Chapter 1
Robust Network Planning

Matthew Roughan
School of Mathematical Sciences
University of Adelaide
matthew.roughan@adelaide.edu.au

1.1 Introduction

Building a network encompasses many tasks: from network planning to hardware installation and configuration, to ongoing maintenance. In this chapter we focus on the process of *network planning*. It is possible (though not always wise) to design a small network by eye, but automated techniques are needed for the design of large networks. The complexity of such networks means that any "ad hoc" design will suffer from unacceptable performance, reliability and/or cost penalties.

Network planning involves a series of quantitative tasks: measuring the current network traffic and the network itself; predicting future network demands; determining the optimal allocation of resources to meet a set of goals; and validating the implementation. A simple example is capacity planning: deciding the future capacities of links in order to carry forecast traffic loads, while minimizing the network cost. Other examples include traffic engineering (balancing loads across our existing network) and choosing the locations of Points-of-Presence (PoPs) though we do not consider this latter problem in detail in this chapter because of its dependence on economic and demographic concerns rather than those of networking.

Many academic papers about these topics focus on individual components of network planning: for instance, how to make appropriate measurements, or on particular optimization algorithms. In contrast, in this chapter we will take a system view. We will present each part as a component of a larger system of network planning. In the process of describing how the various components of network planning interrelate, we observe several recurring themes:

1. *Internet measurements are of varying quality.* They are often imperfect or incomplete and can contain errors or ambiguities. Measurements should not be taken at face value, but need to be continually recalibrated [48], so that we have some understanding of the errors, and can take them into account in subsequent processing. We will describe common measurement strategies in Section 1.2.

2. *Analysis and Modelling* of data can allow us to estimate and predict otherwise unmeasurable quantities. However, in the words of Box and Draper, "Essentially, all models are wrong, but some are useful" [9]. We must be continually concerned with the quality of model based predictions. In particular we must consider where they apply, and the consequences of using an inaccurate model. A number of key traffic models are described in Section 1.3, and their use in prediction is described in Section 1.4.
3. *Decisions based on quantitative data are at best as good as their input data, but can be worse.* The quality of input data and resulting predictions are variable, and this can have consequences for the type of planning processes we can apply. Numerical techniques that are sensitive to such errors are not suitable for network engineering. Discussion of robust, quantitative network engineering is the main consideration of Sections 1.5 and 1.6.

Noting all of the above, it should not be surprising that a robust design process requires validation. The strategy of "set and forget" is not viable in today's rapidly changing networking environment. The errors in initial measurements, predictions, and the possibility for mistakes in deployment mean that we need to test whether the implementation of our plan has achieved our goals.

Moreover, actions taken at one level of operations may impact others. For example, Qiu *et al.* [51] noted that attempts to balance network loads by changing routing can cause higher-layer adaptive mechanisms such as overlay networks to change their decisions. These higher-level changes alter traffic, leading to a change of the circumstances which originally lead us to reroute traffic.

Thus the process of **measure→analyze/predict→control→validate** should not stop. Once we complete this process the cycle begins again, with our validation measurements feeding back into the process as the input for the next round of network planning, as illustrated in Figure 1.1. This cycle allows our planning process to correct problems, leading to a robust process.

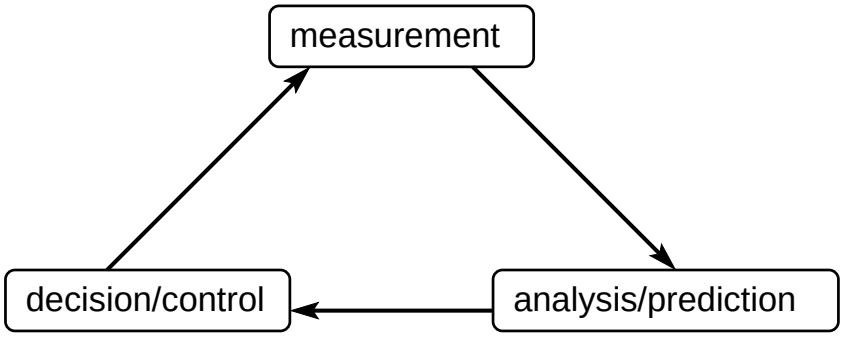

Fig. 1.1 Robust network planning is cyclic.

In many ways this resembles the more formal feedback used in control systems, though robust planning involves a range of tasks not typically modelled in formal control-theory. For instance, the lead times for deploying network components such as new routers are still quite long. It can take months to install, configure and test new equipment when done methodically. Even customers ordering access facilities can experience relatively long intervals from order to delivery, despite the obvious benefits to both parties of a quick startup. So if our network plan is incorrect, we cannot wait for the planning cycle to complete to redress the problem.

We need processes where the cycle time is shorter. It is relatively simple to reroute traffic across a network. It usually requires only small changes to router configurations, and so can be done from day to day (or even faster if automated). Rebalancing traffic load in the short term — in the interim before the network capacities can be physically changed — can alleviate congestion caused by failures of traffic predictions. This process is called *traffic engineering*.

Another aspect of robust planning is incorporation of reliability analysis. Internet switches and routers fail from time to time, and must sometimes be removed from service for maintenance. The links connecting routers are also susceptible to failures, given their vulnerability to natural or man-made accident (the canonical example is the careless back-hoe driver). Most network managers plan for the possibility of node or link failures by including redundant routers and links in their network. A network failure typically results in traffic being rerouted using these redundant pathways. Often, however, network engineers do not plan for overloads that might occur as a result of the rerouted traffic. Again, we need a robust planning process that takes into account the potential failure loads. We call this approach *network reliability analysis*.

We organize this chapter around the key steps in network planning. We first consider the standard network measurements that are available today. Their characteristics determine much of what we can accomplish in network planning. We then consider models and predictions, and then finally the processes used in making decisions, and controlling our network. As noted, robust planning does not stop there, we must continue to monitor our network, but there are a number of additional steps we can perform in order to achieve a robust network plan and we consider them in the final section of this chapter.

The focus of this chapter is backbone networks. Though many of the techniques described here remain applicable to access networks, there are a number of critical differences. For instance, access network traffic is often *very* bursty, and this affects the approaches we should adopt for prediction and capacity planning. Nevertheless, the fundamental ideas of robust planning that we discuss here remain valid.

1.2 Standard network measurements

Internet measurements are considered in more detail in Chapters 10 and 11, but a significant factor in network planning is the type of measurements available, and so we need some planning-specific discussion. In principle it is possible to collect extremely good data, but in practice the measurements are often flawed, and the nature of the flaws are important when considering how to use the data.

The traffic data we might like to collect is a packet trace, consisting of a record of all packets on a subsection of a network along with timestamps. There are various mechanisms for collecting such a trace, for instance, placing a splitter into an optical fiber, using a monitor port on a router, or simply running `tcpdump` on one of the hosts on a shared network segment. A packet trace gives us all of the information we could possibly need but is prohibitively expensive at the scale we require for planning. The problem with a packet trace (apart from the cost of installing dedicated devices) is that the amount of data involved can be enormous, for example, on an OC48 (2.5 Gbps) link, one might collect more than a terabyte of data per hour. More importantly, a packet trace is overkill. For planning we don't need such detail, but we do need good coverage of the whole network. Packet traces are only used on lower speed networks, or for specific studies of larger networks.

There are several approaches we can use to reduce data to a more manageable amount. Filtering, so that we view only a segment of the traffic (say the HTTP traffic) is useful for some tasks, but not planning. A more useful approach is aggregation, where we only store records for some aggregated version of the traffic, thereby reducing the number of such records needed. A common form of aggregation is at the flow-level where we aggregate the traffic through some common characteristics. The definition of "flow" depends on the keys used for aggregation, but we mean here flows aggregated by the five-tuple formed from IP source and destination address, TCP port numbers, and protocol number. Flow data is typically collected within some time frame, for instance, 15 minutes periods. What's more, flow level collection is often a feature of a router, and so doesn't require additional measurement infrastructure other than the Network Management Station (NMS) at which the data is stored. However, the volume of data can still be large (one network under study collected 500 GB of data per day), and the collection process may impact the performance of the router.

As a result, flow-level data is often collected in conjunction with a third method for data reduction: sampling. Sampling can be used both before the flows are created, and afterwards. Prior to flow aggregation, sampling is used at rates of around 1:100 to 1:500 packets. That is, less than 1% of packets are sampled. This has the advantage that less processing is required to construct flow records (reducing the load on the router collecting the flows) and typically fewer flow records will be created (reducing memory and data transmission requirements). However, sampling prior to flow aggregation does have flaws, most obviously it biases the data collection towards long flows. These flows (involving many packets) are much more likely to be sampled than short flows. However, this has rarely been seen as a prob-

lem in network planning where we are not typically concerned with the flow length distribution.

Sampling can also be used after flow aggregation to reduce the transmission and storage requirements for such data. The degree of sampling depends on the desired tradeoff between accuracy of measurements, and storage requirements for the data. Good statistical approaches for this sampling, and for estimating the resulting accuracy of the samples are available [16, 17], though, as noted above these are predominantly aimed at preserving details such as flow-length distributions which are largely inconsequential for the type of planning discussed here, so sampling prior to flow construction is often sufficient for planning.

Of more importance here is the fact that any type of sampling introduces errors into measurements. Any large-scale flow archives must involve significant sampling, and so will contain errors.

An alternative to flow-level data, is data collected via the Simple Network Management Protocol (SNMP) [39]. Its advantage over flow-level data collection is that it is more widely supported, and less vendor specific. However, the data provided is less detailed. SNMP allows an NMS to poll MIBs (Management Information Bases) at routers. Routers maintain a number of counters in these MIBs. The widely supported MIB-II contains counters of the number of packets and bytes transmitted and received at each interface of a router. In effect, we can see the traffic on each link of a network. In contrast to flow-level data, SNMP can only see link volumes, not where the traffic is going.

SNMP has a number of other issues with regard to data collection. The polling mechanism typically uses UDP (the User Datagram Protocol), and SNMP agents are given low priority at routers. Hence SNMP measurements are not reliable, and it is difficult to ensure that we obtain uniformly sampled time series. The result is missing and error-prone data.

Flow-level data contains only flow start and stop times, not details of packet arrivals, and typically SNMP is collected at 5 minute intervals. The limit on timescale of both data-sets is important in network planning. We can only see average traffic rates over these periods, not the variations inside these interval. However, congestion and subsequent packet loss often occur on much shorter timescales. The result is that such average measurements must always be used with care. Typically some overbuild of capacity is required to account for the sub-interval variations in traffic. The exact overbuild will depend on the network in question, and has typically been derived empirically through ongoing performance and traffic measurements. Values are usually fairly conservative in major backbones resulting in apparent underutilization (though this term is unfair as it concerns average utilizations not peak loads), and more aggressive in smaller networks.

In addition to traffic data, network planning requires a detailed view of any existing network. We need to know

- the (layer 3) topology (the locations of, and the links between routers);
- the network routing policies (for instance link weights in a shortest-path protocol, areas in protocols such as OSPF, and BGP policies where multiple inter-domain links exist); and

- the mapping between current layer 3 links and physical facilities (WDM equipment and optical fibers), and the details of the available physical network facilities and their associated costs.

The topology and routing data is principally needed to allow us to map traffic to links. The mapping is usually expressed through the *routing matrix*. Formally, $A = \{A_{ir}\}$ is the matrix defined by

$$A_{ir} = \begin{cases} F_{ir}, \text{ if traffic for } r \text{ traverses link } i \\ 0, \quad \text{otherwise} \end{cases} \tag{1.1}$$

where F_{ir} is the fraction of traffic from source/destination pair $r = (s,d)$ that traverses link i. A network with N nodes, and L links will have an $L \times N(N-1)$ routing matrix.

Network data is also used to assess how changes in one component will affect the network (e.g., how changes in OSPF link weights will impact link loads); determine shared risk-of-failure between links; and determine how to improve our network incrementally without completely rebuilding it in each planning cycle. The latter is an important point because although it might be preferable to rebuild a network from scratch, the capital value of legacy equipment usually prevents this option, except at rare intervals.

For a small, static network, the network data may be maintained in a database, however, best practice for large, complex, or dynamic networks is to use tools to extract the network structure directly from the network. There are several methods available for discovering this information. SNMP can provide this information through the use of various vendor tools (HP Openview, or Cisco NCM, for example), but it is not the most efficient approach. A preferable approach for finding layer 3 information is to parse the configuration files of routers directly, for instance as described in [22, 24]. The technique has been applied in a number of networks [5, 38]. The advantages of using configuration files are manifold. The detail of information available is unparalleled in other data sources. For instance, we can see details of the links (such as their composition should a single logical link be composed of more than one physical link).

The other major approach for garnering topology and routing information is to use a route monitor. Internet routing is built on top of distributed computations supported by routing protocols. The distribution of these protocols is often considered a critical component in ensuring reliability of the protocols in the face of network failures. The distribution also introduces a hook for topology discovery. If any router must be able to build its routing table from the routing information distributed through these protocols, then it must have considerable information about the network topology. Hence, we can place a dummy router into the network to collect such information. Such routing monitors have been deployed widely over the last few years. Their advantage is they can provide an up-to-date dynamic view. Examples of such monitors exist for OSPF [62, 63], and IS-IS [1, 30], as well as for BGP (the Border Gateway Protocol) [2, 3].

1.3 Analysis and Modeling of Internet Traffic

1.3.1 Traffic Matrices

We will now consider the analysis and modelling of Internet data, in particular traffic data. When considering inputs to network planning, we frequently return to the topic of *traffic matrices*. These are the measurements needed for many network planning tasks, and thus the natural structure around which we shall frame our analysis.

A Traffic Matrix (TM) describes the amount of traffic (the number of packets or more commonly bytes) transmitted from one point in a network to another during some time interval, and they are naturally represented by a three-dimensional data structure $T_t(i,j)$ which represents the traffic volume (in bytes or packets) from i to j during a time interval $[t, t+\Delta t)$. The locations i and j are generally considered to be physical geographic locations making i and j spatial variables. However, in the Internet, it is common to associate i and j with logical structures related to the address structure of the Internet, i.e. IP addresses, or natural groupings of such by common prefix corresponding to a subnet.

Origin/Destination Matrices: One natural approach to describe traffic matrices is with respect to traffic volumes between IP addresses or prefixes. We refer to this as an origin/destination TM because the IP addresses represent the closest approximation we have for the end-points of the network (though HTTP-proxies, firewalls, and NAT and other middle-boxes may be obscuring the true end-to-end semantics). IPv4 admits nearly 2^{32} potential addresses, so we cannot describe the full matrix at this level of granularity. Typically, such a traffic matrix would be aggregated into blocks of IP addresses (often using routing prefixes to form the blocks as these are natural units for the control of traffic). The origin/destination matrix is our ideal input for many network planning tasks, but the Internet is made up of many connected networks. Any one network operator only sees the traffic carried by its own network. This reduced visibility means that our observed traffic matrix is only a segment of the real network traffic. So we can't really observe the origin/destination TM. Instead we typically observe the ingress/egress traffic matrix.

Ingress/Egress vs Origin/Destination: A more practical TM, the ingress/egress TM provides traffic volumes from ingress link to egress link across a single network. Note that networks often interconnect at multiple points. The choice of which route to use for egress from a network can profoundly change the nature of ingress/egress TMs, so these may have quite different properties to the origin/destination matrix. Forming an ingress/egress TM from an origin/destination TM involves a simple mapping of prefixes to ingress/egress locations in a network, but in practice this mapping can be difficult unless we monitor traffic as it enters the network. We can infer egress points of traffic using the routing data described above, but inferring ingress is more difficult [22, 23], so it is better to measure this directly.

Spatial Granularity of Traffic Matrices: As we have started to see with origin/destination traffic matrices, we can measure them at various levels of granularity (or resolution). The same is true of ingress/egress TMs. At the finest level,

we measure traffic per ingress/egress link (or interface). However, it is common to aggregate this data to the ingress/egress router. We can often group routers into larger subgroups. A common such group is a Point-of-Presence (PoP), though there are other sub- and super- groupings (e.g. topologically-equivalent edge routers are sometimes grouped, or we may form a regional group). Given subsets S and D of locations, may simply aggregate a TM across these by taking

$$T_t(S,D) = \sum_{i \in S} \sum_{j \in D} T_t(i,j). \tag{1.2}$$

Typical large networks might have 10's of PoPs, and 100's of routers, and so such TMs are of a more workable size. In addition, as we aggregate traffic into larger groupings, statistical multiplexing reduces the relative variance of the traffic and allows us to perform better estimates of traffic properties such as the mean and variance.

Temporal Granularity of Traffic Matrices: We cannot make instantaneous measurements of a traffic matrix. All such observations occur over some time interval $[t, t+\Delta t)$. It would be useful to make the interval Δt smaller (for instance for detecting anomalies), but typically we face a tradeoff against the errors and uncertainties in our measurements. A longer time interval allows more "averaging-out" of errors, and minimizes the impact of missing data. The best choice of time interval for TMs is typically determined by the task at hand, and the network under study, but a common choice is a one hour interval. In addition to being easily understood by human operators, this interval integrates enough SNMP or flow-level data to reduce the impact of (typical) missing data and errors, while allowing us to still observe important diurnal patterns in the traffic.

1.3.2 Patterns in traffic

It is useful to have some understanding of the typical patterns we see in network traffic. Such patterns are only visible at a reasonable level of aggregation (otherwise random temporal variation dominates our view of the traffic), but for high degrees of aggregation (such as router-to-router traffic matrices on a large backbone network) the pattern can be very regular. There are two main types of patterns that have been observed: patterns across time, and patterns in the spatial structure. Each is discussed below.

Temporal Patterns: Internet traffic has been observed to follow both daily (diurnal) and weekly cycles [33–35, 57, 65]. The origin of these cycles is quite intuitive. They arise because most Internet traffic is currently generated by humans whose activities follow such cycles. Typical examples are shown in Figures 1.2 and 1.3. Fig-

ure 1.2 shows a RRD Tool graph[1] of the traffic on a link of the Australian Academic Research Network (AARNet). Figure 1.3 shows the total traffic entering AT&T's North American backbone network at a Point of Presence (PoP) over two consecutive weeks in May 2001. The figure illustrates the daily and weekly variations in the traffic by overlaying the traffic from the two weeks. The striking similarity between traffic patterns from week to week is a reflection of the high level of aggregation that we see in a major backbone network.

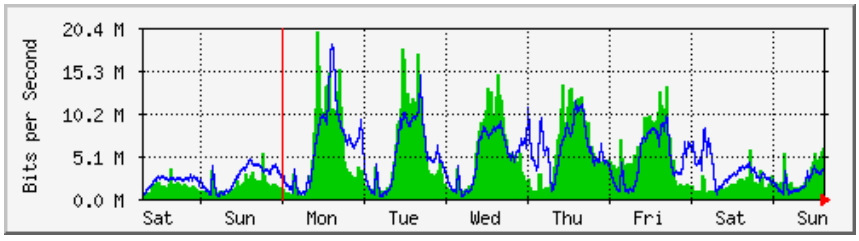

Fig. 1.2 Traffic on one link in the Australian Academic Research Network (AARNet) for just over one week. The two curves show traffic in either direction along the link.

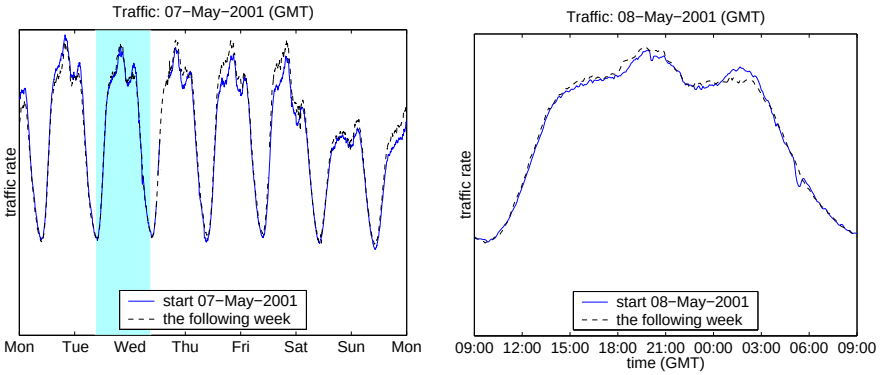

Fig. 1.3 Total traffic into a region over two consecutive weeks. The solid line is the first week's data (starting on May 7th), and the dashed line shows the second week's data. The second figure zooms in on the shaded region of the first.

The observation of cycles in traffic is not new. For many years they have been seen in telephony [13]. Typically telephone service capacity planning has been based on a "busy hour", i.e., the hour of the day that has the highest traffic. The time of the busy hour depends on the application and customer base. Access networks typically have many domestic consumers, and consequently their busy hour

[1] RRDTool (the Round Robin Database tool) [47] and its predecessor MRTG (the Multi-Router Traffic Grapher [46]) are perhaps the most common tools for collecting and displaying SNMP traffic data.

is in the evening when people are at home. On the other hand, the busy hour of business customers is typically during the day. Obviously, time-zones have an effect on the structure of the diurnal cycle in traffic, and so networks with a wide geographic dispersion may experience different busy hours on different parts of their network.

In addition to cyclical patterns, Internet traffic has shown strong growth over many years [45]. This long-term trend has often been approximated by exponential growth, although care must be taken because sometimes such estimates have been based on poor (short or erratic) data [45]. Long-term trends should be estimated from multiple years of carefully collected data.

One public example is the data collected by the Australian Bureau of Statistics (ABS)[2] who have collected historical data on Australian ISP traffic for many years. Figure 1.4 shows Australia's network traffic in petabytes per quarter with a log-y axis. Exponential growth appears as a straight line on the log-graph, so we can obtain simple predictions of traffic growth through linear regression. The figure shows such a prediction based on pre-2005 data. It is interesting to note that the most recent data point does not, as one might assume without analysis, represent a significant drop in traffic growth. Relative to the long-term data the last point simply represents a reversion to the long-term growth from rather exceptional traffic volumes over 2007. We will discuss such prediction in more detail in the following sections.

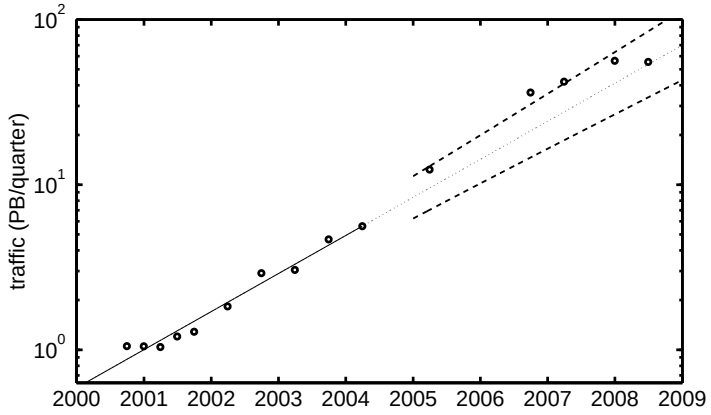

Fig. 1.4 ABS traffic measurements showing Australian Internet traffic, with an exponential fit to the data from 2000-2005. Data is shown by 'o', and the fit by the straight line. Note that the line continuing past 2005 is a prediction based on the pre-2005 data, showing also the 95th percentile confidence bounds for the predictions.

Standard time-series analysis [10] can be used to build a model of traffic containing long-term trends, cyclical components (often called *seasonal* components in other contexts) and random fluctuations. We will use the following notation here:

[2] www.abs.gov.au

$$S(t) = \text{seasonal (cyclical) component} \tag{1.3}$$
$$L(t) = \text{long-term trend} \tag{1.4}$$
$$W(t) = \text{random fluctuations} \tag{1.5}$$

The seasonal component is periodic, i.e., $S(t+kT_S) = S(t)$, for all integers k, where T_S is the period (which is either 24 hours or 1 week). Before we can consider how to estimate the seasonal (and trend) components of the traffic, we must model these components[3]. At the most basic level, consider the traffic to consist of two components, a time varying (but deterministic) mean $m(t)$ and a stochastic component $W(t)$. At this level we could construct the traffic by addition or multiplication of these components (both methods are used in econometric and census data). However, in traffic data, a more appropriate model [43, 56] is

$$x(t) = m(t) + \sqrt{am(t)}\, W(t), \tag{1.6}$$

where a is called the *peakedness* of the traffic, $W(t)$ is a stochastic process with zero mean, and unit variance, and $x(t)$ represents the average rate of some traffic (say a particular traffic matrix element) at time t. More highly aggregated traffic is smoother, and consequently would have a smaller value for a. The reason for this choice of model lies in the way network traffic behaves when aggregated. When multiple flows are aggregated onto a non-congested link, we should expect them to obey the same model (though perhaps with different parameters). Our model has this property: for instance, take N traffic streams x_i with mean m_i, peakedness a_i, and stochastic components which are independent realizations of a (zero mean, unit variance) Gaussian process. The multiplexed traffic stream is

$$x = \sum_{i=1}^{N} m_i + \sum_{i=1}^{N} \sqrt{a_i m_i}\, W_i. \tag{1.7}$$

The mean of the new process is $m = \sum_{i=1}^{N} m_i$, and the peakedness (derived from the variance) is $a = \frac{1}{m}\sum_{i=1}^{N} a_i m_i$, which is a weighted average of the component peakednesses. The relative variance becomes

$$V_x = Var\{x\}/E\{x\} = \frac{1}{m^2}\sum_{i=1}^{N} a_i m_i. \tag{1.8}$$

If we take identical streams, then the relative variance decreases as we multiplex more together, which is to be expected. The result is that in network traffic the level of aggregation is important in determining the relative variance: more highly aggregated traffic exhibits less random behavior. The data in Figure 1.3 from AT&T shows an aggregate of a very large number of customers (an entire PoP of one of North America's largest networks). The consequence is that we can see the traffic

[3] The reader should beware of methods which do not explicitly model the data, because in these methods there is often an implicit model.

is very smooth. In contrast the traffic shown in Figure 1.2 is much less aggregated, and shows more random fluctuations.

The model above is not perfect (none are) but it is useful because it (i) allows us to calculate variances for aggregated traffic streams in a consistent way and to use these when planning our network, and (ii) its parameters are relatively easy to measure, and therefore to use in traffic analysis. To do so, however, we find it useful to spilt the mean $m(t)$ into the cyclic component (which we denote $S(t)$) and the long-term trend $L(t)$ by taking the product

$$m(t) = L(t)S(t). \tag{1.9}$$

We combine the two components through a product because as the overall load increases the range of variation in the size of cycles also increases. When estimating parameters of our models, it is important to allow for unusual or anomalous events, for instance, a Denial of Service (DoS) attack. These events are rare (we hope), but it is important to separate them from the normal traffic. Such terms can sometimes be very large, but we don't plan network capacity to carry DoS attacks! The network is planned around the paying customers. We separate them by including an impulsive term, $I(t)$, in the model, so that the complete model is

$$x(t) = L(t)S(t) + \sqrt{aL(t)S(t)}\,W(t) + I(t). \tag{1.10}$$

We will further discuss this model in Section 1.4, where we will consider how to estimate its parameters, and to use it in prediction.

Spatial Patterns: Temporal models are adequate for many applications: for instance where we consider dimensioning of a single bottleneck link (perhaps in the design of an access network). However, spatial patterns in traffic provide us with addition planning capabilities. For instance, if two traffic sources are active at different times, then clearly we can carry them both with less capacity than if they activate simultaneously.

Spatial patterns refer to the structure of a Traffic Matrix (TM) at a single time interval. It is common that TM elements are strongly correlated because they show similar diurnal (and weekly) patterns. For example, in a typical network (without wide geographic distribution) one will find that the busy hour is almost the same for all elements of the TM, but there is additional structure.

For a start, TMs often come from skewed distributions. A common example is where the distribution follows a rough 80-20 law (80% of traffic is generated by the largest 20% of TM elements). Similar distributions have often been observed, though often even more skewed: for instance 90-10 laws are not uncommon. However the distribution is not "heavy-tailed". Observed distributions have shown a lighter tail than the log-normal distribution [55]. Consequently, traffic matrix work often concentrates on these larger flows, but traditional (rather than heavy-tailed) statistical techniques are still applicable.

Another simple feature one might naively expect of TMs — symmetry — is not present. Internet routing is naturally asymmetric, as is application traffic (a large amount of traffic still follows a client-server model which results in strongly

asymmetric traffic). Hence, the matrix will not (generally) be symmetric [21], i.e., $T(i,j) \neq T(j,i)$.

We observe some additional structure in these matrices. The simplest model that describes some of the observed structure is the *gravity model*. In network applications, gravity models have been used to model the volume of telephone calls in a network [31]. Gravity models take their name from Newton's law of gravitation, and commonly used by social scientists to model the movement of people, goods or information between geographic areas [49, 50, 64]. In Newton's law of gravitation the force is proportional to the product of the masses of the two objects divided by the distance squared. Similarly, in gravity models for interactions between cities, the relative strength of the interaction might be modeled as proportional to the product of the cities' populations, so a general formulation of a gravity model is given by

$$T(i,j) = \frac{R_i \cdot A_j}{f_{ij}}, \tag{1.11}$$

where R_i represents the *repulsive* factors that are associated with leaving from i; A_j represents the *attractive* factors that are associated with going to j; and f_{ij} is a friction factor from i to j. The gravity model was first used in the context of Internet traffic matrices in [68] where we can naturally interpret the repulsion factor R_i as the volume of incoming traffic at location i, and the attractivity factor A_j as the outgoing traffic volume at location j. The friction matrix (f_{ij}) encodes the locality information specific to different source-destination pairs, however, as locality is not as large a factor in Internet traffic as in the transport of physical goods, it is common to assume $f_{ij} = const$. The resulting gravity model simply states that the traffic exchanged between locations is proportional to the volumes entering and exiting at those locations.

Formally, let $T^{\text{in}}(i)$ and $T^{\text{out}}(j)$ denote the total traffic that enters the network via i, and exits via j, respectively. The gravity model can then be computed by

$$T(i,j) = \frac{T^{\text{in}}(i)T^{\text{out}}(j)}{T^{\text{tot}}}, \tag{1.12}$$

where T^{tot} is the total traffic across the network. Implicitly, this model relies on a conservation assumption, i.e., traffic is neither created nor destroyed in the network so that $T^{\text{tot}} = \sum_k T^{\text{in}}(k) = \sum_k T^{\text{out}}(k)$. The assumption may be violated, for instance when congestion causes packet loss. However, in most backbones congestion is kept low, and so the assumption is reasonable.

In the form just described, the gravity model has distinct limitations. For instance, real traffic matrices may have non-constant f_{ij}, (perhaps as a result of different time-zones). Moreover, even if an origin destination traffic matrix matches the gravity model well, the ingress/egress TM may be systematically distorted [7]. Typically, networks use hot-potato routing, i.e., they choose the egress point closest to the ingress point, and this results in a systematic distortion of ingress/egress traffic matrices away from the simple gravity model. These distortions and others related to the asymmetry of traffic and distance sensitivity may be incorporated in

generalizations of the gravity model where sufficient data exists to measure such deviations [13, 21, 68].

The use of temporal patterns in planning is relatively obvious. The use of spatial patterns such as the gravity model is more subtle. The spatial structure gives us the capability to fill in missing values of the traffic matrix when our data is not perfect. Hence we can still plan our network, even in the extreme case where we have no data at all.

1.3.3 Application Profile

We have so far discussed network traffic along two dimensions: the temporal and spatial. There is a third aspect of traffic to consider: its application breakdown, or profile. Common applications on the Internet are email, web browsing (and other server based interactions), peer-to-peer file transfers, video and voice. Each may have a different traffic matrix, and as some networks move towards differentiated Quality of Service (QoS) for different classes of traffic, we may have to plan networks based on these different traffic matrices.

Even where differentiated service is not going to be provided, a knowledge of the application classes in our network can be very useful. For instance

- voice traffic is less variable than data, and so can require less overhead for sub-measurement interval variations;
- peer-to-peer applications typically generate more symmetric traffic than web traffic, and so downstream capacity (towards customer eyeballs) is likely to be more balanced when peer-to-peer applications dominate;
- we may be planning to eliminate some types of traffic in future networks (e.g. peer-to-peer traffic has often been considered to violate service agreements that prohibit running servers).

The breakdown of traffic on a network is not trivial to measure. As noted, typical flow level data collection includes TCP/UDP port numbers, and these are often associated to applications using the IANA (Internet Assigned Numbers Authority) list of registered ports[4]. However, the port numbers used today are often associated with incorrect applications because:

- Ports are not defined with IANA for all applications, e.g. some peer-to-peer applications.
- An application may use ports other than its well-known ports to circumvent access control restrictions,e.g., non-privileged users often run WWW servers on ports other than port 80, which is restricted to privileged users on most operating systems, while port 80 is often used for other applications (than HTTP) in order to work around firewalls.

[4] `http://www.iana.org/assignments/port-numbers`

- In some cases server ports are dynamically allocated as needed. For example, FTP allows the dynamic negotiation of the server port used for the data transfer. This server port is negotiated on an initial TCP connection which is established using the well-known FTP control port, but which would appear as a separate flow.
- Malicious traffic (e.g. DoS attacks) can generate a large volume of bogus traffic that should not be associated with the applications that normally use the affected ports.

In addition, there are some incorrect implementations of protocols, and ambiguous port assignments that complicate the problem. Better approaches to classification of traffic exist (e.g. [59]), but are not always implemented on commercial measurement systems.

Application profiles can be quite complex. Typical Internet providers will see some hundreds of different applications. However, there are two major simplifications we can often perform. The first is a clustering of applications into classes. QoS sometimes forms natural classes (e.g. real-time vs bulk-transfer classes), but regardless we can often group many applications into similarly structured classes, e.g., we can group a number of protocols (IMAP, POP, SMTP, ...) into one class "email". Common groupings are shown in Table 1.1, along with exemplar applications.

Class	example applications
bulk-data	FTP, FTP-Data
database access	Oracle, MySQL
email	IMAP, POP, SMTP
information	finger, CDDBP, NTP
interactive	SSH, Telnet
measurement	SNMP, ICMP, Netflow
network control	BGP, OSPF, DHCP, RSVP, DNS
news	NNTP
online gaming	Quake, Everquest
peer-to-peer	Kazaa, Bit-torrent
voice over IP	SIP, Skype
www	HTTP, HTTPS

Table 1.1 Typical application classes grouped by typical use.

There may be a larger number of application classes, and often there is a significant group of unknown applications, but a typical application profile is highly skewed. Again, it is common to see 80-20 or 90-10 rules. In these cases, it is common to focus attention on those applications that generate the most traffic, reducing the complexity of the profile.

However, care must be taken because some applications which generate relatively little traffic on average may be considered very important, and/or may generate high volumes of traffic for short bursts. There are several such examples in enterprise networks, for instance, consider a CEO's once-a-week company-wide broadcast, or nightly backups. Both generate a large amount of traffic, but in a relative short-time interval, so their proportion of the overall network traffic may be small.

More generally, much of the control-plane traffic (e.g. routing protocol traffic) in networks is relatively low volume, but of critical importance.

1.4 Prediction

There are two common scenarios for network planning:

1. incremental planning for network evolution,
2. green-fields planning.

In the first case, we have an existing network. We can measure its current traffic, and extrapolate trends to predict future growth. In combination with business data, quite accurate assessments of future traffic are possible. Typically, temporal models are sufficient for incremental network planning, though better results might be possible with recently developed full spatio-temporal models [52].

In green-fields planning, we have the advantage that we are not constrained in our network design. We may start from a clean-slate, without concerning ourselves with a legacy network. However, in such planning we have no measurements on which to base predictions. All is not lost, however, as we may exploit the spatial properties of traffic matrices in order to obtain predictions. We discuss each of these cases below.

There are other scenarios of concern to the network planner. For example

- Network mergers, for instance when two companies merge and subsequently combine their networks.
- Network migrations, for instance as significant services such as voice or frame-relay are migrated to operate on a shared backbone.
- Addition (or loss) of a large customer (say a broadband access provider, a major content provider, or a hosting center).
- A change in inter-domain routing relationships. For instance, the conversion of a customer to a peer would mean that traffic no longer transits from that peer, altering traffic patterns.

The impact of these types of event is obviously dependent on the relative volume of the traffic affected. Such events can be particularly significant for smaller networks, but it is not unheard of for them to cause unexpected demands on the largest networks (for instance the migration of an estimated half-million customers from Excite@home to AT&T in 2002[5]). However, the majority of such cases can be covered by one or both of the techniques below.

[5] `http://news.cnet.com/ExciteHome-to-shut-down-ATT-drops-bid/2100-1033_3-276550.html`

1.4.1 Prediction for incremental planning

Incremental planning involves extending, or evolving a current network to meet changing patterns of demands, or changing goals. The problem involves prediction of future network demands, based on extrapolation of past and present network measurements. The planning problems we encounter are often constrained by the fact that we can make only incremental changes to our network, i.e., we cannot throw away the existing network and start from a clean slate, but let us first consider the problem of making successful traffic predictions.

Obviously, our *planning horizon* (the delay between our planning decisions and their implementation) is critical. The shorter this horizon, the more accurate our predictions are likely to be, but the horizon is usually determined by external factors such as delays between ordering and delivery of equipment, test and verification of equipment, planned maintenance windows, availability of technical staff, and capital budgeting cycles. These are outside the control of the network planner, so we treat the planning horizon as a constant.

The planning horizon also suggests how much historical data is needed. It is a good idea to start with historical data extending several planning horizons into the past. Such a record not only allows better determination of trends, but also allows an assessment of the quality of our prediction process through analysis of past planning periods. If such data is unavailable, then we must consider green-fields planning (see Section 1.4.2), though informed by what measurements are available.

Given such a historical record, our primary means for prediction is temporal analysis of traffic data. That is, we consider the traffic measurements of interest (often a traffic matrix) as a set of time-series.

However, as noted earlier the more highly we aggregate traffic, the smaller its relative variance, and the easier it is to work with. As a result, it can be a good idea to predict traffic at a high level of aggregation, and then use a spatial model to break it into components. For instance, we might perform predictions for the total traffic in each region of our network, and then break it into components using the current traffic matrix percentages, rather than predicting each element of the traffic matrix separately.

There are many techniques for prediction, we concentrate here on just one, which works reasonably for a wide range of traffic, but we should note that as in all of the work presented here, the key is not the individual algorithms but their robust application through a process of measurement, planning and validation.

1.4.1.1 Extracting the long-term trend

We will exploit the previously presented temporal model for traffic, and note that the key to providing predictions for use in planning is to estimate the long-term trend in the data. We could form such an estimate simply by aggregating our time-series over periods of one week (to average away the diurnal and weekly cycles) and then performing standard trend analysis. However, knowledge of the cycles in traffic

data is often useful. Sometimes we design networks to satisfy the demand during a "busy hour". More generally though, the busiest hours for different components of the traffic may not match (particularly in international networks distributed over several time-zones), and so we need to plan our network to have sufficient capacity at all hours of the day or night.

Hence, the approach we present provides the capability to estimate both the long-term trend, and the seasonal components of the traffic. It also allows an estimate of the peakedness, providing the ability to estimate the statistical variations around the expected traffic behavior. The method is hardly the only applicable time-series algorithm for this type of analysis (for another example see [44]), but it has the advantage of being relatively simple. The method is based on a simple signal processing tool, the *Moving Average* (MA) filter, which we discuss in detail below.

The moving average can be thought of as a simple low-pass filter as it "passes" low-frequencies, or long-term behavior, but removes short-term variations. As such it is ideally suited to extracting the trend in our traffic data. Although there are many forms of moving average, we shall restrict our attention to the simplest: a rectangular moving average

$$MA_x(t;n) = \frac{1}{2n+1}\sum_{s=t-n}^{s=t+n} x(s), \tag{1.13}$$

where n is the width of the filter, and $2n+1$ is its length. The length of the filter must be longer than the period of the cyclic component in order to filter out that component. Longer filters are often used to allow for averaging out of the stochastic variation as well. The shortest filter we should consider for extracting the trend is three times the period, which in Internet traffic data is typically one week. For example, given traffic data $x(t)$, measured in one hour intervals, we could form our estimate $\hat{L}(t)$ of the trend by taking a filter of length 3 weeks (e.g., $2n+1 = 504 = 24\times 7\times 3$), i.e., we might take $\hat{L}(t) = MA_x(t;252)$ where MA_x is defined in (1.13).

Care must always be taken around the start and end of the data. Within n data points of the edges the MA filter will be working with incomplete data, and so these estimates should be discounted in further analysis.

Once we have obtained estimates for the long-term trend, we can model its behavior. Over the past decade, the Internet has primarily experienced exponential growth (for instance see Figure 1.4 or [45]).

$$L(t) = L(0)e^{\beta t}, \tag{1.14}$$

where $L(0)$ is the starting value, and β is the growth rate. If exponential growth is suspected the standard approach is to transform the data using the log function so that we see

$$\log L(t) = \log L(0) + \beta t, \tag{1.15}$$

where we can now estimate $L(0)$ and β can be estimated from linear regression of the observed data. Care should obviously be taken that this model is reasonable. Regression provides diagnostic statistics to this end, but comparisons to other models (such as a simple linear model) can also be helpful.

Such a model can be easily extrapolated to provide long-term predictions of traffic volumes. Standard diagnostics from the regression can also be used to provide confidence bounds for the predictions, allowing us to predict "best" and "worst" case scenarios for traffic growth, and an example of such predictions is given in Figure 1.4 using the data from 2000-2004 to estimate the trend, and then extrapolating this until 2009. The figure shows the extrapolated optimistic and pessimistic trend estimates. We can see that actual traffic growth from 2005-2007 was on the optimistic side of growth, but that in 2008 the measured traffic was again close to the long-term trend estimate.

This example clearly illustrates that understanding the potential variations in our trend estimate is almost as important as obtaining the estimate in the first place. It also illustrates how instructive historical data can be in assessing appropriate models and prediction accuracy.

Often, in traffic studies, managers are keen to know the *doubling time*, the time it takes traffic to double. This can be easily calculated by estimating the value of t such that $L(t) = 2L(0)$, or $e^{\beta t} = 2$. Again, taking logs we get the doubling time

$$t^* = \frac{1}{\beta} \ln 2. \tag{1.16}$$

The Australian data shown in Figure 1.4 has a doubling time of 477 days.

The trend by itself can inform us of growth rate but modelling the cyclic variations in traffic is also useful. We do this by extending the concept of moving average to the *seasonal moving average*, but before doing so we broadly remove the long-term trend from the data (by dividing our measurements $x(t)$ by $\hat{L}(t)$).

1.4.1.2 Extracting the the cyclical component

The goal of a Seasonal Moving Average (SMA) is to extract the cyclic component of our traffic. We know, *a priori*, the period (typically 7 days) and so the design of a filter to extract this component is simple. It resembles the MA used previously in that it is an average, but in this case it is an average of measurements separated in time by the period. More precisely we form the SMA of the traffic with the estimated trend removed, e.g.,

$$\hat{S}(t) = \frac{1}{N} \sum_{n=0}^{N-1} x(t + nT_S)/\hat{L}(t + nT_S), \tag{1.17}$$

where T_S is the period, and NT_S is the length of the filter. In effect the SMA estimates the traffic volume for each time of day and week as if they were separate time series. It can be combined with a short MA filter to provide some additional smoothing of the results if needed.

The advantage of using a SMA as opposed to a straight forward seasonal average is that the cyclical component of network traffic can change over time. Using the SMA allows us to see such variability, while still providing a reasonably stable

model for extrapolation. There is a natural tradeoff between the length of the SMA, and the amount of change we allow over time (longer filters naturally smooth out transient changes). Typically, the length of filter desired depends on the planning horizon under which we are operating. We extrapolate the SMA in various ways, but the simplest is to repeat the last cycle measured in our data into the future, as if the cyclical component remained constant into the future. Hence, when operating with a short planning horizon (say a week), we can allow noticeable week to week variations, and still obtain reasonable predictions, and so a filter length of 3 to 4 cycles is often sufficient. Where our planning horizon is longer (say a year) we must naturally assume that the week to week variations in the cyclical behavior are smaller in order to extrapolate, and so we use a much longer SMA, preferably at least of the order of the length of the planning horizon.

1.4.1.3 Estimating the magnitude of random variations

Once we understand the periodic and trend components of the traffic, the next thing to capture is the random variation around the mean. Most metrics of variation used in capacity planning do not account for the time-varying component, and so are limited to busy-hour analysis. In comparison, we now have have an estimate of $\hat{m}(t) = \hat{L}(t)\hat{S}(t)$ and so can use (1.6) to estimate the stochastic or random component of our traffic by $z(t) = (x(t) - \hat{m}(t))/\sqrt{\hat{m}(t)}$. We can now measure the variability of the random component of the traffic using the variance of $z(t)$, which forms an estimate $\hat{a}$ for the traffic's peakedness. The estimator for $\hat{a}$ including the correction for bias is given in [58]. Note that it is also important to separate the impulsive, anomaly terms from the more typical variations. There are many anomaly detection techniques available (see [67] for a review of a large group of such algorithms). These algorithms can be used to select anomalous data points that can then be excluded from the above analysis.

1.4.1.4 From traffic matrix to link loads

Once we have predictions of a TM, we often need to use these to compute the link loads that would result. The standard approach is to write the TM in vectorized form $\mathbf{x}$, where the vector $\mathbf{x}$ consists of the columns of the TM (at a particular time) stacked one on top of another. The link loads $\mathbf{y}$ can then be estimated through the equation

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{1.18}$$

where A is the routing matrix. The equation above can also be extended to project observations or predictions of a TM over time into equivalent link loads.

Although there are multiple time-series approaches that can be used to predict future behavior (e.g., Holt-Winters [11]), our approach has the advantage that it naturally incorporates multiplexing. As a result, equation (1.18) can be extended to

other aspects of the traffic model. For instance, the variances of independent flows are additive (the variance of the multiplexed traffic is the sum of the variances of the components), and so the variance of link traffic follows the same relationship, i.e.,

$$\mathbf{v_y} = A\mathbf{v_x}, \tag{1.19}$$

where $\mathbf{v}_y$ and $\mathbf{v}_x$ are the variances of the link loads and TM, respectively. We can use $\mathbf{v}_y$ to deduce peakedness parameters for the link traffic using (1.7).

So far, we have assumed that the network (at least the location of links, and the routing) is static. In reality, part of network planning involves changing the network, and so the matrix A is really a potential variable. When we consider network planning, A appears implicitly as one of our optimization variables. Likewise, A may change in response to link or router failures.

The reason traffic matrices are so important is that they are, in principle, *invariant* under changes to A. Hence predictions of link loads under the changes in A can be easily made. For example, imagine a traffic engineering problem where we wish to balance the load on a network's internal links more effectively. We will change routing in the network in order to balance the traffic on links more effectively. In doing so, the link loads are not invariant (the whole point of traffic engineering is to change these). However the ingress/egress TM is invariant, and projecting this onto the links (via the routing matrix) will predict the link loads under proposed routing changes.

In reality invariance is an approximation. Real TMs are not invariant under all network changes, for instance, if network capacities are chosen to be too small, congestion will result. However, the Transmission Control Protocol (TCP) will act to alleviate this congestion by reducing the actual traffic carried on the network, thereby changing the traffic matrix. In general different sets of measurements will have different degrees of invariance. For instance, an origin/destination TM is invariant to changes in egress points (due to routing changes) whereas an ingress/egress TM is not. It is clearly better to use the right data set for each planning problem, but the desired data is not always available.

The lack of true invariance is one of the key reasons for the cyclic approach to network planning. We seek to correct any problems caused by variations in our inputs in reponse to our new network design.

1.4.2 Prediction for Green-fields planning

The above assumes that we have considerable historical data to which we apply time-series techniques to extrapolate trends, and hence predict the future traffic demands on our network. This has two major limitations:

1. IP traffic is constrained by the pipe through which it passes. TCP congestion control ensures that such traffic does not overflow by limiting the source transmission rate. In most networks our measurements only provide the *carried load*

not the *offered load*. If the network capacities change, the traffic may increase in response. This is a concern if our current network is loaded to near its capacity, and in this case we must discount our measurements, or at least treat them with caution.
2. When we design a new network there is nothing in place for us to measure.

We will start by considering available strategies for the latter case. We can draw inspiration from the spatial models previously presented. The fact that the simple gravity model describes, to some extent, the spatial structure of Internet traffic matrices presents us with a simple approach to estimate an initial traffic matrix.

The first step is to estimate the total expected traffic for the network, based on demographics and market projections. Let us take a simple example: in Australia the ABS measures internet usage. Across a wide customer base the average usage per customer was roughly 3 GB/month (since 2006). The total traffic for our network is the usage per customer multiplied by the projected number of customers. We can derive traffic estimates per marketing region in the same fashion. Note that the figure used above is for the broad Australian market and is unlikely to be correct elsewhere (typical Australian ISPs have an tiered pricing structure). Where more detailed figures exist in particular markets these should be used.

The second step is to estimate the "busy hour" traffic. As we have seen previously the traffic is not uniformly distributed over time. In the absence of better data, we might look at existing public measurements (such as presented in Figures 1.2 and 1.3, or as appears in [44]) where the peak to mean ratio is of the order of three to two. Increasing our traffic estimates by this factor gives us an estimate of the peak traffic loads on the network.

The third step is to estimate a traffic matrix. The best approach, in the absence of other information, to derive the traffic matrix is to apply the gravity model (1.12). In the simple case, the gravity model would be applied directly using the local regional traffic estimates. However, where additional information about the expected application profile exists, we might use this to refine the results using the "independent flow model" of [21]. Additional structural information about the network might allow use of the "generalized gravity model" of [69]. Each of these approaches allows us to use additional information, but in the absence of such information the simple gravity model gives us our initial estimate of the network traffic matrix.

What about the case where we have historical network traffic measurements, but suspect that the network is congested so that the carried load is significantly below the offered load? In this case, our first step is to determine what parts of the traffic matrix are affected. If a large percentage of the traffic matrix is affected, then the only approach we have available is to go back through the historical record until we reach a point (hopefully) where the traffic is not capacity constrained. This has limitations: for one thing, we may not find a sufficient set of data where capacity constraints have left the measurements uncorrupted. Even where we do obtain sufficient data, the missing (suspect) measurements increase the window over which we must make predictions, and therefore the potential errors in these predictions.

However, if only a small part of the traffic matrix is affected we may exploit techniques developed for traffic matrix inference to fill in the suspect values with

more accurate estimates. These methods originated due to the difficulties in collecting flow-level data to measure traffic matrices directly. Routers (particularly older routers) may not support an adequate mechanism for such measurements (or suffer a performance hit when the measurements are used), and installation of stand-alone measurement devices can be costly. On the other hand, the Simple Network Management Protocol (SNMP) is almost ubiquitously available, and has little overhead. Unfortunately, it provides only link-load measurements, not traffic matrices. However, the two are simply related by (1.18). Inferring $\mathbf{x}$ from $\mathbf{y}$ is a so-called "network tomography" problem. For a typical network the number of link measurements is $O(N)$ (for a network of N nodes), whereas the number of traffic matrix elements is $O(N^2)$ leading to a massively underconstrained linear inverse problem. Some type of side-information is needed to solve such problems, usually in the form of a model that roughly describes a typical traffic matrix. We then estimate the parameters of this crude model (which we shall call $\mathbf{m}$), and perform a regularization with respect to the model and the measurements by solving the minimization problem

$$\underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda^2 \mathbf{d}(\mathbf{x}, \mathbf{m}), \tag{1.20}$$

where $\|\cdot\|_2$ denotes the l^2 norm, $\lambda > 0$ is a regularization parameter, and $d(\mathbf{x}, \mathbf{m})$ is a distance between the model $\mathbf{m}$ and our estimated traffic matrix $\mathbf{x}$. Examples of suitable distance metrics are standard or weighted Euclidean distance and the Kullback-Leibler divergence. Approaches of this type, generally called *strategies for regularization of ill-posed problems* are more generally described in [29], but have been used in various forms in many works on traffic matrix inference. The method works because the measurements leave the problem underconstrained, thereby allowing many possible traffic matrices that fit the measurements, but the model allows us to choose one of these as best. Furthermore, through λ the method allows us to trade-off our belief about the accuracy of the model against the expected errors in the measurements.

We can utilize TM structure to interpolate missing values by solving a similar optimization problem

$$\underset{\mathbf{x}}{\operatorname{argmin}} \|\mathscr{A}(\mathbf{x}) - \mathbf{M}\|_2^2 + \lambda^2 \mathbf{d}(\mathbf{x}, \mathbf{m_g}), \tag{1.21}$$

where $\mathscr{A}(x) = M$ expresses the available measurements as a function of the traffic matrix (whether these be link measurements or direct measurements of a subset of the TM elements we don't care), and $\mathbf{m_g}$ is the gravity model. This regularizes our model with respect to the measurements that are considered valid. Note that the gravity model in this approach will be skewed by missing elements, so this approach is only suitable for interpolation of a few elements of the traffic matrix. If larger numbers of elements are missing, we can use more complicated techniques such as those proposed in [51] to interpolate the missing data.

1.5 Optimal Network Plans

Once we have obtained predictions of the traffic on our network we can commence the actual process of making decisions about where links and routers will be placed, their capacities, and the routing policies that will be used. In this section we discuss how we may optimize these quantities against a set of goals and constraints.

The first problem we consider concerns capacity planning. If this component of our network planning worked as well as desired, we could stop there. However errors in predictions, coupled with the long planning horizon for making changes to a network mean that we need also to consider a short-term way of correcting such problems. The solution is typically called *traffic engineering* or simply load balancing, and is considered in Section 1.5.2.

1.5.1 Network Capacity Planning

There are many good optimization packages available today. Commercial tools such as CPLEX are designed specifically for solving optimization problems, while more general purpose tools such as Matlab often include optimization toolkits that can be used for such problems. Even Excel includes some quite sophisticated optimization tools, and so we shall not consider optimization algorithms in detail here. Instead we will formulate the problem, and provide insight into the practical issues. There are three main components to any optimization problem: the variables, the objective, and the constraints.

The variables here are obviously the locations of links, and their capacities.

The objective function — the function which we aim to minimize — varies depending on business objectives. For instance, it is common to minimize the cost of a network (either its capital or ongoing cost), or packet delays (or some other network performance metric). The many possible objectives in network design result in different problem formulations, but we concentrate here on the most common objective of cost minimization.

The cost of a network is a complex function of the number and type of routers used, and the capacities of the links. It is common, however, to break up the problem hierarchically into inter-PoP (Point-of-Presence), and intra-PoP design, and we consider the two separately here.

The constraints in the problem fall into several categories:

1. Capacity constraints require that we have “sufficient” link capacity. These are the key constraints for this problem so we consider these in more detail below.
2. Other technological constraints, such as limited port numbers per router.
3. Constraints arising as a result of the difficulties in multi-objective optimization. For example, we may wish to have a network with good performance and low cost. However, multiobjective optimization is difficult, so instead we minimize cost subject to a constraint on network performance.

4. Reliability constraints require that the network function even under network failures. This issue is so important that other chapters of this book have been devoted to this issue, but we shall consider some aspects of this problem here as well.

1.5.1.1 Capacity constraints and safe-operating points

Unsurprisingly, the primary constraints in capacity planning are the capacity constraints. We must have a network with sufficient capacity to carry the offered traffic. The key issue is our definition of "sufficient". There are several factors that go into this decision:

1. Traffic is not constant over the day, so we must design our network to carry loads at all times of day. Often this is encapsulated in "busy hour" traffic measurements, but busy hours may vary across a large network, and between customers, and so it is better to design for the complete cycle.
2. Traffic has observable fluctuations around its average behavior. Capacity planning can explicitly allow for these variations.
3. Traffic also has unobservable fluctuations on shorter times than our measurement interval. Capacity planning must attempt to allow for these variations.
4. There will be measurement and prediction errors in any set of inputs.

Ideally, we would use queueing models to derive an exact relationship between measured traffic loads, variations, and so determine the required capacities. However, despite many recent advances in data traffic modelling, we are yet to agree on sufficiently precise and general queueing models to determine sufficient capacity from numerical formulae. There is no "Erlang-B" formulae for data networks. As a result, most network operators use some kind of engineering rule of thumb, which comes down to an "over-engineering factor" to allow for the above sources of variability.

We adopt the same approach here, but the term "over-engineering factor" is misleading. The factor allows for *known* variations in the traffic. The network is not over-engineered, it only appears so if capacity is directly compared to the available but flawed measurements. In fact, if we follow a well founded process, the network can be quite precisely engineered[6].

We therefore prefer to use the term *Safe Operating Point* (SOP). A SOP is defined statistically with respect to the available traffic measurements on a network. For instance, with five minute SNMP traffic measurements, we might define our SOP by requiring that the load on the links (as measured by five minute averages) should not exceed 80% of link capacity more than five times per month. The predicted traffic model could then be used to derive how much capacity is needed to achieve this bound.

Traffic variance depends on the application profile and the scale of aggregation. Moreover, the desired tradeoff between cost and performance is a business choice

[6] It is a common complaint that backbone networks are underutilized. This complaint typically ignores the issues described above. In reality many of these networks may be quite precisely engineered, but crude average utilization numbers are used to defer required capacity increases.

for network operators. So there is no single SOP that will satisfy all operators. Given the lack of precision in current queueing models and measurements, the SOP needs to be determined by each network operator experimentally, preferably starting from conservative estimates. Natural variations in network conditions often allow enough scope to see the impact of variable levels of traffic, and from these determine more accurate SOP specifications, but to do this we need to couple traffic and performance measurements (a topic we consider later).

A secondary set of capacity constraints arises because there is a finite set of available link types, and capacity must be bought in multiples of these links. For instance, many high-speed networks use either SONET/SDH links (typically giving 155 Mbps times powers of 4) and/or Ethernet link capacities (powers of 10 from 10 Mbps to 10 Gbps). We will denote the set of available link capacities (including zero) by C.

Finally, most high-speed links technologies are duplex, and so we need to allocate capacity in each direction, but we typically do so symmetrically (i.e., a link has the same capacity from $i \rightarrow j$ as from $j \rightarrow i$ even when the traffic loads in each direction are different).

1.5.1.2 Intra-PoP design

We divide the network design or capacity planning problem into two components and first consider the design of the network inside a PoP. Typically this involves designing a tree-like network to aggregate traffic up to regional hubs, which then transit the traffic onto a backbone[7]. The exact design of a PoP is considered in more detail in Chapter 4, but note that in each of the cases considered there we end up with a very similar optimization problems at this level.

There are two prime considerations in such planning. Firstly, it is typical that the majority of traffic is non-local, i.e., that it will transit to or from the backbone. Local traffic between routers within the PoP in the Internet is often less than 1% of the total. There are exceptions to this rule, but these must be dealt with on an individual basis. Secondly, limitations on the number of ports on most high-speed routers mean that we need at least one layer of aggregation routers to bring traffic onto the backbone: for instance see Figure 1.5. For clarity, we show a very simple design (see Chapter 4 for more examples). In our example, Backbone Routers (BRs) and the corresponding links to Aggregation Routers (ARs) are assigned in pairs in order to provide redundancy, but otherwise the topology is a simple tree.

There are many variations on this design, for instance additional BRs may be needed, or multiple layers. However in our simple model, the design is determined primarily by the limitations on port density. The routers lie within a single PoP, so links are short and their cost has no distance dependence (and they are relatively cheap compared to wide-area links). The number of ARs that can be accommo-

[7] In small PoPs, a single router (or redundant pair) may be sufficient for all needs. Little planning is needed in this case beyond selecting the model of router, and so we do not include this simple case in the following discussions.

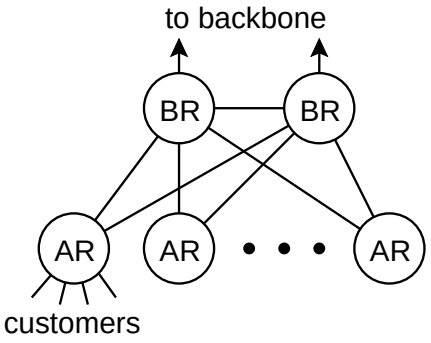

Fig. 1.5 A typical PoP design. Aggregation Routers (AR) are used to increase the port density in the PoP and bring traffic up to the Backbone Routers (BR).

dated depends on the number of ports that can be supported by the BRs, so we shall assume that ARs have a single high-capacity uplink to each BR to allow for a maximum expansion factor in a one-level tree. As a result, the job of planning a PoP is primarily one of deciding how many ARs are needed.

As noted earlier we don't need a TM for this task. The routing in such a network is predetermined, and so current port allocations and the uplink load history are sufficiently invariant for this planning task. We use these to form predictions of future uplink requirements and the loads on each router. When predictions show that a router is reaching capacity (either in terms of uplink capacity, traffic volume, or port usage) we can install additional routers based on our predictions over the planning horizon for router installation.

There is an additional improvement we can make in this type of problem. It is rare for customers to use the entire capacity of their link to our network, and so the uplink capacity between AR and BR in our network need not be the sum of the customers' link capacities. We can take advantage of this fact through simple measurement-based planning, but with the additional detail that we may allocate customers with different traffic patterns to routers in such a way as to leverage different peak hours and traffic asymmetries (between input and output traffic), so as to further reduce capacity requirements.

The problem resembles the bin packing problem. Given a fixed link capacity C for the uplinks between ARs and BRs, and K customers with peak traffic demands $\{T_i\}_{i=1}^{K}$, the bin packing problem would be as follows: determine the smallest integer B, such that we can find a B-partition $\{S_k\}_{k=1}^{B}$ of the customers[8] such that

$$\sum_{i \in S_k} T_i \leq C, \text{ for all } k = 1, \ldots, B. \tag{1.22}$$

[8] A B-partition of our customers is a group of B non-empty subsets $S_k \subset \{1, 2, \ldots, K\}$ that are disjoint, i.e., $S_i \cap S_j = \phi$ for all $i \neq j$, and which include all customers, i.e., $\cup_{k=1}^{B} S_k = \{1, 2, \ldots, K\}$.

The number of subsets B gives the number of required ARs, and although the problem is NP-hard, there are reasonable approximation algorithms for its solution [18], some of which are online, i.e., they can be implemented without reorganization of existing allocations.

The real problem is more complicated. There are constraints on the number of ports that can be supported by ARs dependent on the model of ARs being deployed, constraints on router capacity, and in addition, we can take advantage of the temporal, and directional characteristics of traffic. Customer demands take the form $[I_i(t), O_i(t)]$, where $I_i(t)$ and $O_i(t)$ are incoming and outgoing traffic demands for customer i at time t. So the appropriate condition for our problem is to find the minimal number B of ARs such that

$$\sum_{i \in S_k} I_i(t) \leq C, \text{ and } \sum_{i \in S_k} O_i(t) \leq C, \text{ for all } k,t. \tag{1.23}$$

This is the so-called *vector bin packing* problem, which has been used to model resource constrained processor scheduling problems, and good approximations have been known for some time [15, 28].

The major advantage of this type of approach is that customers with different peak traffic periods can be combined onto one AR so that their joint traffic is more evenly distributed over each 24 hour period. Likewise, careful distribution of customers whose primary traffic flows *into* our network (for instance hosting centers) together with customers whose traffic flows *out of* the network (e.g., broadband access companies) can lead to more symmetric traffic on the uplinks, and hence better overall utilization. In practice, multiplexing gains may improve the situation, so that less capacity is needed when multiple customers' traffic is combined, but this effect only plays a dominant role when large numbers (say hundreds) of small customers are being combined.

1.5.1.3 Inter-PoP backbone planning

The inter-PoP backbone design problem is somewhat more complicated. We start by assuming we know the locations at which we wish to have PoPs. The question of how to optimize these locations does come up, but it is common that these locations are pre-determined by other aspects of business planning. In inter-PoP planning, distance based costs are important. The cost of a link is usually considered to be proportional to its length, though this is approximate. The real cost of a link has a fixed component (in the equipment used to terminate a line) in addition to distance dependent terms derived from the cost to install a physical line, e.g., costs of cables, excavation and right of ways. Even where leased lines are used (so there are minimal installation costs) the original capital costs of the lines are usually passed on through some type of distance sensitive pricing.

In addition, higher speed links generally cost more. The exact model for such costs can vary, but a large component of the bandwidth dependent costs is in the end equipment (router interface cards, WDM mux/demux equipment, etc.). In ac-

tuality real costs are often very complicated: vendors may have discounts for bulk purchases, whereas cutting edge technology may come at a premium cost. However, link costs are often approximated as linear with respect to bandwidth because we could, in principle, obtain a link with capacity $4c$ by combining four links of capacity c.

In the simple case then, cost per link has the form

$$f(d_e, c_e) = \alpha + \beta d_e + \gamma c_e, \tag{1.24}$$

where α is the fixed cost of link installation, β is the link cost per unit distance and γ is the cost per unit bandwidth. As the distance of a link is typically a fixed property of the link, we often rewrite the above cost in the form

$$f_e(c_e) = \alpha_e + \gamma c_e, \tag{1.25}$$

where now the cost function depends on the link index e.

We further simplify the problem by assuming that BRs are capable of dealing with all traffic demands so that only two (allowing for redundancy) are needed in each PoP, thus removing the costs of the router from the problem.

Finally, we simplify our approach by assuming that routes are chosen to follow the shortest possible geographic path in our network. There are reasons (which we shall discuss in the following section) why this might not be the case, however, *a priori*, it makes sense to use the shortest geographic path. There are costs that arise from distance. Most obviously, if packets traverse longer paths, they will experience longer delays, and this is rarely desirable. In addition, packets that traverse longer paths use more resources. For instance, a packet that traverses two hops rather than one uses up capacity on two links rather than one.

As noted earlier, we need to specify the problem constraints, the basic set of which are intended to ensure there is sufficient capacity in the network. When congestion is avoided, queueing delays will be minimal, and hence delays across the network will be dominated by propagation delays (the speed of light cannot be increased). So ensuring sufficient capacity implicitly serves the purpose of reducing networking delays. As noted, we adopt the approach of specifying a SOP, which we do in the form of a factor $\lambda \in (0,1)$, which specifies the traffic limit with respect to capacity. That is, we shall require that the link capacity c_e be sufficient that traffic takes up only λ of the capacity, leaving $1-\lambda$ of the capacity to allow for unexpected variations in the traffic.

The possible variables are now the link locations and their capacities. So, given the (vectorized) traffic matrix $\mathbf{x}$, our job is to determine link locations and capacities c_e, which implicitly defined the network routes (and hence the routing matrix A), such that we solve

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{e\in E} \alpha_e I(c_e > 0) + \gamma c_e \\ \text{such that} & A\mathbf{x} \le \lambda \mathbf{c}, \\ & c_e \in C, \end{array} \tag{1.26}$$

where $A\mathbf{x} = \mathbf{y}$, the link loads, $\mathbf{c}$ is the vector of links capacities, E is the set of possible links, $I(c_e > 0)$ is an indicator function (which is 1 where we build a link, and 0 otherwise), and C is the set of available link capacities (which includes 0).

Implicit in the above formulation is the routing matrix A, which results from the particular choice of links in the network design, so A is in fact a function of the network design. Its construction imposes constraints requiring that all traffic on the network can be routed. The problem can be rewritten in a more explicit form using flow-based constraints, but the above formulation is convenient for explaining the differences and similarities between the range of problems we consider here.

There may be additional constraints in the above problem resulting from router limitations, or due to network performance requirements. For instance, if we have a maximum throughput on each router, we introduce a set of constraints of the form $B\mathbf{x} \leq \eta\mathbf{r}$, where $\mathbf{r}$ are router capacities, and B is similar to a routing matrix in that it maps end-to-end demands to the routers along the chosen path. Port constraints on a router might be expressed by taking constraints of the form $\sum_j I(c_{i,j} > 0) \leq p_i$, where p_i is the port limit on router i. Port constraints are complicated by the many choices of line cards available for high-speed routers, and so have sometimes been ignored, but they are a key limitation in many networks. The issue is sometimes avoided by separation of inter- and intra-PoP design, so that a high port density on BRs is not needed.

The other complication is that we should aim to optimize the network for 24×7 operations. We can do so simply by including one set of capacity constraints for each time of day and week, i.e., $A\mathbf{x}_t \leq \lambda\mathbf{c}$. The resulting constraints are in exactly the same form as in (1.26) but their number increases. However, it is common that many of these constraints are redundant, and so can be removed from the optimization (without effect) by a pre-filtering phase.

The full optimization problem is a linear integer program, and there are many tools available for solution of such programs. However, it is not uncommon to relax the integer constraints to allow any $c_e \geq 0$. In this case there is no point in having excess capacity, and so we can replace the link capacity constraint by $A\mathbf{x} = \lambda\mathbf{c}$. We then obtain the actual design by rounding up the capacities. This approach reduces the numerical complexity of the problem, but results in a potentially suboptimal design. Note though, that integer programming problems are often NP hard, and consequently solved using heuristics which likewise can lead to suboptimal designs. Relaxation to a linear program is but one of a suite of techniques that can be used to solve problems in this context, often in combination with other methods.

Moreover, it is common the mathematical community to focus on finding provably optimal designs, but this is not a real issue. In practical network design we know that the input data contains errors, and our cost models are only approximate. Hence, the mathematically optimal solution may not have the lowest cost of all realizable networks. The mathematical program only needs to provide us with a very good network design.

The components of real network suffer outages on a regular basis: planned maintenance, and accidental fiber cuts are simple examples (for more details see Chapters 3 and 4). The final component of network planning that we discuss here is relia-

bility planning: analyzing the reliability of a network. There are many algorithms aimed at maintaining network connectivity, ranging from simple designs such as rings or meshes, through to formal optimization problems including connectivity constraints. Commonly, networks are designed to survive all single link or node outages, though more careful planning would concern all Shared Risk Groups (SRG), i.e., groups of links and/or nodes who share fates under common failures. For instance, IP links that use wavelengths on the same fiber will all fail simultaneously if the fiber is cut.

However, when a link (or SRG) fails, maintaining connectivity is not the only concern. Rerouted traffic creates new demands on links. If this demand exceeds capacity, then the resulting congestion will negatively impact network performance. Ideally, we would design our network to accommodate such failures, i.e., we would modify our earlier optimization problem (1.26) as follows:

$$\begin{array}{rl} \text{minimize} & \sum_{e \in E} \alpha_e I(c_e > 0) + \gamma c_e \\ \text{such that} & A\mathbf{x} \leq \lambda \mathbf{c}, \\ \text{and} & A_i \mathbf{x} \leq \zeta \mathbf{c}, \ \ \forall i \in \mathscr{F}, \end{array} \tag{1.27}$$

where $\mathscr{F}$ is the set of all failure scenarios considered likely enough to include, and A_i is the routing matrix under failure scenario i. Naively implemented with $\lambda = \zeta$, this approach has the limitation that the capacity constraints under failures can come to dominate the design of the network so that most links will be heavily underutilized under normal conditions. Hence, we allow that the SOPs with respect to normal loads, and failure loads to be different, $\lambda < \zeta < 1$, so that the mismatch is somewhat balanced, i.e., under normal conditions links are not completely underutilized, but there is likely to be enough capacity under common failures. For example, we might require that under normal loads, peak utilizations remain at 60%, while under failures, we allow loads of 85%.

Additionally, the number of possible failure scenarios can be quite large, and as each introduces constraints, it may not be practical to consider all failures. We may need to focus on the likely failures, or those that are considered to be most potentially damaging. However, it is noteworthy that only constraints that involve rerouting need be considered. In most failures, a large number of links will be unaffected, and hence the constraints corresponding to those links will be redundant, and may be easily removed from the problem.

The above formulation presumes that we design our network from scratch, but this is the exception. We typically have to grow our network incrementally. This introduces challenges — for instance, it is easy to envisage a series of incremental steps that are each optimal in themselves, but which result in a highly suboptimal network over time. So it is sometimes better to design an optimal network from scratch, particularly when the network is growing very quickly. In the mean time we can include the existing network through a set of constraints in the form $c_e \geq l_e + c'_e$, where l_e is the legacy link capacity on link e, and c'_e is the additional link capacity. The real situation is complicated by some additional issues: (i) typical IP router load-balancing is not well suited for multiple parallel links of different capacities so

we must choose between increasing capacity through additional links (with capacity equal to the legacy links) or paying to replace the old links with a single higher capacity link; and (ii) the costs for putting additional capacity between two routers may be substantially different from the costs for creating an entirely new link. Some work [40] has considered the problem of evolvability of networks, but without all of the addition complexities of IP network management, so determining long-term solutions for optimal network evolution is still an open problem.

1.5.2 Traffic Engineering

In practice, it takes substantial time to build or change a network, despite modern innovations in reconfigurable networks. Typical changes to a link involve physically changing interface cards, wiring, and router configurations. Today these changes are often made manually. They also need to be performed carefully, through a process where the change is documented, carefully considered, acted upon, and then tested. The time to perform these steps can vary wildly between companies, but can easily be 6 months once budget cycles are taken into account.

In the mean time we might find that our traffic predictions are in error. The best predictions in the world cannot cope with the convulsive changes that seem to occur on a regular basis in the Internet. For instance, the introduction of peer-to-peer networking both increased traffic volumes dramatically in a very short time frame, and changed the structure of this traffic (peer-to-peer traffic is more symmetric that the previously dominant client-server model). YouTube again reset providers' expectations for traffic. The result will be a suboptimal network, in some cases leading to congestion.

As noted, we cannot simply redesign the network, but we can often alleviate congestion by better balancing loads. This process, called *traffic engineering* (or just load balancing) allows us to adapt the network on shorter time scales than network planning. It is quite possible to manually intervene in a network's traffic engineering on a daily basis. Even finer time scales are possible in principle if traffic engineering is automated, but this is uncommon at present because there is doubt about the desirability of frequent changes in routing. Each change to routing protocols can require a reconvergence, and can lead to dropped packets. More importantly, if such automation is not very carefully controlled it can become unstable, leading to oscillations and very poor performance.

The Traffic Engineering (TE) problem is very similar to the network design problem. The goal, or optimization objective is often closely related to that in design. The constraints are usually similar. The major difference is in the planning horizon (typically days to weeks), and as a result the variables over which we have control. The restriction imposed by the planning horizon for TE is that we cannot change the network hardware: the routers and links between them are fixed. However, we can change the way packets are routed through the network, and we can use this to rebalance the traffic across the existing network links.

There are two methods of TE that are most commonly talked about. The most often mentioned uses MultiProtocol Label Switching (MPLS) [54], by which we can arbitrarily tunnel traffic across almost any set of paths in our network. Finding a general routing minimizing max-utilization is an instance of the classical multi-commodity flow problem which can be formulated as a linear program [6, Chapter 17], and is hence solvable using commonly available tools. We shall not spend much time on MPLS TE, because there is sufficient literature already (for instance see [19,36]). We shall instead concentrate on a simpler, less well known, and yet almost as powerful method for TE.

Remember that we earlier argued that shortest-geographic paths made sense for network routing. In fact, shortest-path routing does not need to be based on geographic distances. Most modern Interior Gateway Protocols allow administratively defined distances (for instance Open Shortest Path First (OSPF) [42] and Intermediate System-Intermediate System (IS-IS) [14]). By tweaking these distances we can improve network performance. By making a link distance smaller, you can make a link more "attractive", and so route more traffic on this link. Making the distance longer can remove traffic. Configurable link weights can be used, for example, to direct traffic away from expensive (e.g., satellite) links.

However, we can formulate the TE problem more systematically. Let us consider a shortest-path protocol with administratively configured link *weights* (the link distances) w_e on each link e. We assume that the network is given (i.e., we know its link locations and capacities), and that the variables that we can control are the link weights. Our objective is to minimize the congestion on our network. Several metrics can be used to describe congestion. Network-wide metrics such as that proposed in [25, 26] can have advantages, but we use the common metric of maximum utilization here for its simplicity.

In many cases, there are additional "human" constraints on the weights we can use in the above optimization. For instance, we may wish that the resulting weights don't change "too much" from our existing weights. Each change requires reconfiguration of a router, and so reducing the number of changes with respect to the existing routing may be important. Likewise the existing weights are often chosen not just for the sake of distance, but also to make the network conceptually simpler. For instance, we might choose smaller weights inside a "region" and large weights between regions, where the regions have some administrative (rather than purely geographical) significance. In this case, we may wish to preserve the general features of the routing, while still fine tuning the routes. We can express these constraints in various ways, but we do so below by setting minimum and maximum values for the weights. Then the optimization problem can be written: choose the weights $\mathbf{w}$, such that we

$$\begin{aligned} \text{minimize} \quad & \max_{e \in E} y_e/c_e \\ \text{such that} \quad & A\mathbf{x} = \mathbf{y}, \\ \text{and} \quad & w_e^{\min} \le w_e \le w_e^{\max}, \quad \forall e \in E \end{aligned} \tag{1.28}$$

where A is the routing matrix generated by shortest-path routing given by link weights w_e, and the link utilizations are given by y_e/c_e (the link load divided by

its capacity). The $w_e^{\min}$ and $w_e^{\max}$ constrain the weights for each link into a range determined by existing network policies (perhaps within some bound of the existing weights). Additional constraints might specify the maximum number of weights we are allowed to change, or require that links weights be symmetric, i.e., $w_{(i,j)} = w_{(j,i)}$.

The problem is in general NP-hard, so it is non-trivial to find a solution. Over the years, many heuristic methods [12,20,25,26,37,41,53] have been developed for the solution of this problem.

The exciting feature of this approach is that it is very simple. It uses standard IP routing protocols, with no enhancements other than the clever choice of weights. One might believe that the catch was that it cannot achieve the same performance as full MPLS TE. However, the performance of the above shortest-path optimization has been shown on real networks to suffer only by a few percent [60,61], and importantly, it has been shown to be more robust to errors in the input traffic matrices than MPLS optimization [61]. This type of robustness is critical to real implementations.

Moreover, the approach can be used to generate a set of weights that work well over the whole day (despite variations in the TM over the day) [61], or that can help alleviate congestion in the event of a link failure [44], a problem that we shall consider in more detail in the following section.

1.6 Robust Planning

A common concern in network planning is the consequence of mistakes. Traffic matrices used in our optimizations may contain errors due to measurement artifacts, sampling, inference, or predictions. Furthermore there may be inconsistencies between our planned network design, and the actual implementation through misconfiguration or last minute changes in constraints. There may be additional inconsistencies introduced through the failure of invariance in TMs used as inputs, for example, caused by congestion alleviation in the new network.

Robust planning is the process of acknowledging these flaws, and still designing good networks. The key to robustness is the cyclic approach described in the introduction: measure → predict → plan → and then measure again. However, with some thought, this process can be made tighter. We have already seen one example of this through TE, where a short-term alteration in routing is used to counter errors in predicted traffic. In this section we shall also consider some useful additions to our kitbag of robust planning tools.

1.6.1 Verification Measurements

One of the most common sources of network problems is misconfiguration. Extreme cases of misconfigurations that cause actual outages are relatively obvious (though still time consuming to fix). However, misconfigurations can also result in more

subtle problems. For instance, a misconfigured link weight can mean that traffic takes unexpected paths, leading to delays or even congestion.

One of the key steps to network planning is to ensure that the network we planned is the one we observe. Various approaches have been used for router configuration validation: these are considered in more detail in Chapter 9. In addition, we recommend that direct measurements of the network routing, link loads, and performance be made at all times. Routing can be measured through mechanisms such as those discussed earlier in Section 1.2 and in more detail in Chapter 11. When performed from edge node to edge node, we can use such measurements to confirm that traffic is taking the routes we intended it to take in our design.

By themselves, routing measurements only confirm the direction of traffic flows. Our second requirement is to measure link traffic to ensure it remains within the bounds we set in our network design. Unexpected traffic loads can often be dealt with by TE, but only once we realize there is a problem.

Finally, we must always measure performance across our network. In principle, the above measurements are sufficient, i.e., we might anticipate that a link is congested only if traffic exceeds the capacity. However, in reality, the typical SNMP measurements used to measure traffic on links are five minute averages. Congestion can occur on smaller time scales, leading to brief, but non-negligible packet losses that may not be observable from traffic measurements alone. We aim to reduce these through choice of SOP, but note that this choice is empirical in itself, and an accurate choice relies on feedback from performance measurements. Moreover, other components of a network have been known to cause performance problems even on a lightly loaded network. For instance, such measurements allowed us to discover and understand delays in routing convergence times [32, 62], and that during these periods bursts of packet loss would occur, from which improvements to Interior Gateway Protocols have been made [27]. The importance of the problem would never have been understood without performance measurements. Such measurements are discussed in more detail in Chapter 10.

1.6.2 Reliability Analysis

IP networks and the underlying SONET/WDM strata on which they run are often managed by different divisions of a company, or by completely different companies. In our planning stages, we would typically hope for joint design between these components, but the reality is that the underlying physical/optical networks are often multiuse, with IP as one of several customers (either externally or internally) that use the same infrastructure. It is often hard to prescribe exactly which circuits will carry a logical IP link. Therefore, it is hard in some cases to determine, prior to implementation, exactly what SRG exist.

We may insist, in some cases, that links are carried over separate fibers, or even purchase leased lines from separate companies, but even in these cases great care should be taken. For instance, it was only during the Baltimore train tunnel fire

(2001) [4] that it was discovered that several providers ran fiber through the same tunnel.

Our earlier network plan can only accommodate planned network failure scenarios. In robust planning, we must somehow accommodate the SRGs that have arisen in the implementation of our planned network. The first step, obviously, is to determine the SRGs. The required data mapping IP links to physical infrastructure is often stored in multiple databases, but with care it is possible to combine the two to obtain a list of SRGs. Once we have a complete list of failure scenarios we could go through the planning cycle again, but as noted, the time horizon for this process would leave our network vulnerable for some time.

The first step therefore is to perform a network reliability analysis. This is a simple process of simulating each failure scenario, and assessing whether the network has sufficient capacity, i.e., whether $A_i\mathbf{x} \leq \boldsymbol{\zeta}\mathbf{c}$. If this condition is already satisfied, then no action need be taken. However, where the condition is violated, we must take one of two actions. The most obvious approach to deal with a specific vulnerability is to expedite an increase in capacity. It is often possible to reduce the planning horizon for network changes at an increased cost. Where small changes are needed, this may be viable, but it is clearly not satisfactory to try to build the whole network in this way.

The second alternative is to once again use traffic engineering. MPLS provides mechanisms to create failover paths, however, it does not tell you where to route these to ensure congestion does not occur. Some additional optimization and control is needed. However, we cannot do this after the failure, or recovery will take an unacceptable amount of time. Likewise, it is impractical in today's networks to change link weights in response failures. However, previous studies have shown that shortest-path link weight optimization can be used to provide a set of weights that will alleviate congestive effects under failures [44], and such techniques have (anecdotally) been used in large networks with success.

1.6.3 Robust Optimization

The fundamental issue we deal with is "Given that I have errors in my data, how should I perform optimization?" Not all the news is bad. For instance, once we acknowledge that our data is not perfect, we realize that finding the mathematically optimal solution for our problem is not needed. Instead, heuristic solutions that find a near optimal solution will be just as effective. This chapter is not principally concerned with optimization, and so we will not spend a great deal of time on specific algorithms, but note that once we decide that heuristic solutions will be sufficient, several meta-heuristics such as genetic algorithms and simulated annealing become attractive. They are generally easy to program, and very flexible, and so allow us to use more complex constraints and optimization objective functions than we might otherwise have chosen. For instance, it becomes easy to incorporate the true link costs, and technological constraints on available capacities.

The other key aspect to optimization in network planning directly concerns robustness. We know there are errors in our measurements and predictions. We can save much time and effort in planning if we accommodate some notion of these errors in our optimization. A number of techniques for such optimization have been proposed: oblivious routing [8], and Valiant network design [70, 71]. These papers present methods to design a network and/or its routing so that it will work well for any arbitrary traffic matrix. However, this is perhaps going too far. In most cases we do have some information about possible traffic whose use is bound to improve our network design.

A simple approach is to generate a series of possible traffic matrices by adding random noise to our predicted matrix, i.e., by taking $\mathbf{x}_i = \mathbf{x} + \mathbf{e}_i$, for $i = 1, 2, \ldots, M$. Where sufficient historical data exists, the noise terms $\mathbf{e}_i$ should be generated in such a way as to model the prediction errors. We can then optimize against the set of TMs, i.e.,

$$\begin{aligned} &\text{minimize} \quad \sum_{e \in E} \alpha_e I(c_e > 0) + \gamma c_e \\ &\text{such that} \quad A\mathbf{x}_i \leq \lambda \mathbf{c}, \ \ \forall i = 1, 2, \ldots, M. \end{aligned} \tag{1.29}$$

Once again this can increase the number of constraints dramatically, particularly in combination with reliability constraints, unless we realize that again many of these constraints will be redundant, and can be pruned by preprocessing.

The above approach is somewhat naive. The size of the set of TMs to use is not obvious. Also we lack guidance about the choice we should make for λ. In principle, we already accomodate variations explicitly in the above optimization and so we might expect $\lambda = 1$. However, as before we need $\lambda < 1$ to accomodate inter-measurement time interval variations in traffic, though the choice should be different than in past problems.

Moreover, there may be better robust optimization strategies that can be applied in the future. For instance, robust optimization has been applied to the traffic engineering problem in [66], where the authors introduce the idea of COPE (Commoncase Optimization with a Penalty Envelope) where the goal is to find the optimal routing for a predicted TM, and to ensure that the routing will not be "too bad" if there are errors in the prediction.

1.6.4 Sensitivity Analysis

Even where we believe that our optimization approach is robust, we must test this hypothesis. We can do so by performing a sensitivity analysis. The standard approach in such an analysis is to vary the inputs and examine the impact on the outputs. We can vary each possible input to detect robustness to errors in this input, though the most obvious to test is sensitivity to variations in the underlying traffic matrix. We can test such sensitivity by considering the link loads under a set of TMs generated, as before, by adding prediction errors errors, i.e., $\mathbf{x}_i = \mathbf{x} + \mathbf{e}_i$, for $i = 1, 2, \ldots, M$, and then simply calculating the link loads $\mathbf{y}_i = A\mathbf{x}_i$. There is an obvi-

ous relationship to robust optimization, in that we should not be testing against the same set of matrices against which we optimized. Moreover, in sensitivity analysis it is common to vary the size of the errors. However, simple linear algebra allows us to reduce the problem to a fixed load component $\mathbf{y} = A\mathbf{x}$ and a variable component $\mathbf{w}_i = A\mathbf{e}_i$, which scales linearly with the size of the errors, and which can be used to see the impact of errors in the TM directly.

1.7 Summary

"Reliability, reliability, reliability" is the mantra of good network operators. Attaining reliability costs money, but few companies can afford to waste millions of dollars on an inefficient network. This chapter is aimed at demonstrating how we can use robust network planning to attain efficient but reliable networks, despite the imprecision of measurements, uncertainties of predictions, and general vagaries of the Internet.

Reliability should mean more than connectivity. Network performance measured in packet delay or loss rates is becoming an important metric for customers deciding between operators. Network design for reliability has to account for possible congestion caused by link failures. In this chapter we consider methods for designing networks where performance is treated as part of reliability.

The methodology proposed here is built around a cyclic approach to network design exemplified in Figure 1.1. The process of **measure** → **analyze/predict** → **control** → **validate** should not end, but rather, validation measurements are fed back into the process so that we can start again. In this way, we attain some measure of robustness to the potential errors in the process. However, the planning horizon for network design is still quite long (typically several months) and so a combination of techniques such as traffic engineering are used at different time scales to ensure robustness to failures in predicted behavior. It is the combination of this range of techniques that provides a truly robust network design methodology.

Acknowledgement

This work was informed by the period M.Roughan was employed at AT&T research, and the author owes his thanks to researchers there for many valuable discussions on these topics. M.Roughan would also like to thank the Australian Research Council from whom he receives support, in particular through grant DP0665427.

References

1. Python routing toolkit ('pyrt'). http://ipmon.sprintlabs.com/pyrt/.
2. Ripe NCC: routing information service. http://www.ripe.net/projects/ris/.
3. University of Oregon Route Views Archive Project. www.routeviews.org.
4. CSX train derailment. Nanog mailing list: http://www.merit.edu/mail.archives/nanog/2001-07/msg00351.html, 18th July 2001.
5. Abilene/Internet2. http://www.internet2.edu/observatory/archive/data-collections.html#netflow.
6. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, New Jersey, 1993.
7. D. Alderson, H. Chang, M. Roughan, S. Uhlig, and W. Willinger. The many facets of Internet topology and traffic. *Networks and Heterogeneous Media*, 1(4):569–600, December 2006.
8. D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *ACM SIGCOMM*, pages 313–324, Karlsruhe, Germany, August 2003.
9. G. E. P. Box and N. R. Draper. *Response Surfaces, Mixtures and Ridge Analysis*. Wiley, 2nd edition, 2007.
10. P. Brockwell and R. Davis. *Time Series: Theory and Methods*. Springer-Verlag, 1987.
11. J. D. Brutag. Aberrant behavior detection and control in time series for network monitoring. In *Proceedings of the 14th Systems Administration Conference (LISA 2000)*, New Orleans, LA, USA, December 2000. USENIX.
12. L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup. A memetic algorithm for OSPF routing. In *Proc. 6th INFORMS Telecom*, pages 187–188, 2002.
13. R. S. Cahn. *Wide Area Network Design*. Morgan Kaufman, 1998.
14. R. Callon. Use of OSI IS-IS for routing in TCP/IP and dual environments. Network Working Group, Request for Comments: 1195, December 1990.
15. C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM J. Comput.*, 33(4):837–851, 2004.
16. N. Duffield and C. Lund. Predicting resource usage and estimation accuracy in an IP flow measurement collection infrastructure. In *ACM SIGCOMM Internet Measurement Conference*, Miami Beach, Florida, October 2003. Available at http://www.icir.org/vern/imc-2003/program.html.
17. N. Duffield, C. Lund, and M. Thorup. Flow sampling under hard resource constraints. *SIGMETRICS Perform. Eval. Rev.*, 32(1):85–96, 2004.
18. J. E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation Algorithms for Bin Packing: A Survey. In D. Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing, 1997.
19. A. Elwalid, C. Jin, S. H. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *INFOCOM*, pages 1300–1309, 2001.
20. M. Ericsson, M. Resende, and P. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *J. Combinatorial Optimization*, 6(3):299–333, 2002.
21. V. Erramilli, M. Crovella, and N. Taft. An independent-connection model for traffic matrices. In *ACM SIGCOMM Internet Measurement Conference (IMC06)*, pages 251–256, New York, NY, USA, 2006. ACM.
22. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford. Netscope: Traffic engineering for IP networks. *IEEE Network Magazine*, pages 11–19, March/April 2000.
23. A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, pages 265–279, June 2001.
24. A. Feldmann and J. Rexford. IP network configuration for intradomain traffic engineering. *IEEE Network Magazine*, pages 46–57, September/October 2001.
25. B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. 19th IEEE Conf. on Computer Communications (INFOCOM)*, pages 519–528, 2000.
26. B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

27. P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving sub-second IGP convergence in large IP networks. *SIGCOMM Comput. Commun. Rev.*, 35(3):35–44, 2005.
28. M. Garey, R. Graham, D. Johnson, and A. Yao. Resource constrained scheduling as generalized bin packing. *J. Comb. Theory A*, 21:257–298, 1976.
29. P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1997.
30. G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures over an IP backbone. In *ACM SIGCOMM Internet Measurement Workshop*, Marseilles, France, November 2002.
31. J. Kowalski and B. Warfield. Modeling traffic demand between nodes in a telecommunications network. In *ATNAC'95*, 1995.
32. C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. In *Proceedings of ACM SIGCOMM*, 2000.
33. A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *ACM SIGCOMM Internet Measurement Conference*, Taormina, Sicily, Italy, October 2004.
34. A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM*, 2004.
35. A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *ACM SIGMETRICS / Performance*, 2004.
36. U. Lakshman and L. Lobo. MPLS traffic engineering. Cisco Press `http://www.ciscopress.com/articles/article.asp?p=426640`, 2006.
37. F. Lin and J. Wang. Minimax open shortest path first routing algorithms in networks supporing the SMDS services. In *Proc. IEEE International Conference on Communications (ICC)*, volume 2, pages 666–670, 1993.
38. D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg. Routing design in operational networks: A look from the inside. In *ACM SIGCOMM*, Portland, OR, USA, 2004.
39. D. R. Mauro and K. J. Schmidt. *Essential SNMP*. O'Reilly, 2001.
40. N. F. Maxemchuk, I. Ouveysi, and M. Zukerman. A quantitative measure for comparison between topologies of modern telecommunications networks. In *IEEE Globecom*, 2000.
41. D. Mitra and K.G.Ramakrishnan. A case study of multiservice, multipriority traffic engineering design for data networks. In *Proc. IEEE GLOBECOM*, pages 1077–1083, 1999.
42. J. T. Moy. OSPF version 2. Network Working Group, Request for Comments: 2328, April 1998.
43. I. Norros. A storage model with self-similar input. *Queueing Systems*, 16:387–396, 1994.
44. A. Nucci and K. Papagiannaki. *Design, Measurement and Management of Large-Scale IP Networks*. Cambrigde University Press, 2009.
45. A. M. Odlyzko. Internet traffic growth: Sources and implications. In B. B. Dingel, W. Weiershausen, A. K. Dutta, and K.-I. Sato, editors, *Optical Transmission Systems and Equipment for WDM Networking II*, volume 5247, pages 1–15. Proc. SPIE, 2003.
46. T. Oetiker. MRTG: the multi-router traffic grapher. `http://oss.oetiker.ch/mrtg//`.
47. T. Oetiker. RRDtool. `http://oss.oetiker.ch/rrdtool/`.
48. V. Paxson. Strategies for sound Internet measurement. In *ACM Sigcomm Internet Measurement Conference (IMC)*, Taormina, Sicily, Italy, October 2004.
49. R. B. Potts and R. M. Oliver. *Flows in Transportation Networks*. Academic Press, 1972.
50. P. Pyhnen. A tentative model for the volume of trade between countries. *Weltwirtschaftliches Archive*, 90:93–100, 1963.
51. L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *ACM SIGCOMM*, pages 151–162, 2003.
52. L. Qui, Y. Zhang, M. Roughan, and W. Willinger. Spatio-temporal compressive sensing and Internet traffic matrices. In *to appear in ACM Sigcomm*, August 2009.
53. K. Ramakrishnan and M. Rodrigues. Optimal routing in shortest-path data networks. *Lucent Bell Labs Technical Journal*, 6(1), 2001.
54. E. C. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Network Working Group, Request for Comments: 3031, 2001.
55. M. Roughan. Simplifying the synthesis of Internet traffic matrices. *ACM SIGCOMM Computer Communications Review*, 35(5):93–96, October 2005.

56. M. Roughan and J. Gottlieb. Large-scale measurement and modeling of backbone Internet traffic. In *SPIE ITCOM*, Boston, MA, USA, 2002.
57. M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in measuring Internet backbone traffic variability: Models, metrics, measurements and meaning. In *Proceedings of the International Teletraffic Congress (ITC-18)*, pages 221–230, 2003.
58. M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang. Experience in measuring Internet backbone traffic variability: Models, metrics, measurements and meaning. In *Proceedings of the International Teletraffic Congress (ITC-18)*, pages 379–388, Berlin, Germany, 2003.
59. M. Roughan, S. Sen, O. Spatscheck, and N. Duffield. Class-of-Service Mapping for QoS: A statistical signature-based approach to IP traffic classification. In *ACM SIGCOMM Internet Measurement Workshop*, pages 135–148, Taormina, Sicily, Italy, 2004.
60. M. Roughan, M. Thorup, and Y. Zhang. Performance of estimated traffic matrices in traffic engineering. In *ACM SIGMETRICS*, pages 326–327, San Diego, CA, USA, 2003.
61. M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 248–258, Miami Beach, FL, USA, 2003.
62. A. Shaikh and A. Greenberg. Experience in black-box OSPF measurement. In *Proc. ACM SIGCOMM Internet Measurement Workshop*, pages 113–125, 2001.
63. A. Shaikh and A. Greenberg. OSPF Monitoring: Architecture, Design and Deployment Experience. In *Proc. USENIX Symposium on Networked System Design and Implementation (NSDI)*, March 2004.
64. J. Tinbergen. Shaping the world economy: Suggestions for an international economic policy. The Twentieth Century Fund, 1962.
65. S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1):83–86, January 2006.
66. H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. COPE: traffic engineering in dynamic networks. In *ACM SIGCOMM*, pages 99–110, 2006.
67. Y. Zhang, Z. Ge, M. Roughan, and A. Greenberg. Network anomography. In *Proceedings of the Internet Measurement Conference (IMC '05)*, Berkeley, CA, USA, October 2005.
68. Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *ACM SIGMETRICS*, pages 206–217, San Diego, California, June 2003.
69. Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *ACM SIGCOMM*, pages 301–312, Karlsruhe, Germany, August 2003.
70. R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone. In *HotNets III*, San Diego, CA, November 2004. `http://tiny-tera.stanford.edu/~nickm/papers/index.html`.
71. R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone with Valiant load-balancing. In *Thirteenth International Workshop on Quality of Service (IWQoS)*, Passau, Germany, June 2005. `http://tiny-tera.stanford.edu/~nickm/papers/index.html`.