to know the basics of TCP/IP networking and of communication systems (especially cellular and wireless ones). Most undergraduate students of computer, electrical, telecommunications, and industrial engineering and of computer science take classes on communication and networking at an introductory level, which is more than enough to understand application examples of this textbook. Occasional readers of this textbook could be found among graduate students of data science, transportation engineering, physics, mathematics, statistics as far as they need to manage, design and optimize service systems within their work. As a matter of example, transportation is a prolific field for application of queueing and traffic theory. It is not by chance that some examples presented in this textbook are drawn from transportation systems. Moreover, many scientists make extensive use of networking and computing facilities, which they often need to tailor to their special needs.

A set of exercises is proposed at the end of each chapter. They provide a self-test to assess the comprehension level of the subject of each chapter.

The book can be used modularly, given the dependencies among chapters, as depicted in Figure 1.7.

A directed arc between two chapters represents a major dependence. An arc labeled by section number means that the section of the source chapter is relevant to the target chapter. A label with two section numbers, connected by an arrow, means that the source chapter section is relevant for the target chapter section in the label. If the lecturer wishes to focus on the mathematical tools (with possibly some application examples), Parts I and II can be used. This way, a solid introduction to traffic and queueing theory is provided, moving from first principles

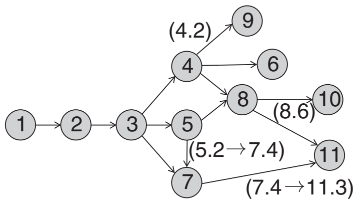

Figure 1.7 Dependencies among chapters.

and basic definition to rather advanced topics. If a short course is to be set up, besides the whole of Part I, Chapters 4 and 5 provide an introductory cornerstone to queueing theory (possibly skipping some more advanced topic, e.g., Sections 4.3, 4.7, 4.8, 5.5). According to available time, class interest and skills, the queueing theory core could be extended by considering advanced topics in Chapters 4 and 5, covering also priority and scheduling in Chapter 6 or generalizing stand-alone queueing systems to network of queues, dealt with in Chapter 7 or choosing topics from Chapter 8, to introduce bounds and approximations that go beyond queueing models yielding to mathematical analysis in closed form.

To offer a course leaning toward applications, Part I plus selected chapters from Part II can be sampled and then Part III can be covered. The minimum set of material to be covered in Part II in this case comprises Section 4.2 (the concept of embedded Markov chain and its application to a single server queue) and Section 8.6 (the fluid approximation). A richer selection of Part II chapters can be organized according to the guidelines mentioned above.

1.4.3 Notation

Scalar variables are denoted usually with plain letters (e.g., x), while vectors are denoted with small-capital boldface letters (e.g., $\mathbf{x}$) and matrices with capital boldface letters (e.g., $\mathbf{X}$). The identity matrix is denoted with $\mathbf{I}$, while $\mathbf{e}$ stands for a column vector of 1's.

Time is usually denoted with t. Hence, $Q(t)$ denotes a function of time. Space variables are usually denoted with x, y, and z.

The usual mathematical notation is used for the sets of integer and real numbers, $\mathbb{Z}$ and $\mathbb{R}$ respectively. The notation $\mathbb{R}^+$ ($\mathbb{Z}^+$) indicates the set of non-negative real (resp., integer) numbers.

We use sometimes the $o(\cdot)$ and $O(\cdot)$ notation. Writing $g(x) \sim o(f(x))$ for $x \to x_0$ means that $\lim_{x\to x_0} g(x)/f(x) = 0$. Instead $g(x) \sim O(f(x))$ for $x \to x_0$ means that the ratio $|g(x)/f(x)|$ remains bounded as $x \to x_0$.

A $-$ ($+$) subscript on a variable x corresponds to approaching x from the left (right). As an example, $f(x_-)$ stands for $\lim_{\epsilon\to 0} f(x - |\epsilon|)$. Analogously $f(x_+) = \lim_{\epsilon\to 0} f(x + |\epsilon|)$. For a continuous function, it is $f(x_+) = f(x_-)$, whereas a function for which the two limits are finite, but $f(x_+) \neq f(x_-)$, is said to have a jump at x.

The probability of event E is denoted with $\mathcal{P}(E)$. $I(E)$ denotes the indicator function of the event E: it is equal to 1 if and only if event E occurs.

Capital letters denote random variables, while sample values are usually denoted with a small capital letter, e.g., the random variable V that takes a specific value x is written as $V = x$.

The cumulative distribution function (CDF) of a random variable V is denoted with $F_V(x) \equiv \mathcal{P}(V \leq x)$, the complementary CDF (CCDF), sometimes also referred to as *survivor function*, with $G_V(x) = \mathcal{P}(V > x) = 1 - F_V(x)$. When existing, also the PDF is used and it is given by $f_V(x) = F'(x) = -G'(x)$.

The expectation operator associated with the random variable X is denoted with $\mathrm{E}_X[\cdot]$. The subscript X is dropped unless it is necessary to avoid ambiguity. Given a function $g : D \mapsto \mathbb{R}$, where D is the domain of the random variable X, it is $\mathrm{E}[g(X)] = \int_D g(x) dF_X(x)$. Specifically, the mean value of X is denoted with $\mathrm{E}[X]$. The variance is denoted with $\sigma_X^2 = \mathrm{E}[X^2] - (\mathrm{E}[X])^2$. Sometimes the notation $Var(X)$ is used for the variance of the random variable X. The coefficient of variation (COV) of the random variable X is defined as $C_X = \sigma_X / \mathrm{E}[X]$. Also the squared COV (SCOV) C_X^2 is used.

The Laplace transform of the PDF of a non-negative random variable V is denoted with $\varphi_V(s)$, and it can be calculated from $\varphi_V(s) = \mathrm{E}[e^{-sV}] = \int_{0_-}^{\infty} f_V(x) e^{-sx}\, dx$.

For random variables defined over the entire real axis, we define the moment generating function (MGF) $\phi_V(\theta) = \mathrm{E}[e^{\theta V}] = \int_{-\infty}^{\infty} f_V(x) e^{x\theta}\, dx$.

As for discrete random variables, the CDF, CCDF, and probability distribution of a discrete random variable N are denoted with $F_N(k) = \mathcal{P}(N \leq k)$, $G_N(k) = \mathcal{P}(N > k)$ and $p_N(k) = \mathcal{P}(N = k)$ for $k \in \mathbb{Z}$. Note that in the discrete case the equality sign in the definition of the CDF is important.

The MGF of a non-negative discrete random variable N is denoted with $\phi_N(z)$. It can be calculated from $\phi_N(z) = \mathrm{E}[z^N] = \sum_{k=0}^{\infty} p_N(k) z^k$.

The notation $A \sim B$ means that the random variables A and B have the same probability distribution. The notation $Z \sim \mathcal{N}(a, b)$ means that Z is a Gaussian random variable with mean a and variance b, while $Z \sim \mathcal{U}(a, b)$ denotes a random variable Z uniformly distributed in the interval $[a, b]$. A random variable Z with negative exponential probability distribution and mean a is denoted with $Z \sim Exp(a)$.

Measure units follow the scientific International System, i.e., meters, seconds and multiples thereof. The symbols and values of multiples and sub-multiples of measure units used in this text are listed in Table 1.1 for reader's ease.

A list of acronyms can be found at the beginning of the book.

Table 1.1 Symbols of multipliers and sub-multipliers of measure units.

Symbol	Name	Value	Symbol	Name	Value
k	kilo	10^3	m	milli	10^{-3}
M	Mega	10^6	μ	micro	10^{-6}
G	Giga	10^9	n	nano	10^{-9}
T	Tera	10^{12}	p	pico	10^{-12}
P	Peta	10^{15}	f	femto	10^{-15}

1.5 Further Readings

Many books can be consulted to integrate the material of this book or to provide in-depth follow-ups.

A first group of references addresses applied probability and queueing theory [54, 86, 94, 121, 130, 131, 196]. This list contains textbooks biased toward or definitely devoted to queueing theory. While being excellent sources for learning on queues, they are not concerned with any specific application. [94, 130, 131] are classic introductory textbooks on queueing theory. [86] is a more recent textbook on queueing theory and stochastic networks, with some emphasis on fluid approximations. [54, 121] are mainly monographs on stochastic and queueing networks. The comprehensive textbook [196] provides an excellent and thorough introduction that covers everything from probability, to Markov chains, queues, and simulation.

Another group of books, [30, 58, 98, 122, 137, 139, 193], leans more toward applications to networked service systems, often in the realm of information and communications technologies. [30] is a classic textbook on communication network performance evaluation. It represents one of the first examples of a perfect mix of technological aspects coupled with rigorous modeling, analysis and dimensioning approaches, based on queueing theory and Markov chains. A much more recent attempt to introduce modeling and performance analysis approaches starting from real-life problems in offered by [58], with specific reference to social networks and communication networks. It uses a wide range of mathematical tools, mostly at an introductory level, and does not give any account of queueing theory, except of elementary notions. The ponderous book of Kumar et al. [137] overviews models and performance results for all aspects of communication networks (multiplexing, switching, routing). It assumes that the reader is already familiar with the required basic theory. It focuses only on performance models. [98] gives a full introductory account of queueing theory, constantly coupling theory and application to computer networks. The book is mostly at an introductory level. [122] is a very nice and concise book on rather advanced modeling, mostly applied to communication and networking problems. This deep book is a neat example of a smart balance between presenting methodological tools and applying them to technical systems, even if examples are rather at high level and do not touch many technical details of real systems. [139] is mainly an introductory level queueing theory book, the last chapter overviewing applications to communication networks. The recent book [193] applies optimization, game and control theories to modeling, analysis and dimensioning of communication networks.

Finally, to expand fundamental theories on probability and Markov chains, the reader might refer to the following classic textbooks, being advised that many other excellent books can be found: for probability theory [76, 90]; for Markov chains and stochastic processes [172, 117, 60, 40]. A concise and rigorous introduction to statistics can be found in Part II of [87], while statistics applied to performance evaluation is presented in [145].

Since networks and networked system have been mentioned several times, it is worth spending a word on a terminology clarification. Here “network” is meant to be a technological network, i.e., the interconnection of service systems (either physical or logical), set up to support a class of applications, e.g., telecommunications, computing, transportation, energy distribution, logistics and inventory, industrial production. The word *network* is also meant sometimes to address graph-based models of interconnected entities. This is more precisely referred to as *network science*. Two reference textbooks on the subject are those by Albert László Barabási [24] and by Mark E. J. Newman [169]. While graphs are a widely used model for any network, the focus of network science is on understanding the properties of graphs and what they mean to the specific “environment” being modeled.

The focus of this book is to apply rigorous mathematical methodology to the performance evaluation, design, and optimization of technological networks. This is essentially the difference between science and engineering (or, more broadly, between fundamental and applied science). The former is concerned with modeling reality to understand it; the second aims at understanding in order to act on reality.

Problems

1.1 Let an audiovisual (AV) frame have a constant length F bigger than the maximum packet payload L, so that each AV frame requires $m = \lceil F/L \rceil$ packet to be conveyed to the destination. Play-out of an AV frame requires all packets carrying the frame information to have been received in due time. Generalize the analysis of the delay equalization buffer of this chapter to the multi-packet per frame case.

1.2 Generalize further the model of Problem 1.1 by letting the number M of packets (each having a fixed length L) per AV frame be a random variable with probability distribution $q_m \equiv \mathcal{P}(M = m),\ m \geq 1$. Give an expression of the starvation probability for an assigned PDF of the network delay $f_D(x)$, under the same assumptions as in Section 1.3.1.