1

Introduction

1.1 What is a Communication Network?

We are surrounded by communication networks, they are part of our life. If asked, we can easily enumerate examples of them: the fixed or mobile telephone network, the Internet, or someone's Ethernet home network would probably be the most popular answers. However, difficulties appear when we try to define the concept "communication network" more formally, without mentioning any specific network technology, looking for a definition applicable to any of them. We will start this book addressing precisely this basic question.

There are two basic elements on which networks are constructed: *telecommunication systems* and *switching systems*. A telecommunication system or "link" consists of a transmitter and one or more receivers connected through a medium that propagates the involved electromagnetic signals. Applying this definition, two telephones A and B directly connected through a bidirectional cable pair (Fig. 1.1) contain two telecommunication systems: (i) one composed of the transmitter at A, the medium A → B, and the receptor at B, and (ii) another system formed by the transmitter at B, the medium B → A and the receiver at A.

Telecommunication systems are the basis for assembling any communications service. However, no *service* can be reasonably built by pure aggregation of telecommunication systems. As an example, imagine we want to provide the telephone service to four users A, B, C, and D, using just "links". To do that, we would need two telephones and one cable *dedicated* for each different *user pair*. The result would be something like Fig. 1.2.

The previous example illustrates that, although possible, it is not *economically feasible* to provide a communication service using just links. Inefficiency in Fig. 1.2 appears because each link is *dedicated* exclusively to a particular user pair, and is thus idle when corresponding users are not talking each other. Improving the efficiency in our example requires adding new elements to the picture that permits a link to be *shared* among several communications. And that is precisely where switching systems come into play.

A switching system or "node" is a device that connects telecommunication systems (links) among them, so that the information from one link can be forwarded to other. We can represent a switching system with a node with N_{in} input ports and N_{out} output ports, as shown in

Optimization of Computer Networks – Modeling and Algorithms: A Hands-On Approach,
First Edition. Pablo Pavón Mariño.
© 2016 John Wiley & Sons, Ltd. Published 2016 by John Wiley & Sons, Ltd.
Companion Website: www.wiley.com/go/PavonMarinoSol16

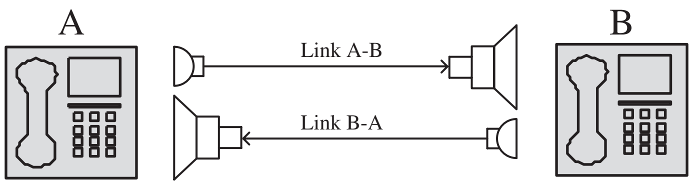

Figure 1.1 Two telecommunication systems

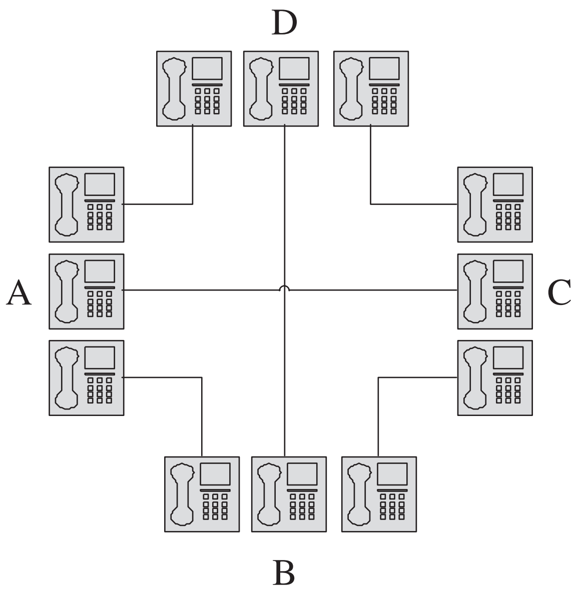

Figure 1.2 Communication service built with dedicated telecommunication systems

Fig. 1.3a. Each input port represents the receiver side of a link, each output port represents the transmitter side of other. The state of the switching system at a given moment is defined by the particular scheme in which input and output ports are internally connected. For instance, Fig. 1.3b represents a node configuration where information from input port 1 is forwarded to output port 2, while input port 3 is connected to output port 1.

The defining aspect to make a system like the one in Fig. 1.3 a switching system, is that it must be *reconfigurable*. That is, using mechanical, electronic, optical, or any other physical procedure, it should be possible to rearrange the internal connections between input and output ports, so that an outgoing link can be carrying at different moments the information received from different input links. Reconfigurability is the enabling feature to make output links become a *shared resource* among the input links.

Figure 1.4 helps us to illustrate how a combination of nodes an links supports a telephone service to seven users, using four nodes and seven (bidirectional) links. Naturally, a network like the one shown, requires extra elements to enable end-to-end communications. First,

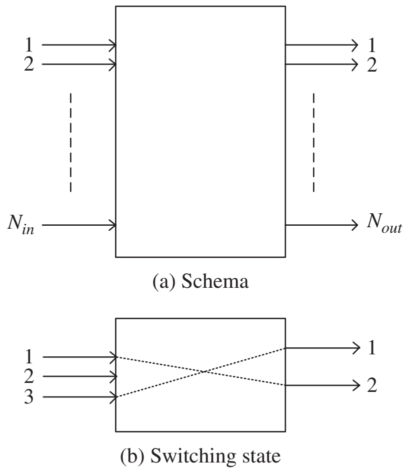

Figure 1.3 Switching system (node)

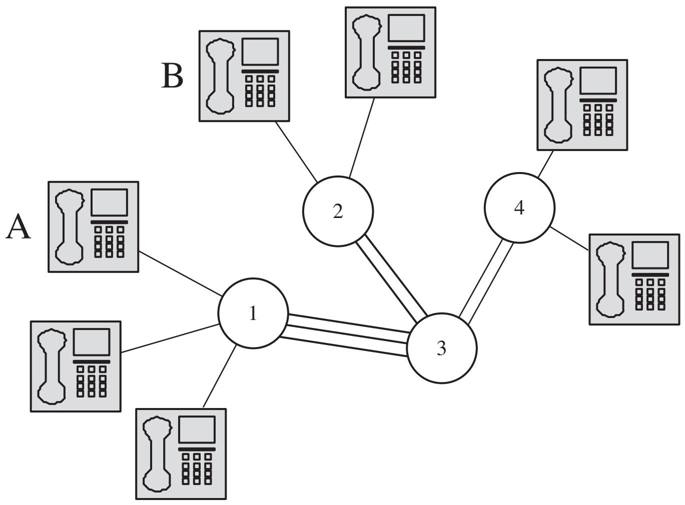

Figure 1.4 Communication network example

telephones become now a more sophisticated device, since a single telephone can be used in Fig. 1.4 to communicate with any other telephone. They are actually the *sources* and *destinations* of information. An addressing or numbering scheme is now required to identify each telephone individually, so that each user can decide the destination telephone to call to. Second, a decision on the sequence of links (route) to be traversed by each telephone call should be taken and signaled to the switching nodes that must reconfigure accordingly.

The previous network scheme should be enriched with the concept of *multiplexed links*. Previous examples have shown links as elements that can either carry one single telephone call, or otherwise be idle. In its turn, a multiplexed link is able to simultaneously carry several communications, so that the aggregated traffic can be de-aggregated again at the link receiver side. Many multiplexing technologies exist, according to the physical form in which the aggregation/deaggregation takes place, the most frequent being frequency multiplexing, time multiplexing, code multiplexing, or a combination of them. The particular multiplexing technique has no importance for the abstract network model we are pursuing. What is important, to capture the essence of multiplexing, is that links should be characterized by a *link capacity*, measured in arbitrary units (e.g., bits-per-second – bps – in digital links). In turn, sources should now be characterized as producers of traffic measured in the same units as the link capacities. Eventually, the link capacities become the shared resource, so that we will be able to compare the capacity of a link with the sum of the traffic of the sources traversing it.

Put together, we have identified four elements in a communication network: (i) information sources and destinations, (ii) links capable of propagating information between their end points, (iii) switching nodes capable of interconnecting links in a reconfigurable form, and (iv) addressing and signaling systems for controlling network operation.

1.2 Capturing the Random User Behavior

A characteristic aspect of communication networks is that they have to deal with the *random nature* of the user behavior. In Fig. 1.4 this means that we do not know beforehand who each user is willing to talk to at each moment. Random behavior is crucial for the dimensioning of resources such as link capacities in the network.

In Fig. 1.4, capacities are set so that two idle users will always be able to call each other, irrespective of what other users are doing. This is called a worst-case network dimensioning. However, this approach is clearly economically unfeasible for nontrivial scenarios. Just imagine that in Fig. 1.4 10,000 phones were connected to nodes 1 and 2. A worst-case dimensioning would need a capacity of 10,000 calls for links 1-3 and 2-3. These links would be clearly underutilized, since the probability of the two sets of 10,000 users using the phone at the same time is small.

For this reason, communication network design has been historically based on probabilistic models that characterize the random user behavior. A statistical characterization of the performances observed by the users, permits dimensioning the network resources so that the service degradation (or the probability of this to happen) becomes small enough. The statistical metrics of interest can be quite variable. However, two main alternatives appear corresponding to the two main strategies network apply for link capacity sharing: *delay systems* and *loss systems*:

- **Delay systems**. Delay systems typically correspond to the form in which *packet switching networks* operate. In packet switching networks, traffic sources split the information into fragments called packets. Each packet is attached to a header, with sufficient control information to permit the packet reach its destination. Nodes process incoming packets one by one and forward them to their corresponding output link. Link capacities are dimensioned so that the *average* flow of packets that traverses a link does not exceed its capacity, potentially with some safety margins. However, the random nature of packet arrival times makes that packets could find their output links busy with other packets. For this reason, nodes incorporate memories for storing packets waiting for their turn to be transmitted. The storage time or queue time is an added delay to the end-to-end communication. Moreover, traffic fluctuations can fill up these memories and force the nodes to discard packets. In these systems, the delay and the packet loss probabilities are the performance metrics of interest.

- **Loss systems.** Loss systems can appear in networks where the traffic takes the form of end-to-end connections. These connections can be circuits, in so-called *circuit switching networks* (e.g., telephone calls in the telephone network, lightpaths in WDM optical networks), or virtual circuits over packet switching networks (e.g., virtual circuits in MPLS networks). Each connection reserves a given amount of capacity in each of the traversed links. The reserved bandwidth is kept throughout the communication. If a connection request does not find a sequence of links with enough capacity, it is discarded (*lost* or *blocked*). The probability that a new connection request is blocked, or *blocking probability*, is the main performance metric for dimensioning loss systems.

1.3 Queueing Theory and Optimization Theory

Queueing Theory has been traditionally the mathematical corpus supporting the design of computer or communication networks, capturing the non-deterministic user behavior and network occupation. This theory focuses on systems where a set of resources (e.g., links in a network) are shared among different users, so that the moment in which each user requests a resource (e.g., a new connection or a packet to transmit), and/or the time in which the resource is to be occupied (connection/packet duration), are known statistically. As such, Queueing Theory is a branch of Applied Probability.

Queueing Theory has been extremely successful and elegant for analyzing network subsystems such as the performances of the traffic traversing a link or a switching node. However, applying this strategy to a network view is extraordinary challenging. Actually, the probability models used in Queueing Theory become intractable when applied to non-trivial communication networks, unless strong simplifying assumptions are made. The essential disadvantage is that the exact statistical characterization of the traffic traversing several links and/or merging with other users traffic, can be mathematically intractable. For these situations, the combination of Queuing Theory and Network Optimization has shown to be a successful approach.

Optimization is a branch of Applied Mathematics, studying the maximization or minimization of functions, subject or not to a set of constraints. Network optimization is just the application of standard general optimization results to network design problems, exploiting any special structure they may have. Decision variables in these problems take the form of, for example, the capacity to assign to each link or the amount of a user traffic to route in each valid path.

The application of network optimization requires getting rid of the unreachable goal of a full statistical characterization of the traffic in the links. In its turn, in most of the cases (as happens in this book), the traffic in network design problems is modeled as a continuous *flow*, not identifying individual connections or packets in it. Flows are simply characterized by a real number: its *average intensity*, measured in bits-per-second, number of active connections, or any suitable unit. Characterizing a traffic flow just by its average yields a significant simplification: *the intensity of the traffic in a link aggregating the contributions of several flows, is just the sum of the intensities of the traversing flows*.

Previous simplification, opens the door to introducing into the optimization models expressions coming from Queuing Theory that estimate performance metrics like delay or blocking probability, as a function of *average* flow intensities. In general, these relations are closed formulas that assume as constants other parameters (e.g., variance or Hurst parameter of the traffic flows), and result in simple expressions. Thanks to that, these network performance estimations can be introduced into the optimization model as objective functions (e.g., minimize the average network delay), or design constraints (e.g., blocking probabilities should be below 1%), without significantly augmenting the problem complexity.

1.4 The Rationale and Organization of this Book

The casuistic of the different network design problems that can exist is unlimited, as the multitude of technologies, protocols and heterogeneous network conditions yield to an unnumbered amount of variants. Since it is impossible to study them all, the target of this book is providing a methodology that can be applied whatever network technology we focus on, and eventually results in (i) insights to understand the network behaviors and (ii) design keys to produce algorithms that solve network problems.

The mathematical corpus of our approach is network optimization, or the application of general optimization theory to network problems. In this strategy, we distinguish two main steps, elaborated into the two parts (Part I and Part II) in which this book is divided: problem modeling and algorithm design. This is followed by a small set of appendices covering basic optimization concepts used throughout the book.

1.4.1 Part I: Modeling

In this part, we pursue the *modeling* of network design problems appearing in communication networks, as *optimization problems*. This means translating them into the problem of finding the values of a vector x of *decision variables* that maximize or minimize an *objective function*, subject to some *constraints*. The reasoning of the approach is that, once a problem is modeled, we can benefit from optimization theory to characterize what their optimum solutions look like and how to reach them.

For didactic purposes, four problem types are addressed separately: (i) traffic routing, (ii) link capacity dimensioning, (iii) bandwidth sharing among network users (i.e., congestion control), and (iv) topology design. These problems appear in different forms in all network technologies. In this part of the book, multiple variants of each are described and analyzed, covering most of the aspects appearing in production networks. The source code for Net2Plan tool of the examples and selected exercises is available for the reader. Then, the he/she will be able to find numerical solutions to these problems in any network instance, using the solvers interfacing with Net2Plan.

Part I chapter organization is described below:

- *Chapter* 2. *Definitions and notation*. Key definitions and the notation that will be used along the book is presented. We elaborate on concepts like network links, nodes, traffic demands (unicast, anycast, multicast, broadcast), and routing (bifurcated, non-bifurcated, integral).
- *Chapter* 3. *Performance metrics*. This chapter describes different performance metrics that are commonly used within network optimization models: delay and blocking probabilities in packet-switched and circuit-switched networks, respectively, average number of hops, network congestion, network cost, network availability in failure scenarios, and fairness

among competing entities in the allocation of shared resources. We pursue, when possible, simplified estimations of these metrics that can be introduced in network design models. In such case, their convexity properties are analyzed, since this is a property that can be exploited when designing optimization algorithms.

- *Chapter* 4. *Traffic routing*. The routing problem consists of deciding the sequence of links that traverse each user's traffic. Three main modeling strategies are addressed: flow-path and flow-link formulations, for flow-based routing (e.g., suitable for MPLS, SDH, WDM networks), and destination-link formulation for destination-based routing (e.g., IP, Ethernet). Main routing problem variants are presented, including routing in anycast and multicast traffic, bifurcated/non-bifurcated routing, routing under protection or restoration schemes, and multi-hour routing.
- *Chapter* 5. *Link capacity dimensioning*. This chapter covers the modeling of the capacity assignment problem in two typical contexts: (i) the capacity assignment, as an offline problem to be solved every, for example, 6 months for updating the capacities in network core links, (ii) capacity assignment as a network control problem, for example, solved at a sub-second pace in wireless networks where each node optimizes its rate to adapt to varying medium and traffic conditions, affecting and being affected by the interferences from neighbor nodes. In offline capacity planning, one of the difficulties is handling the concavity of the link cost with respect to its capacity. We present a case study for which a closed formula is reached to characterize the optimum and give insight on different trade-offs appearing in network design, in particular, between complete and hub- and -spoke topologies. For the wireless case, we first model the medium-access control problems in random-access and in CSMA-based networks (like Wi-Fi). Then, we focus on a network where the link capacities are adjusted by tuning the transmission power in the antennas.
- *Chapter* 6. *Congestion control*. This chapter covers the modeling of the classical network congestion or bandwidth sharing problem, consisting of assigning the amount of traffic to be injected by each traffic demand without oversubscribing the links. Typically, network congestion control is implemented in a decentralized form, by rate adjustment schemes in the traffic sources. We present here a popular model based on the Network Utility Maximization (NUM) framework, where each demand is associated to an utility function. It is shown how fairness in congestion control is related to the concave shape of the utility function, so that by choosing different functions it is possible to enforce different fairness notions. We then concentrate on NUM modeling of the two main versions of TCP protocol (Reno and Vegas), showing how, despite its decentralized nature, it enforces a fair distribution of the link bandwidth among the competing connections. The role of NUM modeling to understand the behavior of active queue management (AQM) techniques is also illustrated.
- *Chapter* 7. *Topology design*. This chapter covers the modeling of topology design problems. First, we address the classical node location problem variants, where the position of the network nodes is optimized. Then, we include some topology design variants where the network links are a part of the problem output. Case studies and numerical examples are provided to illustrate some of the trade-offs in topology design.

1.4.2 Part II: Algorithms

This part targets the development of algorithms to solve network problems like the ones modeled in Part I. In this task, we emphasize the importance of the *algorithm context*, describing the scenario where the algorithm should be implemented. This comprises relevant information like the scale of the network targeted, the time and computing resources available, or the acceptable accuracy. Actually, *good* algorithms in a particular context, can be *bad* or simply inapplicable schemes in other contexts.

In this book, we clearly distinguish two families of algorithms: those targeted to be implemented by distributed protocols and those that are amenable to a centralized execution:

- *Distributed algorithms*. In multiple scenarios it is not possible to communicate to a centralized server the full network picture so that it can run the optimization algorithm. These problems should be solved in a *distributed* fashion by properly designed network protocols, where nodes cooperate and exchange signaling information such that they can iteratively and autonomously adjust its configuration. An example of distributed implementation is congestion control schemes, where traffic sources adjust their rates autonomously and asynchronously receiving a limited signaling information from their traversed links (or estimating it implicitly, e.g., as in TCP, monitoring its packet losses or round-trip-time delays).
- *Centralized algorithms*. A centralized algorithm execution is possible when a central computer can be fed with the full network knowledge required to solve the problem. Centralized algorithms are typical, for example, of those design problems executed offline by network planning departments. For instance, the upgrade plans for the link capacities or the placement of new links for the upcoming year, in order to cope with a forecasted traffic demand. Once the current network topology and traffic forecasts are collected, these algorithms can typically run for hours or even days until an acceptable solution is found.

This distinction between distributed and centralized executions is not gratuitous, since the mathematical approach is different for both. In this book, the theoretical corpus for the design of distributed algorithms is the application of the standard gradient or subgradient projection iterative methods to the primal or the dual version of the network problems. Convergence guarantees are strong in many situations, as long as the optimization problem is convex. In its turn, many offline planning problems to be solved by a centralized algorithm are non-convex and with $\mathcal{NP}$-hard complexity. For them, heuristic and approximation algorithms are needed.

In Part II of the book, we guide the reader in the design of distributed and centralized network algorithms. The source code for Net2Plan tool of the case studies, examples, and exercises is available. In particular, the reader will be able to easily repeat the experiments in the book, or extend them to other network instances, or different contexts in terms of signaling delays or losses.

The organization of the material is described below:

- *Chapter* 8. *Gradient algorithms in network design*. This chapter initiates the reader in the basic but necessary theory related to the standard gradient and subgradient schemes. Their convergence properties are also summarized when they are executed in a distributed form by separated entities that operate asynchronously, potentially using outdated information; for example, because of signaling delays or affected by noisy observations of the network. Techniques and hints for dimensioning the step size are described and later applied in multiple examples in the subsequent chapters.

- *Chapter* 9. *Primal gradient algorithms*. In this chapter we present methods based on the application of the gradient projection iteration to the primal (original) network problem. First, we present penalty techniques to deal with non-separable feasible sets, for which the projection operation is not easy. Then, we apply the primal methodology in several case studies providing distributed primal algorithms for adaptive routing, congestion control, persistence probability adjustment in MAC protocols and transmission power assignment in wireless networks. In each case, the convergence is empirically tested and connected to the theoretical convergence guidelines, under asynchronous executions, handling delayed and noisy observations, and applying convergence improving techniques described in Chapter 8.
- *Chapter* 10. *Dual gradient algorithms*. In this chapter we elaborate on the application of gradient projection iterations to the dual version of network problems. Convergence properties of dual methods are summarized, emphasizing the importance of the strict convexity/concavity of the objective function, and the role of problem regularization when this does not hold. The dual approach is exemplified in several case studies, providing distributed dual algorithms for adaptive routing, backpressure (center-free) routing, congestion control, and decentralized optimization of CSMA window sizes. Empirical convergence tests are provided under realistic asynchronous executions, with delayed and noisy observations.
- *Chapter* 11. *Decomposition techniques*. This chapter presents a framework for applying decomposition techniques to network problems, splitting them into simpler subproblems that can be solved independently, but coordinated by a master program. First, we describe the two baseline decomposition approaches: primal and dual decomposition, together with some reformulation techniques. Then, these strategies are applied in selected examples. We start putting the emphasis on problem decomposition as a theoretical support for so-called *cross-layer algorithms* that make protocols at different network layers cooperate to achieve a common goal. In this case, each network layer is a subproblem, which is solved by a different protocol (potentially itself a distributed protocol), and the master program defines the signaling to coordinate the layers. Three examples with empirical tests are provided: cross-layer congestion control and capacity allocation to traffic sources with different QoS requirements, cross-layer congestion control and backpressure routing, and cross-layer congestion control and power allocation in wireless networks.

  Afterward, we use decomposition techniques to coordinate different agents in a single-layer problem. We present a case study where multiple interconnected network carriers can cooperate in an asynchronous form to globally optimize the routing, for example in the Internet, without disclosing sensitive information like their internal topologies and traffics. Finally, we also include a case study for a $\mathcal{NP}$-hard joint capacity and routing design problem to be solved *offline*. In this case, the target of the decomposition is reducing the overall computational complexity and also permitting parallel executions of the subproblems at a cost of losing some convergence properties. Illustrative numerical tests are included for all the examples.
- *Chapter* 12. *Heuristic algorithms*. This chapter focuses on the design of heuristic algorithms for solving large instances of $\mathcal{NP}$-hard network planning problems in offline and centralized executions. The main heuristic techniques are described in a didactic form: local search, simulated annealing (SAN), tabu search (TS), greedy algorithms, GRASP, ant colony optimization, and evolutionary algorithms. Recommendations for parameter tuning are provided. Each scheme is illustrated by applying it to a common traffic engineering

example: finding the OSPF link weights in IP networks that minimize congestion. Then, a realistic network planning full case study is described where a backbone optical network is optimized for three different protection and restoration schemes. This exemplifies an algorithm where a heuristic scheme is used to guide the search and control the complexity of small ILP formulations.

1.4.3 Basic Optimization Requisites: Appendices I, II, and III

The defining rationale of this book is the systematic application to network problems of selected optimization theory results. The fact that a relatively simple mathematical corpus can be exploited to provide insight on complex network interactions, and guide the design of successful network algorithms, is by all means a strength of this approach.

Still, queuing theory alone has been often the spine of network courses, leaving optimization basics for separated operations research subjects. The fruitful application of optimization to network design along the last decades has changed this inertia, and optimization theory is gaining more and more momentum in the computer networks curricula.

Appendices I, II, and III in this book are motivated by this. As a whole, their target is providing the reader not familiar with optimization theory, the necessary results repeatedly applied throughout the book. Despite being appendices, the material is organized such that it can be used as an introductory part in those computer network courses that prefer to integrate this theory. This is the approach I make in my lectures.

A short description of these appendices follows:

- *Appendix I. Convex sets. Convex functions*. Convexity is a keystone concept in optimization, which helps to later characterize the frontier between those problems that are *easy* to solve (solvable in polynomial time) from those that are not. This appendix defines and summarize the key properties of convex sets, and of convex and concave functions.
- *Appendix II. Mathematical optimization basics*. This appendix introduces optimization theory in a simple and didactic form, suitable for beginners. Optimization problems are first classified according to classical taxonomies, emphasizing the role of convex optimization. The dual of an optimization problem is defined and their main properties exposed. Then, we present and provide a graphical interpretation of Karush–Kunt–Tucker (KKT) optimality conditions for problems with a strong duality property, like the convex network problems appearing throughout the book. KKT conditions are a simple but powerful tool recurrently applied to characterize the optimum of these problems.
- *Appendix III. Complexity theory basics*. Complexity theory helps us to assess the computational complexity of problems and algorithms solving them. For problems, we define the concept of deterministic and non-deterministic machines that helps us to define $\mathcal{P}$ and $\mathcal{NP}$ problem complexity classes. For algorithms, we define algorithm complexity and emphasize the practical difference between polynomial and non-polynomial algorithms. Then, we define $\mathcal{NP}$-complete complexity class and discuss its importance in network design: $\mathcal{NP}$-complete problems cannot be solved to optimality for real network sizes. Then, we elaborate on problems for which it is possible to find algorithms with approximation guarantees ($\mathcal{APX}$ class and $\mathcal{PTAS}$ and $\mathcal{FPTAS}$ approximation schemes), and problems for which it is conjectured that no approximation algorithm exists ($\mathcal{NPO}$-complete problems).

1.4.4 Net2Plan Tool: Appendix IV

This appendix briefly presents Net2Plan, an open-source freeware Java-based software application for network optimization and planning, and its role as a supporting tool in the hands-on approach pursued. The book does not rely on, but is well supported by, Net2Plan for demonstrating competency and practice. The mathematical formulations and algorithms in the book are implemented as Net2Plan algorithms. They are available, indexed and documented, for the interested readers in the website www.net2plan.com/ocn-book. Then, the user will be able to see how the ideas and mathematical approaches take real form in algorithms, reuse, or execute them to obtain numerical solutions, and end up developing his/her own algorithms.