



Network Design Problems – Notation and Illustrations

In Chapter 1, we have presented basic ideas about communication and computer network design problems that a network provider is likely to face. In this chapter, we illustrate a representative set of network design problems through simple numerical examples to help understand different aspects of network design. We will consider only a sampling of design problems here; the later chapters will cover many more models, including in-depth variations and extensions of the samples presented in this chapter.

As you will see, network design problems can be formally formulated using mathematical notations. The advantage of a good mathematical notation is that it can represent a specific design problem in a compact and unambiguous way, and moreover, it eventually helps to understand the problem at hand. In the process, we will show how the same problem can be represented in different ways, discussing advantages and disadvantages of notations; this will lead us to the notation used throughout this book. It is important to understand how we use the notation for different problems as it is essential in understanding different design models covered in this book.

Thus, the purpose of this chapter is basically two-fold: 1) to introduce the mathematical notation used throughout this book for network design problems; and 2) to illustrate the network design problem formulations through a sample of simple numerical examples.

The first three sections, 2.1 through 2.3, of this chapter are the basics on how to formulate a problem for a three-node network example, and then discussing merits and demerits of different notations. This is also where terms such as node-link and link-path formulation are introduced. The experienced reader may skip these three sections. In Section 2.4, we consider a four-node problem and illustrate the network dimensioning problem – this is also where we introduce the formal notation to be used in the rest of the book. In Section 2.5, we discuss shortest-path routing and related network design by considering the same four-node example. Shortest-path routing is important since it is used for packet routing in the Internet. Afterwards, we illustrate fair networks in Section 2.6. Another important problem is topological design – this is illustrated in Section 2.7. We then consider network restoration design (Section 2.8) that addresses failure situations such as a fiber cable cut, and show how the basic notation can be extended to consider restoration design as well. As discussed in Section 1.6 of Chapter 1, an important issue is multi-layer network modeling. Thus, we present a detailed illustration of multi-layer networks in Section 2.9 so that you can understand what extensions are required to be introduced when more than one layer of

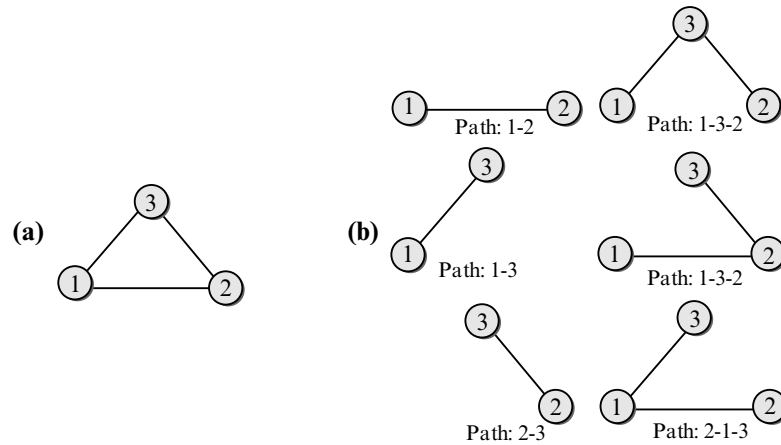


FIGURE 2.1 (a) Three-Node Network Example. (b) All Possible Paths for the Three-Node Example

resources is involved, and how the modeling works. Despite its importance, this section may be skipped in first reading; the reader should come back to it before going into Chapter 12.

2.1 A NETWORK FLOW EXAMPLE IN LINK-PATH FORMULATION

We will start with a simple network that consists of three nodes where each node is connected to the other two nodes, i.e., the network topology looks like a triangle (see Figure 2.1a). In this example, nodes can be routers in the Internet, telephone switches in the telephone network, or digital cross-connects in the SONET network. *Node* is a good generic term to identify different types of routing or switching devices in a network, and we will use this name throughout the book. *Demand volume* represents either the traffic volume (as in the Internet or the telephone network) or the required bandwidth (as in SONET) between a pair of nodes, depending on the considered type of network. Such a pair of nodes is called a *demand pair*, or simply *demand*. Suppose that the demand volume between nodes 1 and 2 is 5, between nodes 1 and 3 is 7, and between nodes 2 and 3 is 8 (units). Note that the demand is assumed to be bi-directional (undirected), given as “between” rather than “from-to”. This is to keep this example simple as we assume here that the links are also undirected (in general the demand and/or links can be directed, i.e., uni-directional). We will use \hat{h} to identify the demand volume:

$$\hat{h}_{12} = 5, \quad \hat{h}_{13} = 7, \quad \hat{h}_{23} = 8.$$

As you can see, subscripts with \hat{h} are used for identifying the associated end nodes of demands. While the use of “hat” ($\hat{}$) notation may seem a bit odd, its appropriateness will become clear as we go through this chapter, especially by the end of Section 2.3.

The demand volume for a pair of nodes can possibly be routed over two paths in this three-node network. For example, for the demand pair with end nodes 1 and 2 (to be denoted

as $\langle 1,2 \rangle$, its demand volume can be routed over the direct-link route 1-2, and the alternate route 1-3-2 via node 3 (Figure 2.1b). How much of the demand volume will be routed on each path really depends on the network design objective (we will discuss this later in this section). So, if we use \hat{x} with an appropriate subscript identifier to denote the unknown *demand path-flow variables* (flow variables, or flows, in short), then for demand pair $\langle 1,2 \rangle$, we can write:

$$\hat{x}_{12} + \hat{x}_{132} = 5 \quad (= \hat{h}_{12}).$$

Note that we have used subscripts with \hat{x} variables to identify the route or the path; in this case, paths 1-2, and 1-3-2. The summation on the left hand of the above equation holds since the demand flow allocated to different paths for a demand pair adds up to the total demand volume for that pair. Similarly, for demand pairs $\langle 1,3 \rangle$ and $\langle 2,3 \rangle$, respectively, we can write the following equations after identifying the possible paths (Figure 2.1b):

$$\hat{x}_{13} + \hat{x}_{123} = 7 \quad (= \hat{h}_{13})$$

$$\hat{x}_{23} + \hat{x}_{213} = 8 \quad (= \hat{h}_{23}).$$

Note that path-flows are non-negative, i.e., $\hat{x} \geq 0$ for all paths. The next item we need to consider is the *link capacity*, sometimes also referred to as *link bandwidth*. In order to distinguish demands from links, we will denote the links by 1-2, 1-3, and 2-3, and the capacity associated with these links by \hat{c}_{12} , \hat{c}_{13} , and \hat{c}_{23} , respectively, where the subscripts denote the end nodes of a link.

It is important to recognize here that the demand volume can be between any pair of nodes while the link connects two nodes directly. Another important point is that the unit of the demand volume needs to be consistent with the unit of link capacities. Thus, if we are considering packets per second (pps) as the unit for demand volume, then the link capacity needs to be expressed in units that can be translated to pps as well. This can be accomplished fairly easily for a link for which the capacity is given in terms of raw link speed such as Megabits per second (Mbps). To relate a link capacity to pps, the traffic measurement process also needs to determine the average packet size. If we have the average packet size, we can obtain the maximal possible total flow on a link in pps by dividing the raw link speed by the average packet size expressed in Mbps; this is then the effective link capacity in pps, sometimes referred to as the service rate of the link in pps. Recall that this was discussed earlier in Section 1.3.1. In any case, we will assume that units are consistent (refer to link capacity units [LCU] and demand volume units [DVU] in Section 1.3.5).

We now want to find out which flows might use different links. Since we are using the subscript identifier for a route, link identifiers are implicit in there. Thus, we can see that flow variables \hat{x}_{12} (for demand pair $\langle 1,2 \rangle$), \hat{x}_{123} (for demand pair $\langle 1,3 \rangle$), and \hat{x}_{213} (for demand pair $\langle 2,3 \rangle$) use link 1-2 which has a capacity of \hat{c}_{12} , expressed in pps. A common sensible requirement in any communication and computer network is that the link load cannot exceed the capacity of the link. Thus, we have the following inequality for link 1-2:

$$\hat{x}_{12} + \hat{x}_{123} + \hat{x}_{213} \leq \hat{c}_{12}.$$

Similarly, for other two links 1-3 and 2-3 we can write:

$$\hat{x}_{132} + \hat{x}_{13} + \hat{x}_{213} \leq \hat{c}_{13}$$

$$\hat{x}_{132} + \hat{x}_{123} + \hat{x}_{23} \leq \hat{c}_{23}.$$

In this illustration, we will now assume that the capacity of the first two links is 10 and the third is 15 (units); thus:

$$\hat{c}_{12} = \hat{c}_{13} = 10, \quad \hat{c}_{23} = 15.$$

Summarizing what we have discussed so far, we have the following set (system) of linear equations and inequalities (called *constraints*) where \hat{x} are unknowns for all three demands considered (Exercise 2.1):

$$\begin{array}{rcccccc} \hat{x}_{12} & + & \hat{x}_{132} & & & & = & 5 \\ & & & \hat{x}_{13} & + & \hat{x}_{123} & & = & 7 \\ & & & & & & \hat{x}_{23} & + & \hat{x}_{213} & = & 8 \\ \hat{x}_{12} & & & & & + & \hat{x}_{123} & & + & \hat{x}_{213} & \leq & 10 \\ & & \hat{x}_{132} & + & \hat{x}_{13} & & & + & \hat{x}_{213} & \leq & 10 \\ & & \hat{x}_{132} & & & + & \hat{x}_{123} & + & \hat{x}_{23} & \leq & 15 \end{array} \quad (2.1.1a)$$

$$\hat{x}_{12}, \hat{x}_{132}, \hat{x}_{13}, \hat{x}_{123}, \hat{x}_{23}, \hat{x}_{213} \geq 0.$$

In fact, system (2.1.1a) has, in general, multiple (continuously many) solutions and defines the set of all *feasible solutions*, i.e., feasible flows \hat{x} . This raises a question as to which specific feasible solution is of best interest. To address this, we now need to realize what is essential as far as the goal of network design is concerned. There can be different goals: minimize the total routing cost, minimize the congestion of the most congested link in the network, and so on. In the context of the mathematical representation, these types of goals are expressed through what is known as the *objective function* which is either minimized or maximized.

Suppose our goal is to minimize the total routing cost. Assuming that the cost of routing one unit of flow on every link along its path is simply set to 1, the total routing cost for all the flow variables is:

$$\mathbf{F} = \hat{x}_{12} + 2\hat{x}_{132} + \hat{x}_{13} + 2\hat{x}_{123} + \hat{x}_{23} + 2\hat{x}_{213} \quad (2.1.1b)$$

and this is our objective function. Note that the unit path cost on path 1-3-2 is 2 since the path is made up of two links where the unit link cost with respect to routing is 1. Another way to look at this is to say that it is twice as expensive to route on a two-link path compared to a one-link direct path. Finally, our goal here is to:

Minimize the objective function (2.1.1b) subject to the requirements or the constraints given by (2.1.1a). This dilemma will be referred to as Problem (2.1.2).

For completeness, we write the entire routing minimization problem discussed so far:

minimize

$$F = \hat{x}_{12} + 2\hat{x}_{132} + \hat{x}_{13} + 2\hat{x}_{123} + \hat{x}_{23} + 2\hat{x}_{213}$$

subject to (constraints)

$$\begin{array}{rcccccccl} \hat{x}_{12} & + & \hat{x}_{132} & & & & & = & 5 \\ & & & \hat{x}_{13} & + & \hat{x}_{123} & & = & 7 \\ & & & & & & \hat{x}_{23} & + & \hat{x}_{213} & = & 8 \\ \hat{x}_{12} & & & & & + & \hat{x}_{123} & & + & \hat{x}_{213} & \leq & 10 \\ & & \hat{x}_{132} & + & \hat{x}_{13} & & & & + & \hat{x}_{213} & \leq & 10 \\ & & \hat{x}_{132} & & & + & \hat{x}_{123} & + & \hat{x}_{23} & & \leq & 15 \end{array} \quad (2.1.2)$$

$$\hat{x}_{12}, \hat{x}_{132}, \hat{x}_{13}, \hat{x}_{123}, \hat{x}_{23}, \hat{x}_{213} \geq 0.$$

A dilemma such as Problem (2.1.2) is an example of a *multi-commodity network flow problem*. The term multi-commodity comes from the fact that there are multiple demands (or commodities) that need to be routed in the network simultaneously and they compete for available resources (link capacities, in this case). This is in fact a very common scenario in communication and computer networks (as opposed to a simpler *single-commodity network flow problem*). In the optimization context, this representation is also known as the *linear programming problem* since all the constraints and the objective function are linear. Finally, we refer to the mathematical description of the problem, as shown in (2.1.2), as the *formulation* (instance) of the problem. This formulation is also called the *arc-path* or *link-path* formulation in the literature; we will use the latter name in this book.

For Problem (2.1.2), we now need to find the *optimal solution*, i.e., feasible values of the decision variables (\hat{x}) that minimize the objective function (2.1.1b). It turns out that the optimal solution in this case is easy to find and does not require any fancy tools; in fact, some common sense is all you need. Since the cost is higher on multi-link paths, we try to route everything on direct, single-link paths. The resulting optimal solution (to be denoted with * in the superscript) is:

$$\hat{x}_{12}^* = 5, \quad \hat{x}_{13}^* = 7, \quad \hat{x}_{23}^* = 8$$

while all the other \hat{x} are 0, and the total (optimal) cost is $F^* = 20$ (Exercise 2.2). This solution is optimal because it is feasible (satisfies all the constraints), while minimizing the total cost. Furthermore, the optimal solution in this case is unique meaning that the flows \hat{x}^* achieving the minimal cost F^* are unique; in general, however, the optimal solution may not be unique.

Seeing a simple way to obtain the optimal solution, like in this case, can give the illusion that these problems are always easy to solve. Although we did not mention this above, we actually really needed to make sure that all constraints are satisfied by the optimal solution and, in particular, the link capacity constraints are not violated. To illustrate this point, let us consider a simple variation of this problem where the routing cost of a unit of flow is a

bit strange; it is twice as expensive to go on the direct path compared to the alternate path. The new objective function can be written as:

$$F = 2\hat{x}_{12} + \hat{x}_{132} + 2\hat{x}_{13} + \hat{x}_{123} + 2\hat{x}_{23} + \hat{x}_{213}. \quad (2.1.3)$$

This certainly seems like an odd cost function compared to the previous cost function (2.1.1b). Observe that in fact the cost function (2.1.1b) is the sum of link costs, where the link cost is the product of the *link load* (link load is the sum of all flows through the link) times the link unit cost (unit link cost are equal to 1 in this case). Actually, in practical world, we do face situations like the new cost function (2.1.3). For example, often an airplane ticket to go from one city to another city, especially in the U.S., is sometimes more expensive to fly directly than to hop through another city! Although you may wonder why anything like this could happen in a communication network, this kind of situation is not unheard of and can be dictated by the routing policy (although, perhaps not as exaggerated as this cost function). Essentially, we need to accept that the routing cost function can be quite arbitrary (Exercise 2.3).

In the revised problem where the specific objective function (2.1.3) is to be minimized with regard to constraints (2.1.1a), an obvious reaction would be to send all the volume for a demand pair on the cheaper of the two paths, i.e., on path 1-3-2 for demand pair (1,2) instead of path 1-2, and so on. Well, we immediately hit a snag which is the capacity of the links! Does this mean that the problem is not feasible, i.e., are there no values of \hat{x} that satisfy (2.1.1a)? Certainly not; otherwise, we would not have found a solution to Problem (2.1.2) with the original objective function. Note that for the revised problem of Problem (2.1.2), only objective function (2.1.3) has changed, *not* constraints (2.1.1a). This means that the optimal solution to the original Problem (2.1.2) is still a feasible solution to the revised problem. Taking a closer look, it is easy to see that this is not the optimal solution to the revised problem, and neither is the case when all demand volumes are routed on the cheaper of the two paths (since feasibility is violated). This means that the true optimal solution lies somewhere in between.

Without keeping you in suspense and without telling you for now how we can obtain the optimal solution, we are happy to report that the optimal solution to the revised problem, where we are minimizing (2.1.3) with regard to constraints (2.1.1a), is as given below:

$$\hat{x}_{12}^* = 0, \quad \hat{x}_{132}^* = 5, \quad \hat{x}_{13}^* = 1, \quad \hat{x}_{123}^* = 6, \quad \hat{x}_{23}^* = 4, \quad \hat{x}_{213}^* = 4$$

and the total cost at the optimal solution is $F^* = 25$.

There are a couple of important lessons that can be learned from the above example besides becoming familiarized with a multi-commodity network flow problem:

1. Changing the objective function can affect the optimal solution to a problem, and the way of finding it, sometimes, quite dramatically;
2. We need to carefully understand and use the right goal or objective function for a particular network; otherwise, the optimal solution obtained may not be meaningful.

2.2 NODE-LINK FORMULATION

The mathematical formulation presented in (2.1.2) uses the notions of link and path to describe the network optimization problem (this is why the name, link-path formulation, makes sense). In fact this formulation is valid for both undirected and directed links and demands. Still, there is another way to represent the same problem.

Assume that both links and demands are directed, and consider a fixed demand pair and a fixed node. Instead of tracing the *path flows* realizing the demand volume of the considered demand, we consider the total *link flow* for this demand on each link (many of such flows will be in general equal to 0). Now if you look at these link flows from the point of view of the fixed node, which is not an end node of the considered demand, you can see that the flows come into this node on the incoming links and are sent out on its outgoing links. Moreover, the total flow realizing the considered demand incoming to the node (called transit or intermediate node in this context) is equal to the total flow for this demand outgoing from the node. This is called *flow conservation law*. If the considered fixed node is the source of the considered demand, then the total outgoing flow minus the total incoming flow must be equal to the demand volume. Finally, if the fixed node is the sink, then the total incoming flow minus the total outgoing flow must be equal to the demand volume.

Coming back to the three-node network example from Figure 2.1a we substitute each of the three undirected links by two directed links (directed links are frequently referred to as *arcs*). For instance, undirected link 1-2 is substituted by two arcs: $1 \rightarrow 2$ and $2 \rightarrow 1$. Also all the three demands must be made directed – this time, however, it is sufficient to choose only one direction out of the two possible. Considering the first demand $\langle 1,2 \rangle$ between nodes 1 and 2, we may assume that its demand volume, \hat{h}_{12} , is originating at node 1 and is destined for node 2 (Figure 2.2), so demand $\langle 1,2 \rangle$ becomes the directed demand $\langle 1:2 \rangle$. As it will soon become clear, this can be viewed in the reverse order as well (using demand $\langle 2:1 \rangle$, instead of $\langle 1:2 \rangle$). This originating demand has two outlets, arc $1 \rightarrow 2$ and arc $1 \rightarrow 3$, to split and realize the demand volume; we will represent the amount of flow to allocate on these arcs by the (unknown) non-negative variables, $\tilde{x}_{12,12}$ and $\tilde{x}_{13,12}$, respectively. Notation “tilde” (\sim) is used to identify this type of the variable, to distinguish it from the path-flow representation given earlier. Here, the first part in the subscript before the comma refers to the arc, and the second part refers to the (directed) demand pair. As for an example, $\tilde{x}_{13,12}$ refers to the flow on arc $1 \rightarrow 3$ for demand pair $\langle 1:2 \rangle$.

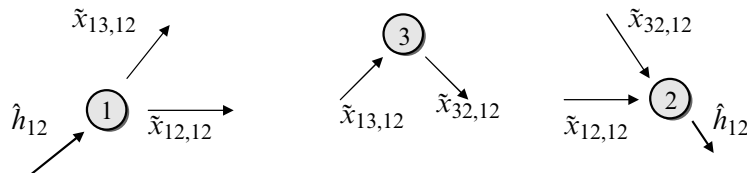


FIGURE 2.2 Flow View for Demand between Nodes 1 and 2

Let us consider demand $\langle 1:2 \rangle$. Due to the flow conservation law and using the convention that anything going into the node is negative and anything going out is positive, we may write the following equation for node 1:

$$-\hat{h}_{12} - \tilde{x}_{21,12} - \tilde{x}_{31,12} + \tilde{x}_{12,12} + \tilde{x}_{13,12} = 0.$$

Note that node 1 is the source of demand $\langle 1:2 \rangle$. For node 3, which is a transit node for demand $\langle 1:2 \rangle$, we have:

$$-\tilde{x}_{13,12} - \tilde{x}_{23,12} + \tilde{x}_{31,12} + \tilde{x}_{32,12} = 0.$$

Finally, at node 2, which is the sink for the considered demand, the flow conservation equation reads:

$$-\tilde{x}_{12,12} - \tilde{x}_{32,12} + \hat{h}_{12} + \tilde{x}_{21,12} + \tilde{x}_{23,12} = 0.$$

It is important to observe that it is sufficient to assume that for each undirected link, at most one out of the two corresponding directed flows is greater than 0 (i.e., one of these two flows is always equal to 0). This is because what really matters is the net flow on a link, e.g., $\tilde{x}_{12,12} - \tilde{x}_{21,12}$. A moment of reflection reveals that in our case we may assume in advance that

$$\tilde{x}_{21,12} = 0, \tilde{x}_{31,12} = 0, \tilde{x}_{23,12} = 0$$

because there is no use to have these flows positive since they are the “backward” flows from the viewpoint of demand $\langle 1:2 \rangle$. (We need to make a remark here that in general it is not obvious which flows can in advance be assumed to be equal to 0.)

Making use of the above observations we can write the set of *flow conservation equations* for demand $\langle 1:2 \rangle$ as:

$$\begin{array}{rcl} \tilde{x}_{12,12} & +\tilde{x}_{13,12} & = \hat{h}_{12} \\ & -\tilde{x}_{13,12} & +\tilde{x}_{32,12} = 0 \\ -\tilde{x}_{12,12} & & -\tilde{x}_{32,12} = -\hat{h}_{12}. \end{array} \quad (2.2.1)$$

(An observant reader may notice that the above system of equations is dependent, and one equation can be eliminated – this is a general fact.)

If we now consider the demand $\langle 1:3 \rangle$ from node 1 to node 3, and assuming $\tilde{x}_{31,13} = 0$, $\tilde{x}_{32,13} = 0$, $\tilde{x}_{21,13} = 0$, we can write the analogous equations as follows:

$$\begin{array}{rcl} \tilde{x}_{12,13} & +\tilde{x}_{13,13} & = \hat{h}_{13} \\ -\tilde{x}_{12,13} & & +\tilde{x}_{23,13} = 0 \\ & -\tilde{x}_{13,13} & -\tilde{x}_{23,13} = -\hat{h}_{13}. \end{array} \quad (2.2.2)$$

Finally, assuming $\tilde{x}_{32,23} = 0$, $\tilde{x}_{31,23} = 0$, $\tilde{x}_{12,23} = 0$, we have the following equations for the demand $\langle 2:3 \rangle$ from node 2 to node 3:

$$\begin{array}{rcl} \tilde{x}_{21,23} & +\tilde{x}_{23,23} & = \hat{h}_{23} \\ -\tilde{x}_{21,23} & +\tilde{x}_{13,23} & = 0 \\ -\tilde{x}_{13,23} & -\tilde{x}_{23,23} & = -\hat{h}_{23}. \end{array} \quad (2.2.3)$$

multi-commodity flow problem; this notation can be best referred to as *node-identifier-based notation*. What we mean is that all demands and paths are easy to follow from a node-reference point of view. However, this notation, which works quite well for a three-node network and in some special cases (Exercises 2.4), does not work very well in the general case since: 1) some node pairs may not have any demand and/or not all nodes are directly connected; 2) paths may contain many intermediate nodes; and 3) flow variables have indices of different length. To illustrate this, we will consider the link-path formulation of the instance (2.1.2) described in Section 2.1 for an undirected network with undirected links and bi-directional demands.

Basically, if every node pair has a demand, then for each pair of nodes i and j , we can represent the demand volume by \hat{h}_{ij} . Now suppose a pair of nodes in a network has no demand. For example, in a network with 50 nodes, suppose the demand does not exist between node 7 and node 15. Imagine now several pairs of nodes that may not have any demand (this is not unusual in real networks) and hence we somehow need to indicate in the model representation that certain pairs of nodes do not have any demand. With the node-identifier-based notation, this would need to be listed explicitly within the context of the model. That is, we have demand \hat{h}_{ij} between two nodes i and j except, for example, pair $\langle i, j \rangle = \langle 7, 15 \rangle$ and so on. At best this takes away from the main flow of understanding a problem formulation and can become a distraction. This distraction concerns also the links. If there is no (direct) link between, for example, nodes 5 and 9, then not only do we have $\hat{c}_{59} = 0$, but we do not need to represent link 5-9 at all in the formulation. In the node-identifier-based notation, you again have the situation of representing such exceptions in the formulation itself.

When the network gets larger, there are many possible paths between two nodes and the paths can be of a variable number of links (hops). Consider a path between nodes i and j of a demand pair going through a third node k ; then, the path is represented by $i-k-j$, and the flow variable is written as \hat{x}_{ikj} . On the other hand, if there is another path that goes via two intermediate nodes m and n for the same demand pair, then the path will be $i-m-n-j$, and the flow variable would be represented as \hat{x}_{imnj} , and so on. This notation creates such problems as:

- there is no easy way to represent multiple paths for a specific demand pair when each path may go through a different number of intermediate nodes;
- to complicate this matter, it so happens that even if a network allows for paths having, for example, a maximum two intermediate nodes, not all paths with two intermediate nodes may be acceptable due to the issues such as the distance between nodes; this again forces us to list exceptions for paths which are not used;
- the notation cannot directly handle the situation when there is more than one link between two nodes, i.e., the multi-graph case;
- also the situation with multiple demands between the same pair of nodes cannot be explicitly handled; and
- there is virtually no way to write summations over paths, which is necessary for expressing constraints (2.1.1a) in the general form.

To avoid such problems, we will use a different notation (which can be called *link-demand-path-identifier-based notation*) for the majority of the discussion in this book. This notation is compact and allows to list only the necessary objects. While the new notation may seem somewhat non-intuitive or awkward at first reading (at least for the three-node network example), in the long run it is much more handy to capture, formulate, and understand multi-commodity network flow problems, as well as to make algebraical manipulations on the formulated problems.

We now discuss how the link-demand-path-identifier-based notation works. All the demand pairs that have non-zero (i.e., non-negligible) demand volume (and only these pairs) are assigned indices from 1 to the total number of such demand pairs. This way any pair of nodes that does not have any demand is not listed at all. Now consider the three-node example of Section 2.1; we can label and map the demand pairs as follows:

$$\begin{aligned} \text{demand pair } \langle 1,2 \rangle &\longleftrightarrow \text{demand label 1} \\ \text{demand pair } \langle 1,3 \rangle &\longleftrightarrow \text{demand label 2} \\ \text{demand pair } \langle 2,3 \rangle &\longleftrightarrow \text{demand label 3.} \end{aligned}$$

In general, we will use the notation D to denote the total number of demand pairs in a network that have positive demand volumes, and index d to label these demands. Thus, in this case $D = 3$ and $d = 1, 2, 3$. Similarly, links that exist in the network are given labels from 1 to the total number of links. As in the case of a demand pair, if there is no link between two specific nodes, we do not need to list it at all in the formulation of the problem. Thus, for the same example, we have

$$\begin{aligned} \text{link 1-2} &\longleftrightarrow \text{link label 1} \\ \text{link 1-3} &\longleftrightarrow \text{link label 2} \\ \text{link 2-3} &\longleftrightarrow \text{link label 3.} \end{aligned}$$

Again, in general, we will use the notation E to denote the total number of actual links in the network, and index e to label the links. In this case, we have $E = 3$ and $e = 1, 2, 3$. Note that if there are multiple demands or links between the same pair of nodes, they can be simply included into the list of demands or links, respectively.

With the above background, we can make the following equivalence mapping for demand volumes and link capacities for the three-node problem of Section 2.1:

$$\begin{aligned} \hat{h}_{12} &\longleftrightarrow h_1, & \hat{h}_{13} &\longleftrightarrow h_2, & \hat{h}_{23} &\longleftrightarrow h_3 \\ \hat{c}_{12} &\longleftrightarrow c_1, & \hat{c}_{13} &\longleftrightarrow c_2, & \hat{c}_{23} &\longleftrightarrow c_3. \end{aligned}$$

Having considered the transformation of the demand pairs and the links to the new notation, we are now ready to discuss identifiers for the paths. Since we now have a demand pair identifier, we will use this as the first subscript in a path-flow variable, and then use the second subscript as the label for the path for that particular demand pair. This means that similar to demand pairs and links, the *candidate paths* for a demand pair are numbered from 1 to the total number of candidate paths for this demand pair. The total number of candidate paths for demand d will be denoted by P_d and the paths will be labeled with index p . For example, for demand $\langle 1,2 \rangle$ identified by label $d = 1$

we have $P_1 = 2$. The two candidate paths, 1-2 and 1-3-2, are labeled with $p = 1, 2$ (as the second subscript in the path-flow variable), respectively, and identified as (1,1) and (1,2), i.e., as paths number 1 and 2 for demand number 1. In the process, we can re-write the flow variables and their equivalence to the previous markers as listed below:

$$\begin{aligned} \hat{x}_{12} &\longleftrightarrow x_{11}, & \hat{x}_{132} &\longleftrightarrow x_{12}, \\ \hat{x}_{13} &\longleftrightarrow x_{21}, & \hat{x}_{123} &\longleftrightarrow x_{22}, \\ \hat{x}_{23} &\longleftrightarrow x_{31}, & \hat{x}_{213} &\longleftrightarrow x_{32}, \end{aligned}$$

You will notice that we are not using the “hat” with the new set of notations anymore. In fact, you might realize that we used the hat-notation to get to this point, and especially to show equivalence to the new notation (for one-to-one mapping). Since, we have accomplished this, from now on we will not be using the hat- or tilde-notation in the general model formulation. Finally, note also that we can easily map path identifiers and paths from the node-identifier-based notation (with node numbering) to the link-demand-path-identifier-based notation (using link numbering). For example:

node-identifier-based		link-demand-path-identifier-based	
path identifier	path	path identifier	path
132	1-3-2	12	{2, 3}
213	2-1-3	32	{1, 2}
23	2-3	31	{3}

and so on, where for the link-demand-path-identifier notation paths are represented as sets of link labels.

Finally, to complete this exercise, we will now re-write the entire formulation for the three-node example of Section 2.1 for the original instance (2.1.2) where we also include the notations for demand volumes and link capacities, instead of using their values on the right-hand side as was shown in (2.1.2):

minimize

$$F = x_{11} + 2x_{12} + x_{21} + 2x_{22} + x_{31} + 2x_{32}$$

subject to

$$\begin{aligned} x_{11} + x_{12} & & & & & = & h_1 \\ & x_{21} + x_{22} & & & & = & h_2 \\ & & x_{31} + x_{32} & & & = & h_3 \\ x_{11} & & & + x_{22} & + x_{32} & \leq & c_1 \\ & x_{12} + x_{21} & & & + x_{32} & \leq & c_2 \\ & x_{12} & + x_{22} & + x_{31} & & \leq & c_3 \end{aligned} \tag{2.3.1}$$

$$x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32} \geq 0.$$

It should be noted that both formulation (2.3.1) and the previous formulation (2.1.2) represent the same problem and use the link-path representation; the only difference is the notation, whether it is node-identifier based or link-demand-path-identifier based. Additional advantages of link-path formulation, especially with link-demand-path-identifier-based representation, is discussed later in Section 4.6.2.

The link-demand-path-identifier based notation can be used for the node-link representation of the problems discussed in Section 2.2 as well. Then the arc-flow variables are identified by the arc and demand labels, as explained in Section 4.1.1.

Let us now come back to the notation and naming issues. Figure 2.1a depicts a *graph* connecting network's nodes and links. Roughly speaking, the *network* is an object with the structure (often called network *topology*) given by its graph, and with many other attributes, as demand volumes, their candidate path lists, link capacities, etc.

As you already know, a demand pair is a pair of nodes and can be uni-directional (directed) or bi-directional (direction-less, undirected) depending on whether the demand volume is directed or not. Thus, the set-notation such as $\{v, v'\}$ (bi-directional demand) or (v, v') (uni-directional demand) for referring to the demand between nodes v and w makes sense. However, in this book, demand (pair) has a special role; thus, not to confuse with the standard set-notation, we will use $\langle v, v' \rangle$ for referring to demand between nodes v and v' (we have already been doing it!) in the bi-directional case, and use $\langle v : v' \rangle$ when referring to the uni-directional demand from node v to node v' . Let us also notice that many books and papers use the name “demand” for the actual volume of demand; in this book we will basically use the term demand to refer to a pair of nodes, and use the expression demand volume to refer to the actual volume (representing traffic or bandwidth) to be realized between the nodes of a particular demand.

We now discuss the case of links. As you know, a link is an object connecting two nodes. In this book, we will use the notation $v-v'$ when referring to the undirected (bi-directional) link between nodes v and v' (if there exists one), and also to distinguish from the notation for a bi-directional demand between the same two nodes. For a directed (uni-directional) link from node v to node v' , we will use the notation $v \rightarrow v'$. Graphs with directed links will be called directed graphs and graphs with undirected links will be called undirected graphs.

Finally, we will discuss the notion of the path. In general, an n -hop path between nodes v and v' is an interlacing sequence of nodes and links of the form

$$(v_1, e_1, v_2, e_2, \dots, v_n, e_n, v_{n+1})$$

where $v_1 = v$, $v_{n+1} = v'$, and link e_i connects nodes v_i and v_{i+1} for $i = 1, 2, \dots, n$. If the links are directed then the path is directed, and if the links are undirected, so is the path. A path can be represented either by its nodes or links. For instance, taking $n = 2$, the corresponding undirected path is represented by its nodes as $v_1-v_2-v_3$, and the directed path as $v_1 \rightarrow v_2 \rightarrow v_3$; in the link representation, we write them as $\{e_1, e_2\}$ and (e_1, e_2) , respectively.

It is important to note that in most problem formulations studied in this book, nodes, demands, links, and paths will be identified by their generic labels (this is possible due to our link-demand-path-identifier-based notation): v for node (nodes are called vertices in

graph theory), d for demand, e for link (links are called edges in graph theory), and p for path. Hence, the somewhat tedious notation needed to distinguish the different cases of the demand and link types will not be much needed. Note that we will use upper-case letters such as D to denote total number of demands, E to denote total number of links, and so on. This is introduced in the next section.

2.4 DIMENSIONING PROBLEMS

In this section, we consider a problem of minimizing the cost of network links with given demand volume between different nodes which can be routed over different paths. We will illustrate this problem through a new four-node example network with three nodes generating demand between each other and one “pure” transit node, not generating any demand. The simple network is shown in Figure 2.3. Our objective function in this case is going to be different than in the previous example.

The structure of this network is represented by the graph depicted in the lower part of Figure 2.3 and consists of $V = 4$ nodes and $E = 5$ undirected links. As shown in the upper part of this figure, there are $D = 3$ bi-directional demands. We have drawn an artificial vertical dotted line between the upper part and the lower part (such as connecting $v = 1$ from the upper part to $v = 1$ in the lower part) to show the corresponding mapping in regard to demand nodes.

We now generalize the labeling a bit more compared to the previous section, and add more formalism as we move forward. Purposefully, we have chosen to repeat certain issues

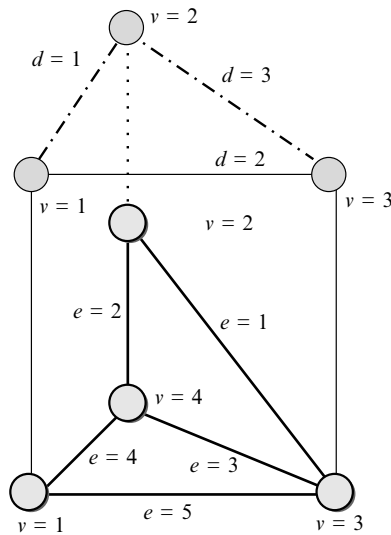


FIGURE 2.3 Four-Node Network Example

discussed in the previous sections for those who skipped reading the first three sections, and to help the beginners better understand mapping, formalism, and formulation. With this, we first introduce generic label for nodes, links, and demands. Nodes (vertices) will be labeled with the generic label v ($v = 1, 2, \dots, V$), links (edges) with label e ($e = 1, 2, \dots, E$), and demands with label d ($d = 1, 2, \dots, D$). Each link connects its end nodes directly, and is identified with the corresponding unordered pair of nodes. For instance, link $e = 1$ is of the form 2-3; here, nodes $v = 2$ and $v = 3$ are the end nodes of link $e = 1$. Similarly, link $e = 5$ connects nodes $v = 1$ and $v = 3$ so is of the form 1-3, and so on. Note that links and demands of the example network are undirected.

Demands correspond to pairs of nodes. In Figure 2.3, demand $d = 1$ corresponds to pair $\langle 1,2 \rangle$, $d = 2$ to pair $\langle 1,3 \rangle$, and $d = 3$ to pair $\langle 2,3 \rangle$. Thus, nodes $v = 1$ and $v = 2$ are the end nodes of demand $d = 1$, and so on. Note that not all pairs of nodes correspond to demands; for instance, there is no demand between nodes $v = 1$ and $v = 4$. In fact, node $v = 4$ is not the end node of any demand in this example and, therefore, is sometimes referred to as a (pure) transit node. Note that end nodes can be transit nodes as well in the sense that they can appear as intermediate nodes for demands with other end nodes. In Figure 2.3 the end nodes are duplicated in the upper part, so the demands between the end nodes are depicted separately.

The capacity of link e ($e = 1, 2, \dots, E$) will be denoted by c_e whenever the capacity is given. On the other hand, for certain design problems, the capacity of a link will not be given and will, in fact, be a design variable. For example, consider the design problem where we are to determine demand flows and link capacities required to carry the given demand volumes, sometimes referred to as the *dimensioning problem*. When link capacity is a variable, we will use the notation y_e ; the notation c_e will be used when the capacity is given. Generically, we will use link capacity unit – LCU – for expressing capacity of a link. One capacity unit (or, 1 LCU) on link e is assigned the unit (or marginal) cost ξ_e (≥ 0).

Each demand d ($d = 1, 2, \dots, D$) is characterized by the demand volume denoted by h_d . Demand volumes are also called commodities in non-telecommunications applications, hence the term multi-commodity when multiple demands are considered, thus leading to the term multi-commodity network problems. Demand volume is expressed in general through demand volume units – DVU. For example, if a DVU is given in Mbps, then for consistency, LCU should be represented in Mbps as well. If however, a DVU is given in pps, LCU need to be mapped to pps even though the actual capacity may be given in link bit-rate (see Section 1.3.1 on how to do proper mapping, and the discussion earlier). In the rest of this section, we shall assume that LCU and DVU are in Mbps, unless stated otherwise.

Figure 2.4 depicts the unit costs and demand volumes assigned to the links and demands, respectively. Each demand d is assigned a list of paths (called also routes) that can carry flows. For demand d the total number of assigned paths is denoted by P_d and they are labeled with p from the first path to the total number of paths, i.e., $p = 1, 2, \dots, P_d$; this sequence is called the *list of candidate paths* (or path list, or routing list, in short). To tie it to generic demand d , we write the list of paths as $\mathbb{P}_d = (\mathcal{P}_{d1}, \mathcal{P}_{d2}, \dots, \mathcal{P}_{dP_d})$.

It is important to note that each path \mathcal{P}_{dp} ($p = 1, 2, \dots, P_d$) connects the end nodes of demand d . Each path is described as the set of links of which the path is composed of.

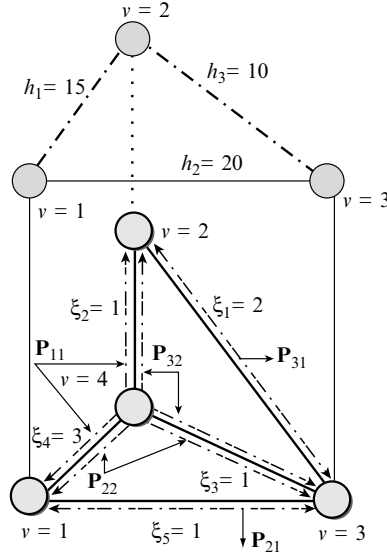


FIGURE 2.4 Four-Node Network Example: Demand Volume and Link Cost

Demand volumes are *realized* by means of flows assigned to paths on their routing lists. The flow realizing demand d on path p is denoted by x_{dp} ($p = 1, 2, \dots, P_d$).

Now, consider the example in Figure 2.4. Here, demand $d = 1$ is allowed to have, for example, just one path: $\mathcal{P}_{11} = \{2, 4\}$ which means that the path consists of link number 2 and link number 4. This then means that $\mathbb{P}_1 = (\mathcal{P}_{11})$. In this case, there is only one flow variable x_{11} . For demand $d = 2$ two paths are allowed: $\mathcal{P}_{21} = \{5\}$, $\mathcal{P}_{22} = \{3, 4\}$ and the associated flow variables are x_{21} and x_{22} , respectively. Note that $\mathbb{P}_2 = (\mathcal{P}_{21}, \mathcal{P}_{22})$. Finally, demand $d = 3$ is also allowed to have two paths: $\mathcal{P}_{31} = \{1\}$, $\mathcal{P}_{32} = \{2, 3\}$ ($\mathbb{P}_3 = (\mathcal{P}_{31}, \mathcal{P}_{32})$), and associated flow variables, x_{31} and x_{32} . Notice that path $\mathcal{P}_{12} = \{1, 5\}$ could have been a possible path for demand $d = 1$, but is not included into the path list \mathbb{P}_1 in the first part of our illustration, and will be considered later.

Since demand volume for each demand d needs to be realized through flows on candidate paths, we can now write the following equations (known as the *demand constraints*):

$$\begin{aligned} x_{11} &= 15 \\ x_{21} + x_{22} &= 20 \\ x_{31} + x_{32} &= 10. \end{aligned} \tag{2.4.1}$$

Note that, in this case, we have $P_1 = 1$, $P_2 = 2$, and $P_3 = 2$. In general, if you notice, the flow variables names and path indices are pretty much the same as in the three-node example from the previous section after we introduced the link-demand-path-identifier-based notation in Section 2.3. This allows you to see the advantage of this notation; although

the actual network topology has changed from the previous example, the representation of the problem remains virtually the same.

The demand constraints (2.4.1) can be written in a general form as follows. Suppose we denote the vector of flows assigned to demand d with $\mathbf{x}_d = (x_{d1}, x_{d2}, \dots, x_{dP_d})$ for path indices $p = 1, 2, \dots, P_d$. Then, we arrive at:

$$x_{d1} + x_{d2} + \dots + x_{dP_d} = h_d$$

for each demand d . In summation notation, we can write this as:

$$\sum_{p=1}^{P_d} x_{dp} = h_d, \quad d = 1, 2, \dots, D. \quad (2.4.2)$$

Since, in general, we know that the paths are numbered from 1 to P_d for each demand d , we can write expression (2.4.2) more compactly as:

$$\sum_p x_{dp} = h_d, \quad d = 1, 2, \dots, D \quad (2.4.3)$$

where, for a fixed demand d , the summation is always carried over all $p = 1, 2, \dots, P_d$, unless explicitly stated otherwise. In its general form, equation (2.4.3) is an important one to note and understand as you will see this throughout the book.

In general, the vector of all flows (path-flow variables), will be called *flow allocation vector*, or simply *flow vector*, which can be written as:

$$\begin{aligned} \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_D) &= (x_{11}, x_{12}, \dots, x_{1P_1}, x_{21}, x_{22}, \dots, x_{2P_2}, \dots, x_{D1}, x_{D2}, \dots, x_{DP_D}) \\ &= (x_{dp} : d = 1, 2, \dots, D; p = 1, 2, \dots, P_d). \end{aligned}$$

By now, you have probably noticed that we are using the bold font to denote a vector, and to distinguish it from a scalar quantity. For example, \mathbf{x}_1 and x_{11} are both valid notations - the first one (bold), \mathbf{x}_1 , is a vector while the second one, x_{11} , refers to the first scalar entity of vector \mathbf{x}_1 . Since with any rule there is an exception, we also have an exception; throughout this book, we will use the bold font, \mathbf{F} , for representing the objective function which takes scalar values.

As you may have guessed by now, there is a second set of constraints which assures that for each link e its capacity c_e (or y_e , if the capacity is a variable) is not exceeded by the flows using this link; we call these *capacity constraints*. This requirement leads to the following inequalities for the four-node network:

$$\begin{array}{rcccc} & & x_{31} & \leq & y_1 \\ x_{11} & & & +x_{32} & \leq & y_2 \\ & & x_{22} & +x_{32} & \leq & y_3 \\ x_{11} & & +x_{22} & & \leq & y_4 \\ & x_{21} & & & \leq & y_5. \end{array} \quad (2.4.4)$$

The sums on the left-hand sides of the above inequalities are called *link loads*, as they give the total flow through a link, expressed in DVUs. This is very similar to what we did in (2.1.1a) (note the last three inequalities). In general, to write down the link loads, we need to

TABLE 2.1 Link-Path Incidence Relation δ_{edp}

$e \setminus \mathcal{P}_{dp}$	$\mathcal{P}_{11} = \{2, 4\}$	$\mathcal{P}_{21} = \{5\}$	$\mathcal{P}_{22} = \{3, 4\}$	$\mathcal{P}_{31} = \{1\}$	$\mathcal{P}_{32} = \{2, 3\}$
1	0	0	0	1	0
2	1	0	0	0	1
3	0	0	1	0	1
4	1	0	1	0	0
5	0	1	0	0	0

know the relationship between links and paths. The relationship can be formally specified by the link-path incidence coefficients. To see this, we refer to Table 2.1.

Observe that Table 2.1 is nothing but an indication of which flow variables appear on the left-hand side of each inequality in (2.4.4). For example, the first path for demand $d = 1$, path \mathcal{P}_{11} does not use link $e = 1$, so this entry is 0. The reason is that \mathcal{P}_{11} consists of links $e = 2$ and $e = 4$; thus, this makes sense. Let us look at another one. The second path for demand $d = 3$ uses link $e = 3$, so this entry is set to 1. If you look at the pattern, you will notice that there are three pieces of information involved, e, d, p ; another way to look at it is to determine whether a generic path p for demand d uses link e ; if it does for a specific case, we set the entry in the table to be 1, otherwise, it is 0. The good news is that this information can be written in a very nice compact manner by using another notation, δ , which really tells us the link-path relation. Formally, a coefficient δ_{edp} is defined for each triple (e, d, p) where $e = 1, 2, \dots, E$, $d = 1, 2, \dots, D$, and $p = 1, 2, \dots, P_d$, and is defined as:

$$\delta_{edp} = \begin{cases} 1 & \text{if link } e \text{ belongs to path } p \text{ for demand } d \\ 0 & \text{otherwise.} \end{cases} \quad (2.4.5)$$

Note that δ_{edp} is a given quantity since the set of possible paths is already given and the links these paths use are fixed; in other words, it is not a variable. Using the introduced coefficients, the link load on e can be nicely written as the following linear expression:

$$\sum_{d=1}^D \sum_{p=1}^{P_d} \delta_{edp} x_{dp}.$$

For the 4-node example, this is nothing but the expression on the left-hand side of (2.4.4) for each link. As you can see, the use of the notation δ_{edp} is extremely helpful since it allows us to write the relation between link load and capacity in a nice compact way. This load for link e will be denoted by \underline{y}_e , and hence is defined as follows:

$$\underline{y}_e = \underline{y}_e(\mathbf{x}) = \sum_d \sum_p \delta_{edp} x_{dp} \quad (2.4.6)$$

where we have skipped the bounds for the summation indices as they are known.

The capacity constraints can be generally written as:

$$\sum_d \sum_p \delta_{edp} x_{dp} \leq y_e, \quad e = 1, 2, \dots, E. \quad (2.4.7)$$

The summation on the left-hand side of inequalities (2.4.7) is taken over all paths appearing in the routing lists of all demands, i.e., over all combinations (d, p) such that $d = 1, 2, \dots, D$ and $p = 1, 2, \dots, P_d$; these are not shown in the above compact form. Now, inequality (2.4.7) is the next important relation to note after equation (2.4.3). For brevity, we will often refer to the capacity variables $y_e, e = 1, 2, \dots, E$ through the vector notation $\mathbf{y} = (y_1, y_2, \dots, y_E)$. It is important to distinguish link capacity variables y_e from the corresponding link loads \underline{y}_e ; in general, we require that $\underline{y}_e \leq y_e$ for each link e .

In this problem, we are interested in minimizing the capacity cost. This objective function can be written as:

$$\begin{aligned} F &= \xi_1 y_1 + \xi_2 y_2 + \xi_3 y_3 + \xi_4 y_4 + \xi_5 y_5 \\ &= 2y_1 + y_2 + y_3 + 3y_4 + y_5. \end{aligned} \quad (2.4.8)$$

In general, this cost function can be denoted as:

$$F = \sum_{e=1}^E \xi_e y_e = \sum_e \xi_e y_e. \quad (2.4.9)$$

Thus, we can pose the following instance of the *dimensioning problem*, referred to as DP in the sequel, for the four-node network example in its entire form as:

minimize

$$F = 2y_1 + y_2 + y_3 + 3y_4 + y_5$$

subject to

$$\begin{array}{rclcl} x_{11} & & & & = & 15 \\ & x_{21} & + & x_{22} & = & 20 \\ & & & x_{31} & + & x_{32} & = & 10 \\ & & & x_{31} & \leq & y_1 \\ x_{11} & & & & + & x_{32} & \leq & y_2 \\ & & & x_{22} & + & x_{32} & \leq & y_3 \\ x_{11} & & + & x_{22} & \leq & y_4 \\ & x_{21} & & & \leq & y_5 \end{array} \quad (2.4.10)$$

$$x_{11}, x_{21}, x_{22}, x_{31}, x_{32} \geq 0, \quad y_1, y_2, y_3, y_4, y_5 \geq 0.$$

In the general form, DP (2.4.10) can be written as:

minimize

$$\text{objective/cost function (2.4.9): } F = \sum_e \xi_e y_e$$

subject to

$$\begin{array}{ll} \text{demand constraints (2.4.3):} & \sum_p x_{dp} = h_d, \quad d = 1, 2, \dots, D \\ \text{capacity constraints (2.4.7):} & \sum_d \sum_p \delta_{edp} x_{dp} \leq y_e, \quad e = 1, 2, \dots, E \\ \text{constraints on variables:} & \mathbf{x} \geq \mathbf{0}, \mathbf{y} \geq \mathbf{0}. \end{array} \quad (2.4.11)$$

Let us pause now to compare this formulation to what we have discussed in earlier sections in regard to the three-node network example (Exercise 2.5). In the instance of Problem (2.3.1) for the three-node example, we have minimized the total routing cost when capacity was given to carry a given demand using flow variables; for the four-node example here, we are minimizing the total link capacity cost required to carry a given demand also using flow variables, but at the same time capacities are variables as well. A problem such as DP where capacity is to be determined is, as already mentioned, referred to as the dimensioning (or uncapacitated) problem. Note that DP (2.4.11) happens to be a linear programming (LP, refer to Section 5.1.1) problem. It may be noted that a common convention for writing a linear programming problem is to list all variables on the left side of the equation or inequality. We favor keeping the logical (and physical) meaning of a constraint intact in its natural form over this convention. Thus, for (2.4.7), i.e., the capacity constraint in (2.4.11), variable y_e is kept on the right side of the inequality instead of moving it to the left side of the inequality.

Now we are ready to discuss the optimal solution for DP (2.4.11). In this model, when variables take continuous values (as in our case for this example), then for any optimal solution of DP, the capacity feasibility constraints become equalities, i.e., for each link, its load is equal to its capacity because otherwise we would pay for unused capacity on the links (implying the solution is not optimal).

In Figure 2.5, we have shown a set of values for the flow allocation vector \mathbf{x} . If we calculate the link loads (and hence, link capacities) for this flow allocation vector, then assuming the equalities in constraints (2.4.7) we arrive at the following loads: $y_1(\mathbf{x}) = y_1 = 5$, $y_2(\mathbf{x}) = y_2 = 20$, $y_3(\mathbf{x}) = y_3 = 10$, $y_4(\mathbf{x}) = y_4 = 20$, and $y_5(\mathbf{x}) = y_5 = 15$. The so defined

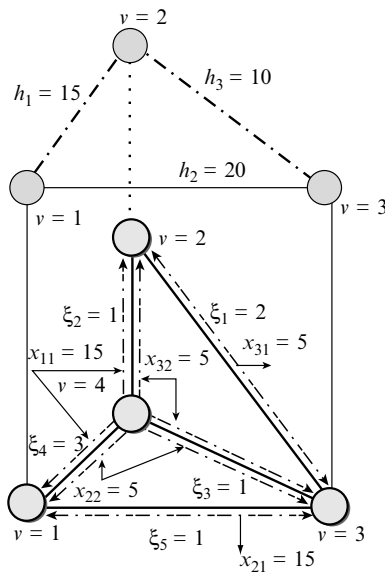


FIGURE 2.5 Four-Node Network Example: Allocation

feasible solution (\mathbf{x}, \mathbf{y}) , where $\mathbf{y} = (y_1, y_2, y_3, y_4, y_5) = (5, 20, 10, 20, 15)$, has the total cost $F = 115$.

It is easy to see that the above solution is not optimal, because of the use of the second path of the second demand, \mathcal{P}_{22} , carrying flow $x_{22} = 5$. This path is composed of links $e = 3$ and $e = 4$, and has the unit cost equal to the sum of the unit costs of its links, i.e., $\zeta_{22} = \xi_3 + \xi_4 = 1 + 3 = 4$. The other allowable path for demand $d = 2$ is its path number $p = 1$, \mathcal{P}_{21} , with $\zeta_{21} = \xi_5 = 1$. In general, the cost of path \mathcal{P}_{dp} for Problem (2.4.11) is given by the formula:

$$\zeta_{dp} = \sum_e \delta_{edp} \xi_e, \quad d = 1, 2, \dots, D \quad p = 1, 2, \dots, P_d. \quad (2.4.12)$$

This also shows another advantage of notation δ_{edp} since it allows us to write the cost of a path in a compact manner if the costs of the links that the path is composed of are available.

In this example, it is profitable then to move all the flow from path \mathcal{P}_{22} to path \mathcal{P}_{21} which gives the saving of $(\zeta_{22} - \zeta_{21}) = 3$ per one unit of flow. In our case, this would give the total saving of 15, since $x_{22}(\zeta_{22} - \zeta_{21}) = 15$. Note that flow x_{11} is trivially optimal since its value is fixed by the demand constraint (2.4.3) for demand $d = 1$; note that there is only one path for demand $d = 1$. Also, the two flows assigned to demand $d = 3$ are optimal since both allowable paths for this demand have the same unit cost $\zeta_{31} = \zeta_{32} = 2$; altogether, we have the optimal cost $F^* = 100$. Note that for this problem, and in the case of demand $d = 3$, any split of its demand volume $h_d = 10$ among the two allowable paths is optimal as well which means that F^* remains the same but there are multiple \mathbf{x} s with the same optimal cost value. The above observations lead to describing optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ in the following way:

$$\begin{aligned} x_{11}^* &= 15 \\ x_{21}^* &= 20, x_{22}^* = 0 \\ x_{31}^* &= a, x_{32}^* = 10 - a \quad \text{for any } 0 \leq a \leq 10 \\ y_1^* &= 10 - a, y_2^* = 15 + a, y_3^* = a, y_4^* = 15, y_5^* = 20 \\ F^* &= 100. \end{aligned}$$

Thus, we can make the following statement in regard to obtaining the optimal solution to our specific problem:

Shortest-Path Allocation Rule for DP:

For each demand, allocate its entire demand volume to its shortest path, with respect to links unit costs and candidate path. If there is more than one shortest path for a demand then the demand volume can be split among the shortest paths in an arbitrary way. (2.4.13)

This rule implies that the optimal solution (i.e., optimal flow allocation vector) of DP, in general, may not be unique. This is an important point to note. Moreover, while the above rule works for DP, it is not (at all) a general solution approach for other multi-commodity flow problems.

We can slightly change the four-node design problem (2.4.10) by putting restrictions on flows. For example, suppose we insist on having an optimal solution with *non-bifurcated* flows (also called *single-path* flows or *unsplittable* flows). This would mean that for $d = 2$

and $d = 3$, we can choose only one path each although each demand has two possible paths. To obtain the optimal solution, no demand split is allowed. The non-bifurcated solutions are not unique either. For instance in the example here if we impose the requirement for non-bifurcated flows, we can have two such solutions corresponding to $a = 0$ and $a = 10$. We want to emphasize that although in the case considered it is easy to obtain a non-bifurcated optimal solution, in general, the single-path flow requirement makes the problem (much) more difficult to solve.

Now we consider another variation to the original example (2.4.10) while allowing bifurcation. Suppose we add path $\mathcal{P}_{12} = \{1, 5\}$ to the candidate path list of demand $d = 1$. Since this path appears for demand $d = 1$, and for links $e = 1$ and $e = 5$, we need to modify the corresponding equations and inequalities. Specifically, the first equation (2.4.10) corresponding to demand $d = 1$ which is $x_{11} = 5$, the inequality corresponding to link $e = 1$ which is $x_{31} \leq y_1$, and the inequality corresponding to link $e = 5$ which is $x_{21} \leq y_5$, need to change, respectively, to:

$$\begin{array}{rclcl} x_{11} & + & x_{12} & & = & 15 \\ & & x_{12} & & + & x_{31} & \leq & y_1 \\ & & x_{12} & + & x_{21} & & \leq & y_5. \end{array}$$

Thus, we have the following modified problem:

minimize

$$F = 2y_1 + y_2 + y_3 + 3y_4 + y_5$$

subject to

$$\begin{array}{rclcl} x_{11} & + & x_{12} & & = & 15 \\ & & & & x_{21} & + & x_{22} & = & 20 \\ & & & & & & x_{31} & + & x_{32} & = & 10 \\ & & x_{12} & & & + & x_{31} & & \leq & y_1 \\ x_{11} & & & & & & + & x_{32} & \leq & y_2 \\ & & & & x_{22} & & + & x_{32} & \leq & y_3 \\ x_{11} & & & & + & x_{22} & & & \leq & y_4 \\ & & x_{12} & + & x_{21} & & & & \leq & y_5 \end{array} \quad (2.4.14)$$

$$x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32} \geq 0, \quad y_1, y_2, y_3, y_4, y_5 \geq 0.$$

The optimal solution to the modified problem (2.4.14) would use the flows $x_{11}^* = 0, x_{12}^* = 15$ and the cost would be further decreased by 15 to $F^* = 85$, since for demand $d = 1$, the cost for the second path ($\zeta_{12} = 3$) is cheaper than the first path ($\zeta_{11} = 4$). This example shows that, as could be expected, the starting candidate path lists can affect the optimal solution and, hence, the path preprocessing, i.e., initializing or augmenting path lists can be an important part of the network design process.

So far, we have assumed that link capacity can be installed in non-integral, continuous values. Often, in real networks, capacities can be installed only in modular units, e.g., T1, E1, OC-3, and so on. Suppose now we change the original design problem DP (2.4.11) to

be addressed in the presence of modular capacity units (i.e., certain modules equal to M DVUs; in this way, one LCU is equal to M DVUs). With such modular links the solution does not, in general, obey the shortest-path allocation rule (2.4.13) (which can be easily anticipated for large values of the module) nor the optimal link load, $\underline{y}_e(\mathbf{x})$, being equal to optimal link capacity, y_e (see the example illustrated in Section 1.4). For instance, with $M = 35$ the cheapest solution for this modularized problem is to install one module of capacity on exactly two links: $e = 1$ and $e = 5$ (assuming that $\xi_e = 1$ is the cost of one module of capacity on link e). Also, the optimal flows in the modular link case are, in general, bifurcated, unless additional constraints forcing the single-path solution are imposed. We note that for large networks, the modular design problems, especially when combined with the single-path requirement, are very difficult to solve even with specialized algorithms/heuristics developed for this purpose.

In DP-like problems, link capacities are subject to optimization and, hence, DP is an example of an *uncapacitated* design problem. Another type of problem is *capacitated* design problem where link capacities are given (note we will use c_e as the given capacity of link e when known, instead of y_e). The problem is to find a feasible flow allocation vector that satisfies the demand constraints (2.4.3) and the capacity constraints (2.4.7) with c_e appearing on the right-hand sides. In such a scenario, there may not be any objective function unless, for example, flow routing cost minimization is required as discussed in Sections 2.1 and 2.3. Thus, this shows the connection between the two different cases which are useful for addressing different network design problems. For completeness, we state the capacitated problem in the compact form below:

$$\begin{aligned} \text{demand constraints:} & \quad \sum_p x_{dp} = h_d, & d = 1, 2, \dots, D \\ \text{capacity constraints:} & \quad \sum_d \sum_p \delta_{edp} x_{dp} \leq c_e, & e = 1, 2, \dots, E \\ \text{constraints on variables:} & \quad \mathbf{x} \geq \mathbf{0}. \end{aligned} \quad (2.4.15)$$

Note that link costs do not come into the picture in this case; however, the routing cost may appear. Such problems often arise in a network scenario when the network capacity is already built up, but the demand may have changed.

Returning to our original example for this section, suppose that the link capacities are given as follows $\mathbf{c} = (c_1, c_2, c_3, c_4, c_5) = (5, 10, 10, 5, 30)$, and that the path lists are extended so that $\mathbb{P}_1 = (\mathcal{P}_{11}, \mathcal{P}_{12}, \mathcal{P}_{13})$, where $\mathcal{P}_{13} = \{4, 3, 1\}$, can also be used for demand $d = 1$ (candidate path lists for the remaining two demands are not altered). It is easy to see that all feasible solutions to the considered capacitated problem are necessarily bifurcated, and an example of one of them is as follows:

$$\begin{aligned} x_{11} &= 5, x_{12} = 0, x_{13} = 10 \\ x_{21} &= 20, x_{22} = 0 \\ x_{31} &= 10, x_{32} = 10. \end{aligned}$$

Note that demand $d = 1$ uses its longest path in terms of the ‘‘hop-count’’.

In general, when non-bifurcated solutions are required, then additional constraints on the flows must be imposed explicitly to force the single-path solution. It may so happen that for given link capacities a bifurcated solution exists but the non-bifurcated does not

(think why). You can see this from the above example when you impose the requirement that flows for a demand pair must be non-bifurcated.

Problems along the lines discussed in this section will be covered in detail in Chapter 4. Optimization algorithms related to these problems will be covered in Chapter 5.

2.5 SHORTEST-PATH ROUTING

Now we will consider another design problem. Computer networks use what is often referred to as *shortest-path routing*; for example, OSPF routing in IP networks (see Section 3.1 and Chapter 7 for additional details). It is important not to confuse shortest-path routing with the term shortest-path allocation discussed in the last section. This will hopefully become apparent once you read through this section. Basically, the shortest-path routing means that for each demand d all its volume h_d is realized on its shortest path, with respect to some given link weight system $\mathbf{w} = (w_1, w_2, \dots, w_E)$ with link weight (cost) w_e for link e , among all possible paths for demand d in the network. Note that the path selection is based on additive calculation of link-weights.

Suppose we are given the link weight system $\mathbf{w} = (1, 3, 1, 2, 4)$ for the example network of Figure 2.3. Then, the following paths will be used to carry the entire demand volumes for different demands as follows:

$$\begin{array}{llll} d = 1 & \mathcal{P}_{13} = \{1, 3, 4\} & \text{path length 4} & x_{13} = 15 \\ d = 2 & \mathcal{P}_{21} = \{1\} & \text{path length 1} & x_{21} = 10 \\ d = 3 & \mathcal{P}_{32} = \{3, 4\} & \text{path length 3} & x_{32} = 10. \end{array}$$

In this case, the rest of the flow variables are equal to 0. Since in this example of a network scenario, the shortest paths are unique for each demand pair, the resulting flow allocation vector $\mathbf{x}(\mathbf{w})$, is also unique, and in this case, non-bifurcated. It is important to note the notation in this case: $\mathbf{x}(\mathbf{w})$ (instead of just \mathbf{x}), which means that flow allocation \mathbf{x} is dictated by the weight system \mathbf{w} . However, this flow vector is not feasible for the link capacities $\mathbf{c} = (5, 10, 10, 5, 30)$ used in the previous example! The link loads, which are denoted by \underline{y}_e , resulting from the allocation vector $\mathbf{x}(\mathbf{w})$, are equal to $\underline{\mathbf{y}}(\mathbf{w}) = (\underline{y}_1, \underline{y}_2, \underline{y}_3, \underline{y}_4, \underline{y}_5) = (25, 0, 35, 35, 0)$; they do not satisfy the link capacities in terms of the capacity constraints listed in (2.4.15). Certainly, if we change the link capacity so that $\mathbf{c} = \underline{\mathbf{y}}(\mathbf{w})$, then this shortest-path allocation, with respect to the weight system \mathbf{w} , will become (trivially) feasible.

In general, the following single shortest-path allocation problem is a complex problem to solve:

For given link capacities \mathbf{c} and demand volumes \mathbf{h} ($\mathbf{h} = (h_1, h_2, \dots, h_D)$), find a link weight system \mathbf{w} such that the resulting shortest paths are unique and the resulting flow allocation vector $\mathbf{x}(\mathbf{w})$ is feasible, i.e., such that $\mathbf{x}(\mathbf{w})$ satisfies (2.4.15).

There are three main reasons for this complexity:

- A non-bifurcated (single-path) feasible flow allocation may not exist while bifurcated feasible flow allocations may exist.

- Even if a single-path solution exists, in most cases it can be hard to determine.
- Moreover, even if we find a single-path flow solution, then the weight system to induce it may not exist.

To illustrate the last (least obvious) item more precisely, we note that there can be a feasible non-bifurcated flow allocation vector, \mathbf{x} , but there may not exist any weight system \mathbf{w} such that $\mathbf{x} = \mathbf{x}(\mathbf{w})$ for any such feasible \mathbf{x} . This issue is illustrated in Figure 2.6. Suppose there are two demands: $d = 1$ between nodes 1 and 7 and $d = 2$ between nodes 2 and 6 with same demand volume $h_1 = h_2 = 1$. Suppose also that all links' capacities are equal to 1 ($c_e \equiv 1$). Here, each demand has two paths; for demand $d = 1$, the first path traverses the nodes 1-3-5-7 (path \mathcal{P}_{11}) and the second path traverses the nodes 1-3-4-5-7 (path \mathcal{P}_{12}). For demand $d = 2$, paths are 2-3-4-5-6 (path \mathcal{P}_{21}) and 2-3-5-6 (path \mathcal{P}_{22}). Now, allocating flow $x_{11} = 1$ to path 1-3-5-7 and flow $x_{21} = 1$ to path 2-3-4-5-6 gives the feasible flow allocation vector $\mathbf{x} = (x_{11}, x_{21})$ with respect to Problem (2.4.15). Now you may wish to check if any link weight system induces this flow (bad news: it is not possible for single shortest path!).

We now return to our original 4-node example and consider the weight system $\mathbf{w} = (1, 1, 1, 1, 1)$. You may recognize this weight system immediately since this is nothing but the system inducing shortest-path routing with respect to hop-count (hop-count rule). Using shortest paths in the sense of the hop count is natural and is a common practice in many actual networks. Consider now demand $d = 1$ for this example; we can immediately find that there are two shortest paths! They are: $\mathcal{P}_{11} = \{2, 4\}$ and $\mathcal{P}_{12} = \{1, 5\}$. In such situations, a common question is which path should be used for routing the traffic. In other words, there must be some additional rule of splitting the demand volume into its different shortest paths.

This problem, for instance, appears in determining link weight in IP-OSPF networks (Section 3.1 and Chapter 7). One such rule used in OSPF routing is the equal-split rule, often referred to as ECMP: “equal-cost multi-path” rule. For a fixed destination, the goal of ECMP is to equally split all the demand outgoing from a node among all its outgoing links

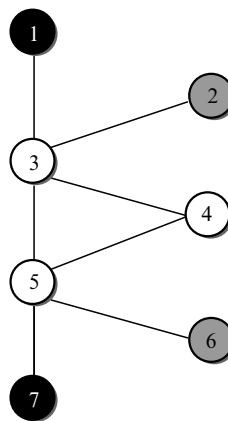


FIGURE 2.6 Infeasible Unique Shortest-Path Case

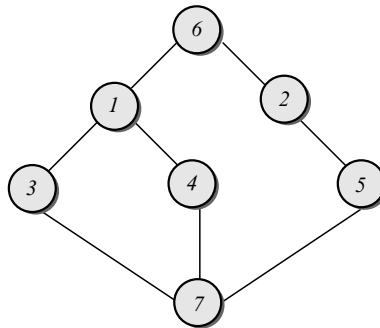


FIGURE 2.7 Equal-Split Rule

that belong to the shortest paths to that destination. This is illustrated through the example network given in Figure 2.7. Assuming link weights to be all equal to 1, there are three shortest paths from node 6 to node 7, i.e., for the directed demand pair $\langle 6:7 \rangle$. The shortest paths 6-1-3-7 and 6-1-4-7 carry one-quarter of the total demand volume from node 6 to node 7, while the shortest path 6-2-5-7 carries the remaining one-half of the volume. To complicate matters, in OSPF network routing is directional. This can imply that demand split depends on the direction: if the demand volume were considered from node 7 to node 6, then the shortest paths would remain the same as in the other direction, but the split would change to the uniform one assigning one-third of the volume to each of the three paths.

The process of determining the weight system under the ECMP rule encounters the same difficulties as the single-shortest-path allocation case (Exercise 2.6). Incidentally, despite these difficulties, in the example shown in Figure 2.6, the weight system assigning weights equal to 2 on all links except for weight equal to 1 on links 3-4 and 4-5 induces a feasible solution under the ECMP rule.

Shortest-path based routing design will be covered in detail in Chapter 7.

2.6 FAIR NETWORKS

We will now consider another yet different type of design problem. So far, we have assumed that the demand volumes are given and the network is required to route these volumes. Consider a network where the demand is *elastic*. Elasticity means that each demand can consume any bandwidth assigned to its path, perhaps within certain predefined bounds. This assumption corresponds, for instance, to a network with demands generating elastic traffic which can accommodate the changing bandwidth currently assigned to it, such as traffic generated during Internet web sessions.

A general problem with such networks is how and how much of demand volume to assign to each demand so that the capacities of links are not exceeded; to simplify, we will first assume that a fixed path is given for each demand to carry its (elastic) demand volume. In other words, in this problem, we have capacity constraints along the line of those listed in Problem (2.4.15), but no particular value for h_d in the demand constraint is assumed, which means that the demand constraint is needed to be satisfied within some predefined bounds. An obvious initial solution could be to assign demand

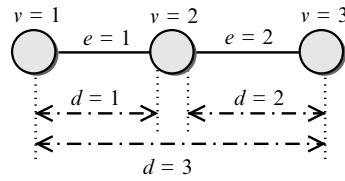


FIGURE 2.8 Two-Link Network

volume on its lower bound if this leads to satisfying the capacity constraints; if not, the problem is not feasible at all. Assuming feasibility, we still want the network to carry more than the lower bound of the demand volume such that some level of fairness among different demands is maintained, raising a question on how to assign volumes to the demands in a fair way. Furthermore, we are interested in understanding the impact on network throughput, which is defined to be the total demand volume carried by the network.

By far, the best known fairness criterion is *Max-Min Fairness* (MMF), also called *equity*. In the pure elastic case with no bounds, the first step of obtaining the MMF solution consists in assigning the same maximal volume to all demands, assuring that minimal assignment is maximized. Consider the network in Figure 2.8. It has $E = 2$ links, $V = 3$ nodes, and $D = 3$ demands. The link capacities are $c_1 = c_2 = 1\frac{1}{2}$. Clearly, all demands have exactly one candidate path. Under the MMF principle, the first step of the solution results in the following flow allocation: $x_{11} = x_{21} = x_{31} = \frac{3}{4}$. In fact, in this case, we are already at the final solution since all capacities are exhausted. Otherwise, if after the first step, some free capacity is still present, the process of increasing demand volumes for those demands (for which it is possible) would be continued. In this way, the second minimal assignment is maximized and so on. To illustrate this point, suppose we increase the capacity of $e = 2$ to 2 units. Then, after the first step, we will have $\frac{1}{2}$ unit of capacity left on link $e = 2$. Thus, demand $d = 2$ (and only this demand) can utilize this unused capacity, thus increasing its assigned demand volume to $1\frac{1}{4}$.

We notice that the MMF solution is fair from the user's viewpoint. On the other hand, if we were to optimize the total throughput in this network, we find that, by returning to the example in Figure 2.8, the maximal throughput is 3 which is achieved with a highly unfair solution $x_{11} = x_{21} = 1.5$ and $x_{31} = 0$. A simple but important observation is that what is good for the network may not necessarily be good for *all* individual users (demand). With the MMF solution, the resulting throughput is equal to $x_{11} + x_{21} + x_{31} = 2\frac{1}{4}$.

Clearly, the throughput degradation observed with the MMF rule is due to the fact that the same volume is assigned to every demand whatever the number of links on its path may be (the demands with long paths use up capacity of many links). Hence, a natural question arises whether there is some compromise solution between MMF and throughput maximization that has better throughput than MMF, yet is not as unfair as pure throughput maximization. The answer is yes, and one such fair allocation principle is called *Proportional Fairness* (PF).

The PF principle uses the *revenue objective* which consists in maximizing the sum of (natural) logarithms of the volumes assigned to demands, i.e., in our case, to maximize $\log x_{11} + \log x_{21} + \log x_{31}$. The rationale behind using the logarithmic function is that it does

not allow the assignment of zero volumes to demands (this would cause the revenue to be equal $-\infty$), and at the same time it makes it not profitable to assign too much volume to any demand (think why). Note that the objective function is non-linear, in contrast to the examples discussed so far in this chapter. You may check that the solution for the PF principle is $x_{11} = x_{21} = 1$ and $x_{31} = \frac{1}{2}$ (Exercise 2.7).

From the user's point of view, the PF solution is less fair than the MMF solution: the long flow (i.e., the flow using both links) $x_{31} = \frac{1}{2}$ is smaller than the two short flows (i.e., the flows using one link each) $x_{11} = x_{21} = 1$. However, because of favoring shorter flows, the PF allocation is more efficient in terms of throughput, in this case equal to $x_{11} + x_{21} + x_{31} = 2\frac{1}{2}$. This is a general observation: the PF solution does better than the MMF solution in terms of throughput, at the expense of fairness given to the users. Another way to examine this is that addressing fairness can reduce overall throughput. Hence, PF can be viewed as a compromise between throughput maximization and MMF.

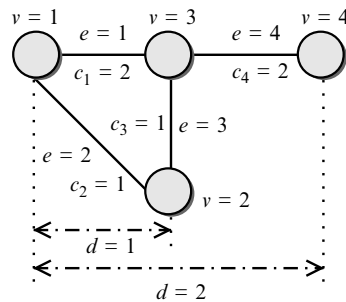
From the above example it follows that the capacitated flow allocation problem in the PF case is to find a flow allocation vector \mathbf{x} satisfying the capacity constraints (2.4.7) and maximizing the (logarithmic) revenue function:

$$R(\mathbf{x}) = \sum_d r_d \log X_d, \tag{2.6.1}$$

where $X_d = \sum_p x_{dp}$ is the total flow allocated to demand d and r_d is a positive weight (revenue) associated with demand d . This is a tractable mathematical formulation which can be solved, for instance, by introducing a linear approximation of the logarithmic function and solving the resulting linear programming problem.

Solving the MMF capacitated allocation problem is more complicated. In general, it is just not enough to find a flow allocation vector \mathbf{x} which maximizes the minimal (over all demands) total flow X_d assigned to all demands; if such a vector \mathbf{x} is found, then, in general, some link capacity will still be free and can be used for increasing flow allocations for at least a subset of demands.

To illustrate this point, consider the network depicted in Figure 2.9. Applying solution \mathbf{x}^1 leaves room for further increase flow for demand $d = 1$, leading to the final (optimal)



Solution \mathbf{x}^1 : $x_{11}^1 = 1$ on path $\{2\}$, $x_{21}^1 = 1$ on path $\{1,4\}$
 Solution \mathbf{x}^2 : $x_{11}^2 = 1$ on path $\{1, 3\}$, $x_{21}^2 = 1$ on path $\{2,3,4\}$

FIGURE 2.9 Multiple Optimal Allocation Vectors

MMF allocation vector:

$$\begin{aligned} x_{11} &= 1, & \mathcal{P}_{11} &= \{2\} \\ x_{12} &= 1, & \mathcal{P}_{12} &= \{1, 3\} \\ x_{21} &= 1, & \mathcal{P}_{21} &= \{1, 4\} \end{aligned}$$

Applying solution \mathbf{x}^2 does not leave room for any further improvement.

The final total allocation vector $\mathbf{X} = (X_1, X_2)$ is equal to $(2, 1)$, and has the crucial property that when sorted, it is *lexicographically maximal* in the set of all feasible sorted total allocation vectors. Recall that a sorted vector $\mathbf{X} = (X_1, X_2, \dots, X_D)$ is lexicographically greater than another sorted vector $\mathbf{Z} = (Z_1, Z_2, \dots, Z_D)$ if there exists $d, 0 \leq d < D$, such that $X_i = Z_i$ for $i = 1, 2, \dots, d$ and $X_{d+1} > Z_{d+1}$. Notice that it is not a straightforward task to find the MMF solution vector, since, as in our example, we do not know in advance which of the possible solutions from the first step leads to the final solution (in general, there can be many steps to perform).

The uncapacitated counterparts of the above presented capacitated allocation problems are also of interest, especially in the PF case. Chapter 8 covers fair network design in depth.

2.7 TOPOLOGICAL DESIGN

Another important class of design problems is *topological design problems*. The basic feature of topological design problems is that in the cost function we take into account not only the capacity-dependent costs of links, ξ_e , but also the link installation (“opening”) costs κ_e ; thus, the augmented cost function to be minimized is of the form:

$$F = \sum_e \xi_e y_e + \sum_e \kappa_e u_e \quad (2.7.1)$$

where the binary variable u_e indicates if link e is installed or not (u_e equals 1 or 0, respectively). The resulting uncapacitated design problem consists in minimizing the augmented cost function (2.7.1) subject to the demand constraints (2.4.3), the capacity constraints (2.4.7), and the additional constraint

$$y_e \leq \Delta u_e, \quad e = 1, 2, \dots, E, \quad (2.7.2)$$

where Δ is an appropriate large constant. Observe that the binary variable u_e , through constraint (2.7.2), forces the result $y_e = 0$ whenever $u_e = 0$, i.e., when link e is not installed.

If the installation costs for the network from Figure 2.4 are assumed to be $\kappa = (\kappa_1, \kappa_2, \dots, \kappa_5) = (50, 1, 1, 1, 50)$, then the optimal network topology becomes a tree composed of links $e = 2, e = 3$, and $e = 4$ ($u_2^* = u_3^* = u_4^* = 1$ and $u_1^* = u_5^* = 0$) with the installation cost $\sum_e \kappa_e u_e^* = 3$ and the capacity-dependent cost $\sum_e \xi_e y_e = 160$.

The difficulty of the topological design problem is considerable and comparable to that of the uncapacitated design problem with modular links described in Section 2.4. Topological design problems, including not only link but also node location, are discussed in Chapter 6.

2.8 RESTORATION DESIGN

So far in our design problems, we have considered the network to be in normal operational state, i.e., no failure consideration, either for a link or a node, is taken into account. We now add a new dimension to the NDPs by taking failures into account which will be referred to as *restoration design problems*. For example, in restoration design problems the link failures are taken explicitly into account in the formulation of the optimization model. For instance, we may assume a failure scenario in which each link can become totally unavailable, still different links do not fail simultaneously. Hence, each of the resulting *failure states* or *failure situations* consists of the failure of one link at a time, i.e., the rest being fully operative. For the network of Figure 2.3 this results in 5 failure states, $s = 1, 2, \dots, 5$ corresponding to each link: in state s link $e = s$ is failed and the remaining links are intact. For convenience, the normal operational state where all components are fully available, is labeled with $s = 0$.

Suppose that we wish to solve the restoration design problem (RDP) for the example network with the link costs and demand volumes given in Figure 2.4. This means that we want to find the cheapest link capacity configuration together with routing and flow allocation in such a way that in all states ($s = 0, 1, \dots, 5$) the demand volumes are fully realized (100% demand protection). Recall the design problem presented in (2.4.10) which was modified by augmenting the candidate path list of demand $d = 1$ (refer to (2.4.14)) and consider the RDP for this modified problem. Due to state s , we now need to introduce an identifier for s . Thus, the demand constraint in (2.4.14) for demand $d = 1$, i.e.,

$$x_{11} + x_{12} = 15$$

is extended to reflect the state s by introducing this in the subscript as follows:

$$x_{11s} + x_{12s} = 15, \quad s = 0, 1, \dots, 5.$$

Simply put, we now have six equations instead of one for demand $d = 1$ due to the normal state plus all the failure states.

We can now generalize the above demand constraints to capture all the states and demands. That is, we can write the demand constraints for all demands and states as:

$$\sum_p x_{dps} = h_d, \quad p = 1, 2, \dots, P_d \quad s = 0, 1, \dots, S. \quad (2.8.1a)$$

Now, let us consider capacity constraints. If we now incorporate failure state $s = 1 (= e)$, then the capacity feasibility constraints for $e = 1$ are of the following form:

$$\begin{aligned} s = 0 : & \quad x_{120} + x_{310} \leq y_1 \\ s = 1 : & \quad x_{121} + x_{311} \leq 0 \\ s = 2 : & \quad x_{122} + x_{312} \leq y_1 \\ & \quad \dots \end{aligned}$$

Again, we have six inequalities for each link. Note that since $s = 1$ corresponds to the failure of link $e = 1$, no capacity should be assigned to link $e = 1$ in this state. This is reflected by the right-hand side of the relevant constraint set to zero (see above).

If we use the notation α_{es} to denote 1 if link e is up and 0 if it is down in state s , then we can compactly write the new set of inequalities for capacity constraints in general as:

$$\sum_d \sum_p \delta_{edp} x_{dps} \leq \alpha_{es} y_e, \quad s = 0, 1, \dots, S \quad e = 1, 2, \dots, E. \quad (2.8.1b)$$

This formulation can be referred to as the scenario where we are allowed to freely rearrange the flows in case of a failure. This may not necessarily be the case in all practical networks (refer to Exercise 2.8). The optimal flows for the robust design case (under full re-arrangement) corresponding to Problem (2.4.14) are as follows:

$$\begin{array}{l} s = 0: \quad x_{110}^* = 0 \quad x_{120}^* = 15 \quad x_{210}^* = 20 \quad x_{220}^* = 0 \quad x_{310}^* = 0 \quad x_{320}^* = 10 \\ s = 1: \quad x_{111}^* = 15 \quad x_{121}^* = 0 \quad x_{211}^* = 20 \quad x_{221}^* = 0 \quad x_{311}^* = 0 \quad x_{321}^* = 10 \\ s = 2: \quad x_{112}^* = 0 \quad x_{122}^* = 15 \quad x_{212}^* = 20 \quad x_{222}^* = 0 \quad x_{312}^* = 10 \quad x_{322}^* = 0 \\ s = 3: \quad x_{113}^* = 0 \quad x_{123}^* = 15 \quad x_{213}^* = 20 \quad x_{223}^* = 0 \quad x_{313}^* = 10 \quad x_{323}^* = 0 \\ s = 4: \quad x_{114}^* = 0 \quad x_{124}^* = 15 \quad x_{214}^* = 20 \quad x_{224}^* = 0 \quad x_{314}^* = 0 \quad x_{324}^* = 10 \\ s = 5: \quad x_{115}^* = 15 \quad x_{125}^* = 0 \quad x_{214}^* = 0 \quad x_{225}^* = 20 \quad x_{315}^* = 0 \quad x_{325}^* = 10. \end{array}$$

The optimal capacity y_e^* of link e implied by the above specified flows is computed as the maximum load of the link over all states $s = 0, 1, \dots, 5$. Hence:

$$y_1^* = 25 \quad y_2^* = 25 \quad y_3^* = 30 \quad y_4^* = 35 \quad y_5^* = 35.$$

This results in the optimal cost of $F^* = 245$, showing that the robust network can be considerably more expensive than the cost of network design for just the normal state; recall from Section 2.4 that the cost of the cheapest network without failure consideration was $F^* = 85$.

In the above example, it so happens that the optimal flow allocations in all situations are non-bifurcated. The following simple example shows that it is not always the case. Consider the network in Figure 2.10 with two nodes (and hence only one demand $d = 1$), three links ($E = 3$), and three failure situations ($S = 3$). The demand volume in all situations is the same, $h_{1s} = 3$, where $s = 0, 1, 2, 3$. The unit cost of all three links is the same and equal to 1. In the failure situations only one link is failed. As shown in the figure, optimal link capacities are all equal to $1\frac{1}{2}$ (figures in parentheses). The optimal flows for all situations are given by the figures without parentheses. The optimal cost is $F^* = 4\frac{1}{2}$.

If we look for non-bifurcated solutions in all situations, then the resulting solution will be more expensive as illustrated in Figure 2.11. The optimal cost $F^* = 6$ is greater than for the previous non-bifurcated solution.

Design problems related to network robustness to failures does not, in general, follow the shortest-path rule (2.4.13) that works for the design of networks for the normal operating state; furthermore, the number of equations, inequalities, and variables grows with the introduction of states to capture different failure situations which can be seen, for example, from (2.8.1). Thus, restoration design problems are often difficult and time consuming to solve; nevertheless, they are an important class of design problems and can help to find network configurations that are resilient and survivable.

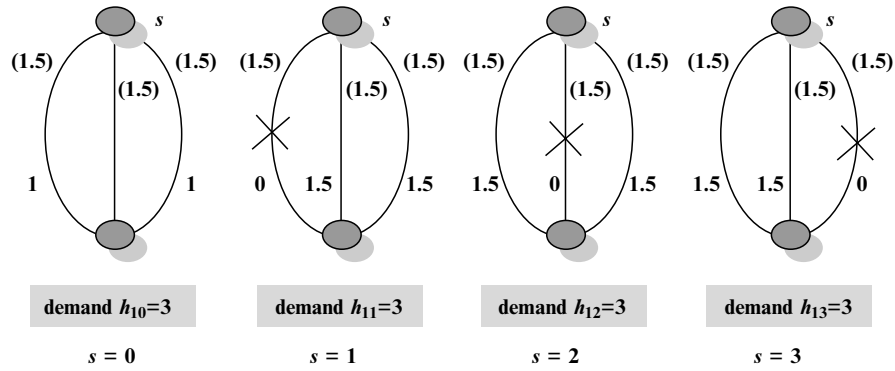


FIGURE 2.10 A Bifurcated Solution

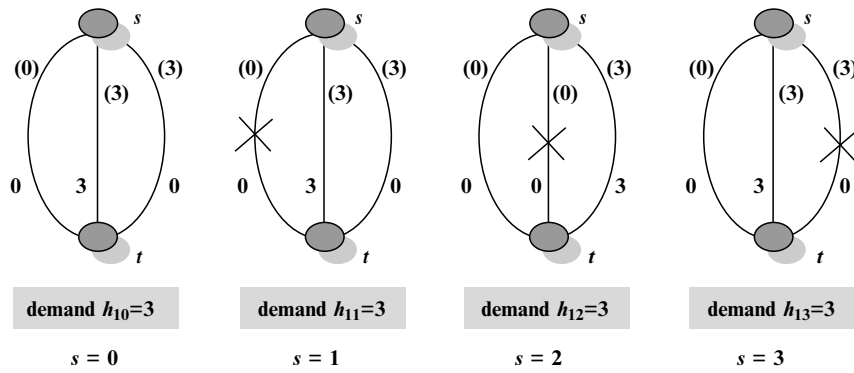


FIGURE 2.11 A Non-Bifurcated Solution

The example restoration design problem considered in this section involves the *path protection mechanism* for flow re-arrangement in failure situations. Note that other mechanisms are possible, for instance *link protection*. It is worth mentioning that these mechanisms make use of a common pool of shared protection capacity. More about restoration design can be found in Chapter 9 and related algorithms are discussed in Chapter 10.

2.9 *MULTI-LAYER NETWORKS MODELING

With respect to design problems presented so far in this chapter, the robustness issue that considers restoration adds a new dimension to NDPs, both in the sense of importance and difficulty. We are now about to introduce yet another important design dimension where the layered structure of the network resources is taken into account in the network modeling. The modeling is going to look complicated when you try to understand the entire relationship for the first time. The best way to think about this is to imagine you have different multi-commodity flow problems for each layer which are combined together in

such a way that the upper layer imposes demand on the neighboring lower layer. If you are not ready for it yet, you may skip this section now and come back to it later (especially when you are about to read Chapter 12 in this book). However, we do not want you to ignore its significance since networks are inherently multi-layered (which is still commonly overlooked in the current literature), and it is important to understand how the relationship works between different layers. Recall that we have described the notion of multi-layer earlier in Section 1.6.

In a nutshell, the resources (links and nodes) of communication and computer networks are configured in a multi-layered fashion, forming a hierarchical structure with each layer being a proper network on its own. The links of an upper layer are formed using paths of the lower layer, and this pattern repeats as one goes down the resources hierarchy. For instance, in a public-switched telephone network there is a layer of trunk groups and the trunk groups are realized by means of transmission paths configured in the transmission facility (or simply, facility) layer forming a layer below [Rey83, p. 83]. The digital links in the facility layer are formed by means of the paths in the optical layer, and so on, until eventually the conduit layer is reached. Similarly, IP networks can be provided over ATM, multi-protocol label switching (MPLS), SONET, or WDM. In fact, we can have more than two layers, for example, in the case of IP over ATM over SONET network.

While some network providers may own resources only at one or two neighboring layers, some large network providers own all of the layers. For the latter, it is essential to address the design of multi-layer networks.

Consider the network example depicted in Figure 2.12. The network consists of two layers of resources (Layer 1: equipment layer, Layer 2: virtual capacity layer) and an additional, auxiliary layer (Layer 3: demand layer) used merely to specify the demands. Altogether the network has two resource layers plus an auxiliary layer for the demand – this can be thought of as three layers; taking a similar view, the networks considered in the previous sections can be considered as two-layer networks composed of one resource layer (the lower, resource layer) and one demand layer (the upper, demand layer). In the case considered here, links are formed in the two lower resource layers, which is a natural extension of the one-layer resource network structure considered so far. For each demand d its demand volume h_d is realized by means of flows assigned to paths of Layer 2. The demands can be treated as the links of Layer 3 so we can say that capacity h_d of each link d of Layer 3 is realized by means of flows x in Layer 2. If we sum up the flows through each link e of Layer 2 then the resulting loads determine link capacity vector y of the layer. The next step is analogous. The capacity of each link e in Layer 2 is realized by means of flow in Layer 1, and the resulting Layer 1 flows z determine the load of each link g of Layer 1, and hence its capacity w_g . The cost of the resulting network configuration is eventually determined as the cost of the capacity of the links of Layer 1. In regard to nodes in different layers, we assume that the relationship is as follows: if a node appears in an upper layer, then it automatically appears downwards in the layer hierarchy. The advantage of our assumption is that it simplifies explanation of the model.

Let us assume that node and link numbering are as specified in Figures 2.13a and 2.13b. We will keep the same node number as we traverse down the layers. The following demand volumes (capacities of the Layer 3 links) and the Layer 1 link unit costs are assumed.

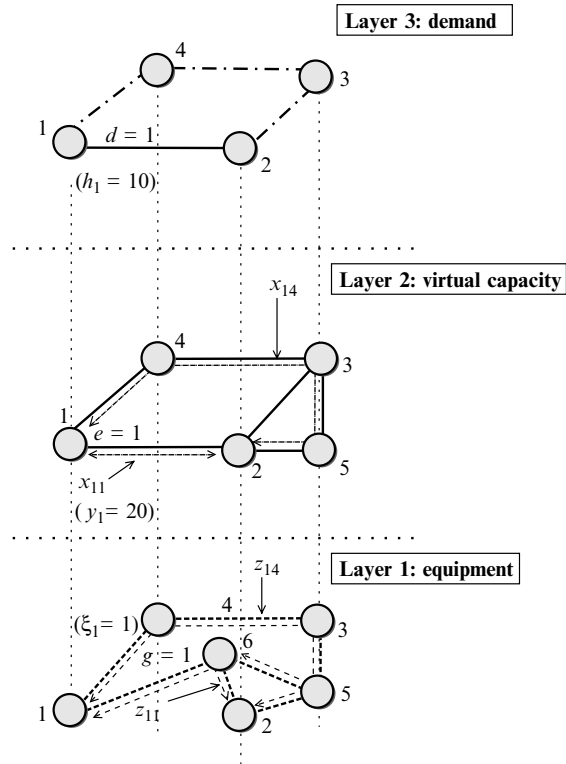


FIGURE 2.12 Three-Layer Network

Layer 3 (demand)

links:	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
end nodes:	1-2	1-3	1-4	2-3	2-4	3-4
capacities:	20	10	10	10	10	10

Layer 2

links:	$e = 1$	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$	$e = 7$
end nodes:	1-2	1-4	2-3	2-4	2-5	3-4	3-5

Layer 1

links:	$g = 1$	$g = 2$	$g = 3$	$g = 4$	$g = 5$	$g = 6$	$g = 7$	$g = 8$
end nodes:	1-4	1-6	2-5	2-6	3-4	3-5	4-6	5-6
unit cost ξ_g :	1	1	1	1	1	1	1	1.

Let us also assume the following candidate path lists where $\mathbb{P}_d = (\mathcal{P}_{d1}, \mathcal{P}_{d2}, \dots, \mathcal{P}_{dP_d})$ is used for denoting the Layer 2 candidate path list for demand d while $\mathbb{Q}_e = (\mathcal{Q}_{e1}, \mathcal{Q}_{e2}, \dots, \mathcal{Q}_{eQ_e})$ is used for denoting the Layer-1 candidate path list for link e :

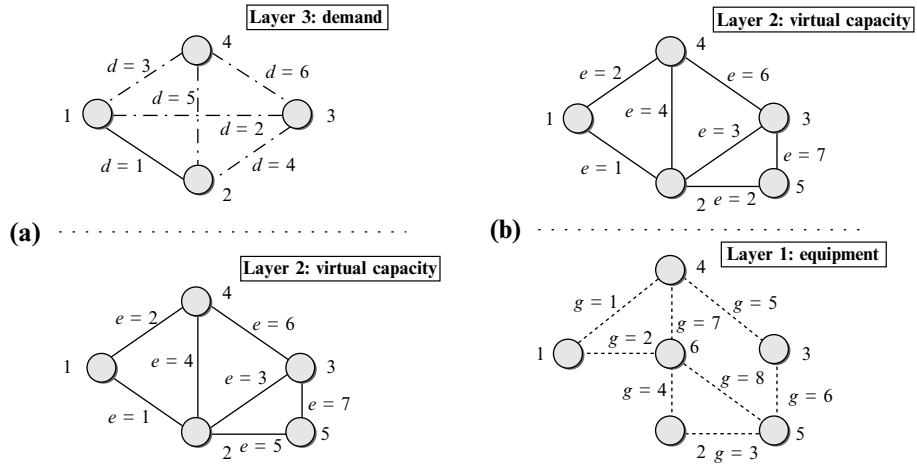


FIGURE 2.13 (a) Node and Link Numbering in Two Upper Layers; (b) Node and Link Numbering in Two Lower Layers

Layer 3/Layer 2

$d = 1$	$\mathcal{P}_{11} = \{1\}$	$\mathcal{P}_{12} = \{2, 4\}$	$\mathcal{P}_{13} = \{2, 3, 6\}$	$\mathcal{P}_{14} = \{2, 5, 6, 7\}$
$d = 2$	$\mathcal{P}_{21} = \{1, 3\}$	$\mathcal{P}_{22} = \{2, 6\}$	$\mathcal{P}_{23} = \{1, 5, 7\}$	
$d = 3$	$\mathcal{P}_{31} = \{2\}$	$\mathcal{P}_{32} = \{1, 4\}$	$\mathcal{P}_{33} = \{1, 3, 6\}$	$\mathcal{P}_{24} = \{1, 5, 6, 7\}$
$d = 4$	$\mathcal{P}_{41} = \{3\}$	$\mathcal{P}_{42} = \{4, 6\}$	$\mathcal{P}_{43} = \{5, 7\}$	$\mathcal{P}_{44} = \{1, 2, 6\}$
$d = 5$	$\mathcal{P}_{51} = \{4\}$	$\mathcal{P}_{52} = \{1, 2\}$	$\mathcal{P}_{53} = \{3, 6\}$	$\mathcal{P}_{54} = \{5, 6, 7\}$
$d = 6$	$\mathcal{P}_{61} = \{6\}$	$\mathcal{P}_{62} = \{3, 4\}$	$\mathcal{P}_{13} = \{1, 2, 3\}$	$\mathcal{P}_{14} = \{1, 2, 5, 7\}$

Layer 2/Layer 1

$e = 1$	$\mathcal{Q}_{11} = \{2, 4\}$	$\mathcal{Q}_{12} = \{1, 4, 7\}$	$\mathcal{Q}_{13} = \{1, 3, 5, 6\}$	
$e = 2$	$\mathcal{Q}_{21} = \{1\}$	$\mathcal{Q}_{22} = \{2, 7\}$	$\mathcal{Q}_{23} = \{2, 5, 6, 8\}$	$\mathcal{Q}_{24} = \{2, 3, 4, 5, 6\}$
$e = 3$	$\mathcal{Q}_{31} = \{3, 6\}$	$\mathcal{Q}_{32} = \{4, 5, 7\}$	$\mathcal{Q}_{33} = \{4, 6, 8\}$	$\mathcal{Q}_{34} = \{1, 2, 4, 5\}$
$e = 4$	$\mathcal{Q}_{41} = \{4, 7\}$	$\mathcal{Q}_{42} = \{1, 2, 4\}$	$\mathcal{Q}_{43} = \{3, 5, 6\}$	
$e = 5$	$\mathcal{Q}_{51} = \{3\}$	$\mathcal{Q}_{52} = \{4, 8\}$	$\mathcal{Q}_{43} = \{4, 5, 6, 7\}$	$\mathcal{Q}_{14} = \{1, 2, 4, 5, 6\}$
$e = 6$	$\mathcal{Q}_{61} = \{5\}$	$\mathcal{Q}_{62} = \{6, 7, 8\}$	$\mathcal{Q}_{63} = \{1, 2, 6, 8\}$	$\mathcal{Q}_{14} = \{1, 2, 3, 4, 6\}$
$e = 7$	$\mathcal{Q}_{71} = \{6\}$	$\mathcal{Q}_{72} = \{5, 7, 8\}$	$\mathcal{Q}_{73} = \{3, 4, 5, 7\}$	$\mathcal{Q}_{74} = \{1, 2, 3, 4, 5\}$

Note that the routing lists do not necessarily contain all possible paths. For instance, the middle layer path $\mathcal{P}_{24} = \{2, 3, 4\}$ is not on the path list of demand $d = 2$, and the lower layer path $\mathcal{Q}_{44} = \{3, 7, 8\}$ is not on the routing list of link $e = 4$.

Now consider the flow allocation vector \mathbf{x} defined by the following individual flows (flows equal to zero are not listed):

$$\begin{aligned} x_{11} = 10 \quad x_{14} = 10 \quad (h_1 = x_{11} + x_{14}, \text{ these two flows are depicted in Figure 2.12}) \\ x_{21} = 10 \quad x_{31} = 10 \quad x_{41} = 10 \quad x_{51} = 10 \quad x_{61} = 10. \end{aligned}$$

The above defined flows realize the assumed demand volumes, i.e., fulfill the demand constraints in the middle layer (Layer2):

$$\sum_p x_{dp} = h_d, \quad d = 1, 2, \dots, D \quad (2.9.1)$$

where the sum is taken over all paths p ($p = 1, 2, \dots, P_d$) on the routing list \mathbb{P}_d of demand d . We are already familiar with this and it has been addressed in Section 2.4. The middle layer link capacities, y_e , are equal (in this case) to the link loads, $\underline{y}_e = \underline{y}_e(\mathbf{x})$, resulting from the middle layer flow allocation vector \mathbf{x} and these are shown below:

$$y_1 = 20 \quad y_2 = 20 \quad y_3 = 20 \quad y_4 = 10 \quad y_5 = 10 \quad y_6 = 20 \quad y_7 = 10.$$

Using the link-path incidence coefficients defined by (2.4.5), the general formula defining the middle layer link loads can be written as:

$$\underline{y}_e = \sum_d \sum_p \delta_{edp} x_{dp}, \quad e = 1, 2, \dots, E. \quad (2.9.2)$$

The other allocation vector, \mathbf{z} , is the lower layer flow allocation vector which specifies how the capacity of each middle layer link e is realized by means of flows z_{eq} allocated to its candidate paths from the routing list in Layer 1. Suppose the vector \mathbf{z} is as follows (zero flows are not listed):

$$\begin{aligned} z_{11} = 15 \quad z_{13} = 5 \quad (y_1 = z_{11} + z_{13}, \text{ these two flows are depicted in Figure 2.12}) \\ z_{21} = 20 \quad z_{31} = 20 \quad z_{41} = 10 \quad z_{51} = 10 \quad z_{61} = 20 \quad z_{71} = 10 \end{aligned}$$

As before, the above defined flows realize the assumed middle layer link capacities, i.e., fulfill the demand constraints in the lower layer:

$$\sum_q z_{eq} = y_e, \quad e = 1, 2, \dots, E, \quad (2.9.3)$$

where the sum is taken over all paths q ($q = 1, 2, \dots, Q_e$) on the routing list \mathbb{Q}_e of link e . The lower link loads resulting from the lower layer flow allocation vector \mathbf{z} are equal to the lower layer link capacities w and are as follows:

$$w_1 = 20 \quad w_2 = 15 \quad w_3 = 35 \quad w_4 = 10 \quad w_5 = 25 \quad w_6 = 35 \quad w_7 = 10 \quad w_8 = 0.$$

Introducing the link-path incidence coefficients for Layer 1

$$\gamma_{geq} = \begin{cases} 1 & \text{if Layer 1 link } g \text{ belongs to path } q \text{ realizing Layer 2 link } e \\ 0 & \text{otherwise} \end{cases} \quad (2.9.4)$$

the general formula specifying the lower layer capacity constraint can be stated as:

$$\sum_e \sum_q \gamma_{geq} z_{eq} \leq w_g, \quad g = 1, 2, \dots, G, \quad (2.9.5)$$

where G is the total number of links in Layer 1 and the summation for each link g is taken over all flows in the lower layer.

Altogether, we have two demand constraints, (2.9.1) and (2.9.3), and two capacity constraints, (2.9.2) and (2.9.5). The demand vector $\mathbf{h} = (h_1, h_2, \dots, h_D)$ is defined through the capacities of the uppermost layer links (auxiliary Layer 3), and the cost vector $\xi = (\xi_1, \xi_2, \dots, \xi_G)$ specifies the unit costs of the lowermost layer (Layer 1) links. The resulting *two-resource layer dimensioning problem* (TRL-DP), being an extension of DP (2.4.11) from Section 2.4, is as follows:

For given vectors \mathbf{h} and ξ , and the network structure specified by the link-path incidence coefficients δ_{edp} and γ_{gek} , find flow vectors \mathbf{x} and \mathbf{z} , and link capacity vectors \mathbf{y} and \mathbf{w} in order to:

$$\begin{aligned} &\text{minimize } F = \sum_g \xi_g w_g \\ &\text{subject to constraints (2.9.1), (2.9.2), (2.9.3) and (2.9.5)} \end{aligned} \quad (2.9.6)$$

The example flow vectors \mathbf{x} and \mathbf{z} , and the corresponding link capacity vectors \mathbf{y} and \mathbf{w} constructed earlier in this section constitute a feasible solution for TRL-DP with $F = 150$ which is not optimal. To see this we observe that the optimal solutions obey the generalized shortest-path allocation rule, that extends the shortest-path allocation rule presented in Section 2.4 for DP (2.4.11):

Shortest-Path Allocation Rule for TRL-DP:

1. For each link e of Layer 2 select one of its shortest paths with respect to the unit costs ξ of the Layer 1 links. Let $q(e)$ be the index of such a shortest path for link e and let λ_e denote its length:

$$\lambda_e = \min \left\{ \sum_g \xi_g \gamma_{geq} : q = 1, 2, \dots, Q_e \right\}, \quad e = 1, 2, \dots, E. \quad (2.9.7)$$

2. For each demand d (link of Layer 3) select one of its shortest paths with respect to the unit costs $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_E)$ of the Layer 2 links defined by (2.9.7). Let $p(d)$ be the index of such a shortest-path for demand d and let μ_d denote its length:

$$\mu_d = \min \left\{ \sum_e \lambda_e \delta_{edp} : p = 1, 2, \dots, P_d \right\}, \quad d = 1, 2, \dots, D. \quad (2.9.8)$$

3. For each demand d allocate its whole demand volume h_d to its shortest path $p(d)$. Compute the resulting capacities (loads) y_e of the Layer 2 links.
4. For each link e allocate its whole capacity y_e to its shortest path $q(e)$. Compute the resulting capacities (loads) w_g of the Layer 1 links.

Any optimal solution \mathbf{x}^* , \mathbf{z}^* , \mathbf{y}^* , \mathbf{w}^* resulting from the application of the generalized shortest-path rule is optimal. In any case when there is more than one shortest-path, the demand volume or link capacity can be split arbitrarily among the shortest paths. Notice that the optimal cost is equal to:

$$\mathbf{F}^* = \sum_d \mu_d h_d. \quad (2.9.9)$$

For the considered network example one of the optimal solutions resulting from the generalized shortest-path rule is shown below:

$$\begin{array}{cccccccc} \lambda_1 = 2 & \lambda_2 = 1 & \lambda_3 = 2 & \lambda_4 = 2 & \lambda_5 = 1 & \lambda_6 = 1 & \lambda_7 = 1 & \\ q(1) = 1 & q(2) = 1 & q(3) = 1 & q(4) = 1 & q(5) = 1 & q(6) = 1 & q(7) = 1 & \\ \mu_1 = 2 & \mu_2 = 2 & \mu_3 = 1 & \mu_4 = 2 & \mu_5 = 1 & \mu_6 = 1 & & \\ p(1) = 1 & p(2) = 2 & p(3) = 1 & p(4) = 2 & p(5) = 1 & p(6) = 1 & & \\ \\ h_1 = 20 & h_2 = 10 & h_3 = 10 & h_4 = 10 & h_5 = 10 & h_6 = 10 & & \\ x_{11} = 20 & x_{22} = 10 & x_{31} = 10 & x_{42} = 10 & x_{51} = 10 & x_{61} = 10 & & \\ y_1 = 20 & y_2 = 20 & y_3 = 0 & y_4 = 10 & y_5 = 10 & y_6 = 10 & y_7 = 10 & \\ \\ z_{11} = 20 & z_{21} = 20 & z_{31} = 10 & z_{41} = 10 & z_{51} = 10 & z_{61} = 10 & z_{71} = 10 & \\ w_1 = 20 & w_2 = 20 & w_3 = 20 & w_4 = 10 & w_5 = 10 & w_6 = 20 & w_7 = 10 & w_8 = 20 \end{array}$$

$$\mathbf{F}^* = 110.$$

The generalization of the described modeling approach and the resulting TRL-DP to more than two resource layers is straightforward. Also, the shortest-path allocation rule easily generalizes to more layers. This will be discussed later in Chapter 12.

The multi-layer design can also be generalized to take the restoration issue into account. It appears that an acceptable failure model, in most cases, is obtained when the failure situations are restricted only to the failures of the Layer 1 (lower layer) links. Several extensions of TRL-DP are possible, depending on the number of layers in which the flow re-arrangement is admissible. One such generalization is obtained when the flow re-arrangement is allowed in both the “true” resource layers (i.e., in Layer 1 and Layer 2). Two more problems arise when the re-arrangement is permitted only in Layer 1 or only in Layer 2, respectively.

Multi-layer design problems and algorithms will be treated in Chapter 12 and partly in Chapter 13.

2.10 SUMMARY

In this chapter, we have discussed a set of representative NDPs using small network examples. We have started with a simple capacitated allocation problem involving minimization of flow routing. This example served as a means for introducing two basic formulation types: link-path formulation (Section 2.1) and node-link formulation (Section 2.2). In both cases we have used a notation system which we call node-based-identifier representation.

Then, in Section 2.3, we have demonstrated that this seemingly (at first sight) nice notation has certain fundamental drawbacks and that, in fact, another notation, called link-demand-path-identifier notation, is much more convenient for general formulations of NDPs, should they be in the link-path or in the node-link formulation.

Having clarified the notational issues, we have proceeded to discuss an uncapacitated network dimensioning problem (Section 2.4), and then to additional and more complicated problems involving the shortest-path routing (Section 2.5), fair networks (Section 2.6), and topological design (Section 2.7), through restoration design (for network protection against failures, Section 2.8) and to end up with the discussion of multi-layer NDPs.

It should be noted that the design problems we have introduced here are classified as multi-commodity network flow problems. While introducing consecutive design problems we have occasionally mentioned that certain problem formulations are simple and others are difficult or complex. As we are not yet prepared for a more precise discussion of these issues, we will now only make some brief comments. Incidentally, multi-commodity network flow problems are frequently linear programming (LP) problems as long as, roughly speaking, the objective function is linear and the bifurcated flows are allowed as independent decision variables. Examples of valid LP design problems can be found in Sections 2.4, 2.8, and 2.9. As you may know, in most cases, LP problems can be effectively solved using the well known simplex algorithm. For some problems (as for DP, refer to Section 2.4), their very special structure permits the creation of even more effective algorithms, such as the shortest-path allocation rule; such problems can be called simple.

Many of the considered problems are not LP problems. These are for example problems involving modular links (Section 2.4), shortest-path routing (Section 2.5), fairness (Section 2.6), and topological design (Section 2.7). These problems are in general difficult; algorithmic approaches to solve such problems will be discussed later in Chapter 5.

Apparently, Kalaba and Juncosa [KJ56] in 1956 were the first to address communication NDPs using a multi-commodity flow approach; incidently, their formulation can be considered as the first node-link-based multi-commodity flow representation where they used the term “message” to generically describe communication demand. The reader may find a comment from this paper rather interesting; to quote “In a system such as the Western Union System, which has some 15 regional switching centers all connected to each other, an optimal problem of this type would have about 450 conditions (constraints) and involve around 3,000 variables”. By referencing the Kalaba-Juncosa paper, Ford and Fulkerson [FF58] were perhaps the first ones to introduce link-path representation to formulate the maximal flow multi-commodity problem; incidently, the origin of the “delta” notation (i.e., δ_{edp}) can be attributed to this work.

For solving LP problems, a commonly known method called simplex, originally developed by G.B. Dantzig [Dan63], can be effectively used in most cases, possibly enforced by some kind of decomposition required by large network instances. In effect, the problems enjoying LP formulations can be roughly classified as effectively solvable, as in practice the simplex methods works in the time proportional to the number of optimization variables.

Deviations from linearity in problem formulations can pose problems. This issue will be discussed in more detail in the summary of Chapter 4 (Section 4.7). Here we only mention

that whenever binary/integral variables are necessary in the problem formulations, resulting in the so-called mixed-integer programming (MIP) formulations (which are basically LP formulations with additional requirements for some variables to be binary/integral) then we can expect with great likelihood that the problems are *NP*-complete (refer to Appendix B) and intrinsically cannot be solved in an exact way in a reasonable time for large networks. For such problems approximate or heuristic methods are used.

No further reading is suggested for this particular chapter, as all of its discussion will be extended in the subsequent chapters of this book.

EXERCISES FOR CHAPTER 2

- 2.1. Consider the formulation (2.1.2). Now suppose there is no demand between node pair 1 and 3, i.e., $\hat{h}_{13} = 0$. Rewrite the complete formulation for this modified problem in link-demand-path-identifier-based notation.
- 2.2. Find other feasible solutions (other than the one already discussed) that satisfy the set of equations and inequalities (2.1.1a).
- 2.3. Identify another goal that can be of interest in a network and formulate the corresponding objective function for the three-node example given by constraints (2.1.2). Determine the optimal solution.
- 2.4. Think of a situation where the node-identifier-based formulation could be advantageous over the link-demand-path-identifier-based formulation. (Hint: consider a situation when a number of hops for a path is fixed at a certain value for all paths and the network nodes are fully connected by links.)
- 2.5. Apply the general formulation (2.4.11) to specify an instance of problem DP for the three-node network (analogous to Problem (2.3.1)).
- 2.6. Consider the example depicted in Figure 2.6 with link capacities altered as follows: all link capacities are equal to 1 except for link 3-5 which has capacity $1\frac{1}{2}$ and links 3-4 and 4-5 which have the same capacity $\frac{1}{2}$. Does a weight system exist that induces a feasible solution under the ECMP rule?
- 2.7. Derive the PF solution for the network in Figure 2.8.
- 2.8. Variation of (2.8.1): consider the restoration design case where the flow that was originally assigned to the normal state may not be re-arranged in the case of a failure. How would the set of equations and inequalities change corresponding to (2.8.1)?